Information Sciences 366 (2016) 150–164

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Flexible queries on relational databases using fuzzy logic and ontologies

Carmen Martínez-Cruz[a,*], José M. Noguera[a], M. Amparo Vila[b]

[a] Department Informatica, University of Jaén, Campus de Las Lagunillas SN, 21073 Jaén, Spain
[b] Department Ciencias de la Computación e Inteligencia Artificial, University of Granada, Periodista Daniel Saucedo Aranda SN, 18073 Granada, Spain

ARTICLE INFO

Article history:
Received 2 March 2013
Revised 16 April 2016
Accepted 22 May 2016
Available online 27 May 2016

Keywords:
Relational databases
Ontologies
Semantic queries
Fuzzy logic
Fuzzy databases
Flexible queries

ABSTRACT

Nowadays there are many proposals that allow users to perform fuzzy queries on relational databases. Regardless of these proposals, fuzzy queries are really useful on scalar values where fuzzy sets can be adjusted to the user needs and domains, but non-scalar values are a more complex task. Here, we extend non-scalar attribute management in fuzzy queries with the use of ontologies. Thus, we allow to compute this kind of queries not only with the similarity relationships defined explicitly on a fuzzy set but with semantically interrelated terms modeled as a domain ontology as well. Moreover, we present the architecture of a novel system that combines both techniques to return an answer as much complete as possible and ordered by a degree of accomplishment. Finally, a qualitative and quantitative study about the use of flexible queries on relational databases is included in this work, as well.

1. Introduction

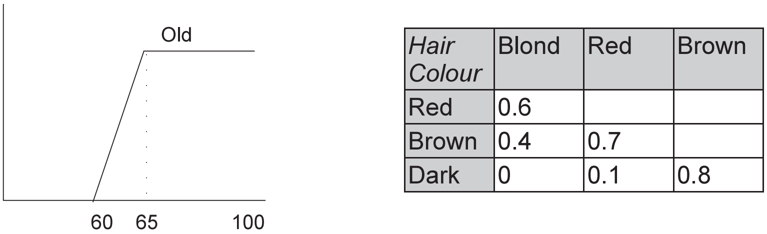
Why do we need flexible queries? In many cases users are unsure about the characteristics of the elements they are trying to retrieve from a database. They cannot provide precise terms or values to perform the query. In those cases, a system that allows users to execute a query in a flexible way can solve their requirements satisfactorily.

There are many proposals in the literature to perform flexible queries on relational databases [28]. Most of them are related with fuzzy logic implementations because they allow users to represent imprecise data instead of crisp values to make a search. In terms of scalar values this practice results very effective because any fuzzy set can be defined according to the domain of the queried attribute. This is the case, for example, of querying: *“Return people around 50 years”*. In this situation, a triangular or gaussian function could represent the *“around 50”* fuzzy set to compute the search. However, non-scalar attributes such as *“qualification”*, *“attitude”* or *“hair color”*, closer to natural language representations are treated differently in fuzzy logic. In most representations similarity relationships among all the domain elements of an attribute must be defined in the design phase. For example, in a search like this: *return blond people*, Could red hair people be included in the answer?. The answer would be affirmative if the relationship between *“blond”* and *“red haired”* were established with a high degree of similarity.

---

[*] Corresponding author. Tel.: +34 953212411.
E-mail addresses: cmcruz@ujaen.es (C. Martínez-Cruz), jnoguera@ujaen.es (J.M. Noguera), vila@decsai.ugr.es (M.A. Vila).

http://dx.doi.org/10.1016/j.ins.2016.05.022
0020-0255/



Hair Colour	Blond	Red	Brown
Red	0.6		
Brown	0.4	0.7	
Dark	0	0.1	0.8

(a) Human Age

(b) Human Hair Colour

Fig. 1. Fuzzy sets representing human age and hair color.

In the past decade, a new technology has appeared to make flexible queries in the frame of the semantics. It consists in using ontologies, a formal representation of the reality which are, theoretically, designed in a consensual and collaborative way [40].

In this paper we aim at answering flexible queries like this one: *return long films about Zombies*, where the concept *Zombie* is not included in any database; or this other: *return all tall students interested in Life Sciences*, meaning Biology, Zoology, Botany, etc. In the last one, there is not any attribute in an academic database that specifies the categorization of a subject, e.g., Life Science, Arts or Social Sciences.

Here, we propose the use of ontologies to solve the disadvantages that fuzzy logic entails on non-scalar domains, i.e. non-scalar fuzzy sets require the establishment of a similarity degree among each pair of domain terms. Consequently, the domain must be closed because of the tediousness of the definition process. Another disadvantage consists in the subjectivity involved in the evaluation process of these similarity relationships, which devolves upon the query or database designer.

In this proposal, when a flexible query is executed, the system uses fuzzy logic techniques if the attribute domain is scalar, but if the domain is not scalar, the system uses semantic or fuzzy logic techniques according to the available information or the user preferences. The principles involved in the designing of a system architecture for processing flexible queries are also described here and as a proof of concept, we also report a thoughtful example of the behavior of our system through the execution of a flexible query using two test-bed relational databases.

This paper is organized as follows: first, we present a brief overview about fuzzy databases and ontologies in Section 2. Afterwards, we present the system architecture and development details in Section 3. Two examples of working and experimentation results are included in Section 4 along with a qualitative and quantitative study of the main features of this proposal. Finally, several conclusions are discussed in Section 5.

2. Background

Two different technologies, further studied in the literature: fuzzy databases and ontologies, have been reviewed in this section to establish the basis of the proposal presented in this contribution.

2.1. Fuzzy queries on relational databases

Several relational database model extensions have been proposed to define fuzzy queries on data during the last four decades. Since the first relational database proposal was defined by Codd [9] and the fuzzy set theory was defined by Zadeh's [44] a frame that extends the relational database model to manage fuzzy data was opened [28]. In Section 4.3, a qualitative comparison among different fuzzy database implementations is presented together with the main features of these models. Some of the latest fuzzy relational data models have tried to include the most relevant characteristics to manage fuzzy data, such as, Rudensteiner et al. [37], Zemancoba and Kandel [45] or Medina et al. [32] proposals. The last one, which is called GEFRED, represents fuzzy data using possibility distributions and similarity relationships. This model extends the relational database model to allow the storage of fuzzy data and the execution of fuzzy queries with the following characteristics:

- *Fuzzy data types* to manage fuzzy data (scalar and non-scalar).
- *Fuzzy operators*, of equality and similarity. Each one has associated a membership degree. For example, *"height FEQ $tall 0.8"* means: values of *height* attribute that are equal to *"tall"* with an accomplishment degree of 0.8.
- *Null* values: *Null*, *Unknown* and *Undefined*.
- *Possibility distributions*. Fuzzy sets can be represented as: *Crisp* values which are ordinary numeric values, *Approximate* values represented by a triangular distribution, *Trapezoidal* values represented by a trapezoidal distribution, *Interval* values represented by an Interval structure. All of them can be associated with a label as shown in Fig. 1(a).
- *Labels* are associated with any domain.
- *Similarity relationships* establish a numeric relationship among different labels based on non-scalar domains, e.g. Fig. 1(b).

An architecture for this theoretical model has been developed in FIRST [32], where a fuzzy database catalogue has been extended to manage this fuzzy model. Also, an extension of SQL, called Fuzzy SQL (FSQL), has been designed to perform queries easily. We can see an example of how this system works in [31].

2.2. Ontologies and relational databases

An ontology is a knowledge representation model which has been increasing in popularity in the last years due to its capability to represent semantics in the Web [40]. Nowadays, they have changed its initial conception consisting of representing only a professional and consensual knowledge of a specific domain in a collaborative way. Some of the main characteristics of ontologies are [29]: (i) ontologies represent semantics of a domain in a formal way, (ii) the Web comsortium *W3C* has accepted OWL and RDF [3] as standard languages, (iii) data are not required to be defined in ontologies but schemas are required, (iv) theoretically, ontologies model a general knowledge of any domain. There are several approaches of generic ontologies that model the whole reality like Cyc [23] but most of them represent the knowledge required to solve specific problems, (v) ontologies require reasoning tools to ensure their data integrity and (vi) ontology number is increasing exponentially due to their use in the Semantic Web and their popularity.

