1998 Second International Conference on Knowledge-Based Intelligent Electronic Systems, 21-23 April 1998, Adelaide, Australia. Editors, L.C. Jain and R.K.Jain

Conceptual Design of Fuzzy Object-Oriented Databases

Adnan YAZICI and Ali ÇINAR

Department of Computer Engineering,
Middle East Technical University (METU), 06531, Ankara / Turkey,
e-mail: {yazici,cinar} @ ceng.metu.edu.tr

Abstract. An important research trend in databases is to handle different types of uncertainty at conceptual level. The trend of incorporating complex objects in databases presents opportunities for representing imprecision and uncertainty that were difficult to integrate cohesively in simple database models. In this study we introduce a conceptual data model by extending ExIFO to handle both complex and uncertain, mainly fuzzy, objects and classes.

Keywords: Uncertainty, Object-Oriented Databases, Conceptual Modeling

1. INTRODUCTION

Modeling requirements for new applications and flaws in the currently used models have lead to the definition of more powerful models that are extended relational and object-oriented database models. But, these models have still a number of serious limitations in implementing increasing number of real-world applications (Message Handling Systems(MHS), Office Automation Systems(OAS), Decision Support Systems(DSS), Geographic Information Systems (GIS), Expert Database Systems(EDS), which involve not only complex information, but also uncertainty and fuzziness.

The conceptual models, semantic data models, and object-oriented data models can represent imprecise and complex object structures without fragmentation of aggregate data and model the complex relationships. Only few models [8, 14] are concerned with handling object-oriented paradigm & uncertain information in a conceptual data model.

In this study a fuzzy object-oriented conceptual data model is developed and called $ExIFO_2$. This model includes representation of fuzzy information and handles object-oriented concepts. In the construction of this model, a previous study on uncertain data representation performed by Yazici and Merdan [2] and a study on object database design performed by Poncelet et. al [3] are combined and further extended for a conceptual data model for fuzzy object-oriented database design. The resulting conceptual model, $ExIFO_2$, attempts to preserve the acquired strengths of semantic approaches, while integrating concepts of the object-oriented paradigm and fuzziness.

2. THE IFO DATA MODEL

The IFO data model [1] is a mathematically defined data model that incorporates the fundamental principles of semantic database modeling within a graph-based representational framework. More formally, an IFO schema is a directed graph with various types of vertices and edges, representing atomic objects, constructed objects, fragments and ISA relationships. A basic IFO schema is a combination of these pieces.

2.1. Objects

The representation of various object structures that are being modeled, called types, constitutes the basis of any IFO schema. Three kinds of atomic types are distinguished. For building more complicated types, there exist two constructs that can be applied to these atomic types.

The first atomic type is called the *printable*, and corresponds to objects of predefined types that serve as the basis for input and output. The second atomic type is the *abstract* type, and corresponds typically to objects in the real world that have no underlying structure. The third type is called *free* and corresponds to entities obtained via ISA relationships. The graphical representations of these atomic types are below in Figure 2.1.

a) Printable b) Abstract c) Free

Figure 2.1: Atomic IFO Types

0-7803-4316-6/98/$10.00 ©1998 IEEE

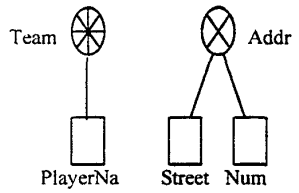


Figure 2.2: Constructed Object Types

Non-atomic objects are formed from an underlying type by applying a finite set of *grouping* or *aggregation* constructors. The *grouping constructor* is used to depict a finite set of objects of a given structure. That is, each set of instances is an "object" having that particular structure. The second constructor is the *aggregation constructor*, which forms ordered n-tuples of instances, which are associated with a type. These two constructs can be applied recursively in any order to form more complex types. Figure 2.2 shows the graphical representation of these two constructs.

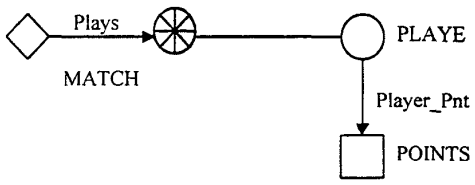


Figure 2.3: Fragments Example

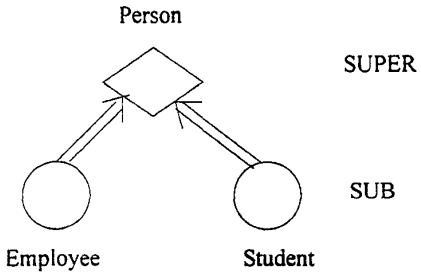


Figure 2.4: Specialization Example

2.3. ISA Relationships

The final structural component of the IFO model is the representation of ISA relationships. Two types of ISA relationships are distinguished: *specialization* and *generalization*. Specialization is depicted using a double arrow, and generalization is depicted using a broad arrow.

