Implementing Imperfect Information in Fuzzy Databases

Afef Bahri, Salem Chakhar, Yosr Naïja
LAMSADE laboratory, University of Paris Dauphine,
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, Paris, France.
Telephone: +33 (1)-44-05-49-18. Fax: +33 (1)-44-05-41-11.
Email: {bahri, chakhar, naija}@lamsade.dauphine.fr
and Rafik Bouaziz
Department of Computer Science, Faculty of Economic Sciences and Management,
Route de l'Aérodrome, BP 1088, 3018 Sfax, Tunisia.
Telephone: +216 74-55-54-44. Fax: +216 74-77-88-99.
Email: raf.bouaziz@fsegs.rnu.tn

***Abstract*— Information in real-world applications is often vague, imprecise and uncertain. Ignoring the inherent imperfect nature of real-world will undoubtedly introduce some deformation of human perception of real-world and may eliminate several substantial information, which may be very useful in several data-intensive applications. In database context, several fuzzy database models have been proposed. In these works, fuzziness is introduced at different levels. Common to all these proposals is the support of fuzziness at the attribute level. This paper proposes first a rich set of data types devoted to model the different kinds of imperfect information. The paper then proposes a formal approach to implement these data types. The proposed approach was implemented within a relational object database model but it is generic enough to be incorporated into other database models.**

***Index Terms*— Fuzzy database, fuzzy set, imperfect information, possibility distribution.**

I. Introduction

Information in real-world applications is often vague, imprecise and uncertain. Ignoring the inherent imperfect nature of real-world will undoubtedly introduce some deformation of human perception of real-world and may eliminate several substantial information, which may be very useful in several data-intensive applications (e.g. CAD/CAM, geographical and environmental information systems, decision support systems). In database context, there are several proposals to develop database models that support fuzziness, uncertainty and impreciseness of real-world [23]. Most efforts have been oriented towards the extension of the conventional relational database models [4], [5], [11], [18] and towards the development of tools that allow for imprecise querying most often in relational database contexts [20]. We enumerate also some extensions of object-oriented [2], [10], [15], [24] and semantic database models [3], [7]–[9], [12], [14], [22].

In these different extended database models, vagueness, imprecision and uncertainty are introduced at different levels. Within object-oriented and semantic database models, we may distinguish three levels as pointed by [25]. The first level concerns classes, relationships and attributes domains which may be fuzzy and may have degrees of membership (d.o.m) in the model. The second level is related with the fuzzy occurrences of objects/entities and relationships. This means that entities/objects and instances of relationships belong to their classes and relationships with a given d.o.m. The third level concerns the attributes where these last ones are authorized to take imprecise, uncertain and vague values. Works within the relational database model propose the addition of uncertainty essentially at the tuple level where tuples belong to their relations with given degrees of membership, and at the attribute level by authorizing values of attributes to be imprecise, uncertain and/or vague.

Common to all these proposals is the support of imperfect information (i.e. uncertain, imprecise, or fuzzy) at the attribute level. This paper proposes first a rich set of data types devoted to model the different kinds of imperfect information as well as conventional crisp data types. To facilitate data manipulation and for computing efficiency, the different types of attributes values (crisp, imprecise, uncertain, fuzzy, unknown, undefined or null) are uniformly represented through possibility distribution. The paper then proposes a formal approach to implement these data types. The proposed approach was implemented within a relational object database model but it is generic enough to be incorporated into other database models, especially in non first-normal-form relational, object-oriented and semantic database models.

The rest of the paper is as follows. Section II briefly describes some proposals of fuzzy databases. Section III enumerates the proposed data types devoted to represent different kinds of imperfect information. Section IV provides our proposal for implementing imperfect information at the attribute level. Section V concludes the paper.

II. Fuzzy databases

Fuzzy information has been extensively investigated in the context of relational database model. The earliest works

focalize on the modelling of incomplete information (i.e. null, undefined and unknown data types) within attributes values. To support the modelling of imperfect information and complex objects, it was necessary to relax the first-normal-form assumption in relational database model by authorizing attributes to be multi-valued. Accordingly, several non first-normal-form relational database models have been proposed.

Later models in fuzzy relational databases are based on the use of similarity relation [4], proximity relation [11], [18], or resemblance relation [17] defined on the domain of scalar data types where each pair of scalars in the attribute domain are mapped, through similarity, proximity or resemblance relation, to the interval [0,1]. For instance, in [4] the authors propose an extended fuzzy relational database model based on similarity relations. In [18] the authors replace the similarity relation used in [4] with a more general proximity relation. This permits to remove the max-min transitivity restriction associated with similarity relation and so gives more freedom to the users for expressing their values structures. Another extension of the relational data model basing on proximity relation is provided in [11] by extending the proposals of [18] through a higher characterization of the proximity relation proposed in [18].

