A Study of Fuzzy Data Bases: an Application to a Peruvian case

Pedro R. Acosta De La Cruz
Universidad Nacional de Ingeniería
Lima, Perú

Miguel A. Meza Pinto
Universidad ESAN
Lima, Perú

José A. Flores Salinas
Universidad ESAN
Lima, Perú

Freddy C. Tineo Córdova
Universidad ESAN
Lima, Perú

Abstract **- This paper aims to show the various types of data that contain in an intrinsically way fuzzy or imperfect data that are presented in the real world. A form of implementation is described that allows extending the capabilities of a database by using layers levels, emphasizing in the concept of inheritance. What is described is illustrated by an example applied to the Peruvian reality, which is shown with a certain level of detail.**

Keywords - Fuzzy Database, Inheritance Of Classes, Data Type, Possibility Of Occurrence, Base Layers, Class Hierarchy.

I. INTRODUCTION

La The information in the applications in the real world is often vague and imprecise. Ignoring these imperfect natural features of the real world, it introduces some imperfections of human perception of the real world and therefore you can delete important information that can be very useful in applications of data processing, such as: geographic systems, environmental, medical systems, academic systems, etc.

In the context of Database (DB), there have been many efforts to develop DB model to support the fuzzy, uncertain and imprecise of the real world. Many efforts have been oriented to the extension of DB conventional models and development tools that allow the consultation of inaccurate information. In this regard we have for example in [1] that develops types of fuzzy data as a new way to manage fuzzy data structures, thus, the properties are sorted in different levels of accuracy according to their relationship with the concept it represents the type.

II. FUZZY MODEL FOR DATABASES

The aim of the Fuzzy Data Bases is to handle imperfect information. In the article [2] five types of imperfect information are distinguished today:

- Inconsistent: is a type of semantic conflict that marks when some aspects of the real world are represented more than once in the Database and in a different way. For example: when the age of a person is stored as 34 and 37.
- Inaccuracy: it means that the choice of an attribute must be taking as basis a range of values (interval or set). For example the old person is in the set {17, 18, 19, 20} or the height of a person is in [1.0, 1.95].
- Vagueness: it is the inaccuracy, but represented as linguistic terms. For example: a person's age is the word “young”.
- Uncertainty: it is related to the grade of certainty of the value of the attribute. For example: to ensure that a person's age is 35 years old with a 90% probability.
- Ambiguity: means that some elements of the model are full semantics, which leads to various possible interpretations (they are given with a synonymous word that can express the same; for example between student and pupil).

One of the ways to handle the fuzzy structures is to characterize different types of data that we believe that it allows to model almost everything imperfect information can be. According to [3] there have been found the following lists:

a) *Fuzzy Range*: This data type handles information that is more or less between two numeric values. Can be written as the following set $\{\mu(z) / z: z \in D\}$ where D is the domain and $\mu(z)$ is the degree of membership in the fuzzy set in which the attribute is defined.

b) *Approximate value*: This data type handles information "almost" of some numerical value. For example age is almost 35 years.

c) *Interval*: It handles information that is in a range. For example the age is between 15 and 20 years.

d) *Minimum or maximum of a value*: Unlike the intervals, these data types concentrate only on one side of the interval. For example the age to have ID card must be or greater or equal to 18 years.

e) *Set of possible scalar assignments*: This allows you to manage attributes defined in a set of scalars. {For example the temperature environmental can be low, medium, high} that is represented by distributions of possibility as 1.0/low, 1.0/average, 1.0/high.

f) *Set of possible numeric assignments*: This data type is similar to the previous one with the difference that is defined on a set of numeric values. For example the temperature can be defined by the set {1.5, 1.6, 1.7} that is represented through a distribution of possibilities to 1.0/1.5, 1.0/1.6, 1.0/1.7.

