A Tool to Perform Semantic and Imprecise Queries on non-scalar Data

Carmen Martínez-Cruz
and Jose María Serrano
Departament of Computer Science, University of Jaén,
23071, Jaén, Spain
Email: cmcruz@ujaen.es and jschica@ujaen.es

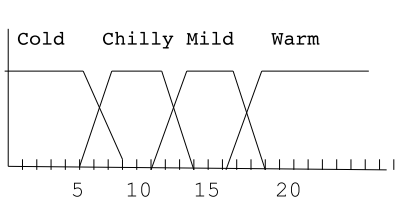
Amparo Vila
Department of Computer Science and A.I.,
University of Granada,
18071, Granada, Spain
Email: vila@decsai.ugr.es

***Abstract*—Imprecision in queries has being managed using fuzzy logic techniques in the last few decades. Fuzzy logic techniques represent uncertainty formally and it allows to manage imprecision on scalar values in an easy and accurate way. The problem arises when users want to deal with semantics and non-scalar data at once. In this situation, fuzzy logic helps us to manage uncertainty, but it lacks of the flexibility that the semantic properties of words imply. Nowadays, ontologies have addressed this problem by the establishment of the semantic relationships among terms. Here, we present a software that allows us to combine fuzzy and semantic queries on non-scalar data. As a proof of concept, we will show some examples of queries performed on a real database about olive trees plantations.**

I. Introduction

Ordinary relational database queries could be considered rigid because user queries require an exact matching between query constraints and database content in order to get results. For example, we can perform a query about student characteristics like this one: *"Return all students with 'big' complexion"*, but the system would return tuples that match with the *"big"* value only. One of the solutions of this rigidness consists in the use of fuzzy logic techniques in database management systems[1], [2]. This solution is really accurate, specially with scalar values modeled by experts. This is the case of temperatures represented in a specific context, e.g. *'Cold', 'Chilly' or 'Warm'* fuzzy sets could be represented by possibility distribution functions as those shown in Figure 1 a). A query like this: "*Return all the cities with 'chilly' temperature*" would return all the tuples ordered by the membership degree associated to the 'chilly' fuzzy set. However, queries that involve non-scalar data are a different issue. A query like this: *"Return all the cities with 'chilly' temperatures"* where the temperature attribute is modeled by words, imprecision could be achieved establishing an explicit similarity relationship between each pair of words. In 1 b), we can see an example of this situation.

This representation that uses fuzzy logic techniques, presents some disadvantages: i) the difficulty of establishing all the similarity degrees between each pair of words, ii) the problem that arises when a new term is included in an already defined domain. iii) The impossibility of querying terms out of the domain, i.e. synonyms, specifications/generalization of a term, etc.



a) Escalar Values

Temp.	Chilly	Mild	Warm
Cold	0.9	0.5	0
Chilly		0.8	0.1
Mild			0.7

b) Non-Escalar Values

Figure 1. Domain of the *Temperature* Attribute using Fuzzy Logic.

Consequently, semantic queries are proposed as an extension to the fuzzy logic database query system to solve this problem [3]. A definition of an ontology[4], [5] that represents the domain of an attribute allows to establish a similarity relationship among database terms and query values. As a result, query clauses would not be restricted to fixed domains, they can include any value anytime, included or not in the domain of the attribute.

The main purpose of this proposal is the description of a software that has been developed to perform queries on an Relational Database Management System (RDMBS) that involves attributes with different domain definitions (ordinary, fuzzy and semantic). This software allows to manage these attributes domains, as well.

A brief summary about fuzzy relational database system implementations is included in Section II together with a study about the use of ontologies in databases. In Section III, the architecture of the system and a description of the main components of the tool are described in detail. Then, the description of the software and an example on a real database is presented in Section IV. Finally, conclusions are included in Section V.

II. Background

In the last decades several tools have been developed to perform queries on databases to manage imprecision [6]. However, these tools use fuzzy logic techniques or semantic techniques to give an answer, but there are not any solution that combines both technologies. A brief summary about the most representative implementations that use fuzzy logic or ontologies for answering queries are described below.

A. Fuzzy queries

There are many implementations in the literature to manage on fuzzy databases[1], [2]. In table I, we can see a summary of the main relevant ones.

