Information Systems Vol. 15, No. 5, pp. 583–590, 1990
Printed in Great Britain. All rights reserved

0306-4379/90 $3.00 + 0.00
Copyright © 1990 Pergamon Press plc

A FUZZY DATABASE-QUERY LANGUAGE

M. H. Wong and K. S. Leung
Computer Science Department, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

(*Received* 14 *September* 1988; *received for publication* 19 *June* 1990)

Abstract—Selection conditions for conventional database-query languages are not natural enough to express criteria with fuzzy concepts. To choose some alternatives from a database requires the techniques of multicriteria decision making or conflict resolution. However, some traditional techniques in this area still have some drawbacks, such as inefficiency and independency of criteria. In this paper a fuzzy database-query language is presented and its position in the area of multicriteria decisions making in database applications is discussed and identified. The selection constraints of this query language can accept both crispy and fuzzy conditions.

Key words: Fuzzy query language, multicriteria decision making, conflict resolution

1. INTRODUCTION

Crispy conditions are used in conventional database-query languages for selecting alternatives. However, they are not natural enough to express criteria with fuzzy concepts. For example, a buyer is looking for a flat for two people with an expected price around $300,000 from the database of an estate agency. The word "around" and the size of the flat for two people are fuzzy concepts, so it is not reasonable to interpret them using traditional crispy query conditions. The buyer certainly would not want to miss a flat with size $= 51\ \text{m}^2$ and price $= \$311{,}000$ because of the constraint set for size is "$40\ \text{m}^2 < \text{size} < 50\ \text{m}^2$" or the constraint set for price is "price $< \$310{,}000$". Therefore, it is important to incorporate fuzzy retrieval capabilities in a database query language to make it more natural and powerful. Moreover, sometimes the criteria of a query condition may be supplemented by each other. For instance, if the location of a flat is not satisfactory, but its prize is low, the flat may also be considered as a possible alternative. However, such a kind of query is difficult to implement using traditional query conditions. Therefore, a more sophisticated query language is required to handle this kind of query.

This kind of query involving considerations of criteria in different dimensions can be classified as multicriteria decision making. Criteria are defined as measures, rules and standards that guide human choices and decision making [1]. Sometimes, the criteria are conflicting with each other and multiple decision making units (decision makers) with different points of view may be involved. A more specific term "conflict resolution" is used to categorize the techniques for multicriteria decision making which involves conflicting criteria and decision making units.

In general, such a query is a decision problem which may be reduced into one of the following three problems [2]:

(1) selecting "the best" alternative;
(2) to select a subset of alternatives which are considered as "good"

and

(3) ranking all the valid alternatives from the "best" to the "worst".

In this paper, a fuzzy query language is presented. This language is designed to retrieve information from an existing database management system (DBMS), VAX Rdb/VMS. The overall architecture is depicted in Fig. 1. The fuzzy retrieval module has two functions. The first function is to translate the fuzzy database-query to a query for VAX Rdb/VMS. The other function is to perform multicriteria decision making based on the information retrieved from the DBMS.

In the next three sections, some classical methods for multicriteria decisions making and conflict resolution are discussed. In Section 5, the DBMS (VAX Rdb/VMS) adopted in the implementation of the fuzzy database-query language is briefly described. The details of the fuzzy retrieval module are given in Section 6. In Section 7, an evaluation of the fuzzy database-query language and a comparison with traditional methods are presented. The last section is the conclusion.

