International Journal of Computational Intelligence Systems, Vol. 5, No. 6 (November, 2012), 1089-1108

An Ontology as a Tool for Representing Fuzzy Data in Relational Databases

Carmen Martinez-Cruz [1] *, Ignacio J. Blanco and M. Amparo Vila [2]

[1] *Computing Department, University of Jaen,*
Campus de las Lagunillas S/N,
Jaen, 23071, Spain
E-mail: cmcruz@ujaen.es

[2] *Computation Science and A.I. Department, University of Granada*
Periodista Daniel Saucedo Aranda S/N,
Granada, 18071, Spain
E-mail: iblanco,vila@decsai.ugr.es

Received 16 July 2012

Accepted 24 July 2012

Abstract

Several applications to represent classical or fuzzy data in databases have been developed in the last two decades. However, these representations present some limitations specially related with the system portability and complexity. Ontologies provides a mechanism to represent data in an implementation-independent and web-accessible way. To get advantage of this, in this paper, an ontology, that represents fuzzy relational database model, has been redefined to communicate users or applications with fuzzy data stored in fuzzy databases. The communication channel established between the ontology and any Relational Database Management System (RDBMS) is analysed in depth throughout the text to justify some of the advantages of the system: expressiveness, portability and platform heterogeneity. Moreover, some tools have been developed to define and manage fuzzy and classical data in relational databases using this ontology. Even an application that performs fuzzy queries using the same technology is included in this proposal together with some examples using real databases.

Keywords: Ontologies, Fuzzy Relational Databases, Schemas, Protégé, Knowledge Representation, Fuzzy Data.

1. Introduction

Several fuzzy database models are defined in the literature [11,23] to represent fuzzy data in relational databases (DB) but none of them have been standardized and commonly accepted. Some fuzzy data representations, such as the proposal of Medina et al. [29], have tried to include most of the characteristics considered in other proposals, i.e, similarity relations[5] or possibility distributions [33,35] to develop a complete fuzzy data representation model. But, in general, fuzzy data representations present some problems: i) these representations are platform dependent, e.g., the proposal of Medina et al. [29]

*Corresponding author.

works with Oracle© platform or Kacprzyk and Zadrozny proposal [16] works with MS. Access ©, ii) some proposals are incomplete, that is, most of them present partial views of uncertainty in DB representations, e.g. proposals that represent fuzziness in tuples and attributes [35], in relationships [43] or values exclusively [5], iii) proposals that only make fuzzy queries, e.g. Bosch et al. [4] or iv) proposals that are not compatible with current Semantic Web technologies because they do not use ontology web languages to be represented, only Martinez-Cruz et al. proposal [25] establishes a first approach that develops such a characteristic.

Thus, a solution to these problems was presented in Martinez-Cruz et al. proposal[25] where an ontology describes a fuzzy database representation model. This ontology acts like an external layer where fuzzy or non-fuzzy data can be represented regardless of any database model restriction, in a flexible way. In this proposal, a prototype was developed to define fuzzy database schemas easily. But, the ontology does not represent fuzzy domains and constraints, which are an important part of the fuzzy relational model. Also, the development lacks of the management of fuzzy data or a further study about tools that are available to be used instead.

In this paper the authors solve the previous work shortcomings and include new research about the establishment of communication between the ontology and the fuzzy relational database management system (RDBMS) as an extension of the paper presented in [24]. Three different database implementations have been identified to establish such communication: RDBMSs that include fuzzy functionality internally, RDBMSs which include procedural features and RDBMSs which cannot execute any SQL procedures. Furthermore, several developments to manage fuzzy data through the ontology have been developed and compared with alternative mechanisms to perform similar operations. This comparison justifies the advantages of using ontologies instead of accessing fuzzy databases directly, such as expressiveness, portability and simplicity.

Moreover, the parallelism between the definition of tuples in the ontology and the definition of a database query has lead to the definition simple fuzzy queries in the ontology. Consequently, an analysis of how a fuzzy query is defined and performed in a fuzzy database is described in this proposal as well. Also, similarly to previous developments, a new Protégé plug-in that helps the user to generate fuzzy queries as ontologies and execute them on fuzzy databases has been developed and included here. In this comparison, the advantages of defining fuzzy queries using the ontology are highlighted because of the decrease of complexity in the definition process and the increase of system independence. Also, a web-understandable database representation is available once the ontology has been defined in OWL (ontology web language) [1]. Thus, fuzzy database schemas and fuzzy queries can be indexed within the Semantic Web when necessary.

Finally, this proposal is tested on real fuzzy databases and consequently, some applications have been developed to make fuzzy schemas and data definition process easier for the user. They are developed using the ontology management tool, *Protégé* and allow transactions to be executed with different and heterogeneous databases platforms at the same time.

A brief review of fuzzy databases and ontologies is presented in section 2, which is followed by a description of the fundamentals of the proposed ontology. The extension and final description of the ontology that defines the fuzzy relational database model is presented in section 3. The architecture of the system and the data flow between the ontology and heterogeneous DB technologies are presented in section 4. Also, a description of the developed tools and their functionality are shown in section 5. A performance analysis of these tools and a comparison with others are presented in this section as well. Finally, conclusions are discussed in section 6.

2. Fundamentals

The proposal presented here is based on two different representation models: fuzzy relational databases and ontologies. Both models have been merged in an ontology that represents the fuzzy relational database model. A brief review of these models and a first approximation to this ontology are described in this section.

2.1. Fuzzy Databases

Several relational database model extensions to define fuzzy data have been proposed during the last four decades. Since the first relational database proposal was defined by Codd [8] and the fuzzy set theory was defined by Zadeh's [45], a frame that extends the relational database model to manage fuzzy data was opened. Compilations like Ma's [23] and Galindo's [11] present a complete overview of these extensions:

- Some proposals represent the uncertainty in table tuples and attributes[35].
- Some proposals represent fuzziness in the relationship among data where non-fuzzy values are presented but operators are fuzzy [5].
- Other proposals allow the questioning of relational databases using fuzzy queries [4].
- Some proposals allow uncertainty to be defined in data. Fuzzy data can be modelled with, e.g., possibility distributions[33], similarity relations [39], etc.

Finally, some fuzzy relational data models have tried to include most characteristics previously described, such as Rudensteiner et al. [36] or Medina et al.[29] proposals. The latter, which is called GEFRED, represents fuzzy data using possibility distributions and similarity relations. This model extends the relational database model with fuzzy data using the elements defined in table 1. An architecture for this theoretical model has been defined in FIRST[29], where a fuzzy database catalog is designed. Furthermore, an extension of SQL called Fuzzy SQL (FSQL) manages fuzzy data [3] through the extension of Data Definition Language (FDDL) and Data Management Language (FDML).

2.2. Ontologies and Databases

Formal ontology definitions, classifications, methodologies, tools, languages and operations are described extensively in literature (see compilations [13,38,41]). In general, an ontology is a knowledge representation model whose popularity has increased due to its capability of representing semantics in the Web. Nowadays, they have nothing in common with their initial definition that represents only a formal and agreed knowledge of a specific domain in a collaborative way. Some of the main characteristics of ontologies are [28]:

- Ontologies represent semantics of a domain in a formal way.
- The Web consortium *W3C* has accepted OWL and RDF as standard languages to represent ontologies.
- Data are not required to be defined in ontologies but schemas are.
- Theoretically, ontologies model a general knowledge of a domain, but they are not working in practice. There are several approaches of generic ontologies that model the whole reality like Cyc [21] or KR ontology [40]. However, current ontologies represent the knowledge required to solve specific problems as databases do.
- Ontologies require reasoning tools to ensure their data integrity.
- The number of ontologies is increasing exponentially due to their use in the Semantic Web and their popularity.

Thus, relational database representation model has certain similarities with ontologies since ontology classes, attributes and axioms can be interpreted as tables, attributes or constraints respectively. However, these correspondences are not so trivial. Some authors consider database schemas as *light weight ontologies* because they lack a semantic description given

Table 1. Fuzzy Basic data types defined in [29]

Fuzzy DT	Name and Example	Description
FType1	Any numeric value	It is only used in fuzzy queries.
FType2	Crisp (5)	A numeric value, e.g. 5.
FType2	Approx (5)	Approximate to 5. It is represented by a triangle membership function where 5 has the highest membership value.
FType2	Interval[4,6]	It is represented by an interval membership function where the gap between 4 and 6 has the highest membership value.
FType2	Trapezoid[4,5,6,7]	This trapezoidal membership function returns the higher value for input values between 5 and 6.
FType2/-FType3	Unknown	Attribute values that are not known.
FType2/-FType3	Undefined	Attribute values that are not applicable.
FType2/-FType3	Null	Attribute values that have not value.
FType2	Label (Tall)	This linguistic label is associated with any of the previous structures, e.g., *Tall* could be associated with Approx(1.75) value.
FType3	Discrete Value (Blond Hair)	A label related to other labels by a similarity relationship, e.g., *Blond Hair* is similar to *Red Head Hair* with a degree of 0.7.
FType3	Discrete Distribution	It represents a set of Discrete Values and their certainty degree. E.g. *Blond Hair, 0.5* and *Brown Hair, 0.7*

by the logical rules represented in ontologies [13]. Moreover, databases present further differences from ontologies:

- Data management is more efficient than ontologies.
- Data consistency is ensured due to the normalization process.
- Any kind of data can be represented in an already developed DBMS: objects, temporal information, multimedia objects, logical rules, spatial data, etc.
- There is a standardized language, SQL.

