Management of an estate agency allowing fuzzy data and flexible queries

José Galindo[1] **Juan M. Medina**[2] **Juan C. Cubero**[2] **Olga Pons**[2]

[1]Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga (Spain). ppgg@lcc.uma.es

[2]Dpto. Ciencias de la Computación e I.A., Universidad de Granada (Spain). {medina,carlos,opc}@decsai.ugr.es

Abstract

As main result of all our previous work, we have now available a FSQL Server *for Oracle Databases, programmed in* PL/SQL. *This server allows us to query either a Fuzzy or Classic Relational Database with the* FSQL *language (Fuzzy* SQL*). The* FSQL *language is an extension of the* SQL *language which permits us to write flexible (or fuzzy) conditions in our queries to a fuzzy or traditional database. In this work we present a management system for a real estate agency in which some attributes of the landed properties may be fuzzy, i.e., we can store imprecise information about it. Besides, our system allows the user to make flexible or fuzzy queries in order to retrieve the most relevant properties of our database, starting with the customer information. The goal is to retrieve the most interesting landed properties according to the initial customer preferences. Of course, we can obtain a membership degree for each landed property in the fuzzy query result.*

KEYWORDS: Fuzzy Management, Information Retrieval, Flexible Queries, Fuzzy SQL, Fuzzy Query Languages, Fuzzy Relational Databases.

1 Introduction

In the last years, the management applications have been popularized and widely extended. In these applications, the database management is very important. Thus, many times the DBMS (Database Management Systems) are used by the final users directly or through an interface program (front-end).

On the other hand, the fuzzy databases have been developed in the last years, rising up different models, among which they highlight the Prade-Testemale model [8], the Umano-Fukami model [9], the Buckles-Petry model [1], the Zemankova-Kaendel model [10] and the GEFRED model by Medina-Pons-Vila [7]. This last model represents an eclectic synthesis of the different models which have appeared to deal with the problem of the representation and management of fuzzy information in relational databases. One of the main advantages of this model is that it consists of a general abstraction which allows us to deal with different approaches, even when these may seem disparate.

Besides, for the GEFRED model, the FSQL language, a fuzzy (or flexible) query language based on the SQL language, has been defined. This language allows us to express sentences taking into account the characteristics of imprecise information. Thus, as we will see, the FSQL queries allow to express fuzzy conditions, to calculate fulfillment degrees, to establish fulfillment thresholds...

In this paper we intend to discuss about how to use FSQL and a FRDB (Fuzzy Relational DataBase) for the management of a company. We will study the case of an estate agency, devoted to the sale and rent of apartments, flats, houses, chalets, semi-detached houses, building sites, coach houses, industrial plants... Other applications for our FSQL Server can be found, for example, in [2, 5].

First, we include a brief explanation of the main advantages of the FSQL `SELECT` sentence, in order to express fuzzy queries (a more detailed description of this and other FSQL sentences can be found in [3, 5]). Next, we will expose briefly the information that is stored in the database (fuzzy attributes types...). Finally, we will study how a FRDB can improve the information management system of an estate agency.

2 Flexible Queries with FSQL

The FSQL language [3, 4, 5] extends the SQL language to allow flexible queries. We have extended the `SELECT` command to express flexible queries and, due to its complex format, we only show here an abstract with the main extensions added to this command:

- **Linguistic Labels**: If an attribute is capable of fuzzy treatment then linguistic labels can be defined on it. These labels will be preceded with the symbol `$` to be easily distinguished. Every label has an associated trapezoidal possibility distribution (Figure 1) or there is a similarity relationship defined between each two labels in the same domain.

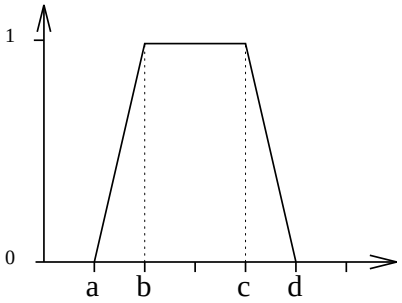

Figure 1: Trapezoidal possibility distribution.