2. MULTIATTRIBUTES UTILITY THEORY

This theory is based on the hypothesis that in any problem there exists a real valued function V defined on those alternatives which the decision maker wishes to maximize [3]. Based on this function,

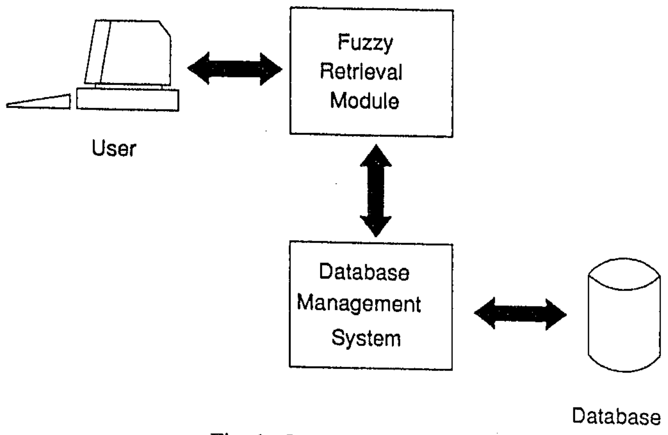

Fig. 1. Overall architecture.

the alternatives can be ranked from the "best" to the "worst". Hence, the "best" alternative or a subset of alternatives which are considered as "good" can be chosen easily.

The function V can be defined in many ways. Some common methods to define the function V are discussed in the following subsections.

2.1. *Additive utility function*

The most common approach for evaluating the multicriteria alternatives is to use an additive representation:

$$V(X) = v_1(X) + v_2(X) + \cdots + v_n(X),$$

where V is the function for evaluating the alternatives and $v_i(i = 1 \ldots n)$ is the function to determine the utility of X with respect to the criterion i.

A fundamental assumption of this representation is that of preferential independence and utility independence (different independence). Preferential independence concerns ordinal preferences among attributes, while utility independence concerns the cardinal preferences of the decision maker.

Let $F = \{v_1, v_2, \ldots, v_n\}$ and $K \subset F$: K is preferentially independent in F if the preferences between alternatives, which are only different for criteria in K, do not depend on the criteria in $F \backslash K$ (the subset of F not belonging to K) [2]. In the case of additive utility functions, the number of elements for the set K is 1. Essentially, preferential independence implies that the indifference curves for any pair of criteria are unaffected by the fixed levels of the remaining criteria.

For example, suppose a buyer prefers a house described by 3-tuple (price, size, location), say, (\$30,000, 400 m^2, good) to one described by (\$30,000, 450 m^2, good), he should also prefer (\$35,000, 400 m^2, fair) to (\$35,000, 450 m^2, fair) if sizes are preferentially independent of costs and locations, where $K = \{\text{size}\}$ and $F \backslash K = \{\text{cost, location}\}$.

An attribute X is said to be utility independent of a set of attributes Y if the decision maker's preferences among alternatives, involving only X and with Y fixed at a particular level, do not depend on the level of Y.

For example, if a decision maker prefers a flat with size 450 m^2 to one with 400 m^2 and the size is utility independent of the price and the location, then the preference difference between (\$35,000, 450 m^2, good) and (\$35,000, 400 m^2, good) must be equal to the preference difference between (\$30,000, 450 m^2, fair) and (\$30,000, 400 m^2, fair), where $X = \{\text{size}\}$ and $Y = \{\text{price, location}\}$.

Although the applications of additive utility functions are very wide, there are still some drawbacks. The main drawback is that criteria expressed by the function are restricted to preferentially independent and utility independent only. However, this hypothesis is unrealistic. Some queries with criteria that are dependent on each other are difficult to model with additive functions only. For example, when choosing a flat for a cripple, one of the requirements may be either the flat is on the ground floor or there is a lift in the building. Therefore the utilities of the two criteria are not independent, since when one criterion is satisfied, the utility of the other criterion will becone zero. Thus additive utility function cannot be applied directly.

Besides, for additive functions, there must also exist functions $W_{i,j}$ (trade-off factor) measuring the amount that the decision maker is willing to concede on the jth criterion to obtain a unit on the ith criterion. Thus the additive function cannot model query with criteria that cannot be replaced or traded-off by other criteria.

2.2. *Utility function based on the concept of ideal point*

Another type of utility function is to find the distance of an alternative from an ideal point (solution). Usually, an ideal point is the point which has the coordinates

$$\max_{x \in A} v_1(x), \max_{x \in A} v_2(x), \ldots, \max_{x \in A} v_n(x),$$

in the criteria space.

