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.

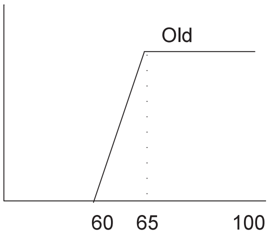
1. Introduction

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/



(a) Human Age

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

(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].