Specialization can be used to define possible roles for members of a given type (e.g. a PERSON might be a STUDENT as in Figure 2.4). However, generalization represents situations where distinct, preexisting types are combined to form a virtual type (e.g. the types CAR and MOTOR-BOAT might be combined to form VEHICLE type).

2.2. Fragments

The IFO data model uses *fragments* as the structural component for the representation of functional relationships. Fragments provide naturally clustered representations of types and their associated functions. Figure 2.3 shows a fragment where the function PLAYS maps a given MATCH to both a set of PLAYERS played in that MATCH and PLAYER_PNTS function maps each player and their POINTS received in the match. (Player is a free type, which is specialization of Person, but Person is not shown in example in order to keep it simple)

Furthermore, specialization is also used to restrict function ranges.

2.4. IFO Schemas

In the preceding subsections the basic building blocks for representing the structure of stored data in the IFO model were described. These building blocks are combined to form IFO schemas. The IFO model provides a natural framework for top-down schema design, beginning with the specification of the major object types arising in the application environment, then specifying subsidiary object types, and finally specifying the functions of all object types of the schema. Related users can have more information from [2] and [4].

3. THE ExIFO$_2$ DATA MODEL

ExIFO$_2$ is the extension of the semantic model IFO$_2$[3]. The extension that is realized in this study could be grouped in two. The first group includes the uncertainty and the second group includes the object-oriented paradigm. In the first group, two types of uncertainty are dealt with. The first type of uncertainty is considered to be at the *attribute-level*, meaning that "attributes" of objects may have values involving uncertainty. The second type of uncertainty is considered to be at the *Object and Class Level*. That is, some objects may have instances whose membership to the object set may be fuzzy (i.e. *fuzzy membership of an instance* of an object type) and/or some subclasses' membership degree is given for superclasses (Class/Subclass level uncertainty). In the second group, an explicit definition of the object identifier, which is object value independent, is integrated. To achieve these, all manipulated elements of IFO$_2$ are redefined. On the other hand, integrating the concepts of

alternative, composition and grouping for complex objects enhances the modeling power of ExIFO.

In the first step, types, functions and ISA relationships are given graphically and simply, in an informal way. The uncertainties and object-oriented paradigm are combined in one conceptual model, in $ExIFO_2$, whose description is given later.

3.1. Graphical Representation

All types of $ExIFO_2$ and object blocks are shown graphically in Figure 2.5, Figure 2.6 and Figure 2.7.

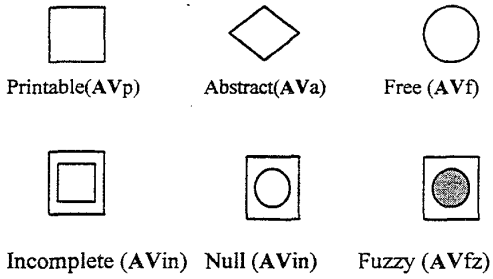


Figure 2.5: Fuzzy, Incomplete and Atomic Types

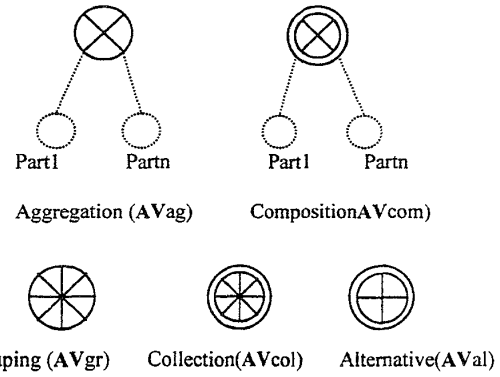
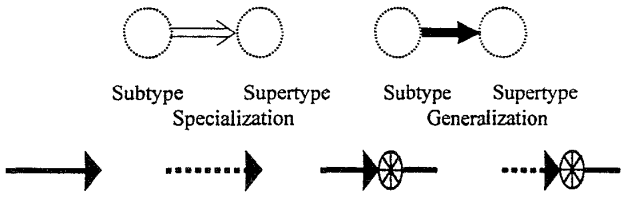


Figure 2.6 : Complex Types



a) Total Func. b) Partial Func. c) Complex Total Func. d) Complex Partial Func.