g) *Probability distributions on a discrete domain*: This data type is represented by its degree of probability standard. These degrees of possibility are in the range [0, 1] and are associated with each domain values. Formally we have the set $\{p_1/d_1, \ldots, p_n/d_n\}$, where pi are the degrees of probability and di are the domain values. For example for temperature you can assign the following degrees of probability {0.6, 0.7, 0.8}, with the domain values {1.5, 1.6, 1.7}, we have the following distribution {0.6/1.5, 0.7/1.6, 0.8/1.7}. This can be set in an orderly manner criterion $p_i \leq p_{i+1}$.

h) *Single number*: This is a well-defined data type that is handled in the usual way. Its probabilistic representation of a single number *n* is $\{1.0/n\}$.

i) *Single scalar*: This is a well-defined data type that is handled in the usual way. Its probabilistic representation of a single scalar s is $\{1.0/s\}$.

j) *Degree assigned*: It is a real number in the interval [0, 1] which is concerned with the degree in which a concept is achieved. For example if quality is equal to 0.7, your probabilistic representation of a degree assigned m is $\{1.0/m\}$.

k) *Unknown*: It means that we cannot decide the value of the attribute among some possible values, and however the attribute can take any value in the domain. Your probability of an unknown data type is $\{1.0 / z: z \in D\}$.

l) *Undefined*: This type of data means that there is no value defined that can be assigned to the attribute. This would mean that none of the values in the domain are authorized. Its probabilistic representation is $\{0 / z: z \in D\}$.

m) *Null*: It means that we can't even know that the value of the attribute is unknown or undefined. Its probabilistic representation is {1.0/unknown, 1.0/not defined}.

III. IMPLEMENTATION OF THE FUZZY MODEL

In previous section we have developed different types of fuzzy data as a way to manage information blurred or imperfect. To implement those new types we have two strategies:

a) A new system that incorporates new fuzzy types as an intrinsic ability.

b) A new layer that implements fuzzy types and add them to the Data Base Oriented to Objects.

In [1] the type of this class and the properties are ordered in different levels of precision according to the relationship that represents the type. They show how typical kinds of a database orientated to objects can implement Fuzzy types and how mechanisms of creation and inheritance are modeled.

In [3] uses the notation ERD (Entity Relationship Data Model) has been extended with a notation that represents the theory blurred and it applies to the problem of management of fuzzy sets.

IV. IMPLEMENTATION

Among the considerations for implementation, we distinguish two parts, which are described below:

A. Implementation of the creation of a fuzzy object

The effect of send the message *new*(α) to a class with a component structural *S* and a component of behavior *B*, consists in creating an object that incorporates the joint Sα of attributes with a behavior defined joint Bα elements. For example, if we consider a blurred class C, represented by the hierarchy $C_1, \ldots, C_{i-1}, C_b, C_{i+1}, \ldots, C_n$, the creation code would read as follows:

```
Begin
    $i \leftarrow 1$;
   while ($i$ < n) and ($\alpha_{i+1} \geq \alpha$) do
      $i \leftarrow i+1$;
   end;
   Ci.new();
End;
```

This procedure consists in finding the class in the hierarchy that incorporates the α corresponding and then create a new instance of this class.

B. Implementation of a fuzzy type inheritance

Two aspects are the form of inheritance is distinguished:

a) Inheritance without the spread of the fuzzy: consists in incorporating the attributes and methods to the Kernel both the component structural as the component behavior of the subclass, eliminating the fuzzy in the inherited properties. This is represented by the following formulation followed by the corresponding graph, see Fig. 1.

$$S_C = S_{sup - \alpha} \cup S_{sub}$$

$$B_C = B_{sup - \alpha} \cup B_{sub}$$

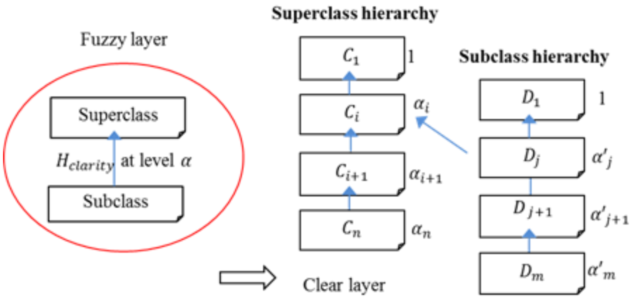

Figura 1. Hierarchy without fuzzy spread

b) Inheritance with spread of the fuzzy: this type of inheritance is to incorporate methods and attributes inherited in the components and structural behavior of the subclasses, keeping the blurriness of the inherited properties. This is represented by the following formulation followed by the corresponding graph, see Fig. 2.

