Fuzzy Quantifiers with and without Arguments for Databases: Definition, Implementation and Application to Fuzzy Dependencies

José Galindo
Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga,
Spain.
jgg@lcc.uma.es

Ramón A. Carrasco
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada,
Spain.
racg@ugr.es

Ana Mª Almagro
I.E.S. Alcaria
Puebla del Río,
Sevilla, Spain
anacurra@hotmail.com

Abstract

This paper studies how fuzzy quantifiers may be defined and implemented in a fuzzy database context. Fuzzy quantifiers are very useful for fuzzy queries, fuzzy constraints and fuzzy data mining applications. Besides, this paper shows different kind of fuzzy quantifiers with and without arguments. Finally, we show how fuzzy dependencies may use these fuzzy quantifiers.

Keywords: Fuzzy Queries, Fuzzy Quantifiers, Fuzzy Databases, FSQL, Fuzzy Dependencies.

1 Introduction

Fuzzy or linguistic quantifiers [5][8][9][11] allow us to express fuzzy quantities or proportions in order to provide an approximate idea of the number of elements of a subset fulfilling a certain condition or the proportion of this number in relation to the total number of possible elements.

As we shall see, fuzzy quantifiers can be *absolute* or *relative*, and some examples are "much more than 10", "close to 100", "a great number of", "the majority" or "most", "the minority" and so on.

In a fuzzy database context [6][7], fuzzy quantifiers are used in fuzzy constraints, fuzzy queries (for example using FSQL language [10]) and fuzzy data mining applications. For example, a fuzzy query is "Give me employees who work for most projects", while a fuzzy constraint is that "An employee must work for many projects". In Section 5 we will see a data mining application.

These quantifiers must be stored in the database data dictionary. Thus, its definition could be used when it is necessary. However, definition of each quantifier depends on the object or context in which it is used and, besides, we find very useful to define fuzzy quantifiers with arguments. This paper studies, how to define these fuzzy quantifiers in a fuzzy database and suggests some fuzzy quantifiers definitions, which may be used as default definitions in any fuzzy database.

Finally, this work shows an application in the data mining area, searching for fuzzy dependencies.

2 Definition of Fuzzy Quantifiers

Fuzzy quantifiers can be *absolute* or *relative*:
Absolute quantifiers express quantities over the total number of elements of a particular set, stating whether this number is, for example, "much more than 10", "close to 100", "a great number of"... Generalizing this concept, we can consider fuzzy numbers as absolute fuzzy quantifiers, in order to use expressions like "approximately between 5 and 10", "approximately –8"... Note that the expressed value may be positive or negative. In this case, we can see that the truth of the quantifier depends on a single quantity. For this reason, the definition of absolute fuzzy quantifiers is, as we shall see, very similar to that of fuzzy numbers.

Relative quantifiers express measurements over the total number of elements, which fulfill a certain condition depending on the total number of possible elements (the proportion of elements). Consequently, the truth of the quantifier depends on two quantities. This type of quantifier is used in expressions such as "the

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 227–234
Torremolinos (Málaga), June 22–27, 2008

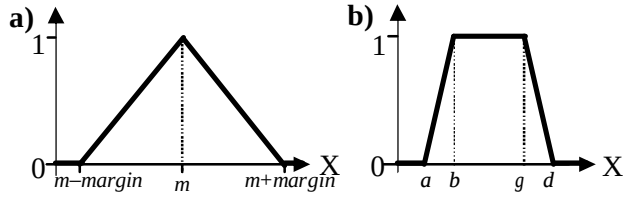

Figure 1: a) Triangular Fuzzy Set (Symmetrical). b) Trapezoidal Fuzzy Set.

