Simple Fuzzy SELECT Query on Crisp Database

Okiriza Wibisono / 13509018
Informatics Engineering
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509018@std.stei.itb.ac.id

ABSTRACT

Fuzzy logic is a form of logic different from our "everyday" logic. It comprises of values in between "true" and "false". Nowadays, fuzzy logic has penetrated its applications into many areas, among others, control system, modeling and simulation, pattern recognition, and optimization. Another important application of fuzzy logic and fuzzy sets is in databases. Fuzzy logic allows database users to give flexible queries: queries with linguistic terms in their conditions (WHERE clause). This paper presents an introduction to fuzzy querying and a simple example of fuzzy querying on a crisp database.

Index Terms: Fuzzy Set, Fuzzy Query, Linguistic Variable, Membership Function

1. INTRODUCTION

Fuzzy logic was first developed by Lotfi A. Zadeh in 1965. The notion of fuzzy logic comes from the concept of "gray" or "vague" area. Whereas normal logic considers only "black-white", "true-false", "yes-no", or "1-0" values, fuzzy logic extends normal logic values to handle the "gray" values: the values in between normal logic values. One direct consequence of extending normal logic notion is fuzzy logic being applicable to problems that contain imprecise, uncertain, noisy, or other kinds of vague values.

Since its first development in 1965, fuzzy logic has found its application in many engineering fields. Among the most important fuzzy logic applications is fuzzy database. A fuzzy database can handle imprecise information given to it by its users. Imprecise information is especially useful if the database is used in a decision-making system such as medical diagnosis, recruitment, and investment. Prevalent in such systems are information extracted from expert's opinions and knowledge, which are often not trivial to express in machine-understandable forms and data types. It is also desirable to enable users to query the database using conversational statements. A query such as "Which restaurants with reasonable price are in walking distance from Jalan Ganesha 10?" may be more helpful to a user than "Which restaurants with prices under Rp30.000,00 have a distance less than 500 m from Jalan Ganesha 10?".

In these past two decades, people have been researching on how to extend existing (crisp) databases to handle fuzzy queries or, even further, to store fuzzy information. There are two outstanding proposals in fuzzy logic application to databases: SQLf and FSQL. SQLf queries may involve linguistic terms defined by users. FSQL is more advanced than SQLf: besides fuzzy querying, it can also store fuzzy information; but FSQL gives its users less freedom over defining fuzzy objects in the database.

2. PRELIMINARIES

2.1 Fuzzy Sets

The concept of **membership** of an element in a **classic set** is straight-forward: an element either fully belongs to the set or it doesn't belong to the set at all. For example, if set $A = \{1, 2, 3, 4, 5\}$, then 3 is a member of A, but 7 is not. Using the notation of **characteristic function**, which is

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

we can write $\chi_A(3) = 1$ and $\chi_A(7) = 0$.

Fuzzy set generalizes this concept of membership to consider the element which is not a full member of a set, but cannot really be excluded from that set. Fuzzy set does this by defining **membership function**. Membership function maps an element in the **universe of discourse** X (i.e. domain of the set) to some value in the interval [0, 1]:

$$\mu_A : X \rightarrow [0, 1]$$

The crisp set membership function is given below:

$$\chi_A : X \rightarrow \{0, 1\}$$

Naturally, we can define three cases of an element's membership:

- If $\mu_A(x) = 1$, then x is a full member of fuzzy set A.
- If $\mu_A(x) = 0$, then x is not a member of fuzzy set A.
- If $\mu_A(x) = c$, $0 < c < 1$, then x is a member of fuzzy set A with **degree of membership** c.

For **finite** universe of discourse, one way of listing the members of a fuzzy set A is

$$A = \{\mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \cdots + \mu_A(x_n)/x_n\}$$

where n = the number of elements in A.

If the universe of discourse is **continuous** and **ordered**, we can define a fuzzy set by its membership function. Examples of the simplest membership functions are:

- Linear membership function

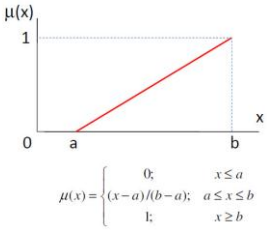

Figure 1. Linear membership function

- Triangular membership function

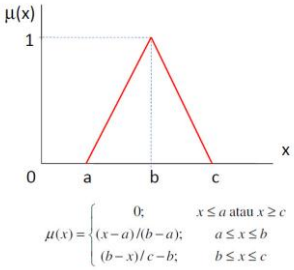

Figure 2. Triangular membership function

- Trapezoidal membership function

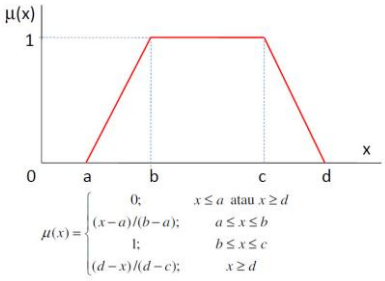

Figure 3. Trapezoidal membership function

For example, consider a linear membership function as depicted in Fig. 1, with a = 20 and b = 80. We can compute the degree of membership of an element with value = 70, which is (70 – 20)/(80 – 20) = 0.8333.