Currently, both technologies coexist together to ensure the best efficiency. Several approaches have been defined to communicate ontologies with databases. These approaches can be summarized below:

- *Generate databases from ontologies.* This approach starts from an ontology with a large amount of information that requires a database definition to ensure system efficiency. There are two alternatives to develop this task but the most usual one uses *Ontologies Based Databases (OBDB)* [14] to store an ontology using a common and unique data model. Jena [34], Sesame[18] and other developments [44] have implemented it. The second alternative defines databases using information extracted from an ontology; classes, relations and constraints. An example of this kind of application is OntoDB.
- *Generate ontologies from databases.* It is the most common approach due to the large amount of existing databases. There are several proposals which use conceptual [14,22], logical schemas [15] or data (tuples)[42] and even database queries[19] to generate the ontology. These proposals generate the domain ontology from the data source automatically. Other proposals implement the relational model as an ontology and database schemas as ontology instances, some examples of this imple-

mentation are in Champing et al.[7], Laborda et al.[31], Calero et al.[6] and Martinez-Cruz et al.[25] proposals.

- *Map databases and ontologies.* This approach is used when a database and the ontology is already developed. Then, only the establishment of the mappings are required. Datagenie application [12] developed by Gennari et al., R2O and Web-PDDL languages developed by Barrasa et al.[2] and Dou and Le Pendu[9] respectively are some examples of this approach.

2.3. *Fuzzy Schema and Data Ontology*

An ontology to represent a fuzzy relational database was firstly presented by Martinez-Cruz et al.[25]. The defined ontology is called *Schema Ontology* or *Fuzzy Catalog Ontology* from now on because it includes all the components of the SQL standard (only relational model) [10] extended with some of the GEFRED[29] structures to represent fuzzy relational database schemas (see section 2.1). The main classes to represent fuzziness in database schemas are: *Fuzzy Table*, *Fuzzy Column*, *Fuzzy Data Type*. They are divided in three: *FDataType1, FDataType2* and *FDataType3*, according to those described in table 1, *Fuzzy Structures or Fuzzy Values.* They are described in table 1, *Fuzzy Labels*, *Fuzzy Discretes.*

In this proposal, database definition process has two stages, the first one defines a fuzzy schema by instantiating the *Fuzzy Catalog Ontology* and the second one generates the database schema as a common *Domain Ontology* (also called *Data Ontology*[25]). The purpose of this *Domain Ontology* definition is the storage of database tuples as ontology instances because it represents a fuzzy schema as a set of classes, attributes and constraints. This ontology is automatically generated and the generation process is addressed in [25].

Finally, this proposal is a first prototype of an ontology that is finished with the inclusion of fuzzy domains and fuzzy constraints. In the next section this ontology is described using a real database example.

3. A Complete Description of the Fuzzy Relational Database Ontology

The *Fuzzy Catalog Ontology* [25], that is summarized above, defines most of the complete SQL description and fuzzy structures defined in the GEFRED model and FIRST architecture. However, two new classes are included in the ontology to complete the definition of fuzzy data in it. These classes, previously introduced in [24], are defined in detail below:

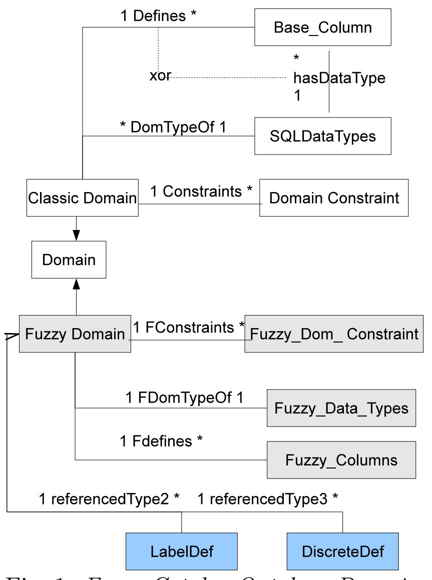


Fig. 1. *Fuzzy Catalog Ontology Domain.* structures.

- *Fuzzy Domain*: it defines any fuzzy column domain. In contrast to ordinary database columns, a fuzzy column requires a domain to be defined instead of a data type. Fuzzy domains are defined not only by data types but also by labels or discrete values and any attribute constraint, as well. Moreover, a fuzzy domain can define several attributes. The relationship between the fuzzy domain class and other classes of the ontology are shown in figure 1 (see [25] for further details of the ontology).
- *Fuzzy Constraints*: represent those constraints defined in fuzzy domains. For example, *Label_Const* constraint means that no la-

bel are allowed in a domain. A description of fuzzy constraints defined in the ontology is shown in figure 2 where SQL and fuzzy classes are in white and grey background respectively.

The *Fuzzy Catalog Ontology* described in OWL is available for downloading in the URL: `http://wwwdi.ujaen.es/cmcruz/research/ontologies/fdtscho.owl`

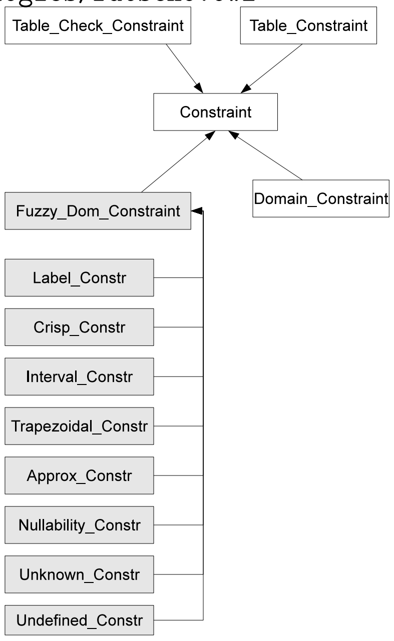


Fig. 2. *Fuzzy Catalog Ontology Constraints.* structures.

The definition process of a database schema that contains fuzzy data consists of instantiating the proper classes and attributes included in the ontology. After this definition, two different operations can be performed: i) the translation of this schema to a RDBMS, ii) the definition of database data (tuples) in the ontology. The last one requires to generate the equivalent *Domain Ontology*. A new guideline to develop this *Domain Ontology* [25] is included to manage the new classes: Each *fuzzy column* is defined as a functional object property. The range of this property depends on the fuzzy domain:

- If domain is defined only by fuzzy data types, the range of the object property goes to the fuzzy structure according to the fuzzy data type. For example, if the attribute *Tavg: Average of temperature* is a *FType2*, the range goes to the super-class *FType2_Structure* that represents: *Label, Trapezoidal, Interval, Approximate, Crisp, Null, Unknown and Undefined* classes.
- If domain includes fuzzy constraints, the range of the object property goes to the specific fuzzy structures that are not constrained. For example, if the fuzzy domain established for the *Tavg: Average of temperature* attribute is a *FType2* but *Unknown* or *Undefined* values are not allowed, the range of this property goes to: *Label, Trapezoidal, Interval, Approximate, Crisp and Null* structures, instead of going to the super-class *FType2_Struct*.

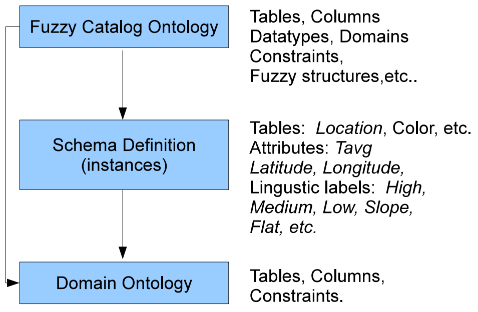


Fig. 3. *Ontologies involved in the definition process.*

The resulting ontology structure is a mixed ontology (see figure 3) containing the *Fuzzy Catalog Ontology*, instances defining a database schema and the *Domain Ontology*. The last ontology is only required if data are defined in the ontology.