Therefore, an idea point is a point in the criteria space to represent an ideal alternative which has maximum utility in each dimension. However, this kind of function also assumes preference and utility independences. Besides, a trade-off function also exist between any two criteria. The only difference is that this function is not linear. Thus, the utility function based on the concept of idea points have the same disadvantages as additive utility functions.

In some systems, such as the conflict resolution approach [4–7], fuzzy concept is introduced into the ideal point, since in most of the conflict environment, the information available is vague and not complete. Besides, the decision making process is always inexact in nature. Therefore, a fuzzy region demarcated by fuzzy intervals is used to represent an ideal solution. The distance of an alternative to the fuzzy ideal solution is determined by a membership function representing a concept such as "close to".

3. SELECT A SUBSET OF ALTERNATIVES WHICH ARE CONSIDERED "GOOD"

This kind of method is designed to select a subset of alternatives which are considered "good". Their main disadvantage is that they cannot be used to find the best alternative nor to rank all the alternatives. Two of the common methods are presented in the following subsections:

3.1. Efficient solution

An alternative a is efficient for the set of criteria $\{v_1, v_2, \ldots, v_n\}$ if there is no alternative b in A (the set of alternatives) which dominates a. Where dominance is defined as follows:

Given a set of criteria $\{v_1, v_2, \ldots, v_n\}$, an alternative a dominates an alternative b iff $v_i(a) \geqslant v_i(b)$ for all i and at least one of the $v_i(a)$ being strictly larger than $v_i(b)$.

For this given method, each alternative has to compare with all the other alternatives. An alternative can only be put into the set of efficient solutions only if it is not dominated by any other alternative. The time complexity of the process is proportional to the square of the number of available alternatives.

3.2. Outranking methods

Outranking method is, in fact, an improvement of the efficient solution. This method consists of two distinct stages. The first stage is to construct outranking relations among the alternatives. An outranking relation is used for two related alternatives, say a and b, such that a is "better" than b or vice versa. The second stage is to select a subset of alternatives based on the outranking relations.

An alternative a outranks b if the following two requirements are satisfied [2]:

(1) There is a sufficient majority among the criteria to consider that a is not worse than b.

(2) No criterion in disagreement with this majority should result in too great a superiority of b over a.

The above two requirements can be implemented by different functions. For example in ELECTRE I [8], the concordance indicator to measure how much an alternative a is better than b is defined as follows:

$$c(a, b) = \frac{1}{\pi} \sum_{i \in C_{ab}} \pi_i,$$

where C_{ab} is the set of criteria for which a is preferred to or indifferent from b, π_i is the weighting factor for criterion v_i and

$$\pi = \sum_{i=1}^{n} \pi_i.$$

The discordance indicator to measure how much an alternative a is worse than b is defined as follows:

$$d(a, b) = \begin{cases} 0, & \text{if } D_{ab} = \varnothing, \\ \dfrac{1}{\delta} \max\limits_{i \in D_{ab}} |v_i(b) - v_i(a)|, & \text{otherwise,} \end{cases}$$

where D_{ab} is the set of criteria for which a is worse than b, and δ is the maximum difference on a particular criterion.

We say that a outranks b (aSb) iff

(1) $c(a, b) \geqslant p$

and

(2) $d(a, b) \leqslant q$,

where p and q are some constants.

From the above definitions, we notice that the outranking relation is not transitive. That is, if a outranks b and b outranks c, a may not outrank c. After the outranking relations have been constructed, it is required to determine a subset E of A such that each element of $A \backslash E$ is outranked by at least one element of A, but the elements of E do not outrank each other. That is

(1) $\forall b \in A \backslash E, \exists a \in E: aSb$,

and

(2) $\forall a, b \in E$: not aSb and not bSa.

The advantage of this kind of method is that it is less subjective. The disadvantage is that it only selects a subset of alternatives which are considered as good or builds a preference tree to show the preference relations. Therefore it only reduces the number of possible alternatives, but does not remove the burden for selecting the best or rank all the alternatives for users.