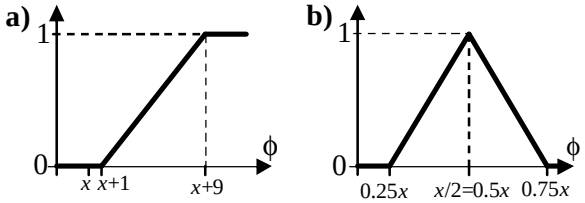

Figure 2: Absolute Fuzzy Quantifiers with one argument (type sum and product):
a) "Much Greater Than x": [x+1, x+9, ∞, ∞],
b)"About half of x": [0.25x, 0.5x, 0.5x, 0.75x].

majority" or "most", "the minority", "little of", "about half of"... In this case, in order to evaluate the truth of the quantifier we need to find the total number of elements fulfilling the condition and consider this value with respect to the total number of elements which could fulfill it (including those which fulfill it and those which do not fulfill it).

Some quantifiers such as "many" and "few" can be used in either sense, depending on the context [9].

In [11] absolute fuzzy quantifiers are defined as fuzzy sets in the positive real numbers and relative quantifiers as fuzzy sets in the interval [0,1]. We have extended the definition of absolute fuzzy quantifiers to all real numbers. Negative fuzzy quantifiers are not very useful but, they are useful for queries like: "Give me those pairs of employees for which if we subtract their corresponding number of project we achieve approximately −7". Negative fuzzy quantifiers express negative quantities, on certain domains that could be negative (for example with subtractions).

Definition 1: *A **fuzzy quantifier** named* Q *is represented as a function Q whose domain depends on whether it is absolute or relative:*

$$\begin{aligned} Q_{abs} &: \Re \rightarrow [0,1] \\ Q_{rel} &: [0,1] \rightarrow [0,1] \end{aligned} \qquad (1)$$

where the domain of Q_{rel} is [0,1] *because the division a/b* ∈ [0,1], *where a is the number of elements fulfilling a certain condition and b is the total number of existing elements.*

In order to know the fulfillment degree of the quantifier over the elements which fulfill a certain condition, we apply the function Q of the quantifier to the value of quantification ϕ (phi):

$$\Phi = \begin{cases} a & \text{if } Q \text{ is absolute} \\ a/b & \text{if } Q \text{ is relative} \end{cases} \qquad (2)$$

Thus, the fulfillment degree is $Q(\phi)$. If the function of the quantifier (absolute or relative) $Q(\phi)$, has the value 1, this indicates that this quantifier is completely satisfied. The value 0 indicates, on the other hand, that the quantifier is not fulfilled at all. Any intermediate value indicates an intermediate fulfillment degree for the quantifier.

Example 1: "Approximately 8" is an absolute fuzzy quantifier, defined as a triangular and symmetrical function just like Figure 1a, with m=8 and *margin*=2, for example. "Approximately between 30 and 40 is another absolute fuzzy quantifier, defined in Figure 1b as a trapezoidal function with b = 30 and g = 40. Sometimes, fuzzy relative quantifier "most" is represented using the function $Q(x) = x$, with x ∈ [0,1].

A survey of methods for evaluating quantified sentences and some new methods are shown exceedingly well in [4] and [8].

There are two important classic quantifiers: The universal quantifier (for all, ∀), and the existential quantifier (exist, ∃). The first of them is relative and the second one is absolute. They are discretely defined as:

$$Q_{\forall}(x) = \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

$$Q_{\exists}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases} \qquad (4)$$

The existential quantifier may be also defined in a fuzzy way with a non-discrete trapezoidal form: [0, 1, ∞, ∞].

3 Fuzzy Quantifiers with Arguments

Some quantifiers (absolute or relative) may have arguments. The arguments are numbers and the meaning and definition of the quantifier depends on these numbers. Most of quantifiers with arguments are absolute, whereas relative ones are rare.

