About the Use of Ontologies for Fuzzy Knowledge Representation

Ignacio J. Blanco, Nicolás Marín, Carmen Martínez-Cruz, M.Amparo Vila
Dept. Computer Science and Artificial Intelligence
University of Granada. Granada, SPAIN.
C/ Periodista Daniel Saucedo Aranda, S/N,18071,
iblanco, nicm, cmcruz *, vila@decsai.ugr.es

Abstract

This paper proposes an ontology which enables fuzzy data to be defined in order to conceptualize them and to represent another type of information. The Fuzzy Knowledge Representation Ontology described is based on the fuzzy data theoretical model, and a method for classifying both classical and fuzzy data is proposed. This ontology defines a framework for storing fuzzy information in fuzzy data types by defining them using classes, slots, and instances.

Keywords: Fuzzy Data, Metadata, DBMS Catalog, Fuzzy Knowledge Representation, Ontology, Databases, Database Modeling

1 Introduction

A Relational Database Management System (RDBMS) extension for representing fuzzy data is obviously not a new problem. The problem arises when the system is extended in order to manage other structured data (logical aspects, data mining operations and data types, etc.) [2]. The extended system increases the number of catalog relations making very complex and tedious to understand and handle.

In recent years, knowledge representation trends have been concerned with the generalization and reuse of modelled problems. An ontology is "a set of objects, concepts, and other entities about which knowledge is being expressed and of relationships that hold among them" [9]. Nowadays, ontologies are used in knowledge representation, including the representation of metadata if necessary. Because of this feature, the use of ontologies could be a good solution for the complexity problem that have just arisen with the extension of a RDBMS.

In Section 2, a brief resume of the architectures that extends the DBMS is presented. In Section 3, a Fuzzy Knowledge Representation Ontology (FKRO) is proposed as the solution to the problem presented. An example of how a relation can be represented in the ontology is then shown in Section 4. Finally, some conclusions and future lines of work are presented as result of this first approach to knowledge representation.

2 Showing the Problem

To date, there have been many proposals for representing fuzzy data in specific RDBMSs. Vila in [6] introduced GEFRED (a fuzzy data representation model) and FIRST (an architecture definition in a real RDBMS in [7]). This architecture defined new fuzzy data types and operations enabling the system to make fuzzy queries to the database using an extension of SQL called FSQL [3, 4]. An implementation for all relations in the system catalog and an example of how structures can be stored in the database are described in [7]

Some extended models have been developed using the GEFRED model mentioned above. The extended FREDDI architecture described in [1] represents logical information in the fuzzy relational model. This architecture stores rules and intensional relations, manages fuzzy information, and implements inference algorithms for making deductions.

DMFIRST* architecture, based in the GEFRED* model[3, 4], incorporates another fuzzy data type that allows to define complex domains. These domains represent other types of information (e.g. XML, tables, etc.) necessary for data mining operations (such as fuzzy clustering, classification operations or fuzzy functional dependencies search) may be performed.

Using the previously described architectures, in [2] the infrastructure for a unified server was proposed. It integrates capabilities of all of these architectures and enables their functionalities to be combined. This integration would be capable of processing several types of queries in the same sentence; for example, queries about deductions with fuzzy data or deductions using the results of a data mining process. Moreover, the proposed architecture establishes a mechanism to add more implementations of new servers with different functionalities.

This proposal is very difficult to achieve, however, because of the complexity of the system and the difficulty of making this system scalable. The huge number of catalog relations, which enables the metadata to establish the structure of the information and domain constraints, also makes it more difficult to understand and to implement.

3 A Fuzzy Knowledge Representation Ontology and its Integration with Existing Infrastructure

In order to solve these problems, the unified server architecture should be isolated from a concrete DBMS representation. Our proposal is based on the fact that all architectures are integrated in a single hierarchy that generalizes the data representation. This solution will consist in an ontology that allows the integration of the existing infrastructures and the representation of the knowledge independently of the context and environment that it uses. In the following section a first prototype of an ontology for representing a part of the unified server is suggested.