Figure 2.7: Function Types and ISA links

3.2. Extending the IFO Data Model for Object-Oriented Paradigm

In order to extend ExIFO data model for object-oriented paradigm, two main extensions are realized. Firstly, an explicit definition of the object identifier, which is object value independent, is integrated. To achieve this, all manipulated elements of the ExIFO model are redefined to be compatible with the object-oriented paradigm.

When an identifier is added for all types then there will be two types of domain for an object type. First one is the value domain that describes the possible values for its objects and second one is the identifier domain that is independent of its

object value. For an example, the object O_{str1} = (id_{str1}, 'Cinnah') is an object of type 'Street'.

Definition of Object Type: TYPESET is an infinite set of object types such that; For every type of T which is an element of TYPESET, Did(T) is an infinite set of symbols called the identifier domain of T and DOM(T) is an infinite set of symbols, including the empty set, called the value domain of T. Objects of type T are defined by pair (id, value) where $\forall$ o, o_1 of type T, $\exists$ id,id_1 $\in$ $Did(T)^2$, $\exists$ value, $value_1$ $\in$ DOM(T) such that: if o=(id, value), o_1 = (id_1,$value_1$) and id $<>$ id_1 then o $<>o_1$. The infinite set of objects of type T in this concept is called Obj(T).

3.3. Extending the IFO for Uncertainty

Attribute-level Uncertainty: For the representation of attribute-level uncertainty in the Extended IFO data model, two new constructors (*FuzzySet* and *IncompleteSet* constructors) are introduced. By using these constructors it is possible to explicitly represent attributes (types in IFO model) having uncertain values. These constructors are taken from the study done by Yazıcı & Merdan [2,19-20]

Three kinds of uncertainty are distinguished;

- The true data may belong to a specific set of values, *incompleteness*,
- The true data value is not known, *null*
- The true data value is available, but in descriptive term in the absence of precise data, *fuzzy*.

FuzzySet type of constructor is used to capture the types that have inherently fuzzy values. With *FuzzySet constructor*, a set of values of a given specified domain of a type (attribute in relational model terminology) can be defined. Only a subset of these values corresponds to the object of that structure type as a true instance.

The *FuzzySet* constructor is said to construct an instance in the form of a set whose elements are related to each other with 'OR' or 'XOR' or 'AND' semantics. The attribute values are differentiated according to their semantics. The conventions used as logical operators are; AND: <...>, OR: {....}, XOR: [...]. For instance, assume the domain of software tools to be **dom(PackSoft)** = {Excel, Access, FoxPro, MS Word, Ami Pro, Wordstar, Win NT, Win 95}. Then the following values for attributes are valid.

Lang attribute of person1 = <Win NT, Win 95 >
Lang attribute of person2 = {Excel, Access }
Lang attribute of person2 = [MS Word, Ami Pro]

Here, person1 use (know) both Win NT and Win 95, person2 can use Excel or Access or both, person3 can use only one word processor either Ms word or Ami Pro. Note that the values of dom(PackSoft) are similar to each other with a degree in [0,1] and these values are represented with a similarity relation [17].

The representation of attribute values that they construct is shown with O_i=(id_i, [O_i,O_{i+1}, ...,O_k]) or O_i=(id_i, <O_i,O_{i+1}, ...,O_k>) or O_i=(id_i, {O_i,O_{i+1}, ...,O_k}), where [O_i,O_{i+1}, ...,O_k], {O_i,O_{i+1}, ...,O_k} and <O_i,O_{i+1}, ...,O_k> denotes a subset of the domain, {O_1,O_2, ...,O_n}, of that attribute, where $1 \leq i \leq n$ and $1 \leq k \leq n$. The *color* attribute, for example, is an inherently fuzzy-valued attribute and defined by using the *FuzzySet* constructor.

Definition of AVfz & AVin in following figures is explained in the formal definition section.