Example 2: Some absolute fuzzy quantifiers with one and two arguments:

- "Much Greater Than x": Represented with function in Figure 2a.
- "About half of x": Represented with function in Figure 2b.
- "Approximately between x and y plus/minus 5" ($x < y$): Represented with a trapezoidal function (Figure 1b) with $[a, b, g, d] = [x–5, x, y, y+5]$.
- "Approximately between x and y" ($x < y$): Another form more free of context may be represented with a trapezoidal function (Figure 1b) with $[a, b, g, d] = [0.75x, x, y, 1.25y]$.
- "Approximately between half of x and half of y" ($x < y$): Represented with a trapezoidal function (Figure 1b) with $[a, b, g, d] = [0.25x, 0.5x, 0.5y, 0.75y]$.

In relative quantifiers, we can use expressions like "half" or "a quarter", for example. From these expressions we get a value $x \in [0,1]$ representing them. This value is computed with the division $1/d$, where d is the significant value in the expression. Thus, the expression "half" gets $x=1/2=0.5$, and "a quarter" gets $x=1/4=0.25$.

Example 3: Some relative fuzzy quantifiers with one and two arguments:

- "Approximately a x-th part" ($x \in [0,1]$): $[a, b, g, d] = [x–0.2, x, x, x+0.2]$.
- "Less than a x-th part" ($x \in [0,1]$): $[a, b, g, d] = [0, 0, x, 1.25x]$.
- "Approximately between a x-th and a y-th part" ($x < y$ and $x, y \in [0,1]$). This is a rare relative quantifier with arguments: $[a, b, g, d] = [0.75x, x, y, 1.25y]$ or $[x–0.1, x, y, y+0.1]$. For example, "approximately between a quarter and the half" is represented with $x = 0.25$ and $y = 0.5$.
- "Approximately between half of a x-th and half of a y-th part" ($x < y$ and $x, y \in [0,1]$). $[a, b, g, d] = [0.4x, 0.5x, 0.5y, 0.6y]$.

Definition 2: *A **general classification of trapezoidal quantifiers** [a, b, g, d], attending to its arguments and the building type, is the following one:*

a) Without arguments: See Definition 1.

b) With one argument x:

- Type Product: $[a * x, b * x, g * x, d * x]$.
- Type Sum: $[a + x, b + x, g + x, d + x]$.

c) With two arguments x and y:

- Type Product: $[a * x, b * x, g * y, d * y]$.
- Type Sum: $[a + x, b + x, g + y, d + y]$.

In relative quantifiers it is not necessary to warranty that all values are in [0,1], because the important is the quantifier definition in [0,1]. If it is an important condition we can use the function min. For example a type product relative quantifier with one argument may be built as: $[\min\{1, a * x\}, \min\{1, b * x\}, \min\{1, g * x\}, \min\{1, d * x\}]$.

4 Fuzzy Quantifiers in the Data Dictionary

Fuzzy quantifiers must be stored in the database data dictionary. Thus, its definition could be used when it is necessary. However, definition of each quantifier depends on the object or context in which it is used. Then, a fuzzy quantifier is always associated to any of the following objects: an attribute, a table (or entity) or the system.

For example, if we look for "employees who belong to most projects", quantifier *most* must be associated with table of projects, i.e., the concept of "*most* projects" depends on the project entity and its meaning. Although most of context-dependent quantifiers are associated to a table, we allow them to be associated to an attribute or column. The database user could have defined different fuzzy quantifiers and then to use the most appropriate for each application.

We call system quantifiers to those quantifiers with a general definition useful in different contexts, such as "approximately 2" or "about half".

Finally, just like any other database object, each database user should be able to define his/her own fuzzy quantifiers or to use those quantifiers defined by other users (specially the database administrator).

We propose the following four basic tables for storing fuzzy quantifiers in the data dictionary of our database:

4.1 Table FUZZY_LABEL_DEF

This table contains the points that define the trapezoidal functions associated to labels and quantifiers. The fields of this table are:

- (OBJ#, COL#, FUZZY_ID): these three fields identify respectively the table, the column of this table and the quantifier to be defined. They are the primary key of this table and foreign key to the table FUZZY_OBJECT_LIST, where, as we will see, the quantifier type is stored.
- ALFA, BETA, GAMMA and DELTA: these define a trapezoidal possibility distribution for the quantifier. The definition depends on the quantifier type.