An ontology for fuzzy information representation is proposed in this paper. The ontology includes knowledge about how to manage and represent uncertain and imprecise data. This representation of the ontology makes the structure of the GEFRED model easier to understand and avoids references to one specific DBMS implementation. The *Protegè* environment tool [8, 5] has been used to implement this. This ontology is a first approach for representing a relational fuzzy information knowledge ontology but its design will enable representation into other database systems (e.g. object-oriented ones).

Figure 1 shows how the system catalog is related to the ontology modeling it. The Ontology Client module carries out the same operations through the Ontology than the DBMS Clients. The connection between the ontology and the database needs an interface, the *Ontology Interface*, which establishes the communication and refreshes the data.

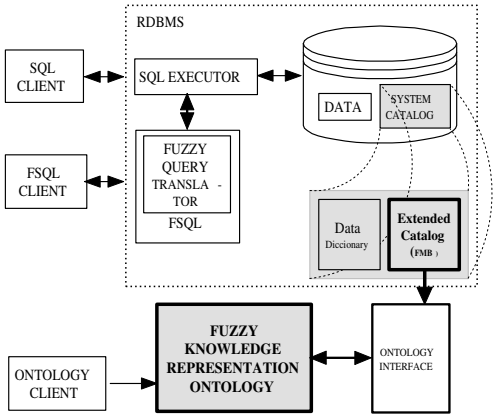

Figure 1: FIRST Architecture and Fuzzy Knowledge Representation Ontology Integration

The fuzzy model representation comprises two well-differentiated parts. Firstly, the ontology must define the necessary classes and slots to represent the metadata. These metadata define intermediate structures (i.e. fuzzy data types, domains, etc.) which are needed for representing the different fuzzy relations. Secondly, the ontology will be able to represent classical or fuzzy information as instances of the relations defined in the previous part.

The metadata definition in an Fuzzy RDBMS allows the system to define the structures, domains, data types, etc. described in the theoretical model. In this ontology, metadata establish how the fuzzy information will be stored. In

Figure 2, the class hierarchy for representing the metadata is shown. Table 1 describes all the classes and slots that cover this part of the ontology.

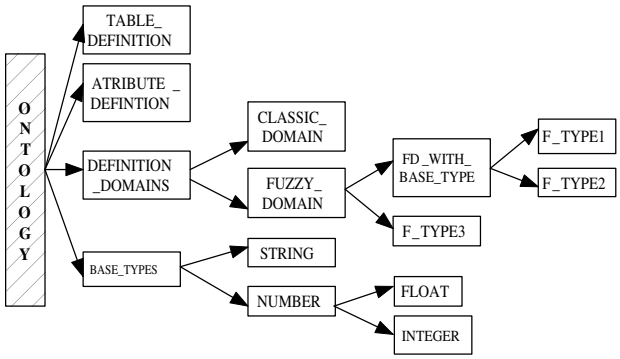

Figure 2: Metadata Representation Ontology

These classes allow the user to define any classic or fuzzy data relation. The relation name, attributes and domain are defined as instances of these classes. The instantiation of the *Table_Definition Class* begins the relation definition process. In this model, it is not necessary to specify concrete data types because the ontology enables generical classes to be defined which refer to them. Once the information structure has been defined, the classes for storing the information must be created in the hierarchy. This class generation process must be automatically developed in order to prevent user involvement. Figure 3 shows the part of the ontology which allows fuzzy information to be stored. The ontology classes and slots are described in Table 2.