- **Fuzzy Comparators**: In addition to the common comparators (=, >, etc), FSQL includes the fuzzy comparators of Table 1. The definition of these fuzzy comparators is shown in [4]. In the same way as in SQL, fuzzy comparators can compare one column value with one constant or two column values of the same type. Necessity comparators are more restrictive than possibility comparators are, i.e. their fulfillment degree is always lower than the fulfillment degree of their corresponding possibility comparator.

- **Fulfillment Thresholds** (γ): For each simple condition a fulfillment threshold may be established (default is 1) with the format: `<condition> THOLD` γ, indicating that the condition must be satisfied with a minimum degree $\gamma \in [0,1]$. The reserved word `THOLD` is optional and it can be substituted by a traditional crisp comparator (=, $\leq$, etc), modifying the meaning of the query. The word `THOLD` is equivalent to the use of the comparator $\geq$.

- `CDEG(<attribute>)` **function**: This function shows a column with the fulfillment degree of the condition of the query for a specific attribute, which is expressed in brackets as the argument. If logic operators appear, the calculation of the compatibility degree is carried out using the minimum T-norm and the maximum T-conorm, but the user may change these functions. If the argument of the `CDEG` function is an attribute, then the `CDEG` function uses only the conditions which include that attribute. We can use `CDEG(*)` to obtain the fulfillment degree of each tuple in the condition (considering all its attributes, not just one of them).

- **Character %**: It is similar to the character `*` of SQL but this one also includes the columns for the fulfillment degrees of the attributes in which they are relevant.

- **Fuzzy Constants**: In FSQL we can use all the fuzzy constants which appear in Table 2.

- **Condition with** `IS`: `<Fuzzy_Attribute> IS [NOT] (UNKNOWN|UNDEFINED|NULL)`

We have a painstaking **FSQL Server** [3, 5] to obtain the answers to FSQL queries. It is programmed in PL/SQL language for Oracle DBMS. The FSQL Server has been programmed to work with fuzzy databases (based on the GEFRED model [7]) and not only crisp databases.

3 Information stored in the database

The data can be classified in two categories:

- **Traditional Database**: It consists of all the data stored in the relations but with a special format in order to represent fuzzy attribute values. The *fuzzy attributes* are classified by the system in 3 types:
  - **Type 1**: These attributes are totally crisp (traditional), but they have some linguistic trapezoidal labels defined on them, which allow us to make the query conditions for these attributes more flexible. Besides, we can use all constants in Table 2 in the query conditions with these fuzzy attributes.
  - **Type 2**: These attributes admit crisp data as well as possibility distributions over an ordered underlying domain. With these attributes we can store and use all the constants shown in Table 2.
  - **Type 3**: These attributes are defined on a not ordered underlying domain, for instance the hair colour. On these attributes some labels are defined and on these labels a similarity relation has yet to be defined. With these attributes we can only use the fuzzy comparator `FEQ`, as they have no relation of order. Obviously, we cannot store or use the constants trapezium, interval and approximate value of Table 2.

- **Fuzzy Meta-knowledge Base** (FMB): It stores information about the FRDB in a relational format. It stores attributes which are capable to fuzzy treatment and it will store different information for each one of them, depending on their type: **a) Type 1**: In order to use crisp attributes in flexible queries we will only have to declare them as being a fuzzy attribute Type 1 and store the following data in the FMB: Trapezoidal linguistic labels, value for the margin of the approximate values (see Table 2) and minimum distance for two values considered *very* separated (used in comparators `MGT/NMGT` and `MLT/NMLT`). **b) Type 2**: They need to store the same data in the FMB as the fuzzy attributes Type 1. **c) Type 3**: They store in the FMB their linguistic labels, the similarity degree amongst themselves...

Table 1: Fuzzy Comparators for FSQL.

Possibility	Necessity	Significance
FEQ	NFEQ	Possibly/Necessarily Fuzzy EQual
FGT (FGEQ)	NFGT (NFGEQ)	Possibly/Necessarily Fuzzy Greater (or Equal) Than
FLT (FLEQ)	NFLT (NFLEQ)	Possibly/Necessarily Fuzzy Less (or Equal) Than
MGT (MLT)	NMGT (NMLT)	Possibly/Necessarily Much Greater (Less) Than

Table 2: Fuzzy constants that may be used in fuzzy comparisons of FSQL queries.