The second constructor, *IncompleteSet* constructor, represents a given specified domain of a type, that the constructor defines, but it involves in different semantics compared to the *FuzzySet* constructor. Only one of the values from a specified range corresponds to an instance of that structure type. Therefore, the *IncompleteSet* constructor has 'XOR' semantics, since one and only one value of the set represented as a range is the *object* of the underlying type. This type of constructor is used for the attributes that have *incomplete* or *null values*.

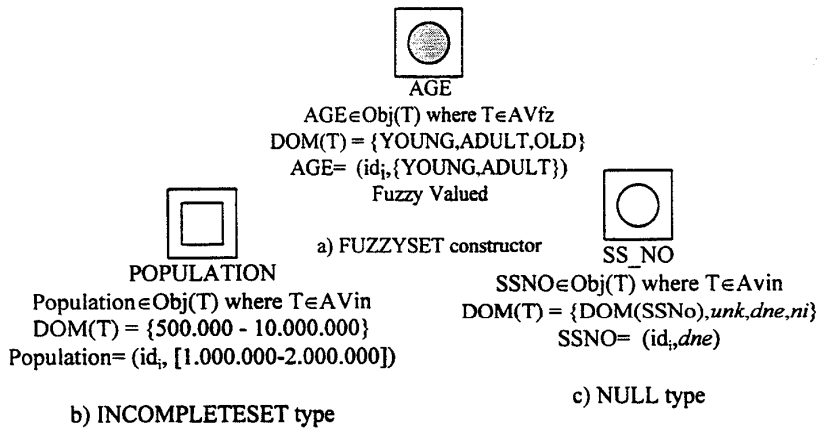


Figure 2.8: Attribute Level Uncertainty

For *incomplete-valued* attributes, $I = O_i$, where O_i is $(id_i, [O_h\text{-}O_m])$ where $[O_h\text{-}O_m]$ is the value of attribute A and $h \leq i \leq m$. The domain of attribute A is also $\{O_1, O_2, ..., O_n\}$, where $1 \leq h \leq n$ and $1 \leq m \leq n$. The value range specifies the minimum and maximum values with O_h and O_m respectively in the set.

For *null-valued* attributes the semantics is similar to *incomplete-valued* attributes, but the representation is different. The definition for *null-valued* attributes is, $I = O_i$, where O_i is one of $\{O_1, O_2, ..., O_n\}$, the domain values of attribute A.

Object and Class Level Uncertainty: The second level of uncertainty considered in this study is the fuzziness of instances of specific objects in the corresponding object set (class/object level) and fuzziness of subclasses in superclasses (class/subclass level).

In class/object level, capital "F" will be used in the representation of the object, to indicate the possibility of fuzzy membership. (e.g. a *minibus* can be considered as a *car* with a membership degree of 0.7).

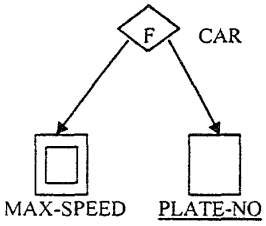


Figure 2.9 a: Fuzzy Membership to an Object Type.

In class/subclass level, again capital "F" will be used on generalization and specialization arrows in the representation of the object, to indicate the possibility of fuzzy membership. (e.g. a *caravan class* can be considered as a subclass of *House* class with a membership degree of 0.6).

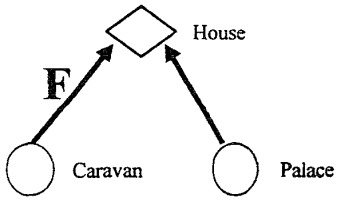


Figure 2.9 b: Fuzzy Membership between class an subclass.

3.4. Aggregation and Composition Types

The aggregation constructor represents the aggregation abstraction of semantic models defined by the Cartesian product. This constructor connects a subtype representing a part of an object to the type representing the entire object; thus, building a higher level object. This abstraction ignores some individual differences of the aggregated types. For example, the object type *motor-boat* is viewed as being an ordered pair of *hull* and *motor*. Semantically, the aggregation constructor derives only objects fully supported by the type, but, unlike the grouping constructor, the objects are ordered. An example is the list of components of an address of a person (i.e., street-name, zip code, city, etc.).

The *composition* constructor represents the aggregation abstraction of semantic models defined by the Cartesian product. The only difference of it from aggregation constructor is that, it provides an exclusive structure.