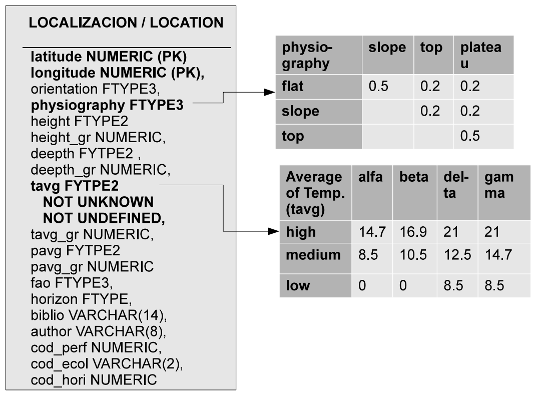
An example of a *land characteristics* database is used throughout this proposal and the complete database description is available in [27]. In this example, instantiation of several *Fuzzy Catalog Ontology* classes to define this database schema is required. A small selection of these instances is shown in table 2 and figure 4. They represent fuzzy and classical attributes: *latitude (Lat), average of temperature (Tavg) and physiography*, fuzzy domains: *Dom_physiog* and *Dom_Tavg*, fuzzy constraints: *FC1_Tavg, FC2_Tavg*, labels and discrete labels (*Low, Flat,* *Slope, etc.*), trapezoidal representations and primary keys. Remaining instances follow the same pattern, no matter whether fuzzy data or not.

ID	**Instance of**	**Value or Range**
Location	Table	Ref: Lat, physiography, Tavg, ...
Lat	Base_Column	Ref: DT_lat
DT_lat	Numeric	Values:2,8
PK_Loc	Primary_Key	Ref: Lat, Long
physiography	Fuzzy_Column	Ref: Dom_physiog
Tavg	Fuzzy_Column	Ref:Dom_Tavg
Dom_physiog	Fuzzy_Domain	Ref: TD_physiog, ..
Dom_Tavg	Fuzzy_Domain	Ref: TD_Tavg, FC1_Tavg, FC2_Tavg,..
TD_physiog	FType2_Struct	Value: 1
TD_Tavg	FType3_Struct	Values: 3,4 Ref: Float
Flat	Discrete_Definition	-
Slope	Discrete_Definition	-
Flat-Slope	Discrete_Relation	Value: 0.5 Ref: Flat and Slope
Low	Label_Definition	Ref: Low_Tavg_TD
Low_Tavg_TD	Trapezoid_Value	Values: [0,0,6.5,8.5]
FC1_Tavg	Nullability_Constraint	Value: true
FC2_Tavg	Unknown_Constraint	Value: true
...	...	...

Table 2. *Land* ontology schema instances.

Due to space limitations, the complete example is available in OWL in the URL:
`http://wwwdi.ujaen.es/cmcruz/research/ontologies/land.owl`

The previous definition of the *land characteristics schema* is translated into a *land domain ontology* following the rules described in [25,24] and above. A partial view of the resulting domain ontology is displayed in table 3 and figure 5. In this example, the range of *physiography* property is a *Ftype3_Structure* class and the range of *Tavg* property is any *FType2_Structure* sub-class but not *Unknown, Undefined* sub-classes. A complete set of instances of this schema is available in OWL from the following URL :
`http://wwwdi.ujaen.es/cmcruz/research/ontologies/landonto.owl`



```
LOCALIZACION / LOCATION

latitude NUMERIC (PK)
longitude NUMERIC (PK),
orientation FTYPE3,
physiography FTYPE3
height FTYPE2
height_gr NUMERIC,
deepth FYTPE2 ,
deepth_gr NUMERIC,
tavg FYTPE2
  NOT UNKNOWN
  NOT UNDEFINED,
tavg_gr NUMERIC,
pavg FYTPE2
pavg_gr NUMERIC
fao FTYPE3,
horizon FTYPE,
biblio VARCHAR(14),
author VARCHAR(8),
cod_perf NUMERIC,
cod_ecol VARCHAR(2),
cod_hori NUMERIC
```

physio-graphy	slope	top	plateau
flat	0.5	0.2	0.2
slope		0.2	0.2
top			0.5

Average of Temp. (tavg)	alfa	beta	delta	gamma
high	14.7	16.9	21	21
medium	8.5	10.5	12.5	14.7
low	0	0	8.5	8.5

Fig. 4. Partial example of *land characteristics* database schema.

Finally, the domain ontology provides a new tool to define simple fuzzy queries. This utility consists of defining the query condition clauses in the *domain ontology* property values. Thus, a query can be represented as an ontology, translated into SQL or Fuzzy SQL and later, query results can be returned in the same ontology structure. This representation tries to take advantage of the ontology language flexibility where data property values are not limited to specific data types and ontology comments allow storage of any value. Therefore, this proposal does not add new elements to the proposed ontology structure, it tries to minimize the sytem complexity.

Thus, a query is represented as set of instances of each table involved in the query in the *domain ontology*. Each instance represents one query condition or the visibility statement of the query attributes. Then, if the attribute is addressed to a data type property in the ontology, the condition clause is set in a string whose value involves writing the condition completely, including the operator and the value or attribute

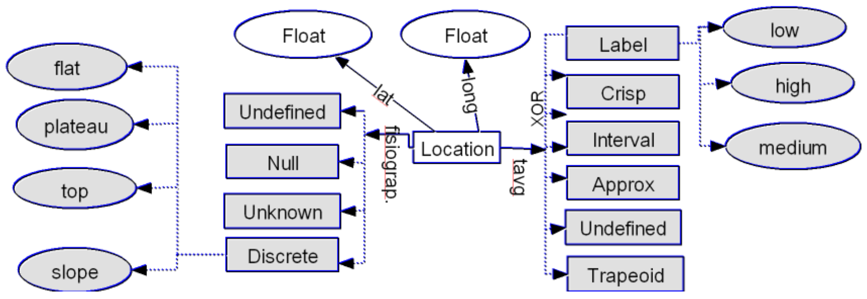


Fig. 5. *Partial example of* land characteristics *database domain ontology.*

to which it is compared. On the other hand, the attribute visibility is set in a data type property storing the ”visibility” string constant. If the attribute is addressed to an object property in the ontology, the condition clause is set in the rdf:comment of the instance that represents the value. If the condition does not involve a value but it involves other attribute, the instance to be included is the one corresponding to the nullability struct. The same operation is made to set the visibility statement in the object properties. About the logical condition joiners AND and OR, this proposal establishes the definition of the conditions in a clausal mode, that is, avoiding the use of brackets.

Table 3. Partial domain ontology of the *Land* database example.

Class	Property	Range or value
Location	Lat	http://www.w3.org /2001/XMLSchema#float
Location	Tavg	Trapezoid, Crisp, Approx, Interval, Null, Label
Location	Orientation	Ftype3_Struct
Location	Pavg	Ftype2_Struct
Location	Slope	Ftype2_Struct
Analytic	Sand	Ftype2_Struct
. . .	. . .	. . .

Following the previous example, a query is represented in the *domain ontology* that is described in figure 5 and table 3. This query gets the *latitude* and *longitude* from the *location* table where the *average of temperature (tavg)* is equal to *“medium”* in a degree of 0.8 and the *physiography* is not equal to *“bottom slope”* in a degree of 0.9. The syntax of the query in FSQL is written below:

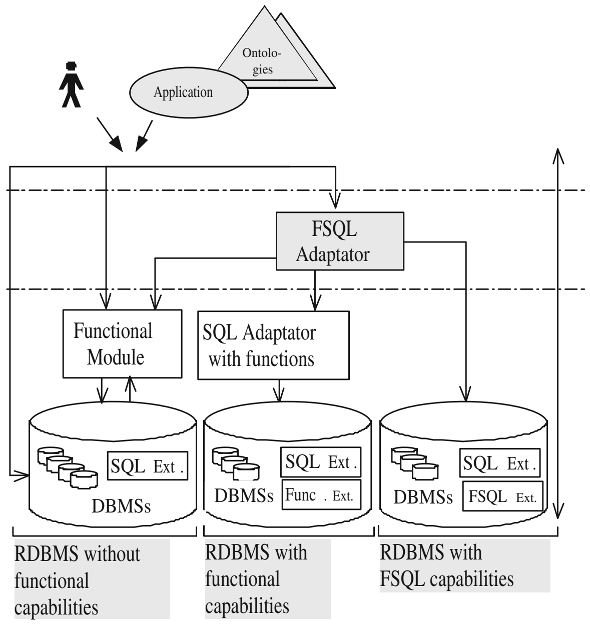
```
SELECT Lat, Lon FROM Location
WHERE
  Tavg FEQ $medium THOLD 0.8 and
  Physiography NEQ '$Bottom Slope'
  THOLD 0.9
```

Table 4. Instances of a query definition in the *domain ontology* of the *land characteristics* schema.