Moreover, the time complexity to build the preference tree or to get the set of efficient solutions is proportional to the square of the number of the alternatives, since each alternative has to compare with all the other alternatives.

4. INTERACTIVE METHODS

Interactive methods require users to be involved in the decision making. This method consists of two stages. The first stage is the calculation stage at which an alternative is selected. The second stage is the discussion stage at which users have to provide supplementary information about his preferences. The additional information is then introduced into the model in the next calculation stage. This two stage is repeated until an acceptable solution is obtained or the user stops the process because no acceptable solution is found.

For example, in the STEM method of Benayoun *et al.* [9], an alternative a is selected at the calculation stage based on the distance between the image point (alternative) and the ideal point. If the decision maker is not satisfied with this solution, he can specify to which criterion v_j he is prepared to make a concession and the maximum amount d_j he is willing to concede. Then the calculation stage is repeated to find another alternative b with the following constraints

$$\vdots$$

(1) $v_i(b) \geqslant v_i(a)$, for all $i <> j$,

(2) $v_j(b) \geqslant v_j(a) - d_j$.

These two stages are repeated until an acceptable solution is obtained, or it can be concluded that no acceptable solution exists.

The advantage of this method is that it can adapt different requirements of different users, therefore it is less subjective. However, it cannot automate the decision process completely, and waste a lot of time in the iterations. Moreover, it does not remove the burden of decision making from users, since the msot difficult tasks are to determine which criterion is to be conceded and how much it should be conceded.

5. DATABASE MANAGEMENT SYSTEM

Basically, the design of the fuzzy query system is independent of the type of the database system used. The only requirement is that the database system must be accessible from the outside world through some standard programming languages. Otherwise, an interface cannot be built to communicate with it.

In the implementation of this system, a VAX Rdb/VMS database system is used. It is a relational database management system for VAX computers that uses the VMS operating system. The details of the VAX information architecture may be obtained from Ref. [10]. Existing DBMSs are chosen simply because it is not practical to build a database system purposely and transfer all the data into it before one can use it.

An Rdb/VMdatabase system provides the Callable RDO utility which can be called from any language that conforms to the VAX procedure calling and condition handling standards [11]. Besides, Rdb/VMS provides a very powerful data retrieving facility with record selection and relation joining capabilities. That is why Rdb/VMS is chosen as an existing DBMS to demonstrate the flexibility and power of the fuzzy database-query language.

To retrieve information from Rdb/VMS database systems through Callable RDO utility, the combinations of START_STREAM, FETCH and GET commands are used. The command START_STREAM is to open a record stream. The records to be included in the record stream are specified by the record selection expression in the START_STREAM statement. START_STREAM places the stream pointer just before the first record in the stream. The FETCH command can be used to advance the stream pointer a record at a time. The GET statement is then used to get the fields in the record.

All the Rdb-statements are passed to Rdb/VMS database system through the procedure RDB$INTERPRET provided by the Callable RDO interface facility. The Rdb/VMS statements are passed to the system as string literals or string variables. The parameter placeholder !VAL is used to reserve a place for each of the host variables. The corresponding host variables are placed in the parameter list that follows the Rdb/VMS statement string.

6. FUZZY INFORMATION RETRIEVAL MODULE

The fuzzy information retrieval module is, in fact, an interface between users and the database system. Its function is to support a simple fuzzy database-query language for users to retrieve information from a database under certain selection conditions. The selection conditions may include both fuzzy and nonfuzzy concepts. The operations of this module can be divided into three phases. In the first phase, the fuzzy query of a user is translated into the query language of the Rdb/VMS database system as described in Section 5. In the second phase, the Callable RDO facility is invoked to retrieve those required data from the database. In the last phase, the data are processed to obtain the degrees of membership as required. Besides, the records are sorted in descending order according to the degrees of membership. Those records with degrees of membership of zero will be eliminated.