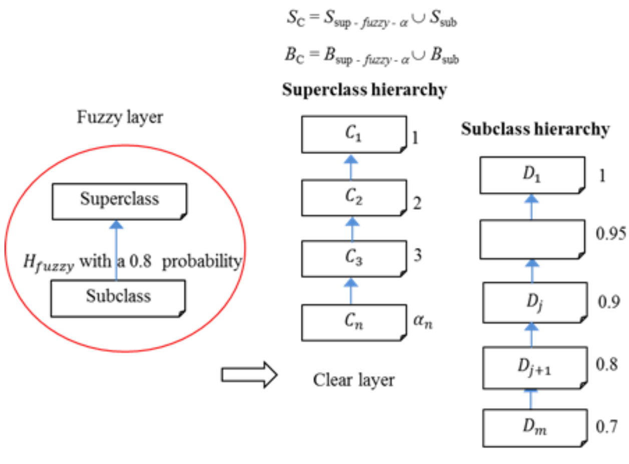

Figura 2. Hierarchy with fuzzy spread

V. APPLICATION TO A PERUVIAN CASE

A. Description

The example that we are taking is referred to a higher education institution. In a particular way we are defining the class Student, in which different attributes defined, for example: *academic performance* (low, average and high), *level of study* (a whole number that goes from 1 to 10 which indicates the relative cycle), *English language proficiency* (Basic, intermediate and advanced).

In a classic Database, each instance can only take one of the values described. However, it is usual that the user requirements require things like the following: that *academic performance* is between average and high; that in *level of study* the student is taking courses of several cycles; for students that is between the eighth and tenth cycles, have *English language proficiency* among intermediate and advanced.

Now let's consider the concept of the object Student, with the following levels of precision:

- Minimum characteristics: ID, surname_and_names, date_of_birth.
- First level of precision: specialty, address, phone_number.
- Second level of precision: academic_performance, level_of_study, English_ language_proficiency.

This structure following the statement [1] will use the following fuzzy set:

- *S* = 1/ID + 1/*surname_and_names* + 1/*date_of_birth*

+ 0.9/*specialty* + 0.9/*address* + 0.9/*date_phone_number*

+ 0.8/*academic_performance* + 0.8/*level_of_study*

+ 0.8/*English_lenguage_proficiency*

All three levels have been defined in this way.

The representation using the concept of class hierarchy will correspond to different levels of precision with which the concept represented by the fuzzy type can be considered, see Fig. 3.

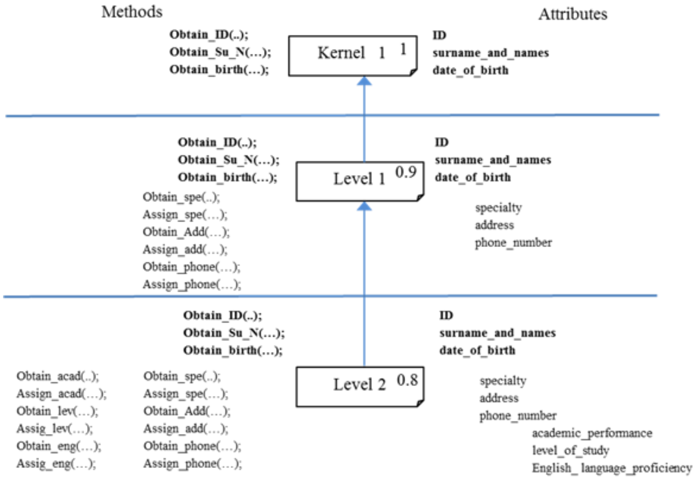

Figura 3. Clarity in the hierarchy of the inheritance of the class Student

B. Creation of subclasses

In the Peruvian case we establish the hierarchy of classes and subclasses shown in the Fig. 4.