Another family of extended fuzzy relational models are based on using fuzzy sets and possibility distributions associated with either tuples or attributes values or with both. At the tuple level, each tuple is associated with either a d.o.m indicating to which extent the tuple belongs to its relation or a possibility distribution measuring the possibility that the tuple belongs to its relation. At the attribute level, the attributes values are represented as possibility distribution of the form $\{\mu(u_1)/u_1, \mu(u_2)/u_2, \cdots, \mu(u_n)/u_n\}$ where $\mu(u_i)$ measures the possibility that the attribute has the value u_i.

More complex fuzzy relational database models are obtained by incorporating proximity relation, possibility distribution and fuzzy sets (see for example [6] for more details). For example, in [13] the authors propose an extended fuzzy relational model where possibility and proximity arise in a relation simultaneously.

On the other hand, the relational database model is often used for implementing other databases models, essentially semantic and object-oriented ones. For instance, in [8] the authors propose a methodology for the design and development of fuzzy relational databases. This methodology was used for mapping a fuzzy ER into a relational one. In [22], the IFO model was extended to the ExIFO (Extended IFO) to represent uncertainty as well as precise information. The authors provide also an algorithm for mapping the schema of the ExIFO model to an extended NF^2 database model. In [12], the IFO data model is extended to support fuzziness. The obtained model, denoted IF_2O, is then mapped into a relational fuzzy database schema.

In semantic and object-oriented database models fuzziness is introduced with one or several of the three levels mentioned in the introduction. These may concern all or a subset of the constructs and relationships of the model. Generally, the entity/class and subclass/superclass relationships are associated with d.o.m that indicate the extent to which entities are encapsuled in their classes or the extent to which subclasses are specializations of their superclasses. These works differ essentially on the ways the different d.o.m are computed.

We enumerate several extensions of the semantic data models. An extension of the graph-based IFO database model for supporting uncertainty and imprecision is provided in [19]. In this paper the authors introduce proposals for handling ill-defined values including values with semantic representation, values with semantic representation and conjunctive meaning, values with semantic representation and disjunctive meaning, representation of uncertainty. The uncertainty is supported at attribute, object and class levels.

Basing on fuzzy set theory, extension of the major constructs and relationships of the well-known ER/EER models—including generalization/specialization, superclass/subclass, and shared subclass and category—to support uncertainty and imprecision of real-world at model/type (i.e. entities, relationships and attributes domains have d.o.m in the model), type/value (i.e. values of entities and relationships have d.o.m in their corresponding entities and relationships types) and attributes levels are introduced in [9].

Another proposal for extending the ER data model to support fuzziness is reported in [8]. The possibility-based Fuzzy ER data model supports fuzziness and uncertainty at attribute, entity, relationships and instance/entity relationships levels.

In [22], based on similarity relations, the IFO model was extended to the ExIFO (Extended IFO) to represent uncertainty as well as precise information. ExIFO support uncertainty at the attribute, entity, and instance/class levels.

The authors in [14] use fuzzy set theory and possibility distribution to extend the EER model into a fuzzy EER (FEER) one to cope with imperfect as well as complex objects at and model/tytpe, type values and attributes levels.

Based on fuzzy set theory and possibility distribution, the author in [12] introduces fuzziness in the different constructs of the semantic IFO data model, including printable type, abstract type, free type, grouping, aggregation, fragment and ISA relationship.

The Fuzzy Semantic Model (FSM) is a recently proposed fuzzy semantic data model [3], [7]. FSM uses basic concepts of classification, association, specialization, generalization, composition, aggregation and grouping that are commonly used in semantic modelling and supports the fuzziness of real-world at attribute, entity, class and intra and inter-classes relationships levels.

There are also several recent extensions to object-oriented database models. The FOOD (Fuzzy Object-Oriented Database) model in [24] is a similarity-based fuzzy object-oriented data model. FOOD supports fuzziness at attribute, object/class as well as subclass/superclass relations levels. One important aspect of FOOD is that it supports the AND, OR and XOR semantics to handle the multi-valued attributes values.

The UFO (Uncertainty and Fuzziness in Object-oriented) in [10] is another object-oriented database model that supports fuzziness and uncertainty at attribute, object, class, and

entity/class and subclass/superclass relations levels. The authors also extend fuzziness to the methods level. In UFO, imprecision and uncertainty are expressed by means of normalized possibility distribution and non-zero plinth within a possibility distribution, respectively; and they are modelled by means of the "role objects" which is a new concept introduced by the authors. As mentioned by the authors, the role objects model uncertain, tentative information about objects and thus the uncertain roles that the objects may play.
A fuzzy object-oriented data model that is a follow-on of the IFO graph-based object model was proposed in [2]. In this paper, the authors use linguistic qualifiers to represent the notion of the strength and associate it with instance and object-class relationships.
In [15], based on possibility distribution and semantic measure method of fuzzy data, the authors extend some basic concepts in object-oriented databases, including attributes, objects, classes, objects-classes relationships, subclass/super-class and multiple inheritances in order to support fuzzy information.
Which is common in all these database models is the fuzziness at the attribute level. Basically, attributes may have one of the following natures:

- *Single-valued*: means that the attribute cannot have more then one value at a given time.
- *Unknown*: means that we cannot decide which is the value of the attribute among several plausible values.
- *Undefined*: means that there is not any defined value that can be assigned to the attribute.
- *Null*: means that we cannot even know whether the attribute's value is unknown or undefined.
- *Multi-valued*: means that the attribute can have several values at a given time.

The *null*, *unknown* and *undefined* data types permit to model incompleteness in databases. Several other data types devoted to model imperfect information will be introduced later in section III.
Note finally that the values of a multi-valued attribute are often related with different logical connectors (i.e. AND, OR or XOR) but this is not dealt with here.

III. Imperfect information representation

The objective of fuzzy databases is primarily to handle imperfect information in databases. In [12], the author distinguishes five types of imperfect information:

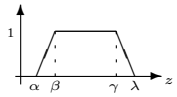
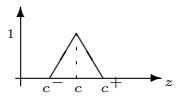
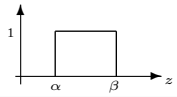
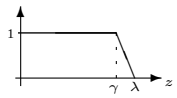
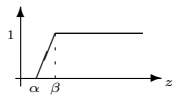
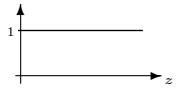
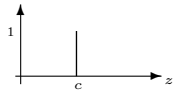
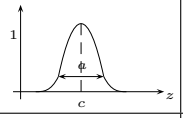
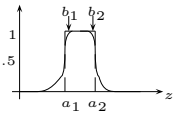
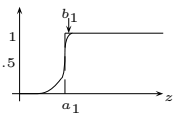
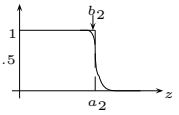
- *Inconsistency*: is a kind of semantic conflict that holds when some aspects of real-world is irreconcilably represented more than once in the database (e.g. when the *age* of a person is stored as 34 and 37);
- *Imprecision*: is relevant to the content of an attribute value and means that a choice must be made from a given range (interval or set) values (e.g. the *age* of a person is the set $\{17, 18, 19, 20\}$ or the *height* is in the interval [1.00-1.95]);
- *Vagueness*: is like imprecision but which is generally represented with linguistic terms (e.g. the *age* of a person is the linguistic "young");
- *Uncertainty*: is related to the degree of truth of attribute value, and it means that we can apportion some, but not all, of our belief to a given value or groups of values. It results from a lack of information and is that related to the designer and not to the object/concept being modelled. (e.g. the possibility that the *age* of a person is 35 right now should be about 90%);
- *Ambiguity*: it means that some elements of the model lack complete semantics leading to several possible interpretations.

The same author adds that imprecision, vagueness and uncertainty are the main types of imperfect information. Fuzziness comes from the impossibility to define sharp or precise borders, and therefore it is often associated with vagueness. In fuzzy database literature, uncertainty and imprecision are often represented through fuzzy sets and possibility distribution. In turn, vagueness is often represented through fuzzy set theory, similarity, proximity and/or resemblance relations. The combination of several approaches is also frequent.
In addition to fuzzy, imprecise and uncertain, values of attributes may be unknown, undefined or null-valued. To facilitate data manipulation and for computing efficiency while giving the maximum flexibility to the users, the different types of attributes values (crisp, imprecise, uncertain, fuzzy, unknown, undefined or null) will be uniformly represented through possibility distribution.
The following list enumerates the different data types which we think permit to model almost all kinds of imperfect information. Note that these types are extensions of the ones proposed in [16]. We also added several new ones. Especially, linguistic labels defined on sinusoidal possibility distributions and the "more than" and "less than" data types are not defined in [16].

- *Fuzzy range*. This data type handles the "more or less" information between two numeric values. The graphical representation of possibility distribution of this data type is shown through Model I.1 in Table I and may be written as $\{\mu(z)/z : z \in D\}$. D is the domain of the attribute values and $\mu(z)$ is the d.o.m of z in the fuzzy set on which the attribute is defined. This set is denoted A in Table I. As it is shown in Table I, four parameters are required to define the possibility distribution of this data type: α, β, γ and λ. The parameters β and γ represent the support of the fuzzy set associated with the attribute values and α and λ represent the limits of the transition zones;
- *Approximate value*. This data type handles the "about" some numeric value information. The graphical representation of possibility distribution of this data type is shown through Model I.2 in Table I and may be written as $\{\mu(z)/z : z \in D\}$. Here, three parameters are required: the central value of the concept c, the limit of left transition zone c^- and the limit of right transition zone c^+;
- *Interval*. Model I.3 in Table I shows the graphical representation of the possibility distribution of a classical crisp range. Mathematically, this possibility distribution