Relational data representation has certain similarities with ontologies since ontology classes, attributes and axioms can be interpreted as tables, attributes or constraints respectively, or queries can be performed using SQL or SPARQL, respectively. However, these correspondences are not so trivial. Some authors [13] consider database schemas as *light weight ontologies* because they lack of a semantic description given by the logical rules represented by ontologies. Moreover, databases present another dissimilarities with ontologies: (i) their data management is more efficient than ontologies, (ii) the data consistency is ensured due to the normalization process and (iii) almost any kind of data can be represented in an already developed Database Management System (DBMS).

Currently, both technologies coexist together to achieve the best efficiency. Several approaches have been defined to communicate ontologies with databases. These approaches are:

- *Generate databases from ontologies*: *Ontologies Based Databases (OBDB)* [17] store an ontology using a common and unique data model: Jena and Sesame [19] have implemented it. Other systems, as OntoDB [16], define databases using information extracted from an ontology.
- *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 schemas [17,27], data (tuples) [1,42] or even database queries [20] to generate an ontology. Other proposals implement the relational model as an ontology: Pérez de Laborda and Conrad [35], Calero et al. [8] and Martinez-Cruz et al. [30] implementations are some examples of this.
- *Map databases and ontologies*. There are many proposals implementing this mapping as described in [22]. A subset of this approach consists in *quering databases using ontologies*. A mapping process must be previously done to stablish the relationship between the queried attribute and the ontology. Necib and Freitag [33] or Lim et al. [26] contributions perform this operation to enrich semantically queries using ontologies. Our contribution is a representation of this approach as well.

Alternative systems related with semantic recognition can be found in [24,25,41].

3. Description of the system

In our proposal, two technologies, namely fuzzy and semantic queries, converge into a single system that solves flexible queries on relational databases. On one hand, fuzzy query solving process consists in defining fuzzy sets associated with the attributes involved in the query. Answered tuples accomplish a membership degree to these fuzzy sets. On the other hand, semantic queries use ontologies to return tuples with information semantically similar to the concepts appeared in the query.

Fuzzy databases, which have been previously described in Section 2, allow to define fuzzy sets based on scalar and non-scalar data. In the case of scalars, fuzzy sets can be defined using any kind of membership functions e.g. trapezoidal, triangular or Gaussian. Performed queries using these fuzzy sets confer a big flexibility on the returned data set. But non-scalar data are different. Explicit relationships between each pair of domain values must be defined in order to establish a similarity relationship among them. Such definitions are a hard task since they depend on the application domain of the non-scalar attribute as well as the subjective perception of the designer that evaluates this similarity degree. Moreover, the attribute domain is limited to the initial domain data set due to a new inclusion in the database requires a new similarity definition between the new term and the remaining elements of the domain. We can see an example of this in Fig. 1 with a representation of a *"Hair Color"* attribute.

Here, we propose an alternative to solve this problem by using an ontology to define the semantic relationships among terms of the same domain. A semantic query searches database content comparing it with the ontology that represents the domain of the attribute queried. Resulting data sets will be ordered by a similarity degree calculated from the subject of the query and the database content in the ontology.

The system architecture and the development process that allow to execute flexible queries in the context of relational databases are described in the following subsections.

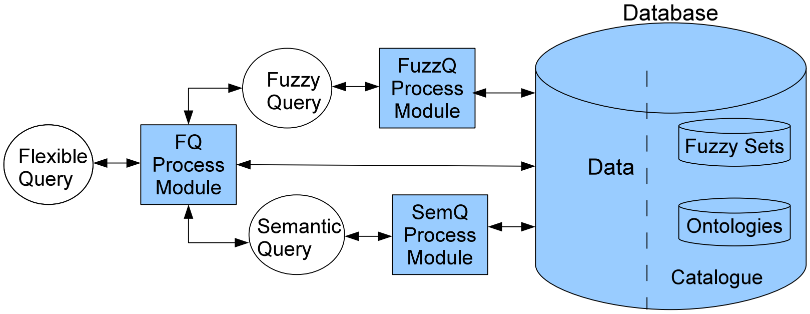


Fig. 2. System architecture.

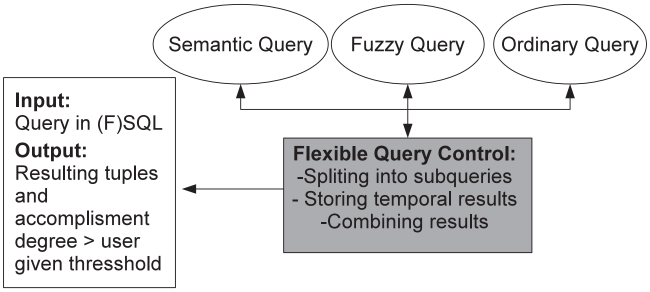

Semantic Query
Fuzzy Query
Ordinary Query
Input:
Query in (F)SQL
Output:
Resulting tuples and accomplishment degree > user given thresshold
Flexible Query Control:
-Spliting into subqueries
- Storing temporal results
-Combining results

Fig. 3. Flexible Query (FQ) process module behavior description.

Table 1
Kind of queries in a flexible query system.

Kind of attributes	Module	Accomplishment deg.
Ordinary attr.	Ordinary DB	1
One semantic attr.	SemQ	Similarity deg.
Two or more semantic attr.	SemQ and FQ	Similarity deg. aggregated
Fuzzy attr.	FuzzQ	Membership deg.
One fuzzy and semantic attr. (coincidence)[a]	FuzzQ, SemQ and FQ	Similarity and membership deg. combined (described in Section 3.2.3)
Fuzzy and semantic attr. (no coincidence)	FuzzQ, SemQ and FQ	Similarity and membership deg. aggregated

[a] This query involves fuzzy and semantic domains and it happens on the same attribute.

3.1. System architecture

The system architecture, which is shown in Fig. 2, has two main parts: (i) a database where data and attribute domains are stored. Attribute domains can be defined by fuzzy sets or ontologies, all of them are stored in the database catalog. (ii) A functional part that defines the behavior of the system with three modules (depicted as blue boxes in Fig. 2):

- *Semantic Queries (SemQ) process module*. This process executes an algorithm that evaluates the similarity degree between the domain ontology associated to an attribute and the database content.
- *Fuzzy Query (FuzzQ) process module*. In this module a fuzzy query is executed performing several subqueries to the fuzzy catalog to get the different fuzzy sets and to perform the evaluation.
- *Flexible Queries (FQ) process module*. In this module (see Fig. 3) two basic operations are performed:
    1. *Input processing* divides the query, if necessary, and each subquery is sent to the corresponding process module according with the query attributes domain.
    2. *Output processing* combines the answers obtained from the different modules. Each row accomplishment degree in the resulting dataset should be higher than a user-given threshold.

A list of attributes and modules that can be involved in each kind of query is shown in Table 1. Notice that all the queries involves the FQ module to analyze the input.

3.2. Development

The system architecture requires a *Database Server* running an instance of an Oracle© database to provide data services and a *Java*-based client application to compute all the operations that are not provided by the database. We can see an illustration and technical details of this architecture in Fig. 4. The database server manages data and computes fuzzy queries

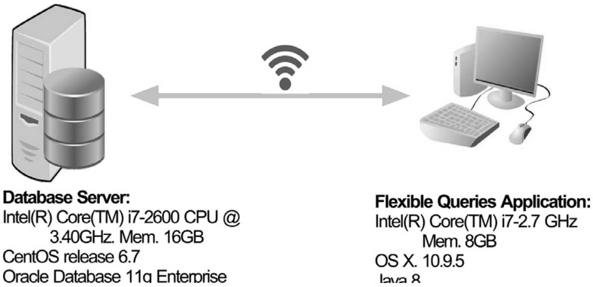

Database Server:
Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz. Mem. 16GB
CentOS release 6.7
Oracle Database 11g Enterprise
Flexible Queries Application:
Intel(R) Core(TM) i7-2.7 GHz Mem. 8GB
OS X. 10.9.5
Java 8

Fig. 4. Technical details of the FlexiQueries system.

thanks to the implementation of a set of stored procedures in PL/SQL. Also, it returns the informational needs (data and metadata) of semantic queries and I/O processing.