Table I
Fuzzy Queries in RDBMS.

Software	Description
Prade and Testemale [7] in *MACLISP* on *DPS8*	This model is mainly theoretical, and it performs a small implementation to manage specially unknown or incomplete information
Umano and Fukami's called FOOBD in SQL [8]	This model is mainly theoretical, and it handles fuzzy and probabilistic information using object oriented databases
Kacprzyk et al. [9] called *FQuery* in Microsoft Access ©	It is a database query system that allows to retrieve database information using fuzzy descriptions and linguistic quantifiers
Bosc et al. [10] called *Sqlf*	This model extends database functionality and SQL to allow imprecise queries
Medina et al. [11] designed and implemented the GEFRED model and FSQL language in Oracle ©	This system proposes a complete extension of the relational database management system to manage imprecision on scalar, non-scalar, unknown, undefined or incomplete information. An extension of the SQL language is also included here

Our proposal is based on Medina et al. [11], [12] implementation, which extends relational database model with fuzzy data types and operations. It has been developed in an Oracle ©system because not only data but also procedures can be included and executed in the system. Moreover, an extension of SQL has ben developed (Fuzzy SQL (FSQL)) to deal with fuzzy data in an easier way [13].

B. Semantic queries

Ontologies have became one of the most popular knowledge representation technique in the last decades thank to the raise of the Semantic Web [4], [14]. Semantic queries allow to perform flexible queries on any kind of data sources because ontologies establish relationships among different terms such as, synonyms, antecedents or is-a relationships in the same structure. Database queries using ontologies have been used in the literature in many implementations. In table II we can see a summary of some of the main representative.

In Table II, there are not any system that uses an ontology from scratch. Most of them perform a mapping process between ontologies and relational databases in a pre-processing stage. In contrast, in our proposal we select the ontology at the moment of the query definition.

Table II
Semantic Queries on RDBMS.

Software	Description
Necib et al.[15] and Lim [16]	This proposal enriches queries on relational databases with the vocabulary provided by an ontology
Barrasa et al.[17]	It develops a system that maps relational DB and ontologies to establish a communication between them
Dragut et al.[18]	This system allows to map automatically an ontology with a database
Buche et al.[19]	This system enlarges a query with ontologies and uses fuzzy completion rules to help users in the formulation process
Konstantinou et al. [20]	They develop a visual tool that maps ontologies and relational databases
Zahng [21] et al.	This implementation performs queries semantically enriched by annotations and ontologies. Queries are executed on the database using keywords.

III. Fuzzy and Semantic Query System

In this paper we describe a software that answer queries that involve non-scalar data on databases using fuzzy logic techniques and ontologies. This idea, that has been described in deepth in [3], combines both kind of domain descriptions in order to provide an enriched answer which includes an accomplishment degree to each returned tuple that satisfies the query clause. This tool has been developed to deal with the following situations:

1) Queries that involve attributes which are described by a set of similarity relationships among each element of the domain. This is the situation described in Fig. 1 b).
2) Queries that involve attributes that have ontologies associated to their domains.
3) Queries that involve both kinds of descriptions at once. Consequently, tuples must be computed in both senses to provide the most accurate answer according with their accomplishment degree.

A. System Architecture

The system architecture consists in three basic modules: fuzzy and semantic modules that are managed by the Query Processing Module (QPM) (see Figure 2). All these modules perform the following operations:

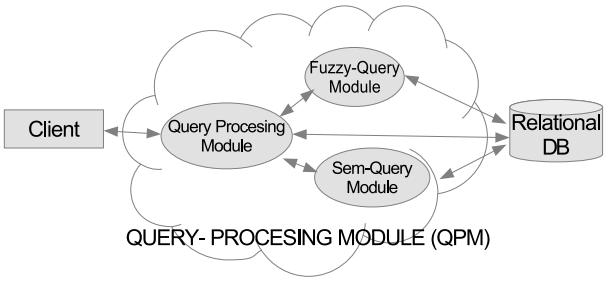


Figure 2. System Architecture.

- *Fuzzy Module*: This module manages fuzzy queries. The input is a sub-query from the QPM, and this module interacts with Relational DB catalog to compute all the

similarity relationships between terms. The output is sent to the query processing module together with each tuple similarity/membership degree.