TABLE I

DIFFERENT DATA TYPES

Data type A	Model	Representation	Parameters	$\mu_A(z)$
Fuzzy range label e.g. *age*= more or less between 20 and 30	I.1		$\alpha, \beta, \gamma, \lambda$	$\mu_A(z) = \begin{cases} 1, & \text{if } \beta \le z \le \gamma; \\ \frac{\lambda - z}{\lambda - \gamma}, & \text{if } \gamma < z < \lambda; \\ \frac{z - \alpha}{\beta - \alpha}, & \text{if } \alpha < z < \beta; \\ 0, & \text{Otherwise.} \end{cases}$
Approximate value e.g. *age*=about 35	I.2		c, c^-, c^+	$\mu_A(z) = \begin{cases} 1, & \text{if } z = c; \\ \frac{c^+ - z}{c^+ - c}, & c < z < c^+; \\ \frac{z - c^-}{c - c^-}, & c^- < z < c; \\ 0, & \text{Otherwise.} \end{cases}$
Interval e.g. *age* $\in [25, 35]$	I.3		α, β	$\mu_A(z) = \begin{cases} 1, & \text{if } \alpha \le z \le \beta; \\ 0, & \text{Otherwise.} \end{cases}$
Less than value e.g. age= less than 35	I.4		γ, λ	$\mu_A(z) = \begin{cases} 1, & \text{if } z \le \gamma; \\ 0, & \text{if } z < \lambda; \\ \frac{\lambda - z}{\lambda - \gamma}, & \text{if } \gamma \le z \le \lambda. \end{cases}$
More than value e.g. age= more than 35	I.5		α, β	$\mu_A(z) = \begin{cases} 1, & \text{if } z \ge \beta; \\ 0, & \text{if } z \le \alpha; \\ \frac{z - \alpha}{\beta - \alpha}, & \text{if } \alpha < z < \beta. \end{cases}$
Unknown	I.6			$\mu_A(z) = 1 \; ; \; z \ge 0$
Undefined	I.7			$\mu_A(z) = 0 \; ; \; z \ge 0$
Real number e.g. *age*=30	I.8		c	$\mu_A(z) = \begin{cases} 1, & \text{if } z = c; \\ 0, & \text{Otherwise.} \end{cases}$
Linguistic label e.g. *age*=young	II.1		a, c	$\mu_A(z) = \frac{1}{(1 + (a(z-c)^2}; \; z \ge 0$
Linguistic label e.g. *age*=young	II.2		a_1, a_2, b_1, b_2	$\mu_A(z) = \begin{cases} \frac{1}{1 + \frac{z - a_1 - b_1}{b_1}}, & \text{if } z < a_1 + b_1; \\ 1, & \text{if } a_1 + b_1 \le z \le a_2 - b_2; \\ \frac{1}{1 + \frac{z - a_2 + b_2}{b_2}}, & \text{if } z > a_2 - b_2. \end{cases}$
Linguistic label *age*=very old	II.3		a_1, b_1	$\mu_A(z) = \begin{cases} \frac{1}{1 + \frac{z - a_1 - b_1}{b_1}}, & \text{if } z < a_1 + b_1; \\ 1, & \text{if } a_1 + b_1 \le z; \end{cases}$
Linguistic label e.g. *age*=very young	II.4		a_2, b_2	$\mu_A(z) = \begin{cases} 1, & \text{if } z \le a_2 - b_2; \\ \frac{1}{1 + \frac{z - a_2 + b_2}{b_2}}, & \text{if } z > a_2 - b_2. \end{cases}$

may be written as $\{\mu(z)/z : z \in D\}$. The parameters required here are the limits of the range α and β;

- *Less/More than value*. These data types focalize only on one side of a value. The graphical representations of the possibility distributions of "less than" and "more than" data types are shown in Models I.4 and I.5 in Table I, respectively. Mathematically, the possibility distribution associated with both of them may be written as $\{\mu(z)/z : z \in D\}$. Two parameters are required to define this data type: the value of interest (γ or β) and the limit of the transition range (λ or α);

