Mathware and Soft Computing 3 (1994) 285-295

An Elemental Processor of Fuzzy SQL*

J. M. Medina, O. Pons, M. A. Vila
Dept. Computer Sciences and Artificial Intelligence
E.T.S. de Ingeniería Informática.
Universidad de Granada 18071. Granada (Spain)
email: medina@robinson.ugr.es

Abstract

This paper reports an alternative for implementing an SQL fuzzy extension on a Fuzzy Relational Database System. This proposal tries to build an FSQL processor using representation and manipulation mechanisms offered by the RDBMS which operates as a host. For this purpose, we are based our work on the formulation of a theoretical model of Fuzzy Relational Databases, on the adoption of a scheme for the representation and implementation of fuzzy information on conventional RDBMS, and on the tools for the development of applications available in the host RDBMS. Finally, the FSQL processor is able to translate sentences formulated in FSQL into classical SQL sentences executed directly by the RDBMS adopted as host.

1 Introduction

Since the appearance of the Relational Database model, proposed in [5], several approaches have tried to provide a theoretical environment for the representation and handling of fuzzy information. These approaches, which are based on the use of fuzzy sets theory [23] as a tool of representation and manipulation, are grouped mainly into two tendencies: *Unification Models by Similarity Relations* and *Possibilistic Models*.

The first approach is described in [1, 2, 3, 4], with additional contributions in [16, 17, 18]. The second approach comprises several proposals. The most important of them were introduced by Umano [20], Prade and Testemale [14] and, Zemankova-Kandel [24].

In [9, 11, 12], we present a model, named GEFRED, which attempts to synthesize the most outstanding aspects of previous approaches within a common theoretical framework.

*This work has been financed by the CICYT charged to Project TIC94-1347

Once the theoretical bases for the fuzzy extension of relational databases have been established, they require mechanisms to build relational systems which operate in accordance with these principles. These systems, called generically Fuzzy Relational Database Management Systems (FRDBMS), require a theoretical model and some construction ideas which make possible to build an operative implementation. Most relevant proposals have been provided by the proponents of different theoretical models [6, 7, 14, 19, 24].

FRDBMS must be built following a scheme based on two basic requirements:

- A suitable theoretical support for a fuzzy relational database model.
- Representation and manipulation criteria of fuzzy information which favor system efficiency.

The first requirement is satisfied by adopting a model which is an extension of the classical relational model (which involves considering it as a particular case of the fuzzy case with precise data) and provides handling for an extensive range of fuzzy information.

The second requirement is referred to certain criteria which take the FRDBMS specification in order to permit an operative implementation.

In [10, 12] some ideas for broaching the building of an FRDBMSs according to both requirements are presented. The general scheme proposed in these works is summarized in the second section.

It is very important, when we are designing an FRDBMS, to use a language which comprises all aspects of the handling of fuzzy information, and preserves the original operations of classical relational algebra. Given the diffusion of SQL as an RDBMS manipulation language, it seems appropriate to broach an extension of it which contemplates these new possibilities.

This paper reports a proposal for an FSQL processor that is perfectly integrated into the scheme shown in the second section, and permits the translation of fuzzy SQL sentences into usual SQL sentences executed directly by the classical RDBMS. Sentences supported by this processor provide suitable mechanisms for expressing the algebra proposed in GEFRED [11].

In the section below, the structural framework necessary for the formulation of the proposed processor is propounded. Later we will describe the FSQL processor and how it operates in the scope of the FRDBMS described in the second section. To do this, we introduce a basic syntax consisting of fuzzy clauses and functions. We will complete the study showing an example of how the FSQL processor transforms a fuzzy sentence into a classical SQL sentence.

2 General Scheme of a FRDBMS

Figure 1 shows the general scheme of the FRDBMS on which the FSQL processor is developed. The initial idea is to build it on a conventional RDBMS, so all