- *Semantic Module*: This module manages semantic queries. The input is a sub-query from the QPM, and this module computes relationships among an ontology and the database content. Resulting tuples are returned together with a similarity degree.
- *Query Processing Module*: This is the main module, where inputs and outputs are managed and formatted.
  - Inputs are analyzed and split according with the query attributes if necessary.
  - If there is an attribute described by fuzzy logic, the subquery is sent to the Fuzzy Module.
  - If there is an attribute described by an ontology, the subquery is sent to the Semantic Module.
  - If there is an ordinary attribute, the subquery is sent to the Relational DB directly.
  - Tuples resulting from the different modules or Relational DB are combined and ordered by an accomplishment degree. This accomplishment degree comes from the returned tuples membership or similarity degree according with the system strategies described in Section III-B.
  - Database Catalog: This element represents the place in the database where not only a description of metadata in the Relational data model but also fuzzy and semantic descriptions are stored as well. In this catalog, the system stores similarity relationships and ontologies to make easier the final user querying process.

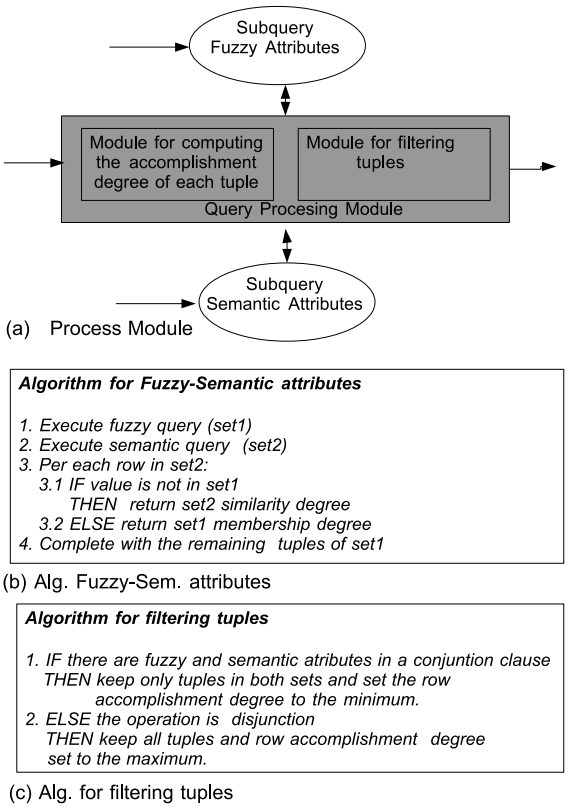


Figure 3. System Algorithms.

B. Programming strategies

When the QPM module receives a query, it analyzes the attributes involved in it to split the query into subqueries and send them to the proper modules (see Figure 3 a)). The answer of these modules consists of this minimum set of attributes:
Tuple_ID, attribute_value, similarity-threshold degree
The output is processed by this module using the strategies below.

1) Decision algorithm: All the results are aggregated and ordered in the output according with each tuple accomplishment degree. If the user has provided a threshold, then results are filtered in the output as shown in the Figure 3 c). Here, it is shown a simplification of the use of a t-norm for the conjunctive operation and a t-conorm for the disjunctive one.

When one tuple has two associated degrees, the QPM is in charge to decide the final accomplishment degree following the algorithm described in Figure 3 b). This situation happens when two different domain definitions are associated with the same attribute to cover all its possible values. It is not very usual but when it happens, the strategy consists basically on prioritizing fuzzy logic module over the semantic one because these relationships are considered semantically stronger than ontology ones.

2) Ontology comparison process: This tool evaluates the semantic relationship between the term included in the query and in the database attribute using the associated ontology. For example, in the query: *"Return all the cities with 'chilly' temperatures"*, if *'chilly'* is a value represented in the ontology, a distance calculation between this value and each term of the database attribute 'Temperature' would be computed. Some of the advantages of using ontologies to perform this calculation are: i) Ontologies can change according with the user interests, ii) If there are not ontologies available we can use alternatives such as the well-known thesaurus *WordNet*, iii) Ontologies provide different kind of relationships between terms that give us a framework to establish a weight system to differentiate them, iv) We can use different methods to calculate a similarity degree among ontology elements [22].