The syntax of the fuzzy query language is defined in extended BNF Grammar as follows:

```
QUERY                  ::= ((relation_name)SYMBOL_LIST
                           NON_FUZZY_CONDITION FUZZY_CONDITION)
SYMBOL_LIST            ::= STAT_EXP | ((field_name)field_name)*)
NON_FUZZY_CONDITION    ::= NON_FUZZY_CON | nil
FUZZY_CONDITION        ::= FUZZY_CON | nil
STAT_EXP               ::= (STAT_OP NON_FUZZY_CONDITION)
NON_FUZZY_CON          ::= (OPERATOR TERM TERM*)
FUZZY_CON              ::= (FUZ_OPERATOR FUZZY_EXP FUZZY_EXP*)
STAT_OP                ::= max〈field_name〉 |
                           min〈field_name〉 |
                           total〈field_name〉 |
                           average〈field_name〉 |
                           count
TERM                   ::= NON_FUZZY_CON |
                           (COMPARATOR〈field_name〉〈value〉) |
                           (COMPARATOR〈field_name〉STAT_EXP)
FUZZY_EXP              ::= FUZZY_CON | FUZZY_TERM
FUZZY_TERM             ::= ((field_name)
                           〈membership_distribution_table〉〈weight〉)
OPERATOR               ::= and | or
FUZ_OPERATOR           ::= and | or | comb |poll
COMPARATOR             ::= < | > | = | <> | >= | <=
```

Where

〈field_name〉 is the name of a field in the database;
〈relation_name〉 is the name of a relation in the database;
value〉 is a value of the type corresponding to a field;
〈membership_distribution_table〉 contains two lists. The first list contains domain values for a field in ascending order. The second list contains the corresponding degrees of membership;
〈weight〉 is a weighting factor by which the degree of membership of the fuzzy term will be multiplied.

For example if a user wants to get the name and age of those students not in the physics department with age <27, height >1.65 m and a good grade in examination, the following query may be used (see the Appendix for the details of the database):

```
(student_rec (name age) (and (<age 27)(>height 1.65)
                    (<>dept "physics"))
(and (grade (       (0.0 0.4 0.8 1.2 1.6 2.0 2.4 2.8 3.2 3.6 4.0)
                    (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.6 0.9 1.0))1)))
```

where the members of the first list of the fuzzy term corresponds to the grades and the members of the second list corresponds to the respective degrees of membership of being a good grade. These two lists form the 〈membership_distribution_table〉 which is followed by the 〈weight〉 1.

After receiving the above query, the fuzzy information retrieval module will generate an Rdb query to get the names, ages and grades from those records which satisfy all the nonfuzzy conditions. The following records are obtained by invoking the RDO facility to access the database:

Name	Age	Grade
John	21	3.8
Peter	19	3.2
David	22	3.6

The degree of membership are then obtained by mapping the grades to the membership distribution table. Then the records are sorted in descending order according to the degrees of membership. Finally, the records are packed in a list and returned to the user. The list is shown as follows:

((John 21 0.95) (David 22 0.90) (Peter 19 0.60))

Moreover, statistical expressions can be nested into nonfuzzy conditions. Five types of statistical functions are provided namely, MAX, MIN, AVERAGE, TOTAL and COUNT. For instance, if the names of those students with grades greater than the average grade of students in physics department is required, the following query may be used:

```
(student_rec (name)
(and (>grade (average grade (and(=dept "physics"))))) nil)
```

Statistical information can also be retrieved by putting a statistical expression as the first list in the query. For example, if the maximum grade in physics department is needed, the following query may be used:

```
(student_rec ((max grade) (and(=dept "physics"))) nil nil)
```