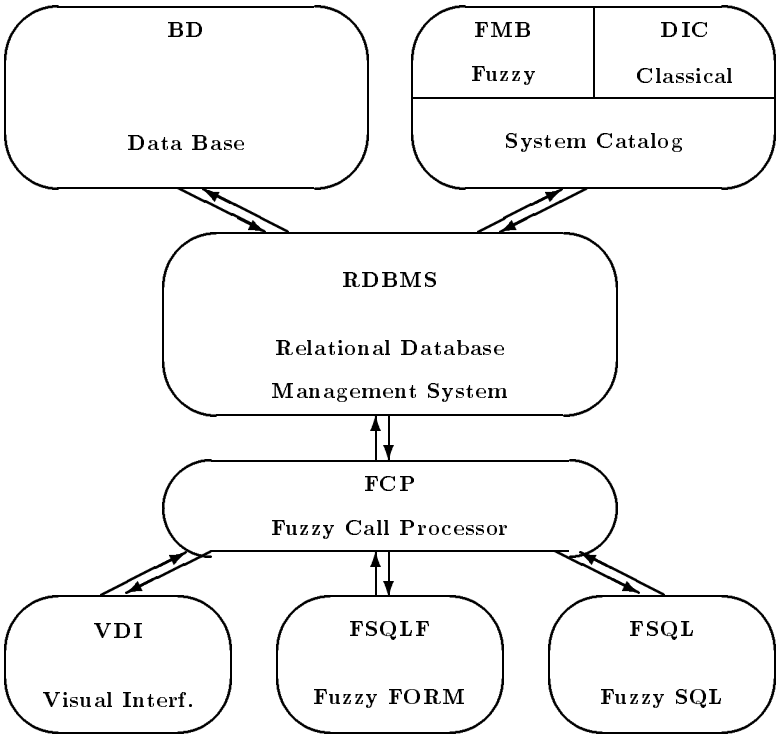

Figure 1: General Scheme of an FRDBMS

the developments take this RDBMS as the element to which all the requests are directed.

Next, we will look over the different modules which constitute the proposed scheme:

- **RDBMS**. As we have said above, all the operations conceived for the fuzzy extension will be translated into requests for the host RDBMS. These requests will be made using a language that the RDBMS supports. Generally, all the requests for the system are solved by classical SQL sentences. Finally, all the processing, whether fuzzy or not, will be translated into a series of requests for the system in SQL language.

- **BD**. The database comprises all the permanent information, whether fuzzy or not, in database tables. Data representation in these tables will depend on

their nature and type. We will use the resorts available in the host RDBMS to represent the required fuzzy information.

- **FMB**. The RDBMS dictionary or catalog represents the part of the system which stores information about the data collected in the database, and other information such as: users, data structure, data control, etc. In our case, we will prolong this part of the system in order for it to collect the necessary information related to the imprecise nature of the new collection of processing data. We will name this extension the Fuzzy Meta-knowledge Base (FMB), and it will be organized following the prevailing philosophy in the host RDBMS catalog.

- **FCP**. With our implementation we will try to transform the supporting group of "fuzzy" operations into a group of requests for the host RDBMS included in the group of instructions collected by it. The FCP mission consists in translating "fuzzy" operations formulated upon FRDBMS into ordinary requests for the host system. This translation takes into account the required "fuzzy" operation, the "fuzzy" item involved in this operation, their representation in the database, and the information the FMB has about them. Thus, the FCP has implemented the way to solve a "fuzzy" request in classical terms.

- **FSQL**. This module contemplates the "fuzzy" extension of SQL. It will be responsible for translating the syntax into operations upon the RDBMS by means of the FSQL processor using FCP routines.

These are the basic modules of the proposed FRDBMS.

With the introduction of a query and manipulation language like FSQL we have solved the problem of the FRDBMS interface with the user. From here, another kind of interface or query languages based on graphic interfaces, 4th Generation languages, etc. can be used.

3 Description of the FSQL Processor

As indicated previously, if we combine a suitable representation for fuzzy information with an adequate implementation of it through the data structure given by the host RDBMS, it is possible to produce processes which translate fuzzy operations into a group of sentences executed directly by a classical RDBMS. In this section we are going to illustrate some aspects of this process, based on a reduced version of an FSQL processor (PFSQL) built upon the Oracle^{TM} RDBMS [13]. The philosophy which supports this processor is to integrate a group of clauses and special functions which permit "fuzzy" requests to the system to be expressed in a certain SQL syntax. According to the proposed scheme, to be expressed in it will be possible to express sentences which involve "fuzzy" and classical clauses.

Next, we show an annotation similar to BNF of some of the new "fuzzy" clauses that FSQL syntax incorporates for DML:

```
fcond_simp      :    fcond_wtout threshold
                |    fcond_wtout
                ;
fcond_wtout     :    column_item fcomp fuz_constant
                ;
threshold       :    THOLD NUMBER
                |    THOLD '$' ID
                ;
fuz_constant    :    '$' ID
                |    NUMBER
                |    '#' NUMBER
                |    '[' NUMBER ',' NUMBER ']'
                |    '$' '[' NUMBER ',' NUMBER ','
                |    NUMBER ',' NUMBER ']'
                ;
fcomp           :    FEQ
                |    FGT
                |    FLT
                |    FGEQ
                |    FLEQ
                ;
comp_deg        :    CDEG '(' fuz_arith_op ')'
                ;
column_item     :    id_user '.' id_table '.' id_attribute
                |    id_table '.' id_attribute
                |    table_alias '.' id_attribute
                |    id_attribute
                ;
```