In this proposal, we extend the semantic relatedness measure proposed by Hirst and St-Onge in [23] to include the following relationships: IS-A , Kind-of, Properties and annotations. Our extension includes Kind-of relationships, 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:

$$sim(a,b) = \frac{(2C - path\_with\_inst(a,b) - k * turns(a,b))}{2C} \quad (1)$$

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 [23] 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.
- *path_with_inst(a,*b) is the minimum distance between two elements in the ontology. We have used the Floyd-Warshall algorithm [24], based in graph analysis to compute this measure.

IV. Results and Discussion

The developed software is described together with a real world example that shows how a database can be queried using different kinds of domains of an attribute.

A. Description of the Software

This development consists in a database client application developed in Java, it has been called "FuzzyQueries2+" version 2.0. The first version was described in [25], [26], [13]. This implementation changes the purpose of the previous version because of the fact that this software is oriented to perform only queries on a Relational Database System. Although, it allows to associate and store fuzzy or semantic domains to specific attributes in the RDBMS to help in the user interaction . This application has the characteristics below:

- It allows to establish the connection/disconnection to an Oracle © database,
- Query Processing Module (QPM) is developed here and the algorithms described in III-B,
- It includes most of the semantic query module with the use of the ontology mananagement library OWLAPI [27] in Java.
- Fuzzy data interaction: SQL or FSQL can be used in this application,
- Query results are shown as temporal tables,
- Response time calculation is included as well.

This software is a development performed in the core of the research group and it is available under request to our group leader. To have a full functionality of this client, it is required to install some Oracle © database scripts with different stored PL/SQL procedures that allow to manage:

- FSQL syntax,
- Temporal fuzzy and ontology structures in the database catalog.

B. Results

A set of queries executed on a real database about land characteristics, which have been made to 542 Spanish olive tree farmers[28], are analyzed in this section. A brief description of some of this database attributes is shown in Figure 4 and 5.

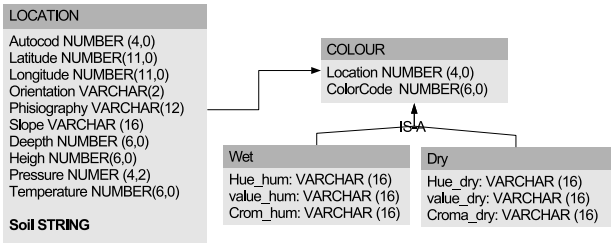


Figure 4. Database Schema.

LAT	LON	ORIENTAC	PHISIOG	SLOPE	SOIL
41118	5453	E	LADERA	STEEP	SINGLE GRAIN
41118	5453	E	LADERA	STEEP	MASSIVE
41098	5509	SW	LADERA	STEEP	MIGAJOSA
420905	60890		LADERA	MODERATLY STEEP	MIGAJOSA
420905	60890		LADERA	MODERATLY STEEP	MIGAJOSA
420905	60890		LADERA	MODERATLY STEEP	MASSIVE
420905	60890		LADERA	MODERATLY STEEP	MASSIVE
422015	61370		LADERA	MODERATLY STEEP	SUBANGULAR BLOCKY
422015	61370		LADERA	MODERATLY STEEP	ROCK STRUCTURE
422255	62340		LADERA	GENTLY SLOPING	GRANULAR
422255	62340		LADERA	GENTLY SLOPING	MASSIVE
422255	62340		LADERA	GENTLY SLOPING	ROCK STRUCTURE
420800	60295		LADERA	SLOPING	GRANULAR
420800	60295		LADERA	SLOPING	SUBANGULAR BLOCKY
420800	60295		LADERA	SLOPING	MASSIVE
420800	60295		LADERA	SLOPING	MASSIVE
42213	61340		LADERA	GENTLY SLOPING	SUBANGULAR BLOCKY
42213	61340		LADERA	GENTLY SLOPING	SUBANGULAR BLOCKY
42213	61340		LADERA	GENTLY SLOPING	PLATY
422010	60870		LLANO	FLAT	GRANULAR

Figure 5. Partial Database Screenshoot.