ID	**Instance of**	**Property**	**Value or Range**
I1	*Location*	Tavg	I2
I2	*Label*	rdf:comment:“FEQ $medium THOLD 0.8”	
I2	*Label*	LabelId	*Medium* instance
I1	*Location*	Physiography	I3
I3	*Simple*	rdf:comment: “FEQ $Bottom Slope THOLD 0.9”	
I3	*Simple*	DiscreteID	*Bottom Slope* instance
I1	*Location*	Lat	“Visible”
I1	*Location*	Long	’“Visible”

The *domain ontology* is instantiated to represent this query, and these instances are shown in table 4. The *where* clauses are included in the ontology as *rdf:comments* because the properties *tavg and physiography* are object properties. Thus, they are included in the instances of the *Label* and *Simple* classes respectively and they have associated the label definition instances: *Medium* and *Bottom Slope* which are obtained from their fuzzy domains. The visibility of this query is set as data type properties, and consequently, no new instances are required to be defined, only the establishment of the *'visible'* constant in the attribute value. Notice that it is not necessary to instantiate *location* table again, because the query does not involve any OR condition nor is one attribute involved in the visibility and where clause at the same time.

4. Architecture Description



Ontolo-gies
Application
FSQL Adaptator
Functional Module
SQL Adaptator with functions
SQL Ext.
DBMSs
SQL Ext.
Func. Ext.
DBMSs
SQL Ext.
FSQL Ext.
DBMSs
RDBMS without functional capabilities
RDBMS with functional capabilities
RDBMS with FSQL capabilities

Fig. 6. Database communication scenarios.

In this proposal, heterogeneous relational database management systems (RDBMS) are involved in data definition and management processes. These RDBMSs require different implementations to build a unified mechanism to manage fuzzy information provided by the ontology. Summarizing, there are three possible communication scenarios with RDBMS that are shown in figure 6 and described below. A deep study of these scenarios can be found in [26]:

Scenario 1. *DB Systems with FSQL capabilities.* These systems do not require translation into SQL because procedures to execute FSQL commands are included. They are the most efficient fuzzy RDBMS implementations because the entire process is done in the same platform. Not all the RDBMSs include programming capabilities.

Scenario 2. *DB Systems with programming capabilities but not FSQL.* Fuzzy queries can be performed within the system by mean of functions and procedures included in SQL sentences but translation from FSQL to SQL has to be done externally. This option improves the statement execution because it can be partially executed within the system, so the response time is reduced.

Scenario 3. *DB Systems with no programming capabilities.* When the system is only capable of representing and storing data, fuzzy data management must be done outside the database completely. An external module is in charge of coordinating the process of executing SQL statements. This is the least efficient system because the entire process is implemented outside the RDBMS and consequently the response time is the highest.

Summarizing, all these scenarios imply a SQL translation, the sentence execution and the results presentation (if applicable). But these operations are performed in different platforms according to the system capabilities and included libraries.

Then, each possible operation that can be performed in a database (schema definition and data management) is analysed in each of the previously defined scenarios.

4.1. Schema and Data Definition Process

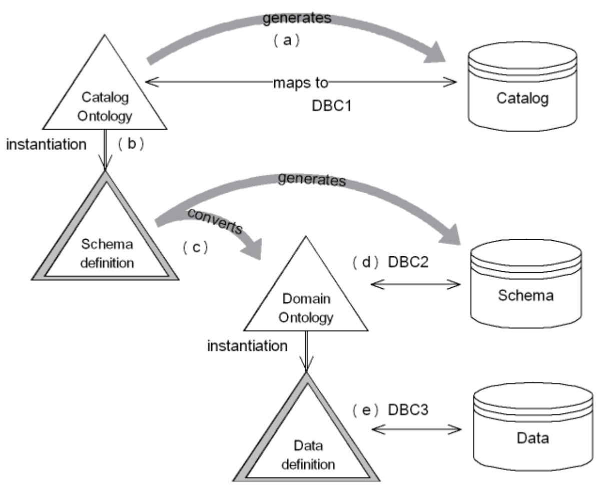


Fig. 7. Data flow description.

Data flow between users and a final storage system is a complex procedure because of heterogeneous formats involved in the schema definition process. The data flow for the definition of fuzzy schemas and data, is shown in figure 7 and is described below.

At first, before any schema definition is performed, the RDBMS catalog should be extended with new structures to store fuzzy data (see section 2.1 for further details). This process should be done once for each RDBMS implementation and such installation depends on the RDBMS system features and library capabilities. Consequently, if a database has no functional capabilities, only the fuzzy catalog structures would be installed and any management operation should be done outside the RDBMS, as described previously. The *DBC1* relation, which is described in figure 7 arrow (a), shows this catalog extension process within a RDBMS. This stage applies to scenarios (1), (2) or (3). Thus, the same catalog structures are stored in any RDBMS regardless of object identification issues and different data types.

Later, a *schema definition* can be done in the RDBMS. This schema is defined in the *Fuzzy Catalog Ontology* using an ontology manager such as the one developed in section 5 of this proposal. The three operations, that the schema definition process involves, are shown in figure 7 and described below:

1. *Instantiation of the Catalog Ontology* is represented by arrow (b) in figure 7. The schema is fully described by the instantiation of the *Fuzzy Catalog Ontology.* Thus, the definition of the schema lacks any DB implementation restriction.

2. *Domain ontology conversion* is represented by arrow (c) in figure 7. The previous *schema definition*, that is available as instances of the *Fuzzy Catalog Ontology*, can be translated into a real ontology, called *Domain Ontology* in figure 7. This ontology corresponds to the schema of a database but internal schema definitions are omitted (section 2.3). Translation rules have been implemented as a Protégé plug-in application (section 3). The resulting OWL Ontology also contains some classes and instances imported from the *Fuzzy Catalog Ontology.*

3. *RDBMS Communication Scenarios.* The database communication, which is called DBC2 and represented by arrow (d) in figure 7, works as follows depending on the scenario[†]

   - Scenario (1) sends FSQL sentences to the RDBMS. These sentences are equivalent to fuzzy DDL. An example of a *create table* sentence that includes fuzzy attributes is shown below. This example is part of the database *land characteristics* followed throughout this paper and further detailed in [27]:

```
CREATE TABLE Location (
    lat NUMERIC NOT NULL,
```

[†]A complete description of the meaning of the FSQL language can be found in [3,11]

```
long NUMERIC NOT NULL,
physiography FTYPE3(1),
Tavg FYTPE2 (4, 10) FLOAT (2)
    NOT UNKNOWN NOT UNDEFINED,
....PRIMARY KEY (lat,long)
)
```

- Scenarios (2) and (3) use only SQL and thus, catalog tables must be updated manually. For example, the translation in SQL of the previous FSQL sentence consists of the following statements (see [29], [11] for further details):

```
CREATE TABLE Location (
    lat NUMERIC NOT NULL,
    long NUMERIC NOT NULL,
    physiographyT NUMERIC,
    physiographyP1 INTEGER,
    physiography1 NUMERIC,
    Tavg1 INTEGER,
    Tavg2 NUMERIC,
    Tavg3 NUMERIC,
    Tavg4 NUMERIC, ....
    PRIMARY KEY (lat,long))
```

After that, the *data definition process* can be performed in the ontology and is shown in arrow (e) in figure 7. As in the schema definition process, an application to manage data has been implemented and described in section 5.2. Database communication in this stage has been called DBC3 in figure 7. Each sentence executed in this layer is part of the fuzzy DML (only insertions or deletions). Database communication behaviour in this process consists of:

- Scenario (1). FSQL sentences can be executed on the RDBMS directly. Then, only a right FSQL sentence formulation is required, for example, a sentence that allows insertion of a row in the previous *Location* table:

```
INSERT INTO
Location ( Lat, Lon,
   Physiography, Tavg, ... )
VALUES( 41045, 5478,
        $slope, $medium, ... )
```

  where $ indicates that it is a label defined within the corresponding domain.