Nonterminal symbols `id_user`, `id_table`, `id_attribute` and `table_alias` identify the object user, table, attribute and alias respectively, and follow the formation rules established in the syntax of the host RDBMS. The terminal symbol `NUMBER` identifies a datum with numerical format; the semantic parser is responsible for verifying the range and type for each occurrence in the entry. Symbols in simple quotes, which belong to the production rule, are literals. The remaining terminal symbols are reserved words in the new syntax which permit "fuzzy" conditions, modifiers and functions to be expressed. The following phrases would be valid statements of the FSQL syntax adopted (assuming the existence of the tables and labels noted):

```
SELECT age, CDEG(age) FROM employees
WHERE age FEQ $young THOLD 0.8

SELECT a.emp#,name,education,CDEG(education)
```

```
FROM employees a, capacity b
WHERE a.emp#=b.emp# AND
education FEQ $graduate THOLD 0.6

SELECT emp#,dept#,job#,salary,
CDEG(salary),commission,CDEG(commission)
FROM jobs
WHERE salary FEQ $high
AND commission FEQ $low THOLD 0.8
```

The nonterminal symbol `fcomp` represents the set of fuzzy comparators in FSQL syntax. The complete syntax of an atomic fuzzy comparison takes the form:

```
<column_item> fcomp <fuz_constant> [<threshold>]
```

where square brackets indicate that the threshold can appear or not in the condition; in the latter case, the default threshold value equals 0.5 here.

Compound conditions are obtained through the use of NOT, AND and OR connectives, and fuzzy atomic conditions can be connected with other conditions which are not fuzzy.

The following syntax is used to visualize the degrees of compatibility that a given attribute or an expression of attributes presents:

```
CDEG(<fuz_arith_op>)
```

where `<fuz_arith_op>` identifies the attribute expression whose degrees of compatibility we want to present.

3.1 Functioning of the FSQL Processor

Figure 2 illustrates the processing of an FSQL sentence through an FSQL processor. This figure shows the modules involved in the processing of a sentence expressed in FSQL, from the entry to the obtaining of the results, if there are any, in the output.

1. For a presumed FSQL statement, the syntax parser verifies that syntax is correct.

2. Simultaneously, through the semantic parser the FSQLP is responsible for checking that all "fuzzy" items involved are correctly used. To do this, it uses the information about these elements present in the FMB.

3. While these processes are being performing, the FSQLP generates a data structure which organizes the entry information in two groups of processes:

Figure 2: Processing of an FSQL statement

- The group of classes present in the original statement, which do not require "fuzzy" treatment. No special operations will be performed on this group.
- The group of "fuzzy" clauses embedded in the entry statement.

From this latter group, the FSQLP takes from the FMB and the "fuzzy" clauses the related parameters that have been defined, and uses the routines of the FCL to generate the fragments of SQL statements corresponding to the function required by these cl auses. All information generated is properly organized by a data structure used for this purpose.

4. When previous steps have been performed, the FSQLP is able to reconstruct the final SQL statement from the available information. The sentence generated is directly executable by the host RDBMS.

5. Finally, this sentence, and if necessary the results obtained from its execution, are sent to the RDBMS in a previously established format.

The steps described above include a mechanism for error detection which works at several levels. At the syntax parser level, the PFSQL detects syntax errors in the entry. Semantic errors are first localized when they arise from the wrong use of "fuzzy" items. During the process of execution by the host RDBMS, the rest of the semantic errors will be detected. In addition, the host manager detects possible execution errors, and communicates them through the appropiate error code.

4 Example of Query Processing by the PFSQL

NAME	ADDRESS	AGE	PERFORMANCE	SALARY
Luis	Recogidas	31	Good	High
Antonio	Reyes Católicos	Middle	Fair	100000
Juan Carlos	Camino Ronda	Young	Bad	90000
Francisco	P. A. Alarcón	Old	Excellent	Low
Julia	Puerta Real	Young	Good	Medium
Inés	Manuel de Falla	#28	Good	125000
Javier	Gran Vía	*30,35	Fair	105000

The symbol # means "approximately", and * denotes an intervalar value.

Table 1: EMPLOYEES

To illustrate the functioning of the PFSQL, we will use the example shown in Table 1, which consists of a relation that collects data, some of a fuzzy nature, about a set of employees.