4.2 Table FUZZY_OBJECT_LIST

This table contains a list of the fuzzy objects that are defined for the columns of the database, including each fuzzy quantifier. The attributes of this table have the following meanings:

- (OBJ#, COL#, FUZZY_ID): The first two values store the identifier of the owner column. FUZZY_ID is an identifier for the fuzzy object, a fuzzy quantifier, a label...
- FUZZY_NAME: the name of the object without spaces.
- FUZZY_TYPE: the type of the object. It may be one of the following codes and each code has an associated object. Codes of quantifiers begin at ten, because we reserve the first numbers to other objects (linguistic labels…). All quantifiers are defined in the table FUZZY_LABEL_DEF with the values α, β, γ and δ, but the interpretation of these values depend on the quantifier type. See Example 1 and 2 for examples about these quantifier types:
  - **10** *Absolute quantifiers* without arguments. In this case, $\alpha, \beta, \gamma, \delta \geq 0$.
  - **11** *Relative quantifiers* without arguments. Now, $\alpha, \beta, \gamma, \delta \in [0,1]$.
  - **12** *Absolute quantifiers* with one *argument x*, type sum. In this case α, β, γ and δ may be negative, and the final quantifier is understood to be defined by adding (or reducing) the argument *x* to each value: $[\alpha+x, \beta+x, \gamma+x, \delta+x]$.
  - **13** *Absolute quantifiers* with one *argument x*, type product. In this case, α, β, γ and δ are usually in the interval [0,1], and the final quantifier is understood to be defined by multiplying each value by the argument *x*: $[\alpha*x, \beta*x, \gamma*x, \delta*x]$.
  - **14** *Relative quantifiers* with one *argument x*, type sum. In this case, α, β, γ and δ may be negative, and the final quantifier is understood to be defined by adding (or reducing) the argument *x* to each value: $[\alpha+x, \beta+x, \gamma+x, \delta+x]$.
  - **15** *Relative quantifiers* with one *argument x*, type product. In this case, α, β, γ and δ are usually in the interval [0,1]. The final quantifier is defined by multiplying each value by the argument *x*: $[\alpha*x, \beta*x, \gamma*x, \delta*x]$.
  - **16** *Absolute quantifiers* with two *arguments x* and *y*, type sum. Here, α, β, γ and δ may be negative, and the final quantifier is defined by: $[\alpha+x, \beta+x, \gamma+y, \delta+y]$.
  - **17** *Absolute quantifiers* with two *arguments x* and *y*, type product. Here, α, β, γ and δ will be usually in the interval [0,1], and the final quantifier is understood to be defined by: $[\alpha*x, \beta*x, \gamma*y, \delta*y]$.
  - **18** *Relative quantifiers* with two *arguments x* and *y*, type sum. Values α, β, γ and δ may be negative, and the final quantifier is understood to be defined by: $[\alpha+x, \beta+x, \gamma+y, \delta+y]$.
  - **19** *Relative quantifiers* with two *arguments x* and *y*, type product. Finally, α, β, γ and δ will be usually in the interval [0,1], and the final quantifier is defined by: $[\alpha*x, \beta*x, \gamma*y, \delta*y]$.

Some examples are shown in Table 1.

4.3 FUZZY_TABLE_QUANTIFIERS

This table stores the definition of quantifiers associated to a relation or table (not to an attribute). These quantifiers are also used in fuzzy constraints, fuzzy queries, and fuzzy data mining applications [6][7]. The columns of this table are:

- OBJ#: this stores the identifier of the table to which the quantifier is associated.
- FUZZY_NAME: the name of the quantifier without spaces.
- FUZZY_TYPE: the type of quantifier. This attribute uses the same codes as the table FUZZY_OBJECT_LIST for quantifiers.
- ALFA, BETA, GAMMA and DELTA: these attributes define the trapezoidal fuzzy quantifier, just like Section 4.2 explains.