- Scenario (2) y (3). SQL sentences are used to manage data on a RDBMS. Some extra queries on the database catalog are required to develop the sentences that populate the database. For example, the previous sentence is executed using SQL language and then fuzzy labels are replaced by some specific catalog references (these references has been previously obtained by other DB queries):

```
INSERT INTO
Location (Lat, Lon,
PhysiographyT, PhysiographyP1,
Physiography1, TavgT, Tavg1,
Tavg2, Tavg3, Tavg4,..
VALUES( 41045, 5478,
  4, 2 , 1, 4, 3,
  NULL, NULL, NULL, ... )
```

4.2. Query Data

This proposal allows the definition of fuzzy schemas and data, but also queries. As in the data definition process described previously, a FSQL query execution requires a lexical, syntactical and semantic analyzer, an operation identifier and also a fuzzy structure searcher. Depending on the RDBMS scenario, the system calls one or more procedures to solve it. These database scenarios are:

- Scenario (1). A FSQL sentence is internally translated and executed. A query about the *latitude and longitude of those locations with a 'medium' temperature and a not 'bottom slope' physiography* is presented following:

```
SELECT Lat, Lon FROM Location
WHERE Tavg FEQ $medium
  THRESHOLD 0.8 and
  Physiography NEQ '$Bottom Slope'
  THRESHOLD 0.9
```

- Scenario (2). FSQL sentences are partially translated into SQL outside the system. Queries can contain functions in the WHERE and SELECT clause that develop fuzzy operator behaviours. Following the previous example, the query in this scenario is:

```
SELECT Lat, lon
```

```
FROM Location WHERE
  FEQ (TavgT, Tavg1, Tavg2, Tavg3, Tavg4,
  '$Medium', 0.8) and
  NEQ (PhysiographyT, Physiography1,
  PhysiographyP1,
  '$Bottom Slope', 0.9)
```

- Scenario (3). A FSQL sentence is translated into basic SQL to get all data tuples. Later, a filtering process is executed using external functions. Following the previous example, the query only receives all the rows of the table *location* and tuple filtering is done outside the DB.

```
SELECT Lat, Lon,
       TavgT, Tavg1, Tavg2, Tavg3, Tavg4,
       PhysiographyT, Physiography1,
       PhysiographyP1, ... (*)
FROM Location
```

Summarizing, the database communication architecture is increasing in complexity according to the fuzzy database management capabilities. Moreover, fuzzy RDMBS requires the system catalog management which also varies according to their implementation. Consequently, the most important issue in this situation is making the data management and definition process transparent to the user and the most portable to different RDBMS as possible.

5. Experimentation

Several tools have been developed to manage the fuzzy data model described in previous sections. These applications try to make fuzzy data definition and management process easier for the user although not necessary. Protégé 3.4 ontology management tool[20] has been chosen because of its intuitive interface and its efficiency in managing ontologies. OWL language is used to describe our ontology system because of its recent standardization and expressiveness. On the other hand, this platform allows the inclusion of different plug-ins in JAVA language to extend its functionalities. Thus, three different Protégé plug-ins have been developed according to their purpose: schema management, data management, and query management.

Connection with heterogeneous databases is handled by wrappers that identify different relational database management systems to use the appropriate connectors. Thus, several connections, no matter the implementation, can be established at once. The scenarios analyzed in section 4 are presented in the following implementations for handling fuzziness:

- *Oracle* ©: This system includes an FSQL library which means that it supports any described operation (scenario 1).
- *PostgreSQL* ©: This system corresponds to scenario 2. Any schema or data definition can be executed on it.
- *MySQL* ©: This system corresponds to scenario 3. Any schema or data definition can be executed on it.

Moreover, the deveoped tools are used to evaluate the advantages of using ontologies to define and manage a fuzzy relational database. These applications are also compared with other tools, i.e, DDL and DML (but not queries) implementations are compared with ontology and text editors and fuzzy queries are compared with *Fuzzyqueries2* [37], which is the last running version of the GEFRED model implementation and that performs FSQL queries in Oracle © systems.

Differences in expressiveness and efficiency between implementations are analysed using two different databases. A first database that contains fuzzy data about *land characteristics* is completely described in[27] and is used throughout this paper. It has four tables: *Location, Color, Structure and Analytics*, classical and fuzzy data types. A second database that contains non-fuzzy data about medical information: surgeries and emergencies. This database is used to perform fuzzy queries. Ir order to do that, the attributes used in fuzzy queries are fuzzified.

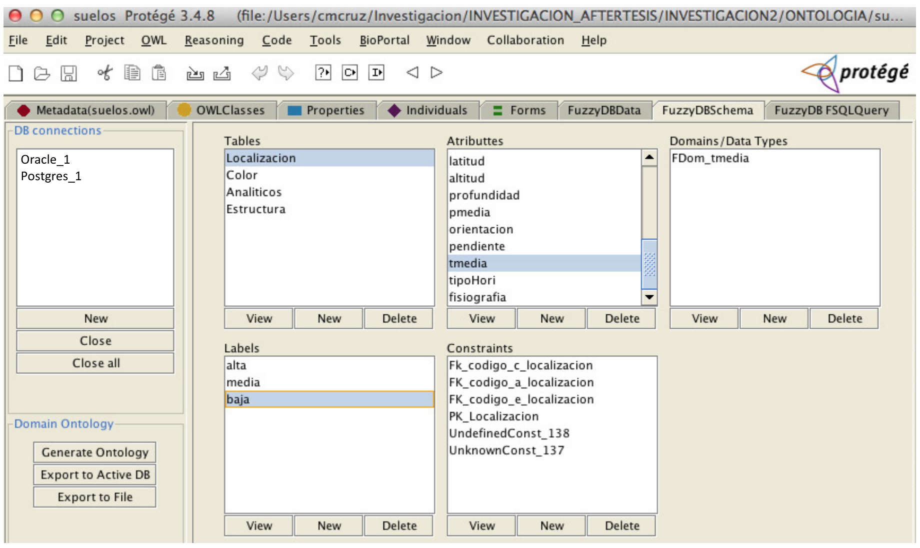


Fig. 8. Protégé plug-in to define fuzzy schemas.

5.1. *Schema Management*

Database definition process can be performed in different ways but it briefly consists of defining tables and domain elements. Usually, a fuzzy schema definition uses a common database client, such as, *sqlplus, squirell, pgAdmin*, etc. where DDL clauses must be defined manually using SQL, PLSQL[30] or PLpgSQL[32] according to the DBMS used. The proposed alternative consist of using an ontology, *Fuzzy Catalog Ontology*, that acts like an external layer of the DB to define fuzzy or non-fuzzy schema as instances. This process can be done in two different ways: The first one consists of instantiating the *Fuzzy Catalog Ontology* in any ontology management tool such as *Protégé, SWOOP* [17] or a common text editor, because a OWL ontology is in plain text. The second one consists of using a *Protégé* plug-un tool (introduced in [25]) that has been extended to make the process of defining schemas easier to the user. This tool manages *Fuzzy Catalog Ontology* structures in a logical, intuitive and tidy interface as can be seen in figure 8 [‡] This tool includes an automatic process to generate the equivalent *Domain Ontology* in OWL language which follows the rules described in Section 2.3. Finally, this application maintains the DB description in OWL language till the user exports the schema to any database. The exportation process is performed to any database connected in the system, that is, those shown on the top left side of the figure 8. In the figure, there is a connection to Oracle © and PostgreSQL © systems.

In order to make the most of this proposal, the best database to be defined in the system is the one that contains fuzzy data, the *land characteristics database*. An example of the *Location* table is shown in the screenshot of the figure 8. In this example, the fuzzy attribute *average of temperature(Tavg)* is selected and thus, its associated: fuzzy domain, *FDom_tavg*, linguistic labels, *high, low, medium* (that is, *alta, baja and media in Spanish*), and fuzzy constraints *Undefined* and *Unknown*. Also, there are other database constraints such as the primary key or some foreign keys.

Advantages and Disadvantages

Some of the advantages that this proposal raises, which are due as much to the use of ontologies as the developed plug-in are:

[‡]Notice that database information is in Spanish but the database description in this proposal is in English to improve its readability. However, examples in the screen shot are shown in the original database language.

- It is not necessary to know any FSQL, that is, fuzzy DDL sentences are generated automatically according to the DBMS. Thus, it is possible to perform any definition on a database regardless of the internal data specification because Protégé application includes different wrappers to manage the most common database systems, such as, Oracle © Postgres © and MySQL ©.
- This plug-in has all the ontology elements in one framework, where they can be created, modified and visualized in a similar way.
- It is possible to define simultaneously the same database in different RDBMS.
- Label definition process is really simple because the user has to choose only the kind of structure they want to use and fill the gaps according to the structure. Thus, if the structure chosen is a trapezoidal one, the user has four edit boxes to insert the trapezoidal edges.

Some disadvantages are:

- It is a requirement in any definition to have some basic knowledge about databases, that is, what are constraints, domains, tables and attributes.
- It is a requirement in any case to have some basic knowledge about what is a fuzzy data type and which data types are included in the database.
- The execution time is higher when ontologies are used because there is a new layer in the process, that is, not only the FSQL translation to SQL but also the ontology translation into a database language.
- There is no drawing interface to define fuzzy structures. Consequently, the user can not visualize the fuzzy membership functions defined for each linguistic labels graphically.