The above membership functions will be used extensively in the experiment section.

2.2 Fuzzy Logic

Fuzzy logic is a form of logic based on the concept of fuzzy sets. An atomic **predicate** (or **proposition**) in fuzzy logic takes the form of "*x* is *A*" (e.g. "Andi is tall", "Budi is fat"). Unlike classic logic, in which the **truth value** of a **predicate** is either 1 (true) or 0 (false), the truth value of a predicate in fuzzy logic is a real number in the interval [0, 1]. This truth value equals to the degree of membership of *x* in *A*.

Compound predicate is formed of atomic predicates joined together with **connectors**. Three most known connectors are AND, OR, and NOT. For example, "Andi is tall AND Budi is fat" and "Chandra is NOT old" are all compound predicates.

The truth value of a compound predicate is computed using some function for each of the connectors. Some particular functions are maximum of two values for OR, minimum of two values for AND, and 1 – value for NOT. Given a compound predicate $C \equiv$ (*x* is S AND *x* is NOT F) OR *x* is M, we can compute its truth value as below:

$T(C) = \max(\min(T(\text{“}x \text{ is S”}), 1 - T(\text{“}x \text{ is F”})), T(\text{“}x \text{ is M”}))$

$T(C) = \max(\min(\mu_S(x), 1 - \mu_F(x)), \mu_M(x))$

2.3 Linguistic Variable

Linguistic variable (**fuzzy variable**) is a variable which holds a **linguistic value**. Linguistic value of a linguistic variable is also known as **linguistic term**, or more commonly, **term**. Examples of these definitions are linguistic variable *speed* with linguistic terms *slow*, *fast*, and *very fast*, and linguistic variable temperature with linguistic terms *freezing*, *cool*, *warm*, and *hot*. Linguistic terms indicate qualitative values of their linguistic variable.

To compute the quantitative value of a linguistic variable, we again use the concept of membership function. One term is represented by one fuzzy set with a certain membership function.

Below is an example of a linguistic variable named *temperature* with its four linguistic terms.

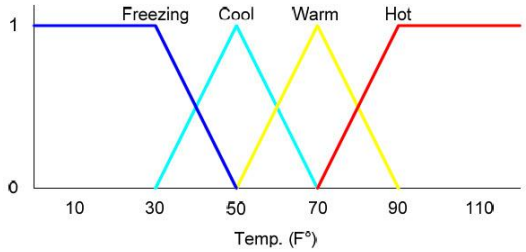

Figure 4. An example of a linguistic variable

2.4 Fuzzy Query

Fuzzy query is a database query involving imprecise or uncertain values. There are two main types of fuzzy query: (1) fuzzy query on crisp database and (2) fuzzy query on fuzzy database. This paper considers only the (simplified) former case, fuzzy query on crisp database.

In a fuzzy query, instead of specifying precise values (e.g. numeric, string, date, etc.), we specify linguistic terms as attribute values. There are at least two advantages in using fuzzy query. First, users can give more human-readable queries to the database. For example, the query "list the young salesmen who have good selling record in the north region" is clearly more comprehensible than "list the salesmen under the age 25 with selling record that is worth more than Rp20.000.000,00 in the regions …". Second, users can decide how "satisfying" a tuple have to be to the given query to be included in the result set. For example, the above normal query would not include a salesman by the age of 26 and has a selling record of Rp25.000.000,00, or

a salesman by the age of 24 and has a selling record of Rp19.000.000,00. Fuzzy query is able to deal with "close to the borders" situations such as these.

A fuzzy query's syntax is very much similar to a normal query's syntax. Each **condition** in the WHERE clause will be evaluated through the term's membership function. The value of these evaluations will then be computed with the appropriate connector functions.

The result of such queries is a set of tuples with the attributes listed in the SELECT clause, plus an additional attribute denoting **satisfaction degree** (also known as **calibration** in SQLf) of a given tuple. This satisfaction degree ranges from 0, meaning that the tuple really doesn't satisfy the given query, to 1, meaning that the tuple perfectly satisfies the given query. Usually tuples with satisfaction degree of 0 are omitted from the result set.