The primary key of this table is (OBJ#, FUZZY_NAME). This indicates that one table cannot have two quantifiers with the same name,

but the same name can be used in different tables, and of course, with possibly different definitions.

4.4 FUZZY_SYSTEM_QUANTIFIERS

This table stores the definition of quantifiers associated to the system (neither an attribute nor a table). These quantifiers may be also used in fuzzy constraints, fuzzy queries, and fuzzy data mining applications [6][7]. The columns of this table are:

- FUZZY_NAME: the name of the quantifier without spaces.
- FUZZY_TYPE: the type of quantifier. This attribute uses the same codes as the table FUZZY_TABLE_QUANTIFIERS.
- ALFA, BETA, GAMMA and DELTA: these define the trapezoidal fuzzy quantifier.

The primary key of this Table is (FUZZY_NAME). This indicates that each system quantifier has a unique name. Table 1 shows an example of this table with interesting fuzzy quantifiers of each type, where MAX is the maximum value in the underlying domain (MAX≡∞).

Some quantifiers are very dependent on the context, and so they are not good system quantifiers. System quantifiers should be relative, or absolute with one or two arguments and type product (types 11, 13, 14, 15, 17, 18 and 19), because these types are not very dependent on the context (particularly the relative quantifiers).

This table should store some default values, but they must be chosen with care, according to the database context. Furthermore, there are two quantifiers (∀ and ∃) which must be implemented directly in the system: **`For_all`** (or **`All`**) and **`Exists`**, with the Equations 3 and 4, respectively.

Note that in relative fuzzy quantifiers with arguments (types 14, 15, 18 and 19), where we use the expression `_xth_part`, the argument should finally be 1/x, because it is relative. For example, if we want to compute approximately the 10^{th} part, we must use x = 0.1. We do not change the quantifier names, because these names are more expressive.

Table 1: Some Interesting System Quantifiers with 0, 1 and 2 Arguments.

FUZZY_NAME	TYPE	ALFA	BETA	GAMMA	DELTA	Final Quantifier
Fuzzy_Exists	10	0	1	MAX	MAX	[0, 1, ∞, ∞]
Approx_8	10	6	8	8	10	[6, 8, 8, 10]
Almost_All / Most	11	0.4	0.9	1	1	[0.4, 0.9, 1, 1]
About_Half	11	0.25	0.5	0.5	0.75	[0.25,0.5,0.5,0.75]
Minority	11	0	0	0.1	0.6	[0, 0, 0.1, 0.6]
Much_Greater_Than_x	12	1	9	MAX	MAX	[1+x,9+x,MAX+x,MAX+x]
About_Half_of_x	13	0.25	0.5	0.5	0.75	[.25x, .5x,.5x, .75x]
Approx_x	13	0.9	1	1	1.1	[0.9x, x, x, 1.1x]
Twice_x / Double_of_x	13	1.75	2	2	2.25	[1.75x, 2x,2x, 2.25x]
Approx_xth_part	14	-0.2	0	0	0.2	[x-0.2, x, x, x+0.2]
Less_Than_xth_part	15	0	0	1	1.25	[0, 0, x, 1.25x]
More_Than_xth_part	15	0.75	1	100	100	[0.75x, x,100x, 100x]
Between_x_and_y	16	-5	0	0	5	[x-5, x, y, y+5]
Approx_Between_x_and_y	17	0.75	1	1	1.25	[0.75x, x, y, 1.25y]
Approx_Between_ Half_x_and_Half_y	17	0.25	0.5	0.5	0.75	[.25x, .5x,.5y, .75y]
Approx_Between_ Twice_x_and_Twice_y	17	1.75	2	2	2.25	[1.75x, 2x, 2y, 2.25y]
Approx_Between_ Xth_and_yth_part	18	-0.1	0	0	0.1	[x-0.1, x, y, y+0.1]
Approx_Between_Half_ Xth_and_Half_yth_part	19	0.4	0.5	0.5	0.6	[0.4x, 0.5x, 0.5y, 0.6y]