$s_e(d,d')$	Bad	Fair	Good	Excellent
Bad	1	0.8	0.5	0.1
Fair	0.8	1	0.7	0.5
Good	0.5	0.7	1	0.8
Excellent	0.1	0.5	0.8	1

Table 2: Proximity Relation on PERFORMANCE.

The NAME and ADDRESS attributes contain "crisp" information, the first attribute being the primary key of the relation. The information they contain is implemented in the Database in the same way as in the host RDBMS, since it will not receive "fuzzy" treatment. AGE and SALARY attributes store "fuzzy" information. The following linguistic labels are defined in AGE: *"Young"*, given by a trapezoidal distribution [16,30,16,10], *"Middle"*, given by [35,45,10,10], and *"Old"*, given by [50,6 5,10,15]. In addition, the concept *"approximately n"* is represented by the distribution [n-5,n,n,n+5]. The labels defined in the SALARY attribute domain are: *"Low"*, represented as [65000,85000,65000,10000], *"Medium"*, as [95000,110000,1 0000,20000], and "High", given by [130000,180000,20000,820000].

The PERFORMANCE attribute accepts "fuzzy" information, but over a scalar domain. To formulate a query which contains this attribute, we need to define a "proximity relation" in its domain. This relation is shown in Table 2.

By way of example, we will explain how the following query can be solved:

"Give me the NAME, ADDRESS, AGE, and the degree to which the condition for the AGE attribute is satisfied, for those employees with a "middle" AGE (with a degree $\geq$ 0.8)"

In terms of the FSQL syntax, this query is formulated as:

```
SELECT name, address, age, CDEG(age) FROM employees
WHERE age FEQ $middle THOLD 0.8
```

From the available information, the PFSQL composes the following classical SQL sentence:

```
select name,address,DECODE(AGET,0,'UNKNOWN',1,'UNDEFINED',2,'NULL',
3,TO_CHAR(AGE1),4,DECODE(AGE1,0,'JOUNG',1,'JOUNG',2,'OLD'),5,'['||
TO_CHAR(AGE1)||',',||TO_CHAR(AGE4)||']',6,'['||TO_CHAR(AGE1)||',',||
TO_CHAR(AGE1+AGE2)||',',||TO_CHAR(AGE4)||']',7,'['||TO_CHAR(AGE1)
||',',||TO_CHAR(AGE2+AGE1)||',',||TO_CHAR(AGE3+AGE4)||',',||TO_CHAR
(AGE4)||']')"AGE",ROUND(DECODE(AGET,0,1,2,1,3,DECODE(SIGN((AGE1-
33)*(47-AGE1)),-1,0,DECODE(SIGN(35-AGE1),1,(AGE1-25)/10,DECODE(
SIGN(AGE1-45),1,(AGE1-55)/-10,1))),4,DECODE(AGE1,1,1.00,0),5,
DECODE(SIGN((AGE4-33)*(47-AGE1)),-1,0,DECODE(SIGN(35-AGE4),1,
(AGE4-25)/10,DECODE(SIGN(AGE1-45),1,(AGE1-55)/-10,1))),6,
DECODE(SIGN((AGE4-37)*(53-AGE4)),-1,0,DECODE(SIGN(35-AGE4+5),
1,(AGE4-25)/(10+5),DECODE(SIGN(45-AGE4+5),1,1,(AGE1-55)/(-10
-5)))),7,DECODE(SIGN((0.80*AGE3+AGE4-33)*(47-0.80*AGE2+AGE1)),
-1,0,DECODE(SIGN((AGE3+AGE4-35)*(45-AGE2+AGE1)),-1,DECODE(SIGN
((0.80*AGE3+AGE4-33)*(35-AGE3+AGE4)),-1,DECODE(SIGN((47-0.80*
AGE2+AGE1)*(AGE2+AGE1-45)),1,(55-AGE1)/(AGE2--10)),(AGE4-25)/
(10-AGE3)),1)),0),2) "CDEG(AGE)" from employees where (AGET=0
OR AGET=2 OR AGET=3 AND AGE1 BETWEEN 33 AND 47 OR AGET=4 AND
AGE1 IN (1) OR AGET=5 AND AGE1<=47 AND AGE4>=33 OR AGET=6 AND
AGE4 BETWEEN 37 AND 53 OR AGET=7 AND AGE4>=33 AND AGE1<=47
AND 0.80*AGE3+AGE4>=33 AND 0.80*AGE2+AGE1<=47)
```

Table 3 shows the results of the execution of this query on the host RDBMS (this sentence and its execution were made using a FIRST prototype developed on the Oracle V6 RDBMS).