Example of a fuzzy query in SQLf:

"SELECT * FROM People WHERE weight=Fat AND height=Tall WITH CALIBRATION 0.75"

where *Fat* and *Tall* are linguistic terms predefined in the database. The clause WITH CALIBRATION 0.75 specifies the minimum satisfaction degree which a tuple must have to be included in the result set. We can say, in other words, a tuple which matches three-fourth of the query should be included in the result.

3. EXPERIMENT

3.1 Simple Fuzzy Query Processing

A simple fuzzy SELECT query feature can be implemented by properly processing the result set of a given query. The experiment in this paper uses MySQL as the relational database and PHP for accessing the database and displaying the query result. For convenience, I created two PHP pages: one for generating the query through HTML input elements and another for processing the query and displaying the result. The query generated will be of the form "SELECT Attrs FROM Tables WHERE $attr_1$=$term_1$ AND|OR … AND|OR $attr_n$=$term_n$". The equality sign ("=") in the WHERE clause may be replaced by inequality ("!=").

The query processing is as follows. First the SELECT query string (more specifically, its WHERE clause) is parsed to get the individual conditions and connectors used in the query. After that, *all* the tuples of the given relations of the query are retrieved. For example, in the query "SELECT * FROM T_1, T_2 WHERE …", *all* tuples from both table T_1 and table T_2 will be retrieved.

For each tuple retrieved, its satisfaction degree is computed by evaluating the tuple's degree of membership for each condition, and appropriately applying each connector's function. Only tuples with satisfaction degree greater than or equals some threshold *T* are included in the result set.

In short, the processing of simple fuzzy SELECT query can be described in the following diagram.

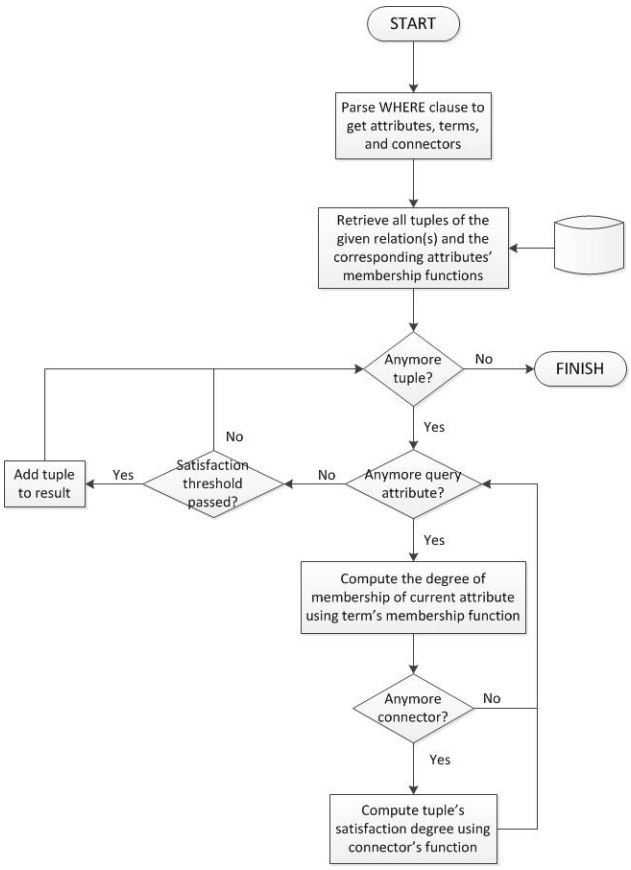

Figure 5. Simple fuzzy SELECT query processing

The implemented fuzzy SELECT query's WHERE clause must follow the linear form "WHERE $attr_1$=$term_1$ AND|OR $attr_2$=$term_2$ AND|OR …". One direct consequence of this limitation is that the conditions in the WHERE clause cannot be nested. To be able to handle nested conditions such as "WHERE NOT ($attr_1$=$term_1$ AND $attr_2$=$term_2$)", an expression tree must be generated from the corresponding WHERE clause.

Another consequence is that the resulting satisfaction degree of a given tuple is dependent on the ordering of the conditions. For example, consider the conditions "$attr_1$=$term_1$ OR $attr_2$=$term_2$ AND $attr_3$=$term_3$" with the first condition evaluated to be 0.5, the second condition 0.3, and the third condition 0.4. If the OR is evaluated first, the resulting satisfaction degree is min(max(0.5, 0.3), 0.4) = 0.4. If the AND is evaluated first, the resulting satisfaction degree is max(0.5, min(0.3, 0.4)) = 0.5. This problem can be solved by taking precedence of connectors into account in computing the satisfaction degree.