Example: Vehicle_Identification type is composed from 3 attributes, which are *Plate_No, Serial_No, Sertificate_No*. Each of these attributes must be unique. For example a plate number or certificate (ownership) number given to a vehicle by traffic, or serial number of the vehicle given by the producer must be unique. In this case instead of the aggregation constructor, composition constructor should be used.

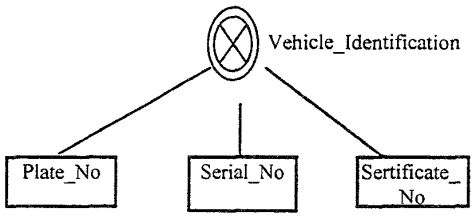


Figure 2.10: Composition example

3.5. Collection and Grouping Types

The grouping constructor is one of five high-level mechanisms of the IFO data model and is a kind of an abstraction in which the relationship among elements is considered as a higher level object known as a set object. This constructor depicts the type corresponding to sets of data values of an attribute. The grouping constructor is said to have 'AND' semantics, since each member of the set necessarily and precisely belongs to the set. Therefore, the grouping type of the constructors can capture the attributes that are multi-valued and crisp. All objects in the grouping abstraction are of the same type and not ordered. An example for this constructor is the set of *authors* of a book. Here a

set of authors for a specific book is precisely known and crisp; therefore, the authors are related with AND-semantics. That is, the whole set of the names of the authors specified is the only authors of the book, not some subset of the names of that set. The only difference of collection from *grouping* constructor is that, it includes an exclusivity constraint for the grouping.

Example: Assume the students in a primary school, each student may be in only one class like 1,2,3,4 or 5. In this case, a class collection is defined for providing exclusive disjunction of student objects.

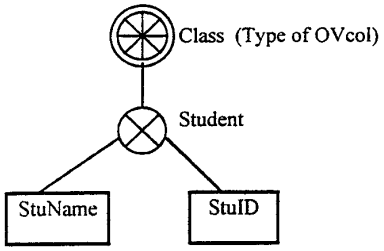


Figure 2.11: Collection example

3.6. Alternative Types

In $ExIFO_2$, structurally different types can be handled by using the alternative type concept. This constructor represents the IS_A generalization link enhanced with a disjunction constraint between the generalized types.

Example: Assume a tour organization that organizes its tours with vehicles like cars or buses. In every different tour, either a car or bus is used. (but not both of them) In this case vehicle type is defined as an alternative type

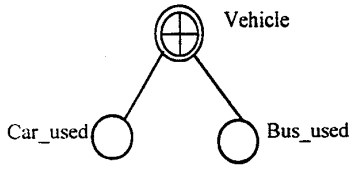


Figure 2.12: Alternative Type

3.7. Fragments & ISA links and Schemas

The definitions of schema and IS_A links are as same as in ExIFO section but now it is necessary to give a brief description of versions of fragments in $ExIFO_2$ [3], which are shown in Figure 2.13.

Using functions (called fragments) could link the types in ExIFO2. The goal of the fragment is to describe the properties of the principal types. Functions can be (simple, complex -multivalued-) or (partial-0:N link-, total-1:N link-) [3]. For instance in the following example; Person must have a name, Person may have ChildNames, Person has at least one telephone number and may have zero or more first names.

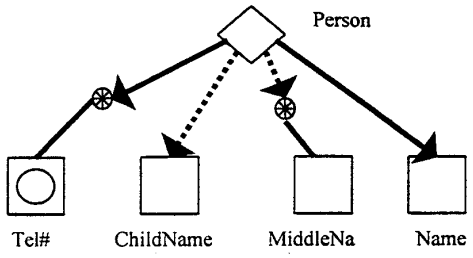


Figure 2.13: Function Example

3.8. Collection, Composition and Alternative Types

Enhanced $EXIFO_2$ includes new types, namely *alternative*, *collection* and *composition* types. To include these types in FOOD, new default method definitions will be created to satisfy exclusivity. For example, in Figure 2.14, consider the *vehicle_identity* example

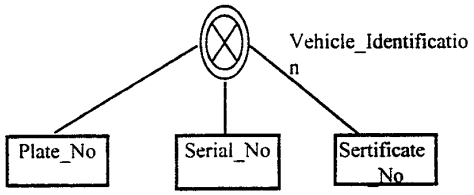


Figure 2.14: Composition example

Since it is *composition*, all the objects of construction types (here, all values of the attributes constructing the type) must be used only once. For this requirement following methods must be added to classes of these types default. These methods are semantics constraint checkers.