F. Constant	Significance
UNKNOWN	Unknown value but the attribute is applicable.
UNDEFINED	The attribute is not applicable or it is meaningless.
NULL	Total ignorance: We know nothing about it.
$[a,b,c,d]	Fuzzy trapezoid ($a \leq b \leq c \leq d$): See Figure 1.
$label	Linguistic Label: It may be a trapezoid or a scalar (defined in FMB).
[n,m]	Interval "Between n and m" (a=b=n and c=d=m).
#n	Fuzzy value "Approximately n" (b=c=n and n−a=d−n=margen).

4 Using FSQL in Management Applications

Of course, the database schema for an estate agency must include classic attributes (not only fuzzy attributes), like the customers' names, telephone numbers, addresses... In general, to give a greater versatility to the system, you can define as many fuzzy attributes Type 2 as you consider, instead of Type 1. However, it is important to take into account that the fuzzy attributes Type 2 require in general more quantity of space to be stored and more time to be processed. So, we must choose between **flexibility** (in the representation and fuzzy treatment) and **eficiency** (in space of storage and time of CPU).

Examples of fuzzy attributes **Type 1** are: number of rooms, number of toilets, price of the community, altitude of the floor... It can be observed that, in general, the values of the previous attributes are usually wellknown and without ambiguity.

In this particular case, most of the attributes are **Type 2**. Thus, the database is as flexible as possible. Among these are the sizes of the coach house, of the garden or of the land (where proceed), highlighting also the following ones:

- **Price:** Many times, the price is not fixed and the salesman (owner) establishes an approximate value.
- **Area (m^2):** Sometimes, it is difficult to access quickly to the title deed of the property or do an exact measurement of its surface. Therefore, the possibility to store approximate values was very interesant for the consulted estate agents.
- **Age:** Perhaps, it is difficult and in general unnecessary to know the exact age of the property, although it is tremendously useful to know its approximate age. So, we can store that a house is new, seminew, old or that it is approximately 8 years old, for example.

The considered fuzzy attributes **Type 3** are the following ones, having each one their corresponding similarity: lightness (sun), noise, sights, quality of the furniture (if furnished apartment) and so on:

- **District:** This attribute has been implemented with length 3, indicating that a landed property may be situated among 3 areas, with different degree. For example, {0.5/Center, 1/North, 0.7/Northwest} indicates that the property is situated in the North district, nearer to the Northwest district than to the town Center. The similarity relation between the different districts depend on the distance between them and on its extension.
- **Kind of landed property:** This attribute distinguishes among apartments, flats, chalets, houses, semi-detached houses, building sites, industrial plants... For example, it can be established that a chalet is similar to a semi-detached house in degree 0.8. A customer that looks for a chalet is a potential customer of the semi-detached houses. Thus, this is taken into account in order to show to our customer all the relevant properties.

With a database schema like this, the type of different queries that can be carried out are immense and among them it highlights the comparison among the relation of available properties and the relation of demands of properties. The first relation stores the available properties we can operate with (to sell, to rent...) and the relation of demands stores the general characteristics of the properties that are being looked for by the customers. Later, the relation of demands is matched with the other relation using fuzzy necessity comparators (Table 1) and thresholds strictly bigger than zero, and ranking the result decreasingly by the compatibility degree of every property. If the query retrieves too many properties then we can augment the

fulfillment threshold and, if the query retrieves too few properties then we can use possibility comparators instead of necessity ones, which are more restrictive.

Another possibility is to consult *on-line* the FRDB at the same time as the customer indicates his preferences. For example, let us suppose that a customer indicates: "I am looking for a big chalet with about 7 rooms and in the Northern area". Then, the following FSQL query retrieves the properties that comply with those conditions, ranking by the first attribute, which is the compatibility degree:

```
SELECT CDEG(*), Sales.* FROM Sales
WHERE Kind     FEQ  $Chalet .5
AND Surface    FGEQ $Big    .5
AND Rooms      FGEQ #7      .5
AND District FEQ  $North  .5 ORDER BY 1 DESC;
```

We have opted to use possibility comparators because quite a lot of elemental conditions are included, in order to select a greater quantity of properties. In the previous query, the semi-detached houses would also be retrieved if this `Kind` of property has a similarity degree greater or equal to 0.5 with regard to chalet. If we look exclusively for chalets we must establish the threshold to 1.