In this experimentation, only one attribute is being analyzed due to space limitations. The description of this attribute, that represents the properties of the soil, can be seen in Figure 6. This attribute domain has been defined twice: i) A fuzzy domain establishes the most relevant terms used in this context. This representation is shown in Figure 7 a). ii) An ontology, that has been generated following the description in [29], covers many other soil structure terms, as we can see in Figure 7 b).

The following set of queries has been performed on the database to analyze the behavior of this software. The syntax of these queries follows this pattern: *"Return all the properties whose soil is ..."*.

1) *'Platy'*: The system only uses the fuzzy logic query module since there are not a *'Laminar'* definition in the ontology.
2) *'Laminar'*: The system uses only the semantic module since there are no other defintion of this term in the fuzzy logic module. In Table III the resulting similarity degree of the query restriction and each ontology term that accomplishes the threshold is shown.

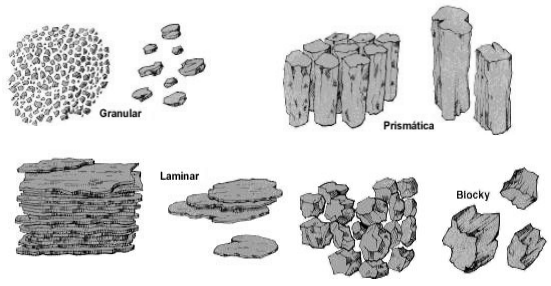
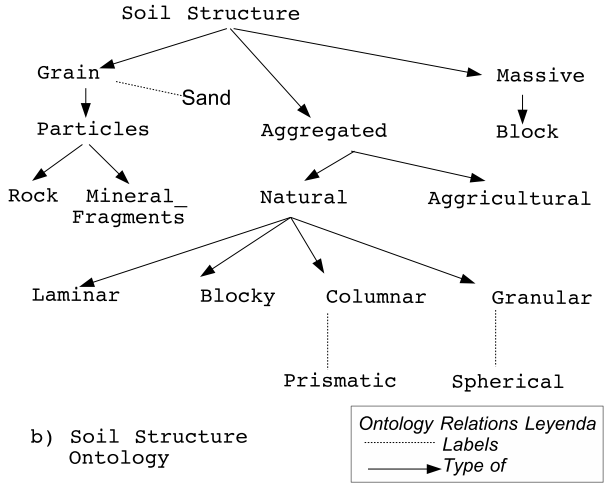


Figure 6. Kind of Soil Quality[30].

SOIL DESCRIPTION

Kind	Granular	Prismatic	Platy
Prismatic	0.5		
Platy	0.5	0.2	
Blocky	0.5	0.8	0.2

a) Fuzzy Description



b) Soil Structure Ontology

Figure 7. Domain Description of Soil Attribute.

3) *'Prismatic'*: Both modules are executed and answered tuples are combined. Results include tuples with these terms and a lower accomplishment degrees: 'Granular', 'Blocky' and 'Platy' and Massive'. [1]
4) *'Laminar'* and *'Platy'*: Both modules are executed but in different modules, and accomplishment degree is got using a t-norm for the conjunctive operation. Results are similar to Query 2, but not accomplishment degrees.
5) *'Migajosa'*: There are not any fuzzy or semantic representation for this term but it is contained in the database, consequently, an ordinary query is performed in the database and all the returned tuples get an accomplishment degree of 1.

In Figure 8), an example of the first query is shown in

[1]we have considered *(Angular/Subangular) 'Blocky'* as *'Blocky'* term.

Table III
SIMILARITY DEG. BETWEEN *'Laminar'* AND EACH ONTOLOGY TERM IN QUERY 2 WITH 0.5 THRESHOLD

Ontology Value	Similarity Deg.
'Blocky' (Angular/Subangular)	0.8
'Granular'	0.8
'Massive'	0.5
'Grain'	0.5
'Prismatic'	0.6

Table IV
SOME OF THE MOST REPRESENTATIVE SEMANTIC AND FUZZY QUERIES.