A *Java*-based application has been developed to implement the main operational functionality described in Section 3.1. Also, this technology has been used to develop the semantic query module together with the OWLAPI library [15] that allows to manage ontologies.

We analyze the system behavior and implementation details throughout this section.

3.2.1. Fuzzy queries

In our research group we have implemented an extension of a relational database management system to manage fuzzy data [4,32]. In this proposal, we have slightly modified this implementation, to execute only fuzzy queries on ordinary relational databases.

When a fuzzy query is intended to be executed on a relational database, the database must be "prepared" i.e. extended with the desired fuzzy domains. For example, if we are querying: *"Return old people"*, we have to associate the *Age* attribute to the fuzzy set called *Old* as part of its domain as shown in Fig. 1(a). If we are dealing with non-scalar attributes, e.g. *Hair Color*, a set of similarity relationships (see Fig. 1(b)) must be defined. These fuzzy domain attributes can remain in the database catalogue as long as the database designer deems necessary. However, temporary results obtained during the membership degree calculation process are not stored in the database, and they must be recalculated each time the same query is executed in the system.

The extension of the SQL language (FSQL) has been used to make us the process of defining fuzzy queries easier. But, FSQL language usage is only for clarity reasons, because all the operations can be performed using SQL commands and calling PL/SQL procedures.

3.2.2. Semantic queries

To perform flexible queries, an ontology that represents the application domain of a database attribute must be searched and associated with an attribute in the database, in a similar way as fuzzy sets are defined in the database to execute fuzzy queries.

Here, we propose several ways to obtain an ontology that better suits to a specific attribute domain:

- Build our own ontology. There are several editors to define ontologies: *Protege, SWOOP,* etc.
- Using an ontology searcher. There is a large number of ontology searchers, e.g. Swoogle, Watson, Ontoselect, Ontosearch, etc. [11].
- Using a large and generic ontology. We can use *WordNet* or *Dbpedia* as a good alternative when a domain is really specific and there is not any ontology developed yet or we do not have enough knowledge to build one.

The ontology chosen must contain the queried value, but not the complete database content for this attribute. For example, if we are looking for *"Bird Dogs"* [1], it is not necessary to include all dog breeds in the ontology, but only those we are interested in.

Similarity degree in semantic queries

Searching process in a semantic query consists in evaluating the database content with the searched term and their relationship in the ontology. This similarity searching process distinguishes three kinds of relationships in an ontology:

- IS-A relationships, between classes and subclasses, e.g. the *Feline* is the superclass of the *Lion* class.
- Kind-of relationships, between classes and instances, e.g. *Francis* is an instance of *Pope* class.
- Properties relationships, e.g. *Margherita Pizza contains Tomato*, or a *Pizza* class *contains* an *Ingredient*.

---

[1] A *"Bird Dog"* is a type of gun dog or hunting dog used to hunt or retrieve birds or other small game animals.

In our system, the establishment of the similarity relationship measure between two terms is a complex task because it cannot be quantified in an objective manner. In terms of semantics, not only the searched concept is subjective but also the selected ontology because its definition can vary in granularity and ways of formalizing the same information domain.

Despite this awkwardness, there are many proposals to compute semantic similarity between two concepts using ontologies as described in [38]. We extend the semantic relatedness measure proposed by Hirst and St-Onge [14] to include all the relationships identified above. Calculation of this measure is simple and efficient, and it can be applied in different kinds of ontologies. The notion of taxonomical edge-counting is extended by considering also non-taxonomic semantic links in the path calculation of this measure. Our extension includes in the formula the Kind-of relationships as well, despite this extension is rarely used in most ontologies because they do not include instances. Also, this similarity measure is normalized to [0,1]. The equation that computes the similarity degree is:

$$similarity(a, b) = \frac{(2C - full\_path\_with\_instances(a, b) - k * turns(a, b))}{2C} \tag{1}$$

where

- C is the maximum distance to be taken into account. In our system, this variable is always the maximum distance between the root or superclass and the farthest instance. If we are using a very large ontology as Wordnet we must establish the distance by hand. In the literature Hirst and St-Onge in [14] set this value to 8.
- *turns(a,b)* is the number of times the path's direction changes.
- k is the weight of *turns(a,b)*. We propose to set this variable to 1 to evaluate turns the same as path distance.
- *full_path_with_instances(a,*b) is the minimum distance between two elements in the ontology. We have used the Floyd–Warshall algorithm [10], based in graphs analysis, to compute this measure.

FSQL extension

FSQL syntax has been slightly modified to define and manage ontologies in the system catalog. Consequently, if a query is performed on semantic attributes, e.g. *"Return films about Zombies"*, this query would be expressed as: *"SELECT name, CDEG(∗) FROM films WHERE category FEQ $Zombi$ THOLD 0.7"*, where *Category* attribute has associated a film ontology that includes the term *Zombie* in it. THOLD clause filters tuples with a similarity degree equal or higher than 0.7 and FEQ means the comparison operator *Fuzzy EQuals*.

Likewise fuzzy queries module, this semantic query module has been designed for querying purposes only, and then, semantic metadata can be kept in the system catalogue as long as the designer deems necessary.

3.2.3. Flexible queries

Flexible queries module is in charge of receiving and analyzing a query, sending it to the corresponding module and aggregating the resulting data set. This module is implemented in Java, and it establishes a database connection to access to the DB system catalog. Implemented aggregation functions and evaluation algorithms are described below.

Aggregation

When a query involves several attributes, each attribute has associated a membership/similarity degree after being processed. To get the *accomplishment degree* of a row in a query we use the Zadeh's set of operations of union (max) or t-conorm for the disjunctive '*OR*' operator and intersection (min) or t-norm for the conjunctive '*AND*' operator:

- Union (max) $\mu_{A\cup B}(x) = max(\mu_A(x), \mu_B(x))$
- Intersection (min) $\mu_{A\cap B}(x) = min(\mu_A(x), \mu_B(x))$

Note that: (i) crisp attributes have always value of 0 or 1 in order to make the evaluation. (ii) Fuzzy attributes have a membership degree and semantic attributes have a similarity degree, but both values are treated as if they were the same, in terms of using the aggregation functions. (iii) When an attribute is evaluated fuzzy and semantically at once, only one value is kept, and the process of calculating this value is explained below.

Fuzzy-semantic attribute management

The most difficult task included in this proposal [2] is to evaluate attributes with two different domains associated to the same attribute: a fuzzy and an ontology one. In this situation the Algorithm 1 is implemented in the FQ process module.

This Algorithm 1 prioritizes the membership degree got from the Fuzzy Query Module (*fuzzy_query_exec()*) because all the similarity relationships of the "fuzzy" attribute domain have been defined explicitly. Then, a lower membership degree got from this module prevails over a higher similarity degree obtained from the semantic query module (*sem_query_exec()*). After that, if a threshold has been given to the algorithm, tuples are filtered according to this value. Developed procedures involved in this algorithm are:

---

[2] Remember that it is required an explicit definition of the query attribute domains and consequently, a double definition (semantic and fuzzy) could imply a big effort.

```
Input  : table, attribute, evalvalue, [threshold]
Output: result: set(rowId, value, degree)
Data: set1, set2 = set(rowId,value,degree), temprow, findrow: set(rowId, value, degree)
set1 ← fuzzy_query_exec(table, attribute, evalvalue);
set2 ← semantic_query_exec(table, attribute, evalvalue);
for each row temprow in set2 do
    findrow = search(temprow, set1);
    if findrow <> null and findrow.degree > 0 then
        add (result, findrow) ;
    end
    else
        add (result,temprow) ;
    end
end
if threshold <> null and threshold > 0 then
    for each row temprow in result do
        if temprow.degree < threshold then
            remove (result, temprow) ;
        end
    end
end
```

Algorithm 1: Fuzz-Sem-Query.