Boolean
Method::Check_Compo_TypeName()
Boolean
Method::Check_Collec_TypeName()

For instance, the method *Check_Compo_TypeName()* checks the exclusive constraint for composition of a type T. When a new instance is created, it verifies if an object contains at least one same-valued attribute in it. It returns true if no element exists before. It returns false if

an element exists before. As an example consider that *Vehicle_identity* has two instances;

('06_AD_216', 'B2341233112', '567_433_345')
(06_BC_541', 'C454544444', '890_452_189')

When a new instance is tried to be created whose attribute values are

(06_AD_216', 'D454545445', '123_234_345')

method *Check_Compo_Vehicle_Identity()* executes and checks if all attributes are used only once or not. Here, since the *plate_no* given is used before (so exclusivity constraint is not satisfied) this is not a valid instance and it can not be created. By the same way, *Check_Collec_TypeName()* checks the exclusivity constraint for collection and alternative types.

4. CONCLUSION

In this study, a conceptual modeling approach for the representation of complex-uncertain information using object-oriented paradigm is described. The ExIFO$_2$ model attempts to preserve the acquired strengths of semantic approaches, while integrating concepts of the object paradigm and fuzziness by adding new constructors. Thus, we can proceed further in design and development of the advanced knowledge-intensive applications such as Decision Support Systems (DSS), Geographic Information Systems (GIS), and Expert Database Systems (EDS).

REFERENCES

1. S. Abiteboul and R. Hull, "IFO: A Formal Semantic Database Model," ACM Transactions on Database Systems, Vol. 12, No.4, Dec. 1987, pp. 525-565.
2. A. Yazici and O. Merdan, "Data Models for the Representation of Complex and Uncertain Information", Technical Report 95-11, METU, Department of Computer Engineering, Ankara/Turkey
3. P. Poncolet., M. Teissere, R. Cicchetti, and L. Lakhal, "Towards a Formal Approach for Object Database Design", Proceedings of the 19th VLDB Conference, Dublin, Ireland, 1993, pp. 278- 289.
4. S.H. Magdy, "A Close Look at the IFO Model", SIGMOD RECORD, Vol. 24, No.1, March 1995, 21-26.
5. A. Heuer, "A Data Model for Complex Object Based on a Semantic Database Model and Nested Relation", pp. 297-312.
6. M. Teisseire, P. Pascal, R. Richetti, "Towards Event-Driven Modeling for Database Design", pp. 285-296, Proceedings of the 20th Int. conference on Very Large Database, Santiago, 1994
7. C. Soutou, "Extraction of an IFO$_2$ Schema", Relational Database Reverse Engineering, pp. 467-478
8. A. Yazici, George R., Aksoy D., "Extending the Similarity-Based Fuzzy Object-Oriented Data Model", Information Sciences (International Journal) (to appear).
9. M. Bouzeghoub, E. Metais et al. A Design Tool for Object Databases LNCS, Vol. 436. Proceedings of the 2nd Conference on Advanced Information Systems Engineering, 1990, pp. 365-392.
10. A. Yazici, B.P. Buckles, and F.E. Petry, "A Semantic Data Model Approach to Knowledge-Intensive Applications," Int. Journal of Expert Systems: Research and Applications, April, 1995, V.8(1), pp:77-91.
11. A. Motro, "Accommodating Imprecision in Database Systems: Issues and Solutions," SIGMOD RECORD, Vol. 19, No.4, December 1990, pp. 69-74.
12. R. Zicari, "Incomplete Information in object-oriented Databases" SIGMOD RECORD, Vol.19, No.3, Sept. 1990, pp. 5-16.
13. A. Yazici, R. George, B.P. Buckles, and F.E. Petry, "Survey of Conceptual and Logical Data Models for Uncertainty Management", in Zadeh, L. A. and Kacprzyk, J. (Eds.) Fuzzy Logic for Management of Uncertainty, John Wiley and Sons, Inc., 1992, New York, USA, pp. 607-644.
14. M.A. Vila, J.C. Cubero, J.M. Medina, O. Pons, " A Conceptual Approach for Dealing with Imprecision and Uncertainty in Object-Based Data Models," International Journal of Intelligent Systems, Vol. 11 (10), 1996, pp.791-806.