- *Set of possible scalar assignments*. This permits to handle attributes defined on a set of scalars. For example, the height of a person may be defined as the set *height*={tall,very tall}, which is represented through possibility distribution as {1.0/tall,1.0/very tall}. A proximity relation is often defined on the domain of this data type. We denote this data type with Model III.1;
- *Set of possible numeric assignments*. This data type is similar to the previous one. It differs only on the fact that it is defined on a set of numeric values. For example, the height of a person may be defined as the set *height*={1.85,1.95}, which is represented through possibility distribution as {1.0/1.85,1.0/1.95}. This data type will be designed as Model III.2;
- *Possibility distribution over discrete domain*. This data type is represented through standard possibility distribution where possibility degrees in [0,1] are associated with each of the domain values. More formally, we have $\{p_1/d_1, \cdots, p_n/d_n\}$; where p_i and d_i for i trough 1 to n are the possibility degrees and the domain values, respectively. Note that the domain values may be numbers as well as scalars. A proximity relation is often associated with scalar-based domains. This data type will be designed as Model III.3;
- *Possibility distribution over a numeric ordered domain*. In this data type, the possibility distribution is defined on an ordered set of numeric values as for example *age*={0.7/25,0.8/26,1.0/27,0.8/28,0.8/30}. More generally, we have $\{p_1/d_1, \cdots, p_n/d_n\}$ with $p_i \leq p_{i+1}$. This data type will be designed as Model III.4.
- *Simple number*. This is a crisp data type which is handled as in conventional databases. The possibilistic representation of a simple number n is $\{1.0/n\}$. Model I.8 in Table I shows the graphical representation of the possibility distribution of this data type;
- *Simple scalar*. This is a crisp data which is handled as in conventional databases. The possibilistic representation of a simple scalar s is {1.0/s}. A proximity relation is often associated with this data type. We denote this data type with Model III.5;
- *Matching degree*. This is a real number in [0,1] that refers to the degree to which a concept is achieved (e.g. Quality=0.7). The possibilistic representation of a matching degree m is $\{1.0/m\}$. This data type will be designed as Model III.6;
- *Unknown*. This data type means that we cannot decide which is the value of the attribute among several plausible values. But the attribute may take any value from its domain. Accordingly, the possibilistic representation of the unknown data type is $\{1.0/z : z \in D\}$. Model I.6 in Table I shows the graphical representation of the possibility distribution of this data type;
- *Undefined*. This data type means that there is not any defined value that can be assigned to the attribute. This means that no one of the domain values is authorized. Accordingly, the possibilistic representation of undefined data type is $\{0/z : z \in D\}$. Model I.7 in Table I shows the graphical representation of the possibility distribution of this data type;
- *Null*. This data type means that we cannot even know whether the attribute's value is unknown or undefined. Accordingly, the possibilsitic representation of undefined data type is $\{1.0/\text{Unknown}, 1.0/\text{Undefined}\}$. This data type will be designed as Model III.7;
- *Symbolic*. This is a crisp data type which takes its values on a set of symbolic values related with the XOR operator. The possibility representation of this data type is $\{0/s_1, \cdots, 1.0/s_i, \cdots, 0/s_r\}$, which means that the attribute value is s_i. This data type will be designed as Model III.8;
- *Linguistic label*. Models II.1-II.4 in Table I are the graphical representation of the possibility distribution of the linguistic label data types. Model II.1 represents the sinusoidal model. The parameters required here are the central value of the attribute c and the parameter that governs the shape of the d.o.m a. Model II.2 is an extension of the previous one that applies when the central value of the concept may take a range of values instead of only one value. Four parameters are required here: the limits of the central range a_1 and a_2; and the left and right transition zones b_1 and b_2, respectively. Note that a_1 and a_2 are the *crossover* (or *transition*) *points* defined such that $\mu(a_1) = \mu(a_2) = 0.5$. Models II.3 and II.4 are the asymmetric extensions of Model II.1 that apply when only the left or right side of the concept is of interest. The required parameters are a_1 and b_1 for Model II.3; and a_2 and b_2 for Model II.4. The mathematical representation of all these data types is $\{\mu(z)/z : z \in D\}$. Attributes defined as linguistic labels need also to be associated with proximity relations defined on their domains.

IV. IMPERFECT INFORMATION IMPLEMENTATION

The authors in [16] enumerate three levels for implementing imperfect information in databases: database system level, database level and metaknowledge base level. Here, we adopt these three levels and we add a new one:

- *Database system level*: this level is associated with extended data manipulation languages devoted to handle different fuzzy operations that the database system should support. This level is not the scope of this paper.
- *Database level*: here we are concerned with the way the imperfect information is internally stored. This concerns both attributes values and extent definition of different fuzzy relations/classes. Our solution to handle imperfect information at the attribute level is provided in the rest of this section.
- *Metadata level*: this level concerns the intent definition of fuzzy relations/classes. Note that this level is called metaknowledge in [16]. This will be introduced in this section.
- *Model base level*: this level groups the definition of all (i) the functions used to compute membership degrees, and (ii) the functions associated with different data types that

are used to generate their possibility distributions. This level is not discussed in this article but it is detailed in [3].