Query	Rows	Time (msec)
1	110	343
2	442	278
3	395	351
4	442	421
5	71	293

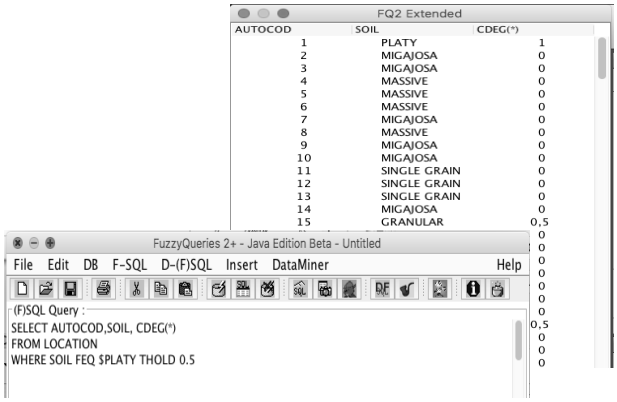


Figure 8. FuzzyQueries 2+ Screenshoot.

FSQL syntax: *'CDEG(*)'* shows the accomplishment degree of each tuple and *'FEQ'* represents the fuzzy operator 'equal'. *'THOLD'* reprepresent the threshold of 0.5. We have considered that this threshold would be enough to get a fair set of tuples to analyze the system. We should remember that all the returned tuples are ordered by a accomplishment degree.

In Table IV, results and execution times of the queries are described. As we can observe, execution times are not very representative because examples are executed on a medium size database, and the selected ontology manages only 16 concepts. In this situation, connection delay takes the most of the time and execution time is almost irrelevant. Consequently these results are not conclusive in terms of efficiency.

V. CONCLUSIONS

A software that implements an extension of a fuzzy database management system is presented in this paper to solve some problems arisen with non-scalar data. The extension has been aimed to include semantic data management to provide an alternative way of establishing relationships among database terms.

With this software we do provide a query framework where ordinary, fuzzy and semantic queries are answered according with the attribute domain involved in the query. Attribute

domains are temporary stored in the database catalog to help the user in the query process.

Others considerations that may be taken into account with the development of this application are:

- The query process is easier when there is not any expert to model similarity relationships among values of an attribute thanks to the Semantic Querying Module. Ontologies can be downloaded from the Semantic Web, and depending of the ontology source, they are considered a consensual knowledge about of the modeled reality.
- The resulting tuples are evaluated and their final results are aggregated and ordered according with each tuple accomplishment degree which has been calculated using membership or/and similarity degree.
- This system allows to combine two definitions of the same attribute at once. This is a rare situation since the definition process is a heavy task, but when it happens the process can be solved accurately.

We would like to stress that this software is a prototype of a combined system that allows to manage two different techniques for dealing with imprecision and uncertainty in data, since there is not any similar system in the literature. In this sense, here it is only analyzed this software behaviour and performance. However, scalability has not been one of the main goals of our proposal because this software has been focused in the management of medium size databases and medium size ontologies in a fair response time. In future versions, we will extend it to NoSQL databases in distributed environments, where not only large databases but also, large or generic Ontologies or Thesaurus can take part in the process.

ACKNOWLEDGMENT

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

REFERENCES

[1] J. Galindo, A. Urrutia, and M. Piattini, *Fuzzy Databases Modeling, Design and Implementation*. Idea Group Publishing, 2006.

[2] J. Kacprzyk, S. Zadrożny, and G. De Tré, "Fuzziness in database management systems," *Fuzzy Sets Syst.*, vol. 281, no. C, pp. 300–307, Dec. 2015.

[3] C. Martínez-Cruz, J. M. Noguera, and M. A. Vila, "Flexible queries on relational databases using fuzzy logic and ontologies," *Inf. Sci.*, vol. 366, pp. 150–164, 2016.

[4] S. Staab and R. Studer, *Handbook on Ontologies*. Springer, 2004.

[5] C. Martinez-Cruz, I. Blanco, and M. Vila, "Ontologies versus relational databases: are they so different? a comparison," *Artificial Intelligence Review*, vol. 38, pp. 271–290, 2012.

[6] J. Alcalá-Fdez and J. M. Alonso, "A survey of fuzzy systems software: Taxonomy, current research trends, and prospects," *IEEE Trans. Fuzzy Systems*, vol. 24, no. 1, pp. 40–56, 2016.

[7] H. Prade and C. Testemale, "Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries," *Information Sciences*, no. 34, pp. 113–143, 1984.