5 Applications to Fuzzy Dependencies

Definition 3: *We say that relation R verifies an* α–ß ***Gradual Functional Dependency (GFD)*** *using* F *and* T*, if and only if* [2][3]:

$\forall t_1,t_2 \in R$, $F(t_1[X],t_2[X]) \geq \alpha \Rightarrow T(t_1[Y],t_2[Y]) \geq ß$

where F *and* T *are fuzzy relations such as: fuzzy greater than, fuzzy greater than or equal to, fuzzy less than, fuzzy equal, etc., like those fuzzy comparators defined for FSQL* [6][7][10].

Often just a few items (objects, tuples or rows) can prevent the GFD from being completed. To avoid this, we can relax the universal quantifier ∀ in such a definition. Thus, all the tuples of the relationship are not forced to fulfill the above condition. Then, we must define a system to know how of interesting is one GFD: the measures of confidence and support:

Definition 4: *The* ***confidence*** *c of a GFD is a value in* [0,1]:

$$c = \begin{cases} 0 & \text{if } Card\{(t_1,t_2)\ t_1,t_2 \in R / F(t_1[X],t_2[X]) \geq a\} = 0; \\ \dfrac{Card\{(t_1,t_2)\ t_1,t_2 \in R / F(t_1[X],t_2[X]) \geq a \ \wedge T(t_1[Y],t_2[Y]) \geq b\}}{Card\{(t_1,t_2)\ t_1,t_2 \in R / F(t_1[X],t_2[X]) \geq a\}} & \\ \text{Otherwise;} & \end{cases}$$

where Ù is the logical operator AND.

The basic idea consists in computing the percentage of objects fulfilling the antecedent and consequent, with respect to those fulfilling only the antecedent.

Definition 5: *The* ***support*** *s of a GFD is the number of items fulfilling the antecedent and consequent, with respect to the total number n of items in R*:

$$s = \begin{cases} 0 & \text{if } n = 0; \\ \dfrac{Card\{(t_1,t_2)\ t_1,t_2 \in R / F(t_1[X],t_2[X]) \geq a \ \wedge T(t_1[Y],t_2[Y]) \geq b\}}{n} & \\ \text{Otherwise;} & \end{cases}$$

The GFD concept may be extended with:

Definition 6: *A fuzzy* ***Global Dependency (GD)****, is a GFD in which the antecedent and the consequent may be a logic expression, using AND, OR and NOT, on a set of attributes, instead of a single attribute with a fuzzy relation. These expressions may include constants and fuzzy constants, classic or fuzzy attributes and for each logic operator (AND, OR and NOT), we must define the fuzzy interpretation function (a t-norm, a s-norm and a negation respectively).*

The two measures, confidence and support, are also available and useful in any GD with similar definitions.

When one or more constants are associated to one elected attribute, the GD inference process avoids the items which do not fulfill such conditions. In this case, a new concept is useful:

Definition 7: *The* ***relative support*** *of a GD is the number of items fulfilling the antecedent and consequent, with respect to the total number n of items fulfilling all the fuzzy conditions with one constant associated to one elected attribute in the antecedent.*

The relative support measures the support of a GD in a particular context, not in the whole database context. The relative support equation is that on Definition 5 but using the value *n* of Definition 7. Obviously, for every GD, the relative support is greater or equal to the support.