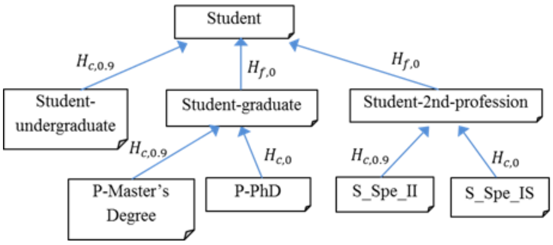

Figura 4. Fuzzy layer

C. Using the inheritance

Inheritance and non-propagation of the fuzzy was considered in the previous section. Here we illustrate this concept as applied to our Peruvian case.

a) The subclass Student-undergraduate, the subclass student - undergraduate, is regarded as a subclass of the class student with a probability of 0.9 and no fuzzy spread in the following way:

```
Class Student-undergraduated
     Structure:
          (1, tutor): chain
          (1, study_location): chain
          (1, number_of_credits_taken): number
     Behavior:
Obtain_tutor(…);
          Assign_tutor(…);
End Student-undergraduated
```

And their graphical (see Fig. 5) presentation is as follows:

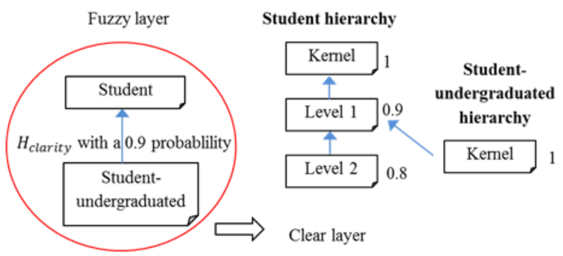

Figura 5. Creation of the class Student-undergraduated

b) The subclass Student-graduated, is considered as a subclass of the class student with a 0.8 possibility with a fuzzy spread of the following form:

```
Class Student-graduated
    Structure:
        (1, weighted_average): number
        (1, number_of_credits_approved): number
        (1, date_of_admission): date
    Behavior:
Obain_weighted_average (…);
Assign_weighted_average (…);
End Student-graduated
```

And their graphical (see Fig. 6) presentation is as follows:

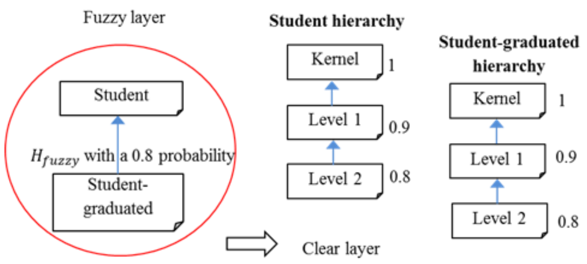

Figura 6. Creation of the class Student-graduated

D. *Coding of the class student*

If we concentrate in the level fuzzy of the class student, a programmer would have to have access to the following fuzzy classes that are based in Fig. 4.

```
Class Student
    Structure:
        (1, ID): chain
        (1, surnames_and_names): chain
        (1, date_of_birth): date
        (0.9, especialty): chain
        (0.9, adress): chain
        (0.9, phone_number): number
        (0.8, academic_performance): number
        (0.8, level_of_study): number
        (0.8, english_language_proficiency): chain
    Behavior:
        Obtain_ID(…);
        …
End Student

Class Student-undergraduated
  Inherits from Student with a 0.9 probability and with fuzzy
spread
    Structure
        (1, tutor): chain
        (1, study_location): chain
        (1, number_of_credits_taken): number
        …
    Behavior:
        Obtain_tutor(…);
       Assign_tutor(…);
        …
End Student-undergraduated

Class Student-graduated
  Inherits from student with a probability close to zero, with a
fuzzy spread
    Structure:
        (1, weighted_average): number
        (1, numbers_of_credits_approved): number
        (1, date_of_admission): date
        …
    Behavior:
        Obtain_weighted_average(…);
        Assign_weighted_average(…);
        …
End Student_graduated

Class Student_2nd_profession
  Inherits from student with a probability close to zero, with a
fuzzy spread
    Structure:
        (1, first_specialty): chain
        (1, origin_institution): chain
        (1, date_of_admission): date
        …
    Behavior:
        Obtain_first_specialty(…);
        Assign_first_specialty(…);
        …
End Student-2nd-profession

Clase P- Master's Degree
    Inherits from Student with a 0.9 probability and without
fuzzy spread
          Structure:
        (1, profesional_title): chain
        (1, origin_institution): chain
                   …
    Behavior:
        Obtain_profesional_title (…);
        Assing_profesional_title (…);
        …
End P-Master's Degree

Clase P-PhD
  Inherits from Student with a 0.9 probability and without
fuzzy spread
    Structure:
        (1, master_title): chain
        (1, origin_institution): chain
```

```
                    …
    Comportamiento:
        Obtain_master_title(…);
        Assign_professional_title(…);
        …
End P-PhD

Class S_Esp_II
  Inherits from Student-2nd-profession with a 0.9 probability
and without fuzzy spread
    Structure:
        (1, title_date): date
        (1, number_of_credits_recognized): number
                    …
    Behavior:
        Obtain_number_of_credits_recognized (…);
        Assign_number_of_credits_recognized (…);
        …
End S_Esp_II

Class S_Esp_IS
  Inherits from Student-2nd-profession with a 0.9 probability
and with fuzzy spread
    Structure:
        (1, title_date): date
        (1, number_of_credits_recognized): number
                    …
    Behavior:
        Obtain_ number_of_credits_recognized (…);
        Asiggn_number_of_credits_recognized (…);
        …
End S_Esp_IS
```

VI. CONCLUSIONS

In this work we have presented them concepts of type of fuzzy data, based mainly in the different types that these can take. Based on article [1] we have worked on the class Student, which is the main objective of an academic institution, establishing the recommended layers and showing what could be a possible definition.

Future work would be oriented to implementing a real database system, extending the facilities that these systems can provide.

REFERENCES

[1] NIcolas Marin, Maria Amparo Vila, Ignacio J. Blanco, Olga Pons: Fuzzy types as a new layer on an Object, Oriented Database System, en International Spain Information Processing and management of uncertainty in knowledge-based system (IPNU)(11) pp 1099-1106.

[2] Abef Bahri, Salem Chakhar, Yosr Naija, Rafik Bouaziz: Implementing Imperfect Information in Fuzzy Databases en The International Symposium on Computational Intelligence an Intelligent Informatics, October 14-16, 2005, Hammametk, Tunisia.

[3] N. Marín , O. Pons, M. A. Vila: A Strategy for Adding Fuzzy Types to an Object-Oriented DataBase System en International Journal of Intelligent Systems, VOL 16, 863-880(2001).

[4] Salem Chakhar and Abdelkader Telmoudi: Extending Database Capabilities:Fuzzy Semantic Model en International Conference: Sciences of Electronic, Technologies of Information and Telecomunications March 15-20, 2004 – Tunisia.

[5] Amel Grissa Touzi and Mohamed Ali Ben Hassine: New Architecture of Fuzzy Database Management Systems in The International Arab Journal of Information Technology, Vol 6, Nro 3, July 2009.

[6] Gregory Vert, Ashley Morris, Molly Stock, Piotr Jankowski: Extending Entity-Relationship Modeling Notation to Manage Fuzzy Datasets en International Spain Information Processing and management of uncertainty in knowledge-based system (IPNU)(11) pp 1131-1138.

[7] Angelica Urrutia, Marcela Varas, Jose Galindo: Diseño de una Base de Datos Difusa Modelada con UML 6to Workshop de Ingeniería de Requisitos y Ambientes Software IDEAS 2003, Asuncion Paraguay, Mayo pp 145-155.

[8] Angelica Urrutia, Marcela Varas: Especificaciones de Bases de Datos aplicando EER Difuso en CibSE (2005) 315-324.