As underlined above, the approach detailed hereafter was implemented within a relational object database model but it is generic enough to be implemented in other database models, especially for non first-normal-form relational, object-oriented and semantic database models. For the two last ones, the meta-relations will be replaced with specific classes. In the following text, the term "meta-relation" should be replaced with "specific class" when the implementing is within object-oriented or semantic database models.

In order to store the specificity of all the attributes, we define a meta-relation, called ATTRIBUTES, at the metadata level with the following attributes:

- *attribute-id*: it uniquely identifies each attribute defined at the database level. It constitutes also the primary key of the ATTRIBUTES meta-relation. Note that the key attribute(s) in this relation and in the other ones are underlined.
- *attribute-name*: it stores the name of the attribute. As for classical databases, the same fuzzy relation/class can not have two attributes with the same name but the same attribute name may appear in different fuzzy relations/classes.
- *defined-in*: denotes the fuzzy relation/class to which the attribute belongs.
- *data-type*: which is a multi-valued attribute that stores the attribute type which may take any one of the list of §III. For crisp attributes, this attribute works as in conventional databases (it may take the values of integer, real, float, etc.). For fuzzy attributes, the *data-type* attribute stores the fuzzy data type itself and the basic crisp data type on which the fuzzy data type is based.

An example of ATTRIBUTES meta-relation is as follows:

attribute-id	*attribute-name*	*defined-in*	*data-type*
attr-15	*star-name*	STAR	{string}
attr-16	*type-of-star*	STAR	{symbolic}
attr-17	*age*	STAR	{linguistic label, integer}
attr-18	*luminosity*	STAR	{linguistic label, real}
attr-19	*location*	STAR	{linguistic label, real}
attr-20	*weight*	STAR	{interval, real}
attr-77	*field-of-research*	SCIENTIST	{scalar}
attr-80	*age*	SCIENTIST	{linguistic label, integer}

These are some attributes associated with the fuzzy class STAR and the class SCIENTIST taken from an example provided in [1].

The parameters associated with different linguistic terms that appear in the domain of any linguistic data type are stored at the metadata level. They will be used to compute the different d.o.m and for query processing. The number of parameters needed is different from one linguistic data type to another and it may vary from zero to four parameters. Thus, several solutions are possible to store these parameters. We can, for example, use one common meta-relation with four attributes devoted to store the different parameters. In that time, we may have "null" values any time the number of parameters associated with one linguistic value is less than four. Another solution is to group data types along the number of required parameters. After that, four relations are needed for data types with one, two, three or four parameters, respectively (we do not have to define a relation for unknown and undefined attribute data and other data types that need no parameters). An ameliorated version of this solution is adopted in [16]. The authors use a common meta-relation similar to ATTRIBUTES and a specific attribute serves as a pointer to two meta-relations. One meta-relation is used to store the "margin" parameter needed for approximately data type (Model I.2 in Table I). The second meta-relation contains a list of fuzzy objects defined in the database columns. This meta-relation contains two specific attributes: one used to store the data type and the other points out to three new meta-relations devoted to store the parameters of qualifier labels defined over the matching of a query, proximity relations associated with scalar data types (Models II.1-II.4 in Table I and Model III.5 in the list of §III), and trapezoidal-based possibility distribution (Models I.1-I3 in Table I) of linguistic labels and query quantifiers, respectively. In the last meta-relation, four attributes (*Alpha*, *Beta*, *Gamma*, *Delta*) are used to store the trapezoidal-based possibility distributions parameters. In the special case of interval data type, the attributes *Alpha* and *Beta* store the same value. This is also true for attributes *Gamma* and *Delta*. The same meta-relation with the four parameters is also used to store undefined, unknown and null data types, which generate an excessive storage space since these data types require no parameters and the different parameters will be "null"-valued.

One drawback of the solutions cited above is that any time we need to add a new linguistic data type or to change the adopted linguistic data types, we may have to update the meta-relations structures. Here, we propose a straightforward solution that does not depend on the parameters number and can be used with any fuzzy model. In fact, we define a common meta-relation with a multi-valued attribute that stores all needed parameters. This meta-relation, denoted by PARAMETERS, contains one line for each linguistic value that appears in the domain of any linguistic data type attribute (or the list of the authorized values for symbolic data type). Its attributes are:

- *attribute-id*: references one attribute that appears in the meta-relation ATTRIBUTES.
- *label*: stores a linguistic term belonging to the attribute domain. For symbolic data types this attribute takes a "nil" value.
- *parameters*: is a multi-valued attribute used to store the parameters required for generating the possibility distribution of the linguistic term. Attributes with no parameters, will not be included in PARAMETERS meta-relation.