3.2 Database and Membership Functions

To enable fuzzy querying on the database, first the linguistic terms for each attribute must be defined, each with its own membership function. The terms' membership functions are then stored in the database. For simplicity, only one real-world relation is defined in this experiment:

Person = {<u>name</u>, age, height, weight}

The terms and their membership functions is listed in the table below. Actually a term refers not only to an attribute name, but also to a relation name. Since this experiment uses only one relation (i.e., *Person*), the column **Relation** is omitted from the table below.

Table 1. Membership functions of linguistic terms on *Person*

Attr.	Term	Inv.	Type	a	b	c	d
Age	Child	false	Tri.	3	10	15	-
Age	Adult	false	Trap.	25	30	45	50
Age	Old	false	Lin.	40	50	-	-
Height	Short	true	Lin.	150	180	-	-
Height	Tall	false	Lin.	160	180	-	-
Weight	Thin	true	Lin.	40	60	-	-
Weight	Normal	false	Trap.	40	50	70	80
Weight	Fat	false	Lin.	60	80	-	-

An explanation of the table above is as follows:

- **Attr.** denotes the attribute name (of the relation *Person*) to which a given term refers
- **Term** denotes the term name
- **Inv.** denotes whether the term's membership function is the complement (i.e. 1 minus the value according to the original membership function) of the one given in section 2.1
- **Type** denotes the type of membership function, i.e. Lin. (linear), Tri. (triangle), or Trap. (trapezoid)
- **a, b, c, d** denotes the parameters to the membership function as depicted in Fig. 1 (for linear membership function), Fig. 2 (for triangular membership function), and Fig. 3 (for trapezoidal membership function).

Below are the plots of each of the linguistic variables (*Age*, *Height* and *Weight*) over their universe of discourses.

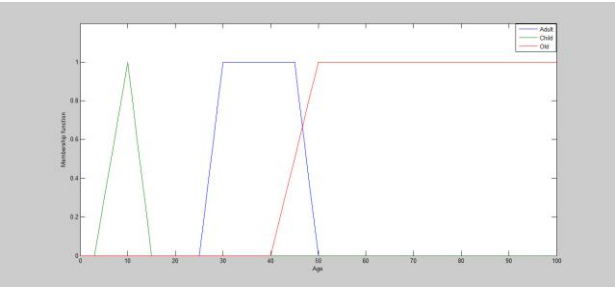

Figure 6. Linguistic variable *Age*'s membership functions

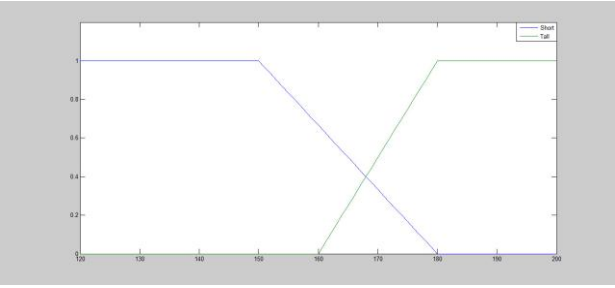

Figure 7. Linguistic variable *Height*'s membership functions

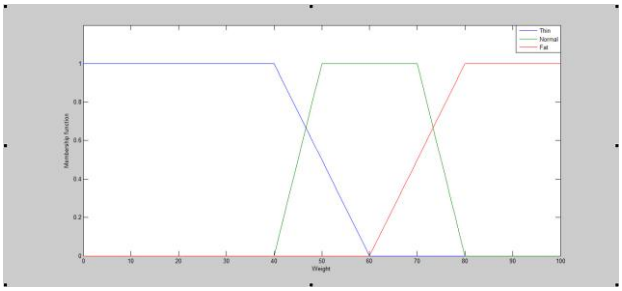

Figure 8. Linguistic variable *Weight*'s membership functions

3.3 Some Examples

Below are examples of some simple fuzzy queries and their corresponding results.