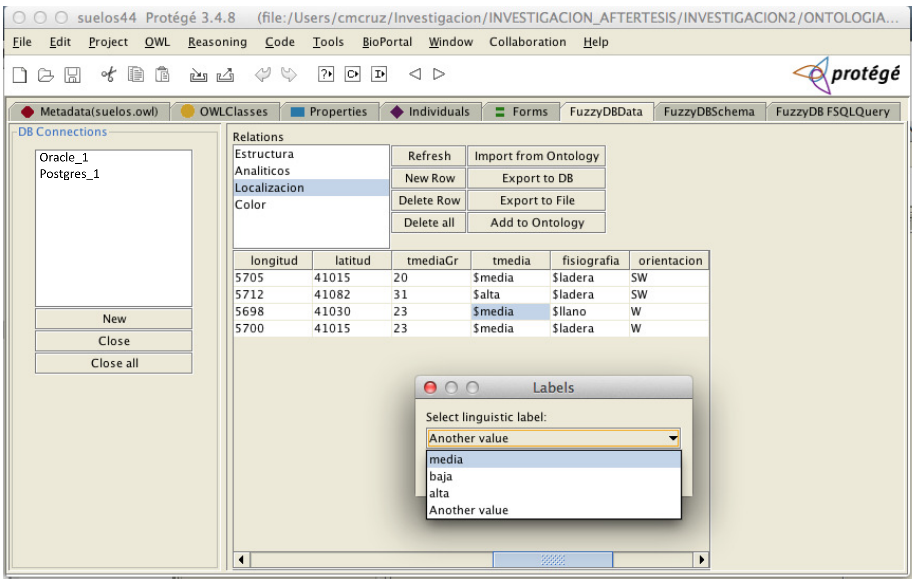
5.2. *Data Management*

This process consists of managing database tuples, e.g, insertions or deletions . Like the previous tool, this operation can be performed using a database client or using the *Domain Ontology* presented in this paper. In the first case, a *insert* statement must be defined manually but, if the DML operation involves fuzzy data, some queries must be performed in order to know the database fuzzy attributes domain (see description of a query in section 4.1). In the second case, if the *Domain Ontology* is used, tuples can be defined using the Protégé management tool or any text editor.

However, fuzzy data insertion process is slightly different from a common ontology instantiation process because some fuzzy data should be attached to this ontology from the previously defined schema instances (see section 2.3 for further details) and consequently, a frame to help the user to instantiate the domain ontology is very useful but not essential. Thus, a *Protégé* plug-in (see figure 9[§]) has been developed to make a user-interactive platform to represent fuzzy data. The main functions developed in this application are: i) the data definition using a table structure, like most usual visual databases clients, e.g. PgAdmin for Postgresql or MS Access ©. ii) loading of tuples from an already defined *Domain Ontology* instances previously saved in OWL language, iii) saving tuples within the OWL ontology iv) exportation of database tuples to a SQL/FSQL format and saving it within a file. Finally, this application maintains the DB information in OWL language till the user exports the data to any database, which are shown on the top left side of figure 9.

Tuple definition process consists of inserting fuzzy data into the cells or choosing the correct domain value from an emerging menu. The last option helps the user to choose a correct value in those fuzzy attributes where linguistic labels are part of the domain. Following the previous database example, this menu is shown at the bottom of figure 9 and the domain values of *average of temperature* attribute: *high/alta,*

[§]Notice that database information is in Spanish but the database description in this proposal is in English to improve its readability. However, examples in the screenshot are shown in the original database language.


suelos44 Protégé 3.4.8 (file:/Users/cmcruz/Investigacion/INVESTIGACION_AFTERTESIS/INVESTIGACION2/ONTOLOGIA...
File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Help
protégé
Metadata(suelos.owl) OWLClasses Properties Individuals Forms FuzzyDBData FuzzyDBSchema FuzzyDB FSQLQuery
DB Connections
Oracle_1
Postgres_1
New
Close
Close all
Relations
Estructura
Analiticos
Localizacion
Color
Refresh
Import from Ontology
New Row
Export to DB
Delete Row
Export to File
Delete all
Add to Ontology
Labels
Select linguistic label:
Another value
media
baja
alta
Another value

longitud	latitud	tmediaGr	tmedia	fisiografia	orientacion
5705	41015	20	$media	$ladera	SW
5712	41082	31	$alta	$ladera	SW
5698	41030	23	$media	$llano	W
5700	41015	23	$media	$ladera	W

Fig. 9. Protégé plug-in to define fuzzy tuples.

low/baja or medium/media.[¶] An alternative to using this plugin is to execute insert clauses within an SQL client, but the syntax is less intuitive. Another one consists of using an ontology editor which shows each ontology constituent and allows easy navigation, but, they do not interact with the user. However, the ontology can be populated without too much difficulty.

Advantages and Disadvantages

Some of this proposal advantages are:

- It is not necessary to know any SQL or FSQL, that is, any fuzzy DML sentence to define data is generated automatically.
- This plug-in allows fuzzy data to be defined in an assisted way similar to other DB management systems.
- It is possible to define simultaneously the same database into different RDBMS.
- It is possible to perform any definition on a database regardless of the internal data specification. The protégé application includes different wrappers to manage the most common database systems.
- Label definition process is really simple because the user has to choose the values they want to use or fill the cells according to the data type manually.

Disadvantages are:

- It is a requirement to have defined the *domain ontology* associated with the defined database schema. Consequently, we need to have both definitions, the schema instances and the domain ontology, at once.
- It is a requirement to have some basic knowledge about what is a fuzzy data type and which data types are included in the database.
- This plug-in is useful when users intend to insert fuzzy data manually because of its expressiveness and intuitiveness. But, the process is slower due to the fact that it must be performed manually. If the user needs to populate the database in a faster way, it is preferable to use a SQL script although its development is more complex and expressiveness poorer.

[¶]Notice that database information is in Spanish but the database description in this proposal is in English to improve its readability. However, examples in the image is shown in Spanish because they are real data.

- The execution time is higher because there is a new layer in the middle of the process. Consequently, not only the FSQL translation to SQL but also the ontology translation into a DB language is performed.

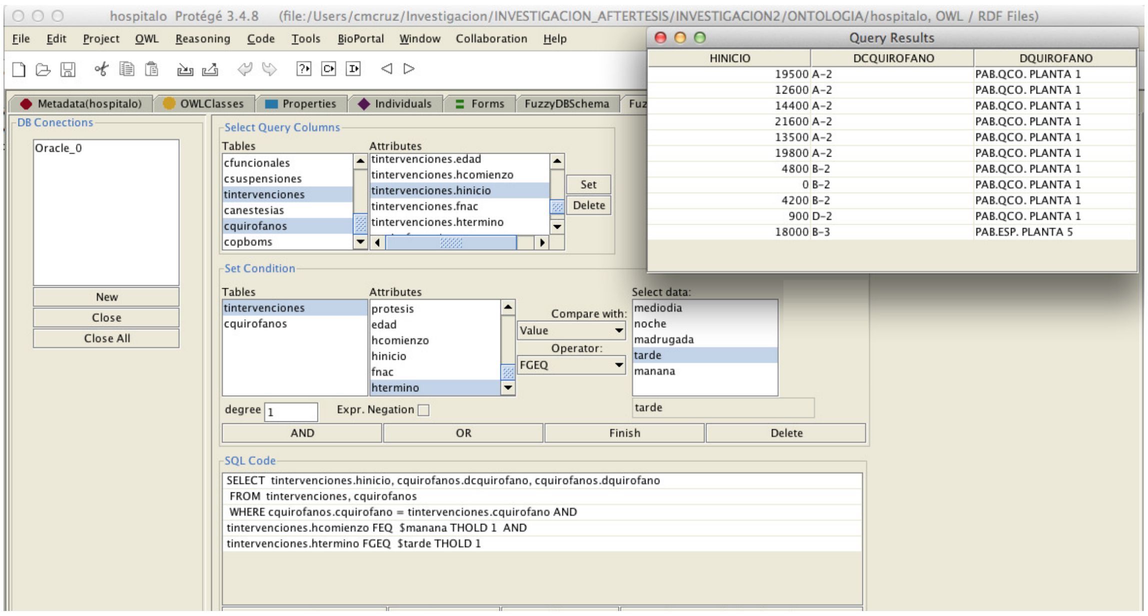
5.3. Fuzzy Query Manager

The fuzzy query builder plug-in helps the user in the definition process of a fuzzy query using an ontology. Similarly to data definition process, a query ontology, which has been described in section 3, is generated from the schema definition instances and is stored as a*domain ontology.* After the ontology is defined, different wrappers are defined to establish the database communication, according to the fuzzy management capabilities required to solve it. In this prototype, only simple SQL queries can be performed on RDBMS or FSQL queries can be made on RDBMS that have been extended with fuzzy management. This plug-in acts like an editor of fuzzy queries using the ontology of the schema that has been previously defined. In the fuzzy query builder many elements are included easily in a fuzzy query from domain values to fuzzy structures. Consequently, the definition requires a considerable decrease of user effort. However, the fuzzy query ontology can be defined using a text editor, similar to what happened with the previous tools.