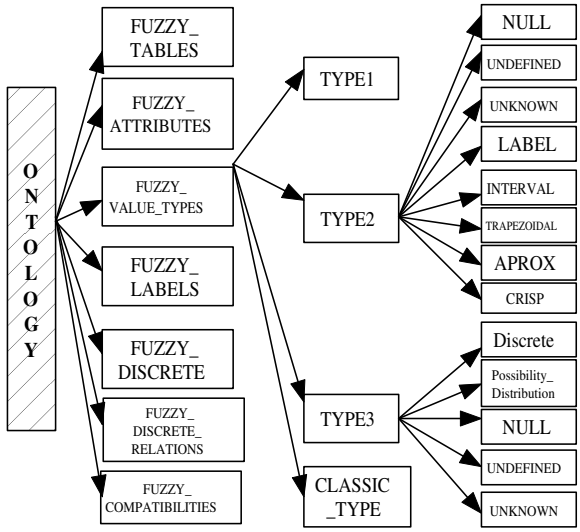

Figure 3: Fuzzy Information Representation Ontology

The *Fuzzy_Tables Subclasses* represent the relations defined in the ontology. There will be one subclass for each relation defined. The instantiation of one of these subclasses enables information to be stored in the ontology as if it were a database relation. This instantiation starts an instantiation process of all the classes involved in the data definition. The following example shows a relation defining process and a tuple storing.

4 An example: Fuzzy Information Description and Storing

The following FSQL sentence defines a new relation with fuzzy and classic attributes. This relation can be represented as a set of instances of the *Metadata Representation Ontology* classes. The *Table_Definition Class* instantiation begins a recursive process that will instantiate all the classes needed to define the relation. Table 3 describes all the instances which allow the *Cats* Relation to be defined.

```
CREATE TABLE Cats (
    Name          STRING,
    Behavior      FTYPE3(1)
    Age           FTYPE2 (1,1)  FLOAT
    Weight        FTYPE1 (3,5)  FLOAT )
```

where FTYPE3(1) represents that the *behavior* can be represented by only one discrete value. FTYPE2(1,1) means that Age is a *fuzzy data type 2* (see [7]) with *margin= 1* and *much = 1* (these values are described in table 1) and *FLOAT* represent the base type of this attribute values. The parameters of *Weight* have the same meaning than the *Age* ones.

In order to store data in the ontology, some new classes must be automatically generated once the relation in the ontology has been defined. The *Cats* relation and its attributes are now subclasses of *Fuzzy_Tables Class* and *Fuzzy_Attributes Class*, respectively, and these are represented in Figure 4 with a gray background.

The subclass *Cats* must define four new slots: *SCat_Name, SCat_Age, SCat_Weight* and *SCat_Behavior* as instances of the *Cat_Name, Cat_Age, Cat_Weight* and *Cat_Behavior* subclasses, respectively. This definition will enable data to be stored by instantiating the *Cats* Class. The relation shown in Figure 4 reflects the connection between the *Cats* relation tuples and the

Table 1: The Metadata Representation Ontology

Class	Description	Slots
Table_Definition	defines all the structures that enable fuzzy and non-fuzzy information to be stores	*Table_name*: relation name and *LColDef*: set of instances of *Attribute_Definition Class*
Attribute_Definition	defines the structure attributes	*Name*: attribute name, *Domain*: instance of *Definition_Domain Class*
Definition_Domain	defines the data type domain. All the existing data types in the fuzzy model are defined in this class	No slots
Fuzzy_Domain	groups all the fuzzy data types defined in the ontology	No slots
Classic_Domain	represents classical data types	*Base_Type*: refers to an instance of *Base_Type Class*
FD_With_Base Type	defines ordered referential data types	*Base_Type*: refers to an instance of *Base_Type Class*, *Margin*: margin of a triangular value (Approx) and *Much*: establishes when two fuzzy numbers are different
F_Type1	represents the fuzzy data types 1	Slots of this class are inherited
F_Type2	represents the fuzzy data types 2	Slots of this class are inherited
F_Type3	represents the fuzzy data type 3	*LEN*: establishes the number of discrete values defined in the data type
Base_Type	represents generic data types. This definition avoids the use of specific data types names	*BT_value*: stores *Any* value
String	represents generic strings	*BT_Value*: changes its inherited type into a String type
Number	represents different types of numbers	*BT_Value*: is inherited
Float	represents a float value	*BT_Value*: changes into a Float type
Integer	represents an integer value	*BT_Value*: changes into an Integer type