- Query 1: "SELECT * FROM Person WITH CALIBRATION 0"

Table 2. Result of Query 1

Name	Age	Height	Weight	Satisfaction Degree (%)
Andi	20	170	60	100
Budi	60	163	62	100
Chandra	40	150	53	100
Deni	10	140	35	100
Eki	70	175	79	100

- Query 2: "SELECT * FROM Person WHERE age=Old WITH CALIBRATION 0.99"

Table 3. Result of Query 2

Name	Age	Height	Weight	Satisfaction Degree (%)
Budi	60	163	62	100

- Query 3: "SELECT * FROM Person WHERE height=Short OR weight!=Thin WITH CALIBRATION 0.5"

Table 4. Result of Query 3

Name	Age	Height	Weight	Satisfaction Degree (%)
Deni	10	140	35	100

- Query 4: "SELECT * FROM Person WHERE age=Adult OR height=Tall AND weight=Normal WITH CALIBRATION 0.01"

Table 5. Result of Query 4

Name	Age	Height	Weight	Satisfaction Degree (%)
Andi	20	170	60	50
Budi	60	163	62	15
Chandra	40	150	53	100
Eki	27	181	79	10

An example of computing the satisfaction degree of the *Person* named "Andi" in Table 6 is as follows:

Query: "SELECT * FROM Person WHERE age=Adult OR height=Tall AND weight=Normal WITH CALIBRATION 0.01"

Referring to Table 5 for the membership functions:
Degree of membership of *Andi*'s age (20) in *Adult*'s membership function = 0

Degree of membership of *Andi*'s height (170) in *Tall*'s membership function = (170-160)/(180-160) = 0.5

Degree of membership of *Andi*'s weight (60) in *Normal*'s membership function = 1

Therefore, satisfaction degree of *Andi* = min(max(0,0.5),1) = 0.5

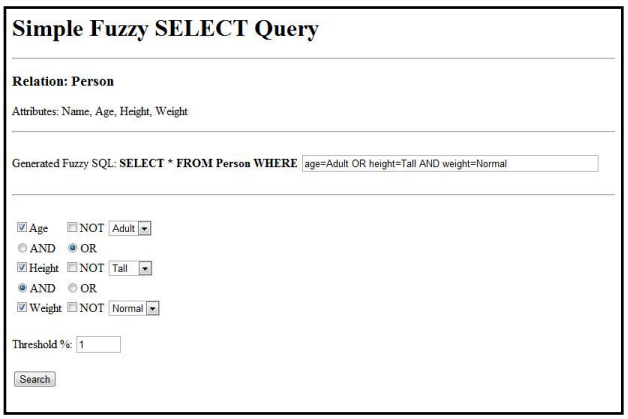

Figure 9. Example screenshot of the query generator page

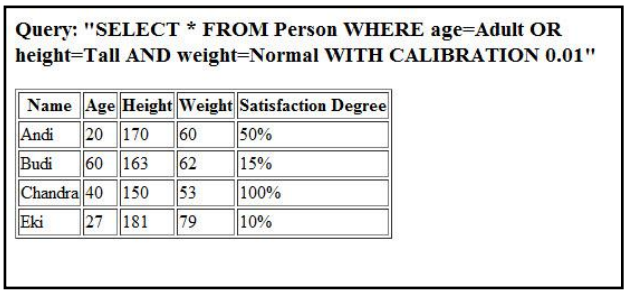

Figure 10. Example screenshot of the query result page

4. CONCLUSION

From the description and examples in the previous sections, several points can be summarized:

1. Fuzzy query is an extension to the classic SQL query. It enables users to query a database, either a crisp or a fuzzy one, more flexibly using linguistic terms common to humans.
2. Each tuple in a fuzzy query result has a corresponding satisfaction degree, ranging from 0 to 1. The more closely a tuple matches a fuzzy query, the greater its satisfaction degree. A fuzzy query result may include tuples that doesn't perfectly match the query conditions.
3. The fuzzy querying mechanism described in this paper demonstrates how to compute the truth value of a (compound) predicate in fuzzy logic. In this case, the predicate is the WHERE clause of a fuzzy query.
4. The fuzzy querying mechanism described in this paper is a very simplified one. The limitations are mainly considered about the absence of expression tree and precedence of connectors. These limitations imply that the resulting satisfaction degree may not resemble the actual satisfaction degree.

STATEMENT

Hereby I declare that this paper is my own writing, not an adaptation, a translation, nor a plagiarism.

REFERENCES

[1] Munir, Rinaldi. Slide Kuliah IF4058 Topik Khusus Informatika I. 2012.

[2] Galindo, Jose. *Handbook of Research on Fuzzy Information Processing in Databases*. New York: Information Science Reference. 2008.

[3] Galindo, Jose, et al. *Fuzzy Databases: Modeling, Design and Implementation*. Hershey: Idea Group Publishing. 2006.

[4] Klir, George J. and Bo Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Application*. New Jersey: Prentice Hall. 1995.

Bandung, May 15th 2012

Okiriza Wibisono
13509018