An example of a PARAMETERS meta-relation is as follows:

attribute-id	*label*	*parameters*
attr-16	nil	{nova, supernova}
attr-17	very young	$\{0.0, 0.0, 0.5, 1\}$
attr-17	young	$\{0.8, 1.7, 2, 2.5\}$
attr-17	old	$\{2.3, 5, 10, 15\}$
attr-17	very old	$\{12, 17, 50, 60\}$

The meta-relation PARAMETERS permits also to generate the domain of linguistic or symbolic data types. This needs only to group together all the linguistic labels having the same *attribute-id* in the meta-relation PARAMETERS. For example, the domain of attribute *attr-17* above is {very young, young, old, very old}. The domain of a symbolic data type is the list of the terms in the *parameters* attribute.

The attribute values are stored at the database level along with the extent definition of their relations/classes. As mentioned above, to facilitate data manipulation and for computing efficiency, the different types of attributes values are uniformly represented through possibility distribution. However, these distributions are not explicitly stored in the database but generated automatically during data manipulation and query processing by means of specific functions associated with different data types.

Attributes values may be crisp, fuzzy or both. This need only to be indicated in the intent definition of the fuzzy relations/classes the attributes belong to. The database system should allow users to insert values of any data type that is consistent with the formal definition of the attribute. At the extent definition of the fuzzy relation/class, each fuzzy attribute is mapped into a new composite one composed of three component attributes:

- *attr-value*: stores the value of the attribute as provided by the user.
- *data-type*: stores the data type of the value being inserted.
- *parameters*: is a multi-valued attribute used to store parameters associated with the attribute value that are used to generate its possibility distribution.

The *data-type* attribute is used both at the extent definition and in the intent definition to allow users insert values of different data types, which may have different number of parameters. This will offer more flexibility to the user. Nevertheless, the different data types defined at the extent level should be consistent with the formal definition of the attribute at the intent level. For instance, the formal definition of the attribute may be a trapezoidal-based possibility distribution with four paraments but the user may introduce a crisp value (with no parameter at all), an interval (with two parameters only) or an approximate value (with three parameters only). Remark that attribute *data-type* at the extent definition is not a multi-valued one.

The extent definitions of two attributes taken from an example in [1] are as follows:

luminosity *attr-value*|*data-type*|*parameters*
{high, linguistic label model II.1, {25,5}} {0.1L_s, real,{nil}} {more than 10L_s, more than linguistic label, {7.5L_s,10}}

weight *attr-value*|*data-type*|*parameters*
{10W_s, real, {nil}} {[12W_s-15W_s], interval, {12,15}} {about 17W_s, approximate value, {15,17,18}}

The symbols L_s and W_s are the luminosity and the weight of the Sun, respectively; they are often used as measurement units.

Some data types (Models II.1-II.4, III.1, III.3 and III.5) require also to define the proximity relation between the elements of their respective domains. Proximity relations are stored at the metadata level through the meta-relation PROXIMITY which has the following attributes:

- *attribute-id*: references the attribute for which the proximity relation is defined.
- *label-1* and *label-2*: denote two linguistic terms belonging to the attribute domain.
- *degree*: stores the similarity degree between two linguistic terms denoted by *label-1* and *label-2*.

The following is an example of a meta-relation PROXIMITY:

attribute-id	*label-1*	*label-2*	*degree*
attr-17	very young	young	0.7
attr-17	very young	old	0.1
attr-17	very young	very old	0.0
attr-17	young	old	0.1
attr-17	young	very old	0.0
attr-17	old	very old	0.8

Proximity relations are reflexive and symmetric. Thus, there is no need to handle the proximity degrees for pairs of the type (x, x) and only one pair from (x, y) and (y, x) should be stored for any two linguistic labels x and y.

V. Conclusion

Information in real-world applications is often vague, imprecise and uncertain. In database context, several fuzzy database models have been proposed. In these works, fuzziness is introduced at different levels. Common to all these proposals is the support of fuzziness at the attribute level. This paper proposes first a rich set of data types devoted to model the different kinds of imperfect information. To facilitate data manipulation and for computing efficiency, the different types of attributes values are uniformly represented through possibility distribution. The paper then proposes a formal approach to implement these data types. The proposed approach was implemented within a relational object database model but it is generic enough to be implemented in other database models, especially for non first-normal-form relational, object-oriented and semantic database models.

Acknowledgment

The authors would like to thank Mr. Abdelkader Telmoudi for his valuable comments on an earlier version of this paper.

References

[1] A. Bahri, R. Bouaziz, S. Chakhar, Y. Naïja and A. Telmoudi, *Implementing the fuzzy semantic model through a fuzzy relational object database model*. IEEE Transactions on Fuzzy Systems, Submitted.

[2] G. Bordogna, G. Pasi and D. Lucarella, *A fuzzy object-oriented data model for managing vague and uncertain information*. International Journal of Intelligent Systems, Vol. 14, pp. 623-651, 1999.

[3] R. Bouaziz, S. Chakhar, V. Mousseau, R. Sudah and A. Telmoudi, *Database design, implementation and querying within the fuzzy semantic model*. Information Science, (In revision).