The above examples have illustrated how information is retrieved from the database. However, the most powerful function of this module is the manipulation of fuzzy conditions. As defined by the extended BNF grammar above, a fuzzy condition is composed of an operator and a series of fuzzy expressions. A fuzzy expression consists of a simple fuzzy term, (with a field name, a membership distribution table and a weighting factor), or consists of another fuzzy condition with an operator and several fuzzy expressions. This syntax provides a simple way for constructing a nested condition. For a simple fuzzy condition, the degree of membership is obtained by mapping and interpolation using the curve corresponding to the distribution table and then multiplied by a weighting factor. For a compound fuzzy condition, the degree of membership is obtained by a function $f_n(a_1, a_2, \ldots a_n)$ and then multiplied by a weighting factor, where $a_1, a_2, \ldots a_n$ are the degrees of membership of the fuzzy terms contained in the fuzzy condition. The function used is determined by the operator specified in the fuzzy condition.

For example, if the names of those students who are either young and tall or with good sport-grade are needed, the following query with nested fuzzy conditions can be used.

```
(student_rec (name) nil
(or(and      (age             ( (35 33 31 39 27 25 23 21 19 17 15)
                              (0.0 0.0 0.0 0.0 0.0 0.1 0.2 0.4 0.7 0.9 1.0)) 1)
             (height          ( (0.9 1.0 1.1 1.2 1.3 1.4 1.6 1.7 1.8 1.9 2.0)
                              (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.6 0.9 1.0)) 1))
        (sports-grade         ( (0.0 0.4 0.8 1.2 1.6 2.0 2.4 2.8 3.2 3.6 4.0)
                              (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.6 0.9 1.0)) 1)))
```

Four operators are supported in this fuzzy information retrieval module. They are "and" "or", "evidence combination" and "polling". The differences of these four operations are illustrated in the following examples.

Suppose the names of those students in the physics department with a good grade and a good sports-grade are required. The following fuzzy database-query will be used.

```
(student_rec (name) (and (=dept "physics"))
(FUZ_OPERATOR (grade (  (0.0 0.4 0.8 1.2 1.6 2.4 2.8 3.2 3.6 4.0)
                        (0.0 0.0 0.0 0.0 0.0 0.0  0.2 0.6 0.9 1.0))
            1.0)
            (sports-grade (  (0.0 0.4 0.8 1.2 1.6 2.4 2.8 3.2 3.6 4.0)
                        (0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.6 0.9 1.0))
            1.0)))
```

where the FUZ_OPERATOR may be any one of the "and", "or", "comb" and "poll" operators.

The names, grades and sport-grades of those students which satisfy the nonfuzzy selection condition will be obtained from the database. After receiving the data from the database, the fuzzy information retrieval module will find out the degree of membership for each fuzzy term and multiply it by the weighting factor of the corresponding condition.

The details of the information obtained are shown as follows:

Name	Grade	Good grade	Sport-grade	Good sport-grade
Paul	3.8	0.95	2.6	0.10
Jimmy	2.6	0.10	3.6	0.90
Alfred	2.8	0.20	2.8	0.20

If the operator is "and", the operation is to take the minimum degree of membership from that of good grade and good sport-grade, i.e.

$$f_n(a_1, a_2, \ldots a_n) = \min(a_1, a_2, \ldots, a_n), \text{ for all } n.$$

If the operator is "or", the operation is to take the maximum degree of membership from that of good grade and good sport-grade, i.e.

$$f_n(a_1, a_2, \ldots a_n) = \max(a_1, a_2, \ldots a_n), \text{ for all } n.$$

If the operator is "comb", the overall degrees of membership is obtained from the formula $a_1 + a_2(1 - a_1)$ [12]. Where a_1 and a_2 are the degrees of membership for good grade and good sport-grade respectively. In general, if there are n fuzzy terms, the function f_n is defined as follows:

$$f_n(a_1, a_2, \dots a_n) = \begin{cases} a_1, & \text{if } n = 1, \\ a_1 + a_2(1 - a_1), & \text{if } n = 2, \\ f_2(f_{n-1}(a_1, a_2, \dots, a_{n-1}), a_n), & \text{otherwise.} \end{cases}$$

Lastly, if the operator is "poll", a polling method is used. The operation is to take the average degree of membership of the fuzzy terms, i.e.

$$f_n(a_1, a_2, \dots a_n) = (a_1 + a_2 + \cdots + a_n)/n, \text{ for all } n.$$

The degree of membership for the above four operations are summarized in the following table. (Assuming equal weighting.)