- *fuzzy_query_exec(table, attribute,evalvalue)* procedure: returns a set of tuples with three attributes: id_row, evaluated attribute value and membership degree of this evaluation. It only returns those tuples that satisfy the condition.
- *semantic_query_exec(table,attribute,evalvalue)* procedure: returns a set of tuples with three attributes: id_row, evaluated attribute value and similarity degree of this evaluation. It returns all tuples, not only the ones that match with the ontology. Attribute values without any coincidence get a value of 0 in the similarity calculation.

4. Results and discussion

In this section, we show the system behavior and its performance through two examples. Moreover, a qualitative comparison among different fuzzy and semantic systems in the literature is included to stress the strengths and drawbacks of our contribution.

4.1. Experimentation 1

The system behavior, when a flexible query is executed in the system, is analyzed in this example. A small database about animals has been developed for experimentation purposes only. A brief schema of this database is shown in Fig. 5.

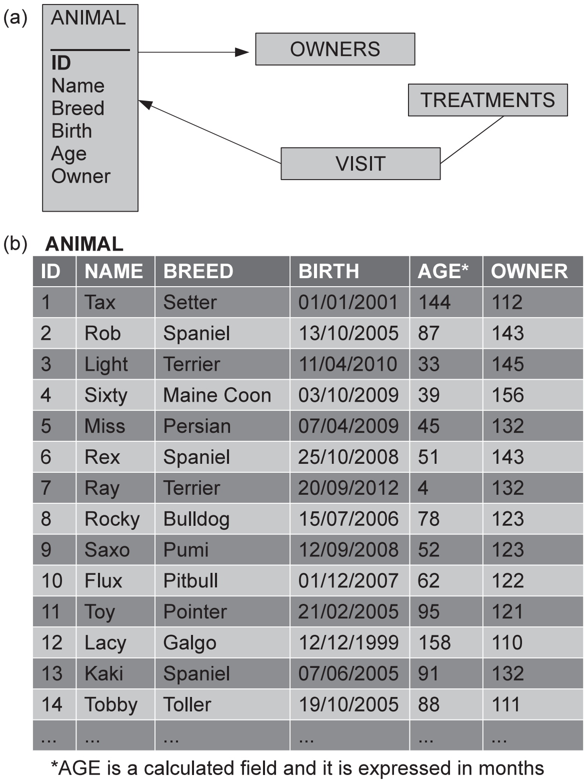
The flexible query: *“Return all Old Bird Dogs”* implies the definition of two attribute domains in the DB:

- Age: this attribute represents the age of an animal measured in months. We have defined *‘Old’* as a fuzzy set represented by the *right shoulder* membership function shown in Fig. 6.
- Specie: it means the specie of animal. The work of defining a similarity relationship between each pair of animals according with their specie does not make sense because it is too laborious. We can simplify this task using the ontology about animal species that is shown in Fig. 7.

The query introduced in the system in FSQL is: *“SELECT id, specie, age, CDEG(∗) FROM animal WHERE age GEQ Old AND specie FEQ BirdDog;”* where *GEQ* means Greater or Equal, *FEQ* means Equal and *CDEG(∗)* means that each returned row must include the calculated accomplishment degree.

The database searching process starts when the query is sent to the FQ process module and the query is split into two subqueries:

- Subquery 1: *return “Old” animals*: this query is solved by the FuzzQ process module. Resulting data set is shown in Table 2. Animals between 0 and 84 months (7 years old) are returned by the query with a membership degree of 0. Animals from 8 years onwards get a membership degree of 1. Remaining animals have a membership degree that is calculated using this formula (see Fig. 6): $y = \frac{x-84}{12}$
- Subquery 2: *return “Bird Dog” species*: this query is sent to SemQ process module. Resulting data are shown in Table 3. Similarity degree of 1 is obtained only if the searched value (“*Bird Dog*”) matches with a database register. Otherwise similarity degrees decrease according with the distance between the searched value and the animal breed found in the database. Below, some of the distances are calculated:



ANIMAL

ID	NAME	BREED	BIRTH	AGE*	OWNER
1	Tax	Setter	01/01/2001	144	112
2	Rob	Spaniel	13/10/2005	87	143
3	Light	Terrier	11/04/2010	33	145
4	Sixty	Maine Coon	03/10/2009	39	156
5	Miss	Persian	07/04/2009	45	132
6	Rex	Spaniel	25/10/2008	51	143
7	Ray	Terrier	20/09/2012	4	132
8	Rocky	Bulldog	15/07/2006	78	123
9	Saxo	Pumi	12/09/2008	52	123
10	Flux	Pitbull	01/12/2007	62	122
11	Toy	Pointer	21/02/2005	95	121
12	Lacy	Galgo	12/12/1999	158	110
13	Kaki	Spaniel	07/06/2005	91	132
14	Tobby	Toller	19/10/2005	88	111
...	...	...	...	...	...

*AGE is a calculated field and it is expressed in months

Fig. 5. Veterinary database: (a) Entity-relationship diagram and (b) Some tuples of table *Animal*. Total number of tuples: 40.

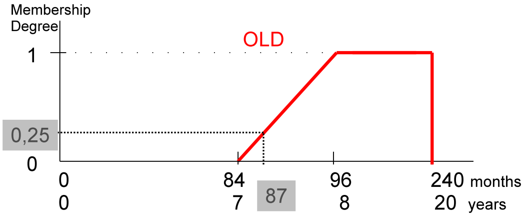


Fig. 6. Calculation process of *"Old Dogs"* membership degree.

Table 2
Results of querying *"Old" animals* in FuzzQ module.

ID	Age	Membership degree	ID	Age	Membership degree
1	144	1	8	78	0
2	87	0.25	9	52	0
3	33	0	10	62	0
4	39	0	11	95	0.91
5	45	0	12	158	1
6	51	0	13	91	0.58
7	4	0	14	88	0.33
			…	…	…

- $similarity(Setter, BirdDog) = \frac{8-1-0}{8} = 0.87$
- $similarity(Terrier, BirdDog) = \frac{8-4-1}{8} = 0.375$
- $similarity(Pumi, BirdDog) = \frac{8-6-1}{8} = 0.125$
- $similarity(Toller, BirdDog) = \frac{8-3-1}{8} = 0.5$

Both results are aggregated using the specified t-norm. Resulting tuples are shown in Table 4. As we can observe, tuples of *"Old Bird Dogs"* with the highest degrees are: *Dog 1 (12 years)* and *Dog 11 (8 years)*. However, *Dog 13 (7 years and 8 moths)* almost accomplished the query because its age is close to *Old*, *Dog 14 (7 years and 3 moths)*, *Dog 2 (7 years and 3 moths)* and