NAME	ADDRESS	AGE	CDEG(AGE)
Antonio	Reyes Catolicos	Middle	1
Javier	Gran Via	[30,35]	1

Table 3: Results of the Example Query

5 Conclusions

In this work we have been presented the scheme of a module which for translating sentences expressed in FSQL into standard SQL expressions. This module is integrated into a scheme of FRDBMS built over the theoretical concepts formulated in GE FRED [11]. Thanks to this system, it is possible to develop "fuzzy" extensions of classical RDBMS that are available at present, thus preserving the investments made in these systems and enhancing their characteristics of representation and handling of information.

Future work will address the specification and development of a complete prototype of an FSQL processor. We also plan to undertake the theoretical study of fuzzy information processing in other Database models, such as Object Oriented ones.

References

[1] Anvari M., Rose G.F. "Fuzzy Relational Databases". Analysis of Fuzzy Information. Bezdek ed. Vol II CRC Press. (1987)

[2] Buckles B.P., Petry F.E. "A Fuzzy Representation of Data for Relational Databases". *Fuzzy Sets and Systems,* **7**. 213-226. (1982)

[3] Buckles B.P., Petry F.E. "Extending the Fuzzy Database with Fuzzy Numbers". *Information Sciences*, **34**. 145-155. (1984)

[4] Buckles B.P., Petry F.E., Sachar H.S. "A Domain Calculus for Fuzzy Relational Databases". *Fuzzy Sets and Systems*, **29**. 327-340. (1989)

[5] Codd E.F. "A Relational Model of Data for Large Shared Data Banks". Commun. ACM, 13. (6). pp. 377-387 (1970)

[6] Hamon Guy. "Extension d'un langage d'interrogation de Base de Données en vue de l'utilisation de questions imprécises", Ph.D. Thesis, (1986).

[7] Li D., Liu. "A Fuzzy Prolog Database System". ed. John Wiley & Sons. New York. (1990)

[8] Medina J. M., Vila M.A., "Un Modelo de Bases de Datos Difuso Aplicado a Información Médica". 1^{er} Congreso Español sobre Tecnologías y Lógica Fuzzy. Granada. (1991)

[9] Medina J. M., Pons O., Vila M.A., "GEFRED. Un Modelo Generalizado de Bases de Datos Relacionales Difusas". 2º Congreso Español sobre Tecnologías y Lógica Fuzzy. Madrid.(1991)

[10] Medina J. M., Vila M.A., Cubero J.C., Pons O. "Towards the Implementation of a Generalized Fuzzy Relational Database Model". To appear in *Fuzzy Sets & Systems.*

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

[12] Medina J. M. "Bases de Datos Relacionales Difusas: Modelo Teórico y Aspectos de su Implementación". Tesis Doctoral. Universidad de Granada. (1994)

[13] Oracle RDBMS. "SQL Language Reference Manual vs. 6.0". (1990).

[14] Prade H., Testemale C. "Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries". *Information Sciences*, **34**. 115-143. (1984)

[15] Prade H., Testemale C. "Representation of Soft Constraints and Fuzzy Attribute Values by means of Possibility Distributions in Databases". Bezdek ed. Analysis of Fuzzy Information. Vol II CRC Press. (1987)

[16] Rundensteiner E.A., Hawkes L. W., Bandler W. "On Nearness Measures in Fuzzy Relational Data Models". *International Journal of Approximate Reasoning.* 3. pp 267-298. (1989)

[17] Shenoi S., Melton A. "Proximity Relations in the Fuzzy Relational Database Model". *Fuzzy Sets and Systems,* **31**. 285-296. (1989)

[18] Shenoi S., Melton A. "An Extended Version of the Fuzzy Relational Database Model". *Information Sciences*, **52**. 35-52. (1990)

[19] Umano M., Mizumoto M. and Tanaka. "FSTDS System: A Fuzzy-Set Manipulation System", *Information Sciences.* **14**, pp. 115-159 (1978).

[20] Umano M. "Freedom−0 : *A* Fuzzy Database System". Fuzzy Information and Decision Processes. Gupta-Sanchez edit. North-Holland Pub. Comp. (1982)

[21] Vila M.A., Cubero J.C., Medina J. M., Pons O. "A Logic Approach to Fuzzy Relational Databases", IPMU'92 (1992).

[22] Zadeh L.A. "Similarity Relations and Fuzzy Orderings". *Information Sciences,* **3**, 177-200. (1971)

[23] Zadeh L.A. "Fuzzy Sets as a Basis for a Theory of Possibility". *Fuzzy Sets and Systems,* **1**, 3-28. (1978)

[24] Zemankova M., Kandel A. "Fuzzy Relational Data Bases - A Key to Expert Systems". Verlag TUV Rheinland. (1984)