A screen shot of the implementation is shown in figure 10. Query definition process in the plug-in is very intuitive and it consist of the following steps: Firstly, the user selects the table and attributes that he wants to be visualized. Secondly, the user chooses the conditions to be accomplished in the query, if any. There are two different kinds of comparisons defined in this interface: a) *Attribute-attribute* comparison compares two database attributes. b) *Attribute-value* comparison involves one attribute from the database with: on one hand, a value that exists previously in the database, i.e. a domain linguistic label defined in the ontology such as *slope* in the *physiography* attribute. On the other, a value that is included manually in the system, i.e., the *average of temperature* attribute is compared with an interval defined as [20,25]. The comparison operators are fuzzy or non-fuzzy according to the kind of data involved in the condition, p.e. FEQ means fuzzy equal (a list of fuzzy operators and their meaning can be found in [29]). Moreover, any fuzzy condition should be accomplished in a degree between [0,1] that must be specified by the user. The default value of this accomplishment degree is 1. Non-fuzzy conditions do not include this accomplishment degree. iii) At the same time, the query is being generated in SQL or FSQL (SQL extended to manage fuzzy data described in [29]) in order to help the user in the construction process and also, the final *query domain ontology* is being populated and stored in the system. This application maintains the DB information in OWL language till the user performs the query execution to any database connected in the system as previous developments do.

FuzzyQueries2 tool is specialized in executing fuzzy queries on Oracle ©but the syntax of a fuzzy select clause must be known in advance. An assistant is available to navigate along the database elements and a drawing tool helps to define trapezoidal structures. In contrast to this tool, the Protégé fuzzy query plug-in is developed to avoid the writing of the FSQL clause by a visual windowed interface as is shown in figure 10. In order to analyse this plug-in a comparison between both applications has been performed throughout this example. We are using the *hospital database* where fuzzy data are included in the query but not in the database tables. The query made to the system consist of knowing all the surgery rooms data (room name and denomination) and the time of the surgery, where these surgeries have lasted for at least two shifts, from morning to afternoon, or even longer. The syntax of this query is expressed in FSQL in order to make it shorter and has been translated into English:

```
SELECT  Start_Time, SurgeryRoom,
            SurgeryRoomName
FROM CSurgeryRoom CSR, TSurgeries TS
```



Compare with: Value
Operator: FGEQ
Select data: tarde
degree 1
Expr. Negation

HINICIO	DCQUIROFANO	DQUIROFANO
19500	A-2	PAB.QCO. PLANTA 1
12600	A-2	PAB.QCO. PLANTA 1
14400	A-2	PAB.QCO. PLANTA 1
21600	A-2	PAB.QCO. PLANTA 1
13500	A-2	PAB.QCO. PLANTA 1
19800	A-2	PAB.QCO. PLANTA 1
4800	B-2	PAB.QCO. PLANTA 1
0	B-2	PAB.QCO. PLANTA 1
4200	B-2	PAB.QCO. PLANTA 1
900	D-2	PAB.QCO. PLANTA 1
18000	B-3	PAB.ESP. PLANTA 5

SQL Code

```
SELECT  tintervenciones.hinicio, cquirofanos.dcquirofano, cquirofanos.dquirofano
 FROM  tintervenciones, cquirofanos
 WHERE cquirofanos.cquirofano = tintervenciones.cquirofano AND
tintervenciones.hcomienzo FEQ  $manana THOLD 1  AND
tintervenciones.htermino FGEQ  $tarde THOLD 1
```

Fig. 10. Protégé plug-in to perform fuzzy queries.

```
WHERE  CSR.SurgeryRoomNum =
            TS.SurgeryRoomNum
AND TS.RealStart_Time FEQ $Morning
AND TS.EndTime FGEQ $Afternoon
```

Resulting data set is the same using the Protégé plugin or the *FuzzyQueries2* tool, as can be observed in figures 10 and 11 respectively. But, using the FuzzyQueries2, the query is written by hand meanwhile, query definition made in the Protégé plug-in is more intuitive. In addition to the semantics, Protégé plug-in allows more fuzzy structures to be represented without using FSQL syntax and queries are not limited to Oracle ©. Concerning the efficiency, both applications uses the same API to perform fuzzy queries. Thus, system efficiency rates are similar. Advantages and disadvantages of using this Protégé plug-in are analysed below.

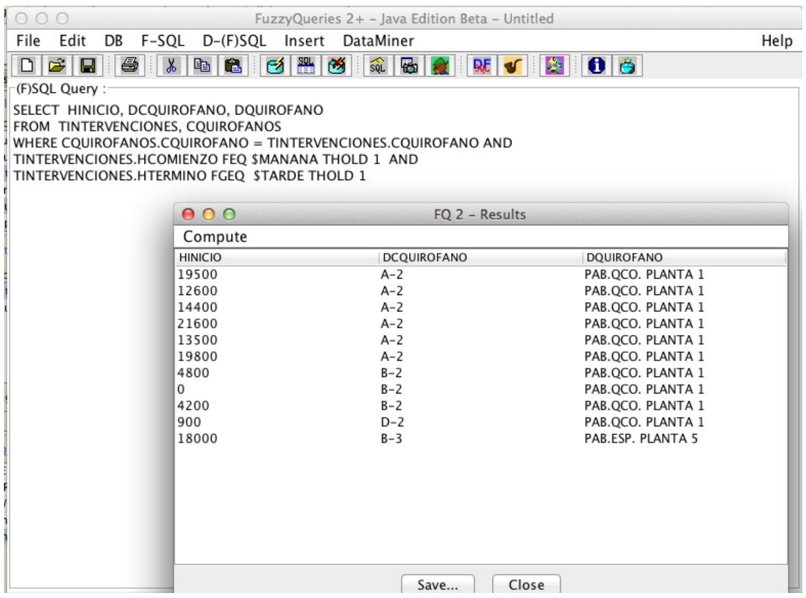
Advantages and Disadvantages

This proposal presents some advantages over those presented above:

- The query definition can be done without any knowledge about DML language.
- The query definition can be performed in OWL using a common text editor, regardless of the plugin developed.
- Queries can be executed on several databases, that share the same schema, and the results are obtained at once.
- Queries can be performed on non fuzzy databases but without the use of fuzzy data or fuzzy comparators.

Disadvantages of this proposal are:

- The interface is still fixed to the definition of fuzzy data types and consequently, some basic knowledge about the fuzzy model is necessary.
- The query can be performed simultaneously on heterogeneous databases but sharing the same schema.
- The ontology data is fully stored in memory and consequently, if query the resulting set is too large, memory problems can occur.
- Fuzzy queries can be made in non-fuzzy DB but several schema modifications should be performed to allow fuzzy managing.


FuzzyQueries 2+ – Java Edition Beta – Untitled
File Edit DB F-SQL D-(F)SQL Insert DataMiner Help
(F)SQL Query :
FQ 2 – Results
Compute
Save... Close

```
SELECT  HINICIO, DCQUIROFANO, DQUIROFANO
FROM  TINTERVENCIONES, CQUIROFANOS
WHERE CQUIROFANOS.CQUIROFANO = TINTERVENCIONES.CQUIROFANO AND
TINTERVENCIONES.HCOMIENZO FEQ $MANANA THOLD 1  AND
TINTERVENCIONES.HTERMINO FGEQ  $TARDE THOLD 1
```

```
HINICIO        DCQUIROFANO        DQUIROFANO
19500          A-2                PAB.QCO. PLANTA 1
12600          A-2                PAB.QCO. PLANTA 1
14400          A-2                PAB.QCO. PLANTA 1
21600          A-2                PAB.QCO. PLANTA 1
13500          A-2                PAB.QCO. PLANTA 1
19800          A-2                PAB.QCO. PLANTA 1
4800           B-2                PAB.QCO. PLANTA 1
0              B-2                PAB.QCO. PLANTA 1
4200           B-2                PAB.QCO. PLANTA 1
900            D-2                PAB.QCO. PLANTA 1
18000          B-3                PAB.ESP. PLANTA 5
```

Fig. 11. Fuzzyqueries2 program.

6. Conclusions

Ontologies have been used as a frame to solve the problems associated with fuzzy data representation in the database relational model. Heterogeneous platforms and fuzzy data complexity are two of these problems that have been solved with this ontology proposal and tools developed.

Specifically, in this paper, the fuzzy relational database specification as an ontology has been finished with the definition of fuzzy domains and constraints. Also, the generation process of the *Domain Ontology* which defines the schema as an ontology has been also modified to attend to these new added structures.

An architecture with different database scenarios has been analysed to define and manage fuzzy data. This analysis shows the complexity of managing fuzzy data in heterogeneous relational database management systems. The analysis has also shown how the system acts when a fuzzy operation is performed in a database, the importance of keeping this process transparently to the user and the difficulty of making the fuzzy database model portable.