It is easy to note that the number of possible queries and the utility of its answers is tremendous. Thus, we will show first to each customer the property that has a greater compatibility degree. In the case that none of the retrieved properties satisfies the customer, we can make a more flexible query, putting down the thresholds (until 0), changing the fuzzy constants on the right of the simple conditions, changing fuzzy comparators, eliminating not very important conditions or exchanging some logical comparator `AND` by `OR`.

Naturally, the flexible query system does not assures the accomplishment of operations, but it assures that we find the property most accordant with the customer needs and tastes. It is necessary to take into account that when somebody looks for any type of property, he rarely has a fixed idea, but looks for something with some initial basic characteristics. It is frequent that what the customer acquires finally is not very similar to what he looked for at first.

Besides, this system allows the real estate agency to maintain a large database without having to remember the characteristics of the properties. This problem makes impossible to handle many properties effectively, and it is solved by our proposed system.

5 Conclusions

We have presented a discussion in which it remains clear how the fuzzy databases systems can provide a lot of advantages to the enterprises that use traditional databases. This is possible because we have a flexible query language, the FSQL language, which is very similar to SQL and therefore it is easy to learn and use. The FSQL language has a very similar syntax to standard SQL and, furthermore, it is easily installable on a system which uses Oracle as a DBMS. Of course, the FSQL queries may be implemented in a program in such a way that the user does not need to know the FSQL syntax. Furthermore, we are now working on the creation of a Visual FSQL Client program for Internet (in Java). Others applications for FSQL are shown in [2, 5, 6].

The power of this language, the great quantity of implemented fuzzy comparators, the flexibility to establish fulfillment thresholds, the possibility to be installed and used on traditional DBMS and other shown characteristics make that the advantages of FRDB are easily evaluated and therefore, this means an important step in the transfer of results of research to the business world.

References

[1] B.P. Buckles, F.E. Petry, "Extending the Fuzzy Database with Fuzzy Numbers". Information Sciences 34, pp. 45–55, 1984.

[2] R.A. Carrasco, J. Galindo, M.A. Vila, J.M. Medina, "Clustering and Fuzzy Classification in a Financial Data Mining Environment". 3th International ICSC Symposium on Soft Computing, SOCO'99, pp. 713–720, Genova (Italy), June 1999.

[3] J. Galindo, J.M. Medina, O. Pons, J.C. Cubero, "A Server for Fuzzy SQL Queries", in "Flexible Query Answering Systems", eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in A.I. (LNAI) 1495, pp. 164–174. Ed. Springer, 1998.

[4] J. Galindo, J.M. Medina, A. Vila, J.C. Cubero, "Fuzzy Comparators for Flexible Queries to Databases". Iberoamerican Conference on Artificial Intelligence, IBERAMIA'98, Lisbon (Portugal), 1998.

[5] J. Galindo, "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales". Ph. Doctoral Thesis, University of Granada (Spain), March 1999.

[6] J. Galindo, M.C. Aranda, "Gestión de una Agencia de Viajes usando Bases de Datos Difusas y FSQL". Turismo y tecnologías de la información, TuriTec'99, Málaga (Spain), September 1999.

[7] J.M. Medina, O. Pons, M.A. Vila, "GEFRED. A Generalized Model of Fuzzy Relational Data Bases". Information Sciences, 76(1–2), pp. 87–109, 1994.

[8] H. Prade, C. Testemale, "Fuzzy Relational Databases: Representational issues and Reduction Using Similarity Measures". J. Am. Soc. Information Sciences 38(2), pp. 118–126, 1987.

[9] M. Umano, S. Fukami, "Fuzzy Relational Algebra for Possibility-Distribution-Fuzzy-Relational Model of Fuzzy Data". Journal of Intelligent Inform. Systems, 3, pp. 7–28, 1994.

[10] M. Zemankova-Leech, A. Kandel, "Implementing Imprecision in Information Systems". Information Sciences, 37, pp. 107–141, 1985.