5.1 Automatically Finding Interesting GD

Summarizing, the system is built implementing the following four steps: In the first step one expert in the database context must choose some interesting attributes for the antecedent and consequent respectively. For each attribute, the expert must choose a fuzzy comparator, like those defined for FSQL [6][7][10], and a fulfillment threshold for each one. Optionally, the expert may associate different constants to some of the elected attributes. These constants may be fuzzy constants like those defined for FSQL: fuzzy sets like “approximately 5”, fuzzy labels like “big”, fuzzy intervals, etc.

When one constant is associated to one elected attribute, the expert wants to fix this attribute to that constant. The system only will work with items fulfilling this kind of (fuzzy) conditions, and the inference process avoids the other items. In this case, the relative support is useful.

After that, in an optional second step the expert may choose a minimum confidence and a minimum support for the future discovered dependencies. Instead of such two values, the expert may choose two relative fuzzy quantifiers with two fulfillment thresholds for each one.

In the third step, the system will try each possible combination of the antecedent, with

each possible combination of the consequent. For each one, the system computes the confidence and the support. If these values are greater or equal to the minimum values, or these values fulfill the relative fuzzy quantifiers with degrees greater or equal to the respective fulfillment thresholds, then the GD is said to be an interesting GD.

In the fourth and last step, the system will show the discovered interesting global dependencies, showing the confidence, the support and the relative support for each one, showing the fulfillment degree of these three values with respect to the relative quantifiers in the system. Thus, we achieve an easy to understand sentence.

Keep in mind that confidence measures in what extend the GD is satisfied, and support measures whether it is satisfied by an enough number of cases or only for some few ones. Of course, a strong GD is when confidence and support are 1, but, generally we must only demand a "big" confidence, and an "enough" support (or at least an "enough" relative support).

5.2 Example

Let us suppose a database about reforestations. In this database, we store data about planting of forest trees made by different organisms (like NGO's), in different climatic conditions, especially in the Mediterranean area.

For each reforestation we store some attributes such as: planted species (pines, evergreen oak, wild olive tree, carob tree, mastic tree…), plantation date, age of planted trees, number of planted trees, number of alive specimens after the first summer and after the second summer, survival percentage after the first and after the second summer, amount of irrigations (if they exist), kind of soil, kind of climatic conditions in the week of the reforestation or planting (including temperature, rain and other climatic measurements), etc.

In our first step, the expert chose the following conditions for attributes of the antecedent:

- Attribute age of planted trees, with the fuzzy relation FGT (Fuzzy Greater Than).
- Attribute amount of irrigations, with the fuzzy relation FEQ (Fuzzy Equal) and the fuzzy constant "approximately 0", indicating that we are interested in reforestations with 0, or near 0 artificial irrigations.
- Attribute rain in the week of the reforestation, with the FEQ and two fuzzy constants: "Normal_Rain" or "Very_Little_Rain".

The expert set the following conditions for attributes of the consequent:

- Attribute survival percentage after the first summer, with FGT.

The expert chose a 0.75 threshold for all the thresholds. This threshold guaranties a fulfillment degree not too small.

The system may answer information like the following, showing the confidence, support and relative support for each interesting dependency according to some fuzzy quantifiers stored in the system. We use only some of the quantifiers showed on Table 1. Then, one fuzzy dependency says that: With few irrigation units and during a period of "Normal_Rain", the greater age of planted trees, the lesser survival percentage after the first summer.

- Confidence: "Most", with degree 0.95.
- Support: "Approximately the fifth part", with degree 0.8, "Most" with 0.2.
- Relative Support: "Most", with 0.7.

On the other hand, another unexpected dependency set that: With few irrigation units and during a period of "Very_Little_Rain", the greater age of planted trees, the greater survival percentage after the first summer.

- Confidence: "Most", with degree 0.82.
- Support: "Approximately the fourth part", with degree 0.8, "Most" with 0.3.
- Relative Support: "Most", with 0.6.