Several tools that implement the system architecture have been developed as Protégé plug-ins to achieve an intuitive and accessible fuzzy data definition and management framework. These tools have been mainly designed to manage fuzzy data structures easily, in contrast to the current database systems where this process is performed by hand using SQL or FSQL languages. Moreover, the use of ontologies have allowed that a fuzzy databases can be expressed in any ontology editor as well. Also, this proposal defines fuzzy database schemas and data in OWL making the fuzzy database structure and final data readable in the semantic web. Consequently, semantic web agents, searchers or any other service can access the information represented in fuzzy databases. On the other hand, these tools establish database communication with database schemas allocated in different RDBMS implementations simultaneously.

A new analysis about how to represent a fuzzy query in the ontology is included in this paper as well. This representation is similar to the data definition process in the ontology because it uses the same *domain ontology* as the skeleton of the sentence. A new Protégé plug-in that helps the user in the query building process has been developed in Protégé. This application allows to execute fuzzy queries on databases which share the same schema at the same time, but only RDBMS with FSQL capabilities. A comparison between a tool that performs fuzzy queries in the Oracle © database management system, called *Fuzzyqueries2* and the developed Protégé plugin, has addresed the advantages of using ontologies in the process, specially, the increase of expressiveness and usability regardless of database languages.

Finally, this proposal is the first approximation to a complete fuzzy RDBMS representation as an ontology. Future extensions include fuzzy objects management, fuzzy logic information representation and data mining operations management. Fuzzy queries are being extended to be executed in heterogeneous databases and schemas as well. Thus, matching operations should be performed in order to control different granularities and attribute meanings. This task is also programmed in a near future.

Acknowledgments

The research reported in this paper has been supported by Andalusian Government (Junta de Andalucía, Consejería de Innovación, Ciencia y Empresa) under projects P07-TIC-03175 and TIC-6109, and by Spanish Government (Ministerio de Ciencia e Innovación) under project TIN2008-02066. Finally, we are specially grateful to Jose María Serrano Chica for his support with the *FuzzyQueries2* tool.

References

1. G. Antoniou and F. van Harmelen. *Handbook on Ontologies in Information Systems*, chapter Web Ontology Language: OWL, pages 67–92. Springer-Verlag, 2003.
2. J. Barrasa, O. Corcho, and A. G. . Perez. Fund finder: A case study of database to ontology mapping. In *International Semantic Web Conference, number 2870 in Lecture Notes in Computer science,*, pages 17–22. Springer-Verlag., 2003.
3. I. Blanco, N. Marín, O. Pons, and M. A. Vila. An extension of data description language (ddl) for fuzzy data handling. In *Flexible Query Answering Systems, Recent Advances*, Advances in Soft Computing, pages 54–64. Physica-Verlag, 2000.
4. P. Bosc, M. Galibourg, and G. Hamon. Fuzzy querying with sql: Extensions and implementation aspects. *Fuzzy Sets and Systems*, 28:333–349, 1988.
5. B. P. Buckles and F. E. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, (7):213–226, 1982.
6. C. Calero, F. Ruiz, A. Baroni, F. Brito e Abreu, and M. Piattini. An ontological approach to describe the sql:2003 object-relational features. *Computer Standards and Interfaces Journal*, pages 1–28, 2005.
7. P. A. Champin, G.J. Houben, and Ph. Thiran. Cross: An owl wrapper for reasoning on relational databases. In C. Parent, K.D. Schewe, Veda C. Storey, and Bernhard Thalheim, editors, *ER*, volume 4801 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 2007.
8. E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
9. D. Dou and P. LePendu. Ontology-based integration for relational databases. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 461–466, New York, NY, USA, 2006. ACM Press.
10. International Organization for Standardization (ISO). Information Technology. Database language sql. parts 1 to 4 and 9 to 14. *9075-1:2003 to 9075-14:2003 International Standards*, (Standard No. ISO/IEC 9075:2003), September,2003.
11. J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases Modeling, Design and Implementation.* Idea Group Publishing, 2006.
12. J. Gennick. *SQL Pocket Guide.* O'Reilly, 2006.
13. A. Gómez-Pérez, M. Férnandez-López, and O. Corcho-García. *Ontological Engineering.* Springer-Verlag New York, Inc., 2003.
14. S. Jean, G. Pierra, and Y. AitAmeur. Domain ontologies: A database-oriented analisys. In *Proceedings of the Web Information Systems and Technologies (WEBIST'2006)*, April 2006.
15. D. Juric and Z. Skocir. Building owl ontologies by analyzing relational database schema concepts and wordnet semantic relations. In *The 9th International Conference on Telecommunications. ConTEL 2007*, June 13-15 2007.
16. J. Kacprzyk and S. Zadrozny. Sqlf and fquery for access. *IFSA World Congress and 20th NAFIPS International Conference. Joint 9th*, 4:2464–2469, 2001.
17. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James Hendler. Swoop: A web ontology editing browser. *Journal*

of Web Semantics, 4:2005, 2005.
18. Arjohn Kampman and Jeen Broekstra. Sesame. http://www.openrdf.org/, 2007.
19. V. Kashyap. Design and creation of ontologies for environmental information retrieval, 1999.
20. H. 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.
21. D.B. Lennat. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 33(8):33–38, 1995.
22. L. Lubyte and S. Tessaris. Extracting ontologies from relational databases.krdb research centre technical report krdb07-4. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, 2007.
23. Z. Ma. *Fuzzy Database Modeling with XML*. Springer, 2005.
24. C. Martinez-Cruz, I. Blanco, and M.A. Vila. An ontology to represent queries in fuzzy relational databases. In *Proc. of the 2011 IEEE International Conference ISDA, Cordoba, Spain*, page 1317, 2011.
25. C. Martinez-Cruz, I. J. Blanco, and M. A. Vila. The use of ontologies for representing database schemas of fuzzy information. *International Journal of Intelligent Systems*, 23(4):419–445, February 2008.
26. C. Martinez-Cruz, J.M. Serrano, , I. Blanco, and M.A. Vila. A first approach to multipurpose relational database server. *Mathware and Soft Computing*, 12(2-3):129–153, 2005.
27. Carmen Martínez Cruz. *Sistema de Gestión de Bases de Datos Relacionales Difusas Multipropósito. Una Ontología para la Representación del Conocimiento Difuso.* PhD thesis, Department of Computer Science and Artificial Intelligence, University of Granada, España, 2008.
28. Carmen Martínez-Cruz, Ignacio Blanco, and M. Vila. Ontologies versus relational databases: are they so different? a comparison. *Artificial Intelligence Review*, pages 1–20. 10.1007/s10462-011-9251-9.
29. 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.
30. Oracle. Plsql for oracle 11g. http://www.oracle.com/technetwork/database/-features/plsql/index.html, January 2012.
31. C. Pérez de Laborda and S. Conrad. Relational.owl: a data and schema representation format based on owl. In *CRPIT '43: Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, pages 89–96, 2005.
32. PostgreSQL. Postgresql documentation. http://www.postgresql.org/docs/8.3/static/-plpgsql.html, January 2012.
33. H. Prade and C. Testemale. Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. *Information Sciences*, (34):113–143, 1984.
34. HP Labs Semantic Web Programme. Jena/ a semantic web framework for java. http://jena.sourceforge.net/, 2007.
35. K. V. S. V. N. Raju and A. K. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2):129–166, 1988.
36. E. A. Rundensteiner, L. W. Hawkes, and W. Bandler. On nearness measures in fuzzy relational data models. *International Journal Approximate Reasoning*, (3):267–298, 1989.
37. J.M. Serrano, M.A. Mara Amparo Vila, V. Aranda, and G. Delgado. Using fuzzy relational databases to represent agricultural and environmental information an example within the scope of olive cultivation in granada. *Mathware and Soft Computing*, 8(3):275–289, 2001.
38. R. Sharman, R. Kishore, and R. Ramesh. *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems (Integrated Series in Information Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
39. S. Shenoi and A. Melton. Proximity relations in the fuzzy relational databases. *Fuzzy sets and Systems*, 31(3):285–296, 1989.
40. John F. Sowa. Ontology, metadata, and semiotics. In *ICCS '00: Proceedings of the Linguistic on Conceptual Structures*, pages 55–81, London, UK, 2000. Springer-Verlag.
41. S. Staab and R. Studer. *Handbook on Ontologies.* Springer, 2004.
42. Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, Y. Ding, and G. Nagy. Towards ontology generation from tables. *World Wide Web*, 8(3):261–285, 2005.
43. M. Umano. *Fuzzy Information and Decision Processes*, chapter FREEDOM-0: A Fuzzy Database System, pages 339–347. North Holland Pub. Co., 1982.
44. Tijn van Hofwegen. A survey of ontology based databases. http://referaat.cs.utwente.nl/new/-paper.php?paperID=294.
45. L. A. Zadeh. Fuzzy sets. *Information and Control*, 83:338–353, 1965.