[8] M. Umano, I. Hatono, and H. Tamura, "Fuzzy database systems," in *Proccedings of Fuzzy Systems, 1995. International Joint Conference IEEE Int.*, vol. 5, Mar 1995, pp. 35–36 vol.5.

[9] J. Kacprzyk and S. Zadrożny, "Sqlf and fquery for access," *IFSA World Congress and 20th NAFIPS International Conference. Joint 9th*, vol. 4, pp. 2464–2469, 2001.

[10] P. Bosc and O. Pivert, "Sqlf: a relational database language for fuzzy querying," *Fuzzy Systems, IEEE Transactions on*, vol. 3, no. 1, pp. 1–17, Feb 1995.

[11] J. M. Medina, O. Pons, and M. A. Vila, "Gefred. a generalized model of fuzzy relational databases," *Information Sciences*, vol. 76, no. 1-2, pp. 87–109, 1994.

[12] C. Martinez-Cruz, I. J. Blanco, and M. A. Vila, "An ontology as a tool for representing fuzzy data in relational databases," *International Journal of Computational Intelligence Systems*, vol. 5, no. 6, pp. 1089–1108, 2012.

[13] I. Blanco, J. C. Cubero, O. Pons, and M. A. Vila, "An implementation for fuzzy relational databases," in *Recent Research Issues on the Management of Fuzziness in Databases*, ser. Studies in Fuzziness and Soft Computing, G. Bordogna and G. Passi, Eds. Physica-Verlag, 2000, pp. 183–207.

[14] D.-E. Spanos, P. Stavrou, and N. Mitrou, "Bringing relational databases into the semantic web: A survey," *Semant. web*, vol. 3, no. 2, pp. 169–209, Apr. 2012.

[15] C. B. Necib and J. C. Freytag, "Ontology based query processing in database management systems." in *CoopIS/DOA/ODBASE*, 2003, pp. 839–857.

[16] L. Lim, A. Kementsietsidis, and M. Wang, "Ontology-based searching in database systems," Mar. 13 2012, uS Patent 8,135,730. [Online]. Available: http://www.google.com/patents/US8135730

[17] 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,*. Springer-Verlag., 2003, pp. 17–22.

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

[19] P. Buche, C. Dervin, O. Haemmerle, and R. Thomopoulos, "Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules," *Fuzzy Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 373–383, June 2005.

[20] N. Konstantinou, D. Spanos, M. Chalas, E. Solidakis, and N. Mitrou, "Visavis: An approach to an intermediate layer between ontologies and relational database contents," in *Proceedings of the CAISE WISM, Luxemburg, June 5-9*, 2006.

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

[22] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Syst. Appl.*, vol. 39, no. 9, pp. 7718–7728, 2012.

[23] G. Hirst and D. St Onge, *Lexical Chains as representation of context for the detection and correction malapropisms*. The MIT Press, May 1998.

[24] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[25] I. Blanco, D. Sánchez, J. Serrano, M. Vila, and J., "Fuzzyqueries 2+, una herramienta para la integración de consultas flexibles, cálculo de agregaciones y resúmenes, y extracción de conocimiento," in *Actas del XI Congreso EspaÃ±ol sobre tecnologÃas y LÃ³gica Fuzzy (ESTYLF02)*, 2002, pp. 337–342.

[26] I. Martinez-Cruz C., Blanco, J. Serrano, and M. Vila, "A first approach to multipurpose relational database server," *Mathware and Soft Computing*, vol. 12, no. 2-3, pp. 129–153, 2005.

[27] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies," *Semantic Web Journal*, vol. 2, no. 1, pp. 11–21, 2011.

[28] J. Calero, G. Delgado, M. Sánchez-Marañón, D. Sánchez, M. A. V. Miranda, and J. M. Serrano, "An experience in management of imprecise soil databases by means of fuzzy association rules and fuzzy approximate dependencies," in *ICEIS (2)*, 2004, pp. 138–146.

[29] E. Gregorich and M. Carter, Eds., *Soil Quality for Crop Production and Ecosystem Health*. Elsevier, 1997, vol. 25.

[30] "Fao description," ftp://ftp.fao.org/fi/cdrom/fao_training/fao_training/general/x6706s/x6706s07.htm, accessed: 2017-01-30.