The first dependency says that supposing few or none artificial irrigations, if we expect a year of normal rain, we must plant young trees (according to the usual planting techniques). The second dependency sets that in the same conditions, if we expect a year of very little rain, we must plant trees as adult as possible.

Conclusions and Future Lines

We have showed one approach for defining and storing fuzzy quantifiers in a fuzzy database context, in order to use these quantifiers in fuzzy queries, fuzzy constraints or fuzzy data mining

applications. This paper defines different kind of fuzzy quantifiers with zero, one and two arguments.

This work is linked with the FIRST-2 approach for designing fuzzy databases and the FSQL language [6][7][10]. In particular, this definition is coherent with the approach presented in [1] for achieving fuzzy databases starting from classical databases. These works show some interesting algorithms and ideas addressed to DBA's (database administrators) in order to translate classical databases to fuzzy ones.

Many future research lines arise from here. Some of them are how to use all these ten types of fuzzy quantifiers in fuzzy constraints and fuzzy queries (with FSQL or with other language or application), to study other possible types of fuzzy quantifiers with arguments and to apply fuzzy quantifiers of fuzzy databases in fuzzy data mining applications. This paper presents an application to gets fuzzy dependencies (particularly fuzzy GD) and to measure, using fuzzy quantifiers, how good these GD are. We have included an example about a reforestation database. The example shows that fuzzy dependencies may be very useful in a decision support system.

Acknowledgements

This work has been partially supported by the "Ministry of Education and Science" of Spain (projects TIN2006-14285 and TIN2006-07262) and the Spanish "Consejería de Innovación Ciencia y Empresa de Andalucía" under research project TIC-1570.

References

[1] Ben Hassine, M.A., Touzi, A.G., Galindo, J., & Ounelli, H. (2008). How to Achieve Fuzzy Relational Databases Managing Fuzzy Data and MetaData. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science Reference (http://www.igi-pub.com).

[2] R.A. Carrasco, M.A. Vila, J. Galindo, J.C. Cubero (2000). FSQL: a Tool for Obtaining Fuzzy Dependencies. 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU, pp. 1916-1919. Madrid (Spain).

[3] R.A. Carrasco, M.A. Vila, J. Galindo, J.C. Cubero (2000). Fuzzy Global Dependencies in Databases. 10th Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF), pp. 175-180. Sevilla (Spain).

[4] Delgado M., Sánchez D., Vila M.A. (2000). Fuzzy Cardinality Based Evaluation of Quantified Sentences. International Journal of Approximate Reasoning, 23, pp. 23-66.

[5] Galindo J., Medina J.M., Cubero J. C., García M.T. (2001). Relaxing the Universal Quantifier of the Division in Fuzzy Relational Databases. International Journal of Intelligent Systems. Vol. 16-6, June, pp. 713-742.

[6] Galindo, J., Urrutia, A., & Piattini, M., (2006). Fuzzy Databases: Modeling Design and Implementation. Hershey, USA: IDEA Group.

[7] Galindo, J. (Ed.), (2008). Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science Reference (http://www.igi-pub.com).

[8] Liétard, L., & Rocacher, D. (2008). Evaluation of Quantified Statements using Gradual Numbers. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science Reference (http://www.igi-pub.com).

[9] Liu Y., Kerre E.E. (1998). "An overview of fuzzy quantifiers (I). Interpretations" and "An overview of fuzzy quantifiers (II). Reasoning and applications". Fuzzy Sets and Systems, Volume 95, Issue 1, pp. 1-21 and Issue 2, pp. 135-146.

[10] Urrutia, A., Tineo, L., Gonzalez, C. (2008). FSQL and SQLf: Towards a Standard in Fuzzy Databases. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science Reference (http://www.igi-pub.com).

[11] Zadeh L.A. (1983). A Computational Approach to Fuzzy Quantifiers in Natural Languages. Computer Mathematics with Applications, 9, pp. 149-183.