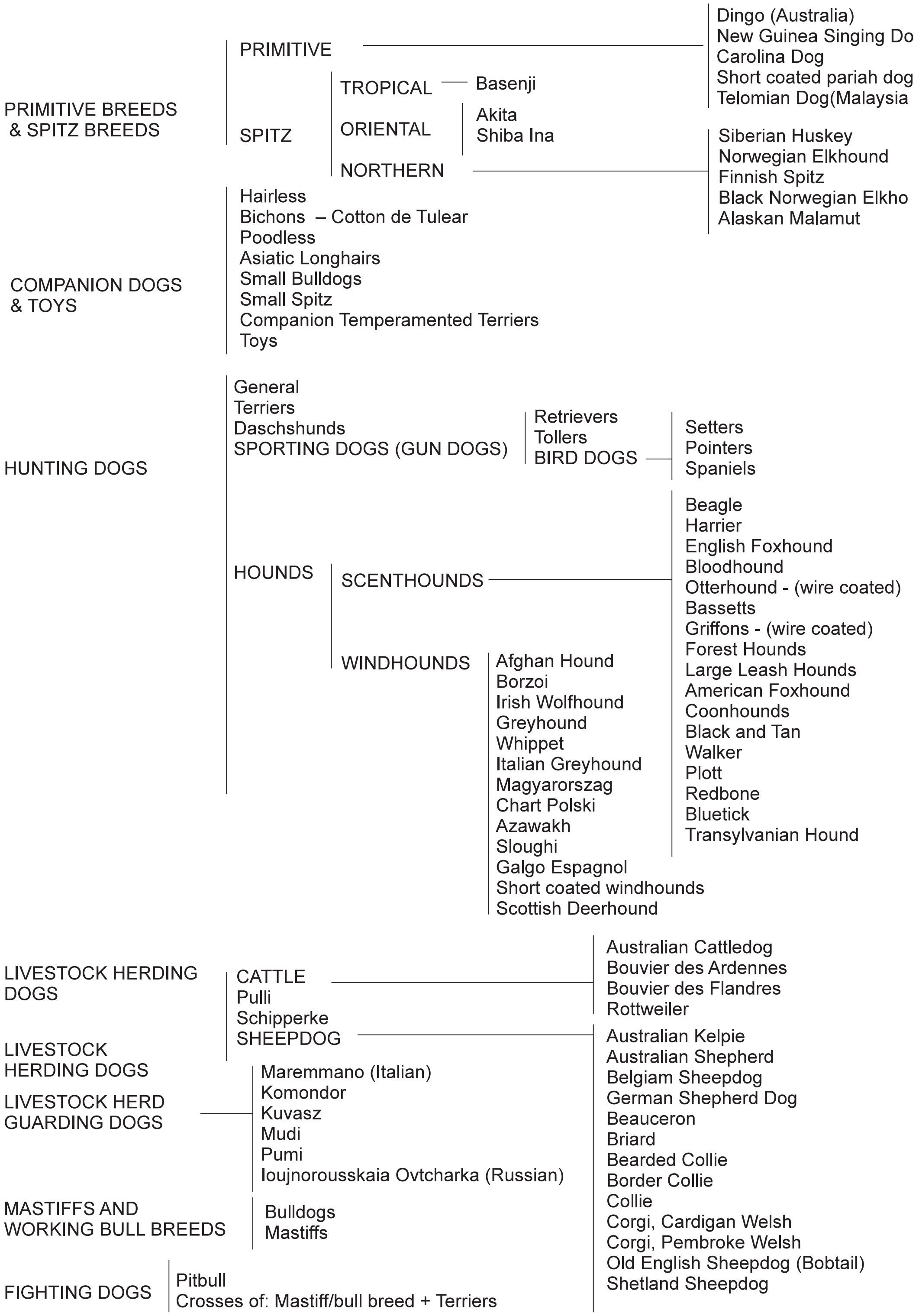


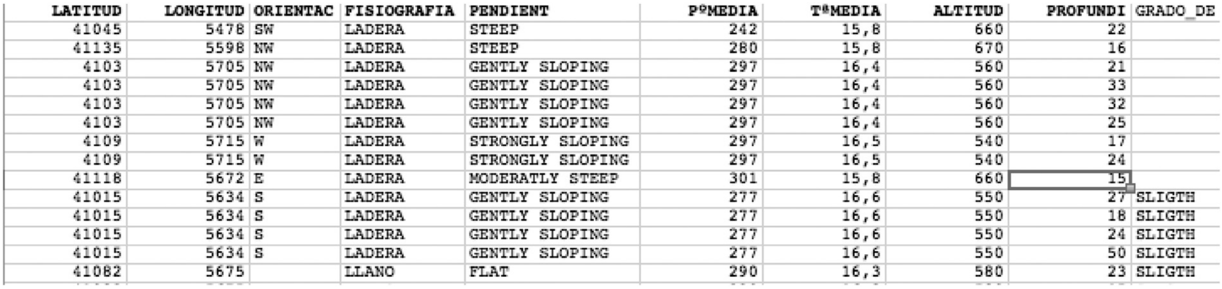
Fig. 7. Ontology about dogs breed based on Dalzell/Russell classification. Root class: *Dogs*.

Table 3
Results of querying about *"Bird Dogs"* in the DB in SemQ module.

ID	Specie	Similarity degree	ID	Specie	Similarity degree
1	Setter	0.87	8	Bulldog	0.25
2	Spaniel	0.87	9	Pumi	0.125
3	Terrier	0.375	10	Pitbull	0.25
4	Maine Coon	–	11	Pointer	0.87
5	Persian	–	12	Galgo	0.25
6	Spaniel	0.87	13	Spaniel	0.87
7	Terrier	0.375	14	Toller	0.5
			…	…	…

Table 4
Results of query: "*Return all the Old Bird Dogs*".

ID	Specie	Age	Accomp. deg.	Description
1	Setter	144	0.87	It accomplished the query
11	Pointer	95	0.87	It accomplishes the query
13	Spaniel	91	0.58	It accomplishes partially the query
14	Toller	88	0.33	It accomp. a little the query
2	Spaniel	87	0.25	It accomp. a little the query
12	Galgo	158	0.25	It accomp. a little the query
3	Terrier	33	0	Not old
4	Maine Coon	39	0	Not in the ontology
5	Persian	45	0	Not in the ontology
6	Spaniel	51	0	Not old
7	Terrier	4	0	Not old
8	Bulldog	78	0	Not old
9	Pumi	52	0	Not old
10	Pitbull	62	0	Not old
…	…	…	…	…



LATITUD	LONGITUD	ORIENTAC	FISIOGRAFIA	PENDIENT	PºMEDIA	TªMEDIA	ALTITUD	PROFUNDI	GRADO_DE
41045	5478	SW	LADERA	STEEP	242	15,8	660	22	
41135	5598	NW	LADERA	STEEP	280	15,8	670	16	
4103	5705	NW	LADERA	GENTLY SLOPING	297	16,4	560	21	
4103	5705	NW	LADERA	GENTLY SLOPING	297	16,4	560	33	
4103	5705	NW	LADERA	GENTLY SLOPING	297	16,4	560	32	
4103	5705	NW	LADERA	GENTLY SLOPING	297	16,4	560	25	
4109	5715	W	LADERA	STRONGLY SLOPING	297	16,5	540	17	
4109	5715	W	LADERA	STRONGLY SLOPING	297	16,5	540	24	
41118	5672	E	LADERA	MODERATLY STEEP	301	15,8	660	15	
41015	5634	S	LADERA	GENTLY SLOPING	277	16,6	550	27	SLIGTH
41015	5634	S	LADERA	GENTLY SLOPING	277	16,6	550	18	SLIGTH
41015	5634	S	LADERA	GENTLY SLOPING	277	16,6	550	24	SLIGTH
41015	5634	S	LADERA	GENTLY SLOPING	277	16,6	550	50	SLIGTH
41082	5675		LLANO	FLAT	290	16,3	580	23	SLIGTH

Fig. 8. Example of data.

Dog 12 (12 years) accomplish the query with a very low degree, because they are not for bird hunting purposes or because they are not old enough.

Note that we have not included a threshold in this query, and consequently, we have obtained all the rows of the database with an associated accomplishment degree in the answer.

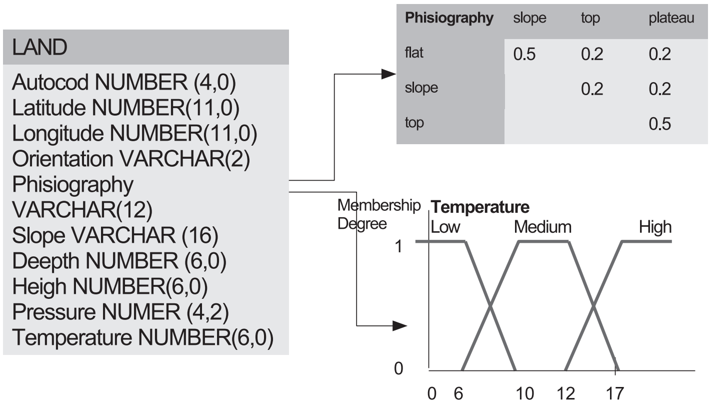
4.2. Experimentation 2

Different tests to measure the system efficiency have been performed on a database about *Land Characteristics*, which has been provided by 542 surveys made to Spanish farmers (see Fig. 8) [39]. For the shake of the brevity we are going to describe a partial view of the database and only the attributes used in our experimentation:

1. *Physiography Attribute*: describes the physiography of the land with this set of labels: *[Flat, Slope, Top, Plateau]*. All of them have a fuzzy definition which is shown in Fig. 9 and a semantic definition, represented by an ontology of land characteristics which is partially shown in Fig. 10[3].
2. *Tavg attribute*: describes the average of temperature in this location. The fuzzy sets: *[High, Medium, Low]* are illustrated in the Fig. 9.