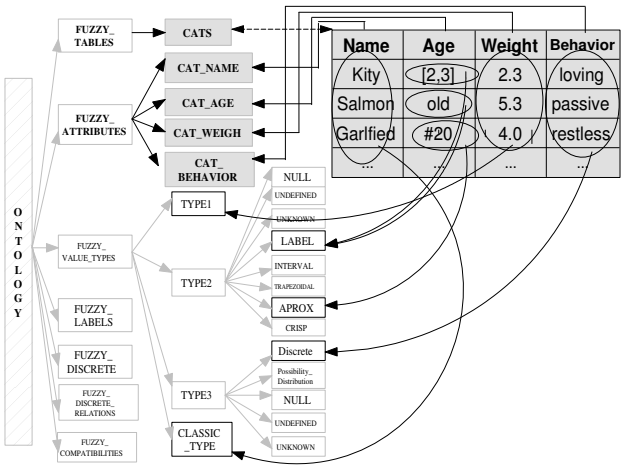

Figure 4: Relation between the Ontology and the Cats relation

ontology classes which allow these values to be stored. In Table 4, all the necessary instances for defining the first tuple of Figure 4 are shown.

A single name is associated to each ontology instance. This name is not always relevant in the ontology, as we can see in Table 4, but allows us to know the order in which the instances were created according to their sequence number. The information is stored in the instances of the *Base_Type* Subclasses: Float, String, etc. Labels and Discrete values must also be previously defined in the ontology.

5 Conclusions

The ontology in this paper attempts to represent fuzzy data regardless of implementations in concrete database models. Moreover, the structure of this ontology allows scalability so new data types as well as fuzzy ones can be represented. A first approach to a fuzzy information representation ontology has been proposed. This ontology is important as an intuitive tool which allows non-expert users to define specific information (fuzzy information) without the help of an expert knowing about catalog structures in the database. Ob-

Table 2: The Fuzzy Information Representation Ontology

Class	**Description**	**Slots**
Fuzzy _Tables	represents the structures previously defined as instances in the metadata ontology	*Name*: relation name
Fuzzy_Attributes	represents all the knowledge system attributes (fuzzy or not) and everything defined in the metadata ontology as instances of *Attribute_Definition Class*	*Contains*: instance of *Fuzzy_Value_Types Class*
Fuzzy_Value _Types	represents the values that an attribute can store	No slots
Type1	represents the fuzzy data type 1 values	*FT1_Base_Type*: instance of *Base_Type Class*
Type2	represents the fuzzy data type 2 values	No slots
Null, *Unknown*, *Undefined*	represents the values Null, Undefined and Unknown, respectively	No slots
Crisp	represents a crisp value	D: instance of the *Base_Type Class*
Approx	represents an approximate value	x: instance of the *Base_Type Class*
Interval	represents an interval value	m and n: instances of the *Base_Type Class.*
Trapezoid	represents a trapezoidal value	*alfa*, *beta*, *delta* and *gamma*: instances of the *Base_Type Class*
Label	represents a previously defined label in the *Fuzzy_Labels Class*	*Label_Name*: instance of the *Fuzzy_Labels Class*
Discrete	represents a previously defined discrete value in the *Fuzzy_Discrete Class*	*DiscreteName*: instance of the *Fuzzy_Discrete Class*
Possibility _Distribution	represents a possibility distribution with multiple discrete values	*Par*: set of instances of the *Discrete Class*
Classic Types	represents all the classic values	*Classic_Base_Type*: is an instance of *Base_Type Class*
Fuzzy_Labels	maintains a register of each defined label	*Attribute*: refers to the *Fuzzy_Attributes Class*, *Name*: label name and *Type*: instance of *Type1* or *Type2 Classes*
Fuzzy_Discrete	maintains a register of each defined discrete value	*Attribute*: refers to the *Fuzzy_Attributes Class* and *Discrete*: discrete value name
Fuzzy_Discrete _Relations	establishes a similarity relationship between two discrete values	*Discrete1* and *Discrete2*: instances of the *Fuzzy_Discrete Class* and *Similitude*: similarity degree between two discrete values