Name	"and"	"or"	"comb"	"poll"
Paul	0.100	0.950	0.955	0.525
Jimmy	0.100	0.900	0.910	0.500
Alfred	0.200	0.200	0.360	0.200

The ranking obtained from different operators may or may not be different. Any combination of the four operations is allowed to be nested in a fuzzy expression. Records with a final degree of membership of zero will be eliminated from the output. The final degree of membership obtained may be subjective. Sometimes it cannot be interpreted directly, but it can serve as an indicator for comparison.

7. EVALUATION AND COMPARISON

The fuzzy database-query language presented in this paper has two functions. The first function is to select alternatives from a database according to the nonfuzzy (crispy) conditions of a query. The other function is to find the degree of membership based on the fuzzy conditions. Moreover, the alternatives selected are arranged in descending order with respect to their degrees of membership. In other words, the fuzzy database-query language not only possesses the selecting power like some common database-query languages, but also possesses the capabilities for multicriteria decision making.

By using the fuzzy-query language, a degree of membership will be calculated for each alternative selected. A degree of membership is a real value ranged from 0 to 1.0 to indicate how much an alternative satisfies fuzzy conditions. According to the definition in Section 2, the fuzzy database-query language belongs to the multiattributes utility approach, since each alternative is mapped to a real value (degree of membership).

When compared with outranking methods and interactive methods, the advantages of multiattributes utility method are that after the utility of an alternative is found, we can choose the best alternative, select a subset of good alternatives or rank all the alternatives easily and efficiently. However, in traditional multiattributes utilities methods, their assumptions are not realistic. They cannot be applied to the problems involving criteria that are not independent. For the fuzzy database-query language, this kind of drawback has been overcome.

In the fuzzy database-query language, the member distribution table of a fuzzy condition is used to determine the utility of an attribute. It is similar to the function v_i in the additive utility function presented in section 2.1. In fact, this table is used to represent a fuzzy term such as "good", "very good", "strong", etc. The member distribution table may be subjective, since different people may have different interpretations on an attribute. However, the tables of some well-defined linguistic fuzzy terms may be used. Besides, some statistical methods may be used to obtain the member distribution table of a fuzzy term objectively when required.

Four operators are supported in the fuzzy database-query language which operate on the utility of each attribute and give the total utility of the alternative. The four operators are "AND", "OR", "COMB" and "POLL". Using the "POLL" operator only, the power of the query language is equivalent to additive utility methods. Since the operator of "POLL" is in effect, to sum up the utility of each attribute and then to normalize the overall utility.

On the other hand, the operator "AND", "OR" and "COMB" provides some means to model query with criteria which are not independent. The "AND" operator is used to express the situation where some attributes are useful only when they exist together. For example, a cigarette is useless when there is no match or lighter. The "OR" operator is used to model the situation where some attributes can be substituted by each other. For example, the utility of a match can be treated as zero if there is a lighter and vice versa. The "COMB" operator can be used to represent the situation where each attribute has a certain contribution to a goal even though the attributes exist independently. However, the effect of applying all the attributes is less than the sum of the effect of each attribute used separately. This approach can also be viewed as combining evidences to draw a conclusion. For example, different kinds of medical treatments may be used to cure a disease. Each of the treatments may have its own effect. When these treatments are used together, their effects may not be additive, but the overall effect may be greater than that of using a single treatment.

With the help of the weighting factors for each attributes together with mixing and nesting of the four operators, most of the queries can be expressed. In addition, using the fuzzy database-query language, a user can express his requirements in a natural way, since the concepts of the four operators are very close to human thinking and fuzzy terms frequently appear in our conversations. Therefore, a query can be expressed by this language easily.