---

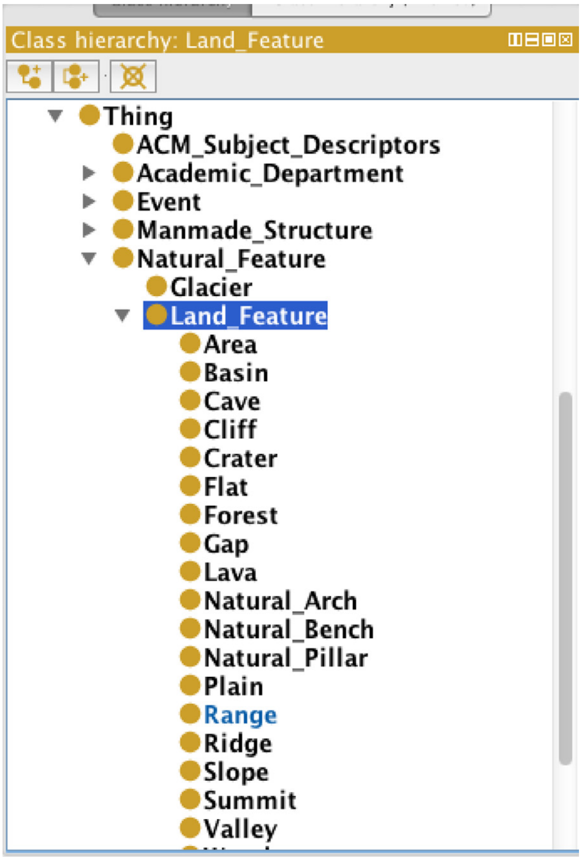
[3] URL: http://lsdis.cs.uga.edu/proj/traks/ontologies/ontology_9.owl



LAND
Autocod NUMBER (4,0)
Latitude NUMBER(11,0)
Longitude NUMBER(11,0)
Orientation VARCHAR(2)
Phisiography VARCHAR(12)
Slope VARCHAR (16)
Deepth NUMBER (6,0)
Heigh NUMBER(6,0)
Pressure NUMER (4,2)
Temperature NUMBER(6,0)

Phisiography	slope	top	plateau
flat	0.5	0.2	0.2
slope		0.2	0.2
top			0.5

Fig. 9. Partial example of *Land Characteristics* database with fuzzy data definitions.



- Thing
  - ACM_Subject_Descriptors
  - Academic_Department
  - Event
  - Manmade_Structure
  - Natural_Feature
    - Glacier
    - Land_Feature
      - Area
      - Basin
      - Cave
      - Cliff
      - Crater
      - Flat
      - Forest
      - Gap
      - Lava
      - Natural_Arch
      - Natural_Bench
      - Natural_Pillar
      - Plain
      - Range
      - Ridge
      - Slope
      - Summit
      - Valley

Fig. 10. Partial ontology view about *Land Features* in *Protege* © editor.

We have performed two different tests on the database to analyze our proposal efficiency:

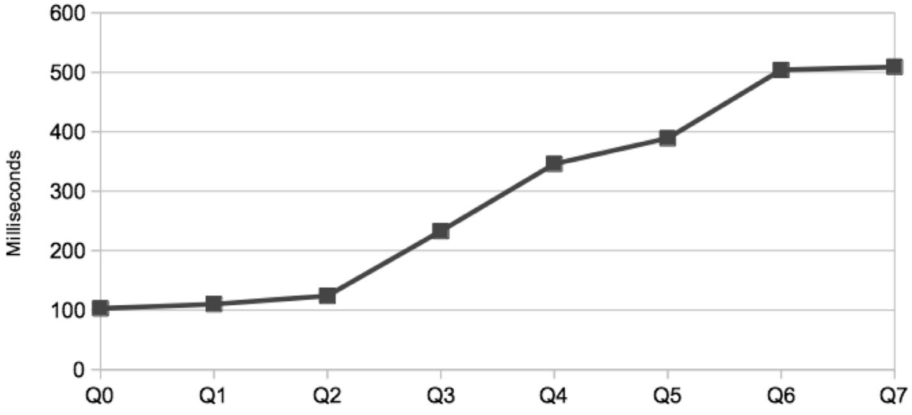
- Varying complexity of the query.
- Varying the number of tuples computed on a same query, to analyze the system scalability.

A set of different queries has been designed to measure the performance of the system. These queries, that vary in complexity, are represented in Table 5 together with a description of them, the number of tuples returned and their execution time in milliseconds. In Fig. 11(a), we can observe how the execution time increases when the number of attributes or the complexity of the query increases. It is important to notice that the network latency is variable and it is around the value given in Q0.

Table 5
Set of flexible queries.

ID	FSQL	Description	Rows	Time
1	Number of pieces of land with *high* average of temperature	*SELECT AUTOCOD, TMEDIA, CDEG(∗) FROM LOCATION WHERE TMEDIA FEQ $HIGH THOLD 0.5*	273	110
2	Number of pieces of land with *high or medium* average of temperature	*SELECT AUTOCOD, TMEDIA, CDEG(∗) FROM LOCATION WHERE TMEDIA FEQ $HIGH THOLD 0.5 OR TMEDIA FEQ $MEDIUM THOLD 0.5*	424	124
3	Number of pieces of land with *valley* physiography	*SELECT PHISIOGRAPHY, AUTOCOD,CDEG(∗) FROM LOCATION WHERE PHISIOGRAPHY FEQ $VALLEY THOLD 0.5*	528	233
4	Number of pieces of land with *slope* physiography	*SELECT PHISIOGRAPHY, AUTOCOD,CDEG(∗) FROM LOCATION WHERE PHISIOGRAPHY FEQ $SLOPE THOLD 0.5*	528	346
5	Number of pieces of land with *slope* physiography and *high* average of temperature	*SELECT PHISIOGRAPHY, AUTOCOD,CDEG(∗) FROM LOCATION WHERE PHISIOGRAPHY FEQ $SLOPE THOLD 0.5 OR TMEDIA FEQ $HIGH THOLD 0.5*	528	389
6	Number of pieces of land with *slope or valley* physiography	*SELECT PHISIOGRAPHY, AUTOCOD, CDEG(∗) FROM LOCATION WHERE PHISIOGRAPHY FEQ $SLOPE THOLD 0.5 OR PHISIOGRAPHY FEQ $VALLEY THOLD 0.5*	528	504
7	Number of pieces of land with *slope or flat* physiography	*SELECT PHISIOGRAPHY, AUTOCOD, CDEG(∗) FROM LOCATION WHERE PHISIOGRAPHY FEQ $SLOPE THOLD 0.5 OR PHISIOGRAPHY FEQ $FLAT THOLD 0.5*	528	509
0	Reference query to measure network latency	*SELECT AUTOCOD FROM LOCATION*	542	103

a) Execution Time



b) Escalability in Execution Times

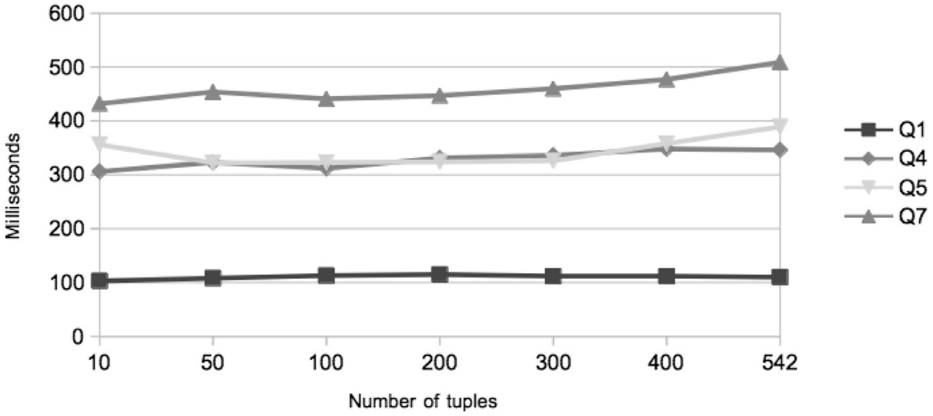


Fig. 11. Execution times in queries performed on the *Land Features* DB.