Table 3: Metadata Representation Ontology Example

Class	Instance Name	Slots
Table _Definition	Table_Cats	*Table_name*: Cats. *LColDef*: Attr_Cat_Name Instance, Attr_Cat_Age Instance, Attr_Cat_Weigh Instance, Attr_Cat_Behavior Instance
Attribute _Definition	Attr_Cat_Name	*Name*: Cat_Name. *Domain*: Name_Domain Instance
	Attr_Cat_Age	*Name*: Cat_Age. *Domain*: CatAge_Domain Instance
	Attr_Cat_Weigh	*Name*: Cat_Weigh. *Domain*: CatWeigh_Domain Instance
	Attr_Cat_Behavior	*Name*: Cat _Behaviour. *Domain*: CatBehaviour_Domain Instance
Classic _Type	Name _domain	Base_Type: Reference to Class String
F_Type2	CatAge_Domain,	*Base_Type*: Reference to Class Float and *Margin*: 1 and *Much*: 1
F_Type1	CatWeigh_Domain	*Base_Type*: Reference to Class Float, *Margin*: 1 and *Much* 1
F_Type3	CatBehaviour _Domain	*LEN*: 1

Table 4: Fuzzy Information Representation Ontology Example

Class	Instance Name	Slots
Cats	Instance15	*Name*: Cats. *SCat_Name*: Instance24, *SCat_Age*: Instance27, *SCat_Weigh*: Instance31, *SCat_Behaviour*: Instance34
Cat_Name	Instance24	*Content*: Instance25
Cat_Age	Instance27	*Content*: Instance28
Cat_Weigh	Instance31	*Content*: Instance32
Cat _Behaviour	Instance34	*Content*: Instance35
Classic _Type	Instance25	Base_Type: Instance26
String	Instance26	BT_Value: Kity
Interval _Type	Instance28	N: Instance29 and M: Instance30
Float	Instance29	BT_Value: 2.0
Float	Instance30	BT_Value: 3.0
Type1	Instance32	FT1Base_Type: Instance33
Float	Instance33	BT_Value: 2.3
Discret	Instance35	Possibility: 0.9 and Discret_Name: Loving

viously every implementation in a concrete DBMS requires an interface able to translate the ontology elements into database language. Furthermore, this ontology acts as an interface between the users and the database system. It simplifies the way in which users creates their own fuzzy elements even if they know nothing about the database underlying the ontology.

In our future work, logical and data mining aspects will be modelled, and the integration of all data types into an unified ontology will be studied. We think that the use of knowledge representation methods are useful for representing complex structures which database model environments complicate.

Acknowledgments

This research has been partially supported by the Spanish "Ministerio de Ciencia y Tecnología" (MCYT) with projects TIC2002-04021-C02-02 and TIC2002-00480.

References

[1] I. Blanco, M. J. Martin-Bautista, O. Pons, and M. A. Vila. A mechanism for deduction in a fuzzy relational database. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11:47–66, September 2003.

[2] I. Blanco, C. Martinez-Cruz, J.M. Serrano, and M. A. Vila. A first approach to multipurpose relational database server. *Mathware, In Press*, 2005.

[3] R. A. Carrasco, M. A. Vila, and J. Galindo. Fsql: a flexible query language for data mining. *Enterprise information systems IV*, pages 68–74, 2003.

[4] R.A. Carrasco, J. Galindo, M.C. Aranda, J.M. Medina, and M.A. Vila. Classification in databases using a fuzzy query language. In *9th International Conference on Management of Data, COMAD'98*, 1998.

[5] Holger Knublauch. An ai tool for the real world. knowledge modeling with protègè. Technical report, http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html.

[6] J. M. Medina, O. Pons, and M. A. Vila. Gefred. a generalized model of fuzzy relational databases. *Information Sciences*, 76(1-2):87–109, 1994.

[7] 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*, 75:273–289, 1995.

[8] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, http://protege.stanford.edu/publications/ontology_development/ontology101.html.

[9] Natalya Fridman Noy and Carole D. Hafner. The state of the art in ontology design. a survey and comparative review. *The American Association for Artificial Intelligence*, pages 53–74, 1997.