[4] W. P. Buckles and F. E. Petry, *Fuzzy representation of data for relational databases*. Fuzzy Sets and Systems, Vol. 7, pp.213-226, 1982.

[5] W. P. Buckles and F. E. Petry, *A fuzzy model for relational databases*. Fuzzy Sets and Systems, Vol. 33, pp.213-226, 1992.

[6] W. P. Buckles and H. Prade, *An introduction to fuzzy set and possibility theory based approaches to the treatment of uncertainty and imprecision in database management systems*. Proceedings of the 2nd Workshop on Uncertainty Management in Information Systems: From Needs to Solutions.

[7] S. Chakhar and A. Telmoudi, *Extending database capabilities: Fuzzy semantic model*. Proceedings of the International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT'04), Sousse, Tunisia, March 15-20, 2004.

[8] N. Chaudhry, J. Moyne and E.A. Rundensteiner, *An extended database design methodology for uncertain data managmenet*. Journal of Database Management, Vol. 121, No. 1/2, pp. 83-112, 1999.

[9] G. Q. Chen and E. E. Kerre, *Extending ER/EER concepts towards fuzzy conceptual data modeling*. Proceedings of the 1998 IEEE International Conference on Fuzzy Systems, pp. 1320-1325, 1998

[10] N. V. Gyseghem and R.D. Caluwe, *Imprecision and uncertainty in UFO database model*. Journal of American Society of Information Sciences, Vol. 49, No. 3, pp. 236-252, 1998.

[11] S. Kuma De, R. Biswas and A. R. Roy, *On extended fuzzy relational database model with proximity relations*. Fuzzy Sets and Systems, Vol. 117, pp. 195-201, 2001.

[12] Z. M. Ma, *A conceptual design methodology for fuzzy relational databases*. Journal of Database Management, Vol. 16, No. 2, pp. 66-83, 2005.

[13] Z. M. Ma, W. J. Zhang and W. Y. Ma, *Assessment of data redundancy in fuzzy relational databases based on semantic inclusion degree*. Information Processing Letters, Vol. 72, pp. 25-29, 1999.

[14] Z. M. Ma, W. J. Zhang, W. Y. Ma and G. Q. Chen, *Conceptual design of fuzzy object-oriented databases using extended entity-relationship model*. International Journal of Intelligent Systems, Vol. 16, pp. 697-711, 2001.

[15] Z. M. Ma, W. J. Zhang and W. Y. Ma, *Extending object-oriented databases for fuzzy information modeling*. Information Systems, Vol. 29, pp. 421-435, 2004.

[16] J. M. Medina, M. A. Vila, J. C. Cubero and O. Pons, *Towards the implementation of a generalized fuzzy relational database model*. Fuzzy Sets and Systems, Vol. 75, pp. 273-289, 1995.

[17] E. A. Rundensteiner, L. W. Hawkes and W. Bandler, *On nearness measures in fuzzy relational data models.* International Journal of Approximate Reasoning, Vol. 3, pp. 267-298, 1989.

[18] S. Shenoi and A. Melton, *Proximity relations in the fuzzy relational database model*. Fuzzy Sets and Systems (Supplement), Vol. 100, pp. 51-62, 1999.

[19] M. A. Vila, J. C. Cubero, J. M. Medina and O. Pons, *A conceptual aproach for deal with imprecision and uncertainty in object-based data model*. International Journal of Intelligent Systems, Vol. 11, pp. 791-806, 1996.

[20] Q. Yang, C. Liu, J. Wu, C. Yu, S. Dao, H. Nakajima and N. Rishe, *Efficient processing of nested fuzzy SQL queries in fuzzy databases*. International Conference on Data Engineering, pp. 131-138, Taipei, Taiwan, March, 1995.

[21] A. Yazici, B. P. Buckles and F. E. Petry, *A semantic data model approach to knowledge-intensive applications*. International Journal of Expert Systems: Research and Applications, Vol. 8, No. 1, pp. 77-91, 1995.

[22] A. Yazici, B. P. Buckles and F. E. Petry, *Handling complex and uncertain information in the ExIFO and NF^2 data models*. IEEE Transactions on Fuzzy Systems, Vol. 7, No. 6, pp. 659-676, 1999.

[23] A. Yazici and R. George, *Fuzzy database modeling*. Heidelberg: Springer-Verlag, 1999.

[24] A. Yazici, R. George and D. Aksoy, *Design and implementation issues in fuzzy object-oriented data model*. Information Science, Vol. 108, pp. 241-260, 1998.

[25] A. Zvieli and P. P. Chen, *Entity-relationship modeling and fuzzy databases*. Proceedings of the 1986 IEEE International Conference on Data Engineering, Los Angeeles, California, USA, Februray 5-7, 1986, pp. 320-327.