Table 6
Execution times varying number of tuples.

N. tuples	*Q1*	*Q4*	*Q5*	*Q7*
10	103	306	356	432
50	108	323	322	454
100	113	312	323	441
200	115	331	324	447
300	112	336	326	460
400	112	348	358	477
542	110	346	389	509

Table 7
Comparison of most relevant features in fuzzy query systems.

Model	Zemankova [45]	Buckles [7]	Prade [36]	Umano et al. [43]	Kacprzyk [18]	Bosc [5]	Medina [32]	Our proposal
Manage scalar data	X	X	X	X	X	X	X	X
Manage non-scalar data	X	X	X				X	X
Similarity relationship	X	X					X	X
Possibility distributions	X		X		X	X	X	X
Degree in attributes level		X	X	X			X	X
Degree in tuples level	X		X	X	X	X	X	X
Fuzzy modifiers	X					X		
Fuzzy quantifiers					X	X	X	
Fuzzy comparison operators	X	X	X	X	X	X	X	X
Fuzzy group by op.					X	X		
Fuzzy joins			X	X		X	X	
Nesting						X	X	
Store fuzzy data	X	X	X	X			X	
Fuzzy queries	X	X	X	X	X	X	X	X
Extension SQL lang.				X	X	X	X	X

Table 8
Comparison of most relevant features in semantic query systems.

Model	Necib [34]	Lim [26]	Barrasa [2]	Dragut [12]	Buche [6]	Konstantinou [21]	Zahng [46]	Our proposal
Mapping BD-onto	X	X	X	X		X		
Create ontology								
Enriched query	X	X				X	X	X
Degree of similarity					X		X	X
Query language			X	X	X	X		X
Manage fuzziness								X

Scalability is measured varying the number of tuples in the database from 10 to 542. To do that, the most representative queries have been executed. Execution times are shown in Table 6 and they have been illustrated in Fig. 11(b). In these figures, we can observe how the scalability grows according with the number of computed rows in the most complex queries, that is, those that have more computational needs. However, it is noticeable that varying the number of rows is not enough to distinguish the delay provoked by the latency of the network or the changes in the number of computed rows, specially in the most simplest queries.

4.3. Qualitative comparison

A comparison between the features of the main fuzzy relational databases in the literature and our proposal is shown in Table 7. First models were mainly theoretical proposals of fuzzy relational databases, as implementations of Prade and Testemale [36] in *MACLIPS* on *DPS8* and Umano and Fukami's called FOOBD [43] in SQL. The most complete implementations were provided by Kacprzyk and Zadrozny [18] and Bosc and Pivert [5], called *FQuery* in Microsoft Access © and *Sqlf* respectively. Also, Medina et al. [32] designed and implemented the GEFRED model and FSQL (SQL extension) language in Oracle ©. Our implementation is a modification of Medina's proposal, but in ours data are stored in the database catalogue for querying purposes only. Moreover, there are a few functionalities, that GEFRED has defined theoretically, that have been never included in the implemented version, i.e. fuzzy joins and fuzzy quantifiers, nor in our implementation.

A comparison among query systems that uses ontologies to perform flexible queries on relational databases is a hard task due to the big amount of developments in the last two decades, as we described in Section 2. In Table 8, we focus in some of existing developments that perform queries on Relational databases through an external pre-existing ontology.

Necib's [34] and Lim et al.'s proposals use ontologies to enrich the query with more vocabulary. Barrasa et al.'s proposal develops a language to represent mappings between Ontologies and RDB. Similarly, Draug et al.'s proposal that maps relational databases to an Ontology automatically. Buche et al. [6] enlarge semantically a query and uses fuzzy completion rules to help the user to formulate it. These fuzzy rules are implemented in *Sqlf*. Tuples have associated an'*adequation*' degree according with the fuzzy degree between any pair of concepts in the ontology. Konstantinou et al. [21] develop a visual tool for mapping an ontology to a relational database to perform queries using an ontology interface. And, Zahng et al.'s [46] implementation performs a semantic enlargement of a query with annotations and ontologies. But queries are performed in the RDB using keywords on databases. These features are summarized in Table 8.

However, we have not included in Table 8 any system that uses an ontology from scratch. In general, most of them need to execute a mapping process between the ontology and the RDB in a preprocessing stage. In contrast to our system and Zahng's. In particular, our system only requires to establish the relationship between the queried attribute and the ontology before starting the querying process explicitly. This situation simplifies the vision of the mapping process, but also, it could lead us to degrade the performance depending of the size of the ontology. However, the user is the one who chooses the ontology and consequently, it does not have to be very large to get good results.

Regarding to query enrichment, our proposal do not enrich queries in terms of including synonyms, hypernyms, etc. in the *SELECT* clause. We focus in the relational databases content and we use the ontology to make more flexible the query when no terms are found in the relational databases or these terms are very restrictive.

Finally, the similarity degree is not a common measure in the context of ontology mapping with relational databases. Maps are defined and queries are performed according with these maps with no accomplishment degree. Draug et al.'s and Zahng et al.' proposal are an exception in this summary because the first one defines fuzzy rules, and the second compares keywords to ontologies and annotations. Our proposal acts similarly to Zahng's but our similarity degree calculation process involves less information.

5. Conclusions

In this paper we provide a mechanism to perform flexible queries on relational databases where answers are not limited to the searched object but also, to similar information to this object. Answers are ordered according with the accomplishment degree to the query.

Therefore, one of the main goals of this proposal consists in solving the fuzzy logic drawback managing non-scalar values. A query requires the definition of each similarity relationship among each pair of the attribute domain values to be solved. Thanks to the use of ontologies, such relationships are established by a set of experts in a consensual and collaborative way and they are represented in terms of their semantic relationship.

We have presented the architecture of this novel system that performs semantic and fuzzy queries on relational databases. To do that, all the methods and algorithms involved in the process of generating a combined answer with different kind of attributes have been studied in depth. As a proof of concept, the proposal has been tested on two different databases where not only the behavior but also a quantitative study about the system efficiency have been provided. Moreover, a qualitative comparison between the most relevant systems in the literature and our proposal has addressed the strengths and drawbacks of our contribution.

Finally, this approach has settled the basis to implement a less rigid query frame for alternative database models, i.e. the non-structured databases as those based in documents where schema-less information are stored without any domain restriction. Also, this implementation can be included in different environments to help users to get information from databases such as Natural Language Avatars, E-Shops searchers, Information Recommendation Systems and any other database where a certain imprecision in the query formulation would be necessary. Future steps in this research line are leaded to its inclusion in these databases.

Acknowledgment

This work has been partially supported by the Spanish Government under the project TIN2014-58227: "Descripción lingüística de información visual mediante técnicas de minería de datos y computación flexible".