8. CONCLUSION

The fuzzy database-query language combines both crispy and fuzzy conditions for database queries. It provides a natural way for a user to retrieve information from a database. Moreover, the capability of multicriteria decisions making is supported by this language which can help user to make decisions. This fuzzy query language not only provides a powerful tool for human as a database-query tool, but also support some other software systems for information retrieval from a database. For example, this fuzzy database-query system has been successfully integrated with an expert system shell, Z-II [13], to support data and information for inference.

This fuzzy logic based approach is superior to the traditional multicriteria decision methods in terms of its flexibility in manipulating the logic relationship between the criteria. However, like utility functions, fuzzy sets representing the criteria are subjective and must be formulated with great care. In some cases, the degrees of membership can only be used for comparison purpose and not to be interpreted directly.

REFERENCES

[1] B. Roy. Multicriteria analysis: survey and new direction. *Eur. J. Op. Res.* **8**, 207–218 (1981).

[2] M. Zeleny. *Multicriteria Decision Making*. In *System & Control Encyclopedia* (Ed. M. G. Singh), pp. 3116–3121. Pergamon Press, Oxford (1987).

[3] J. S. Dyer and R. K. Sarin. Multicriteria decision making. *Encyclopedia of Computer Science and Technology* (Eds J. Belzer, A. G. Holzman and A. Kent), Vol. 11, pp. 51–76. Marcel Dekker, New York.

[4] Y. Leung. Dynamic conflict resolution through a theory of a displaced fuzzy idea. *Approximate Reasoning in Decision Analysis*, pp. 381–390. North-Holland, Amsterdam (1982).

[5] Y. Leung. A concept of a fuzzy idea for multicriteria conflict resolution. In *Advances in Fuzzy Sets, Possibility Theory, and Applications* (Ed. P. P. Wang), pp. 387–403. Plenum Press, New York (1982).

[6] Y. Leung. A value-based approach to conflict resolution involving multiple objectives and multiple decision-making units. In *International and Regional Conflict: Analytic Approaches* (Eds W. Isard and Y. Nagao), pp. 55–71. Ballinger, Cambridge, Mass. (1982).

[7] Y. Leung. Conflict resolution: fuzzy-ideal concepts. In *System & Control Encyclopedia* (Ed. M. G. Singh), pp. 773–776. Pergamon Press, Oxford (1987).

[8] B. Roy. Classment et choix en presence de points de vue multiples (la methode ELECTRE). *Rev. Fr. Automat. Inf. Rech. Op.* **8**, (1968).

[9] R. Benayoun, J. de Montgolfier, J. Tergny and O. Larichev. Linear programming with multiple objective functions: step method (STEM). *Math. Prog.* **1**(3), pp. 366–375 (1971).

[10] *VAX Information Architecture Summary Description*. Digital Equipment Corp. (1987).

[11] *VAX Rdb/VMS Guide to Programming*. Digital Equipment Corp. (1985).

[12] B. G. Buchanan and E. H. Short-liffe. *Rule-based Expert Systems*. Addison Wesley, Reading, Mass.

[13] K. S. Leung and W. Lam. Fuzzy concepts in expert systems. *IEEE Comput.* **21** (9), 43–56 (1988).

APPENDIX

A part of the sample database used in this paper is given as follows:

Name	Department	Age	Grade	Sports-grade	Health	Height
John	Comput. Sci.	21	3.8	3.2	Very good	1.76
Lucia	Electronics	24	2.8	2.8	Good	1.61
Peter	Comput. Sci.	19	3.2	3.6	Quite good	1.72
Paul	Physics	18	3.8	2.6	Fair	1.82
Mary	Comput. Sci.	23	3.6	2.8	Fair	1.59
Joe	Electronics	27	2.4	3.8	Very good	1.85
Jimmy	Physics	25	2.6	3.6	Good	1.80
David	Electronics	22	3.6	3.2	Good	1.90
Alfred	Physics	22	2.8	2.8	Rather good	1.72