References

[1] I. Astrova, Reverse engineering of relational databases to ontologies, in: Proceedings of the First Europan Semantic Web Symposium (ESWS), in: Lecture Notes in Computer Science, vol. 3053, 2004, pp. 327–341.
[2] J. Barrasa, O. Corcho, A.G.. Perez, Fund finder: A case study of database to ontology mapping., in: Proceedings of the 2003 International Semantic Web Conference, in: Lecture Notes in Computer science, vol. 2870, Springer-Verlag., 2003, pp. 17–22.
[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. Mcguinness, P.F. Patel-Schneider, L.A. Stein, OWL Web Ontology Language Reference, Technical Report, W3C, 2007.
[4] I. Blanco, J.C. Cubero, O. Pons, M.A. Vila, An implementation for fuzzy relational databases, in: G. Bordogna, G. Passi (Eds.), Recent Research Issues on the Management of Fuzziness in Databases, Studies in Fuzziness and Soft Computing, Physica-Verlag, 2000, pp. 183–207.
[5] P. Bosc, O. Pivert, SQLF: A relational database language for fuzzy querying, IEEE Trans. Fuzzy Syst. 3 (1) (1995) 1–17.
[6] P. Buche, C. Dervin, O. Haemmerle, R. Thomopoulos, Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules, IEEE Trans. Fuzzy Syst. 13 (3) (2005) 373–383.
[7] B.P. Buckles, F.E. Petry, A fuzzy representation of data for relational databases, Fuzzy Sets Syst. (7) (1982) 213–226.
[8] C. Calero, F. Ruiz, A. Baroni, F. Brito e Abreu, M. Piattini, An ontological approach to describe the SQL:2003 object-relational features, Comput. Stand. Interfaces 28 (6) (2006) 695–713.
[9] E.F. Codd, A relational model of data for large shared data banks, Commun. ACM 13 (6) (1970) 377–387.

[10] T.H. Cormen, C. Stein, R.L. Rivest, C.E. Leiserson, Introduction to Algorithms, second, McGraw-Hill Higher Education, 2001.

[11] M. d'Aquin, E. Motta, Watson, more than a semantic web search engine, Semant. Web 2 (1) (2011) 55–63.

[12] E. Dragut, R. Lawrence, Composing mappings between schemas using a reference ontology, in: Proceedings of the 2004 International Conference ODBASE, Springer, 2004, pp. 783–800.

[13] A. Gómez-Pérez, M. Férnandez-López, O. Corcho-García, Ontological Engineering, Springer-Verlag New York, Inc., 2003.

[14] G. Hirst, D. St Onge, Lexical Chains as Representation of Context for the Detection and Correction Malapropisms, The MIT Press.

[15] M. Horridge, S. Bechhofer, The OWL API: A Java API for OWL ontologies, Semant. Web J. 2 (1) (2011) 11–21.

[16] S. Jean, D. Hondjack, D.N. Xuan, G. Pierra, L. Bellatreche, Y.A. Ameur, Ontodb: It is time to embed your domain ontology in your database, in: Proceedings of the Twelfth International Conference on Database Systems for Advanced Applications (DASFAA), 2007, pp. 1119–1122.

[17] S. Jean, G. Pierra, Y. AitAmeur, Domain ontologies: A database-oriented analysis, in: Proceedings of the 2006 Web Information Systems and Technologies (WEBIST), 2006.

[18] J. Kacprzyk, S. Zadrozny, SQLF and FQuery for access, in: Proceedings of the Twentieth NAFIPS International Conference and Joint Ninth IFSA World Congress, vol. 4, 2001, pp. 2464–2469.

[19] J. Broekstra, A. Kampman, F. van Harmelen, Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, in: The Semantic Web(ISWC 2002): First International Semantic Web Conference, 2002, pp. 54–68. 2003 pp. 71–89.

[20] V. Kashyap, Design and creation of ontologies for environmental information retrieval, in: Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management, 1999, pp. 1–18.

[21] N. Konstantinou, D. Spanos, M. Chalas, E. Solidakis, N. Mitrou, VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents, in: Proceedings of the 2006 CAISE Third International Workshop on Web Information System Modeling (WISM), 239, 2006.

[22] N. Konstantinou, D.E. Spanos, N. Mitrou, Ontology and database mapping: A survey of current implementations and future directions, J. Web Eng. 7 (1) (2008) 1–24.

[23] D. Lennat, CYC: A large-scale investment in knowledge infrastructure, Commun. ACM 33 (8) (1995) 33–38.

[24] Z. Li, J. Liu, J. Tang, H. Lu, Robust structured subspace learning for data representation, IEEE Trans. Pattern Anal. Mach. Intell. 37 (10) (2015) 2085–2098.

[25] Z. Li, J. Liu, Y. Yang, X. Zhou, H. Lu, Clustering-guided sparse structural learning for unsupervised feature selection, IEEE Trans. Knowl. Data Eng. 26 (9) (2014) 2138–2150.

[26] L. Lim, A. Kementsietsidis, M. Wang, Ontology-based Searching in Database Systems, 2012, US Patent 8,135,730 URL http://www.google.com/patents/US8135730.

[27] L. Lubyte, S. Tessaris, Extracting Ontologies from Relational Databases.KRDB Research Centre, Technical Report KRDB07-4, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, 2007.

[28] Z. Ma, Fuzzy Database Modeling of Imprecise and Uncertain Engineering Information, Springer, 2006.

[29] C. Martinez-Cruz, I. Blanco, M. Vila, Ontologies versus relational databases: are they so different? a comparison, Artif. Intell. Rev. 38 (2012) 271–290.

[30] C. Martinez-Cruz, I. Blanco, M.A. Vila, Handbook of Research on Web Information Systems Quality, Idea Group Ref., pp. 300–324.

[31] C. Martinez-Cruz, I.J. Blanco, M.A. Vila, An ontology as a tool for representing fuzzy data in relational databases, Int. J. Comput. Intell. Syst. 5 (6) (2012) 1089–1108.

[32] J.M. Medina, O. Pons, M.A. Vila, GEFRED: A generalized model of fuzzy relational databases, Inf. Sci. 76 (1-2) (1994) 87–109.

[33] C. Necib, J.-C. Freytag, Query processing using ontologies, in: O. Pastor, J. Falcão e Cunha (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 3520, Springer Berlin Heidelberg, 2005, pp. 167–186, doi:10.1007/11431855_13.

[34] C.B. Necib, J.C. Freytag, Ontology based query processing in database management systems., in: Proceedings of the OTM Confederated International Conferences on CoopIS/DOA/ODBASE, 2003, pp. 839–857.

[35] C. Pérez de Laborda, S. Conrad, Relational.owl: a data and schema representation format based on owl, in: Proceedings of the Second Asia-Pacific Conference on Conceptual Modelling (CRPIT '43), 2005, pp. 89–96.

[36] H. Prade, C. Testemale, Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries, Inf. Sci. 34 (1984) 113–143.

[37] E.A. Rundensteiner, L.W. Hawkes, W. Bandler, On nearness measures in fuzzy relational data models, Int. J. Approx. Reason. 3 (1989) 267–298.

[38] D. Sánchez, M. Batet, D. Isern, A. Valls, Ontology-based semantic similarity: A new feature-based approach, Expert Syst. Appl. 39 (9) (2012) 7718–7728.

[39] J. Serrano, M. María Amparo Vila, V. Aranda, G. Delgado, Using fuzzy relational databases to represent agricultural and environmental information an example within the scope of olive cultivation in granada, Mathw. Soft Comput. 8 (3) (2001) 275–289.

[40] S. Staab, R. Studer, Handbook on Ontologies, Springer, 2004.

[41] J. Tang, Z. Li, M. Wang, R. Zhao, Neighborhood discriminant hashing for large-scale image retrieval, IEEE Trans. Image Process. 24 (9) (2015) 2827–2840.

[42] Y.A. Tijerino, D.W. Embley, D.W. Lonsdale, Y. Ding, G. Nagy, Towards ontology generation from tables, World Wide Web 8 (3) (2005) 261–285. http://dx.doi.org/10.1007/s11280-005-0360-8.

[43] M. Umano, I. Hatono, H. Tamura, Fuzzy database systems, in: Proccedings of the IEEE International Joint Conference on Fuzzy Systems, vol. 5, 1995, pp. 35–36.

[44] L.A. Zadeh, Fuzzy sets, Inf. Control 83 (1965) 338–353.

[45] M. Zemankova, A. Kandel, Implementing imprecision in information systems, Inf. Sci. 37 (1985) 107–141.

[46] J. Zhang, Z. Peng, S. Wang, H. Nie, Si-seeker: Ontology-based semantic search over databases, in: Knowledge Science, Engineering and Management, in: Lecture Notes in Computer Science, vol. 4092, Springer Berlin Heidelberg, 2006, pp. 599–611.