New Architecture of Fuzzy Database Management Systems

Amel Grissa Touzi and Mohamed Ali Ben Hassine
Faculty of Sciences of Tunis, Tunisia University, Tunisia

Abstract: *Fuzzy relational data bases have been extensively studied in a theoretical level. Unfortunately, the repercussions of these works on the practical plan are negligible. Medina et al. have developed a server named fuzzy SQL, supporting flexible queries and based on a theoretic model called GEFRED. This server has been programmed in PL/SQL language under Oracle database management systems. To model the flexible queries and the concept of fuzzy attributes, an extension of the SQL language named fuzzy SQL has been defined. The FSQL language extends the SQL language, to support the flexible queries, with many fuzzy concepts. The FRDB is supposed has already been defined by the user. In this paper, we extend the work of medina et al. to present a new architecture of fuzzy DBMS based on the GEFRED model. This architecture is based on the concept of weak coupling with the DBMS Oracle. It permits, in particular, the description, the manipulation and the interrogation of FRDB in FSQL language.*

Keywords: *Fuzzy DB, FSQL, FIRST, FSQL serve, GEFRED.*

Received March 18, 2007; accepted May 30, 2007

1. Introduction

The research area of fuzziness in Data Base Management Systems (DBMS) has resulted in a number of models aimed at the representation of imperfect information in DataBases (DB), or at enabling non-precise queries (often called flexible queries) on conventional database schemas [1, 2].

However, few works have been done from a practical point of view. The majority of these works used the fuzzy sets formalism to model the linguistic terms as "moderate", "means" and to value the predicates including such terms. The essential idea in these works consists in extending the SQL language and adding a supplementary layer to the relational DBMS to evaluate the fuzzy predicates [3].

In this paper, we are interested to the works of Medina *et al*. who introduced the GEFRED model [14] in 1994 and its associated language named FSQL [5, 6, 7, 13]. This language presents new concepts such as fuzzy comparators, fuzzy attributes, fuzzy constants, *etc.* The use of this language is through a software named Fuzzy Query (FQ) [10]. Even though it solved several problems related to the flexible queries modeling, FQ presents several limits: (1) it allows only the flexible querying of FRDB, (2) the FRDB is supposed already implemented under Oracle, (3) the implementation of the DB is made manually by the user, (4) FQ is not suitable in practice for FRDB made up of more than ten tables.

In this paper, we propose a new architecture of the Fuzzy Relational DBMS (FRDBMS) based on the GEFRED model. This architecture is based on the weak coupling principle with the RDBMS Oracle. This FRDBMS offers all functionalities of a classic DBMS, in particular the description, the manipulation and the querying of FRDB.

Besides this introduction, this paper includes five sections. Section 2 presents the basic concepts of FRDB. Section 3 presents the architectures already used for the flexible querying modeling. Section 4 presents the architecture type of FRDBMS. Section 5 presents our new architecture of the FRDBMS as well as its implementation. Section 6 makes an evaluation of this work and gives some future perspectives of it.

2. Basic Concepts

In this section, we present the basis concepts of FRDB.

2.1. Definitions

A FRDB is an extension of the relational database. This extension introduces fuzzy predicates under shapes of linguistic expressions that, at the time of a flexible querying, permits to have a range of answers (each one with a membership degree) in order to offer to the user all intermediate variations between the completely satisfactory answers and those completely dissatisfactory [2].

A FRDBMS is an extension of the relational DBMS in order to treat, store and interrogate imprecise data. The FRDB models are considered in a very simple shape and consist in adding a degree, usually in the interval [0, 1], to every tuple. It allows maintaining the homogeneity of the data in DB. The main models are

those of Prade-Testemale [16], Umano-Fukami [17], Buckles-Petry [4], Zemankova-Kaendel [19] and GEFRED of Medina *et al.* [14]. This last model constitutes an eclectic synthesis of the various models published so far with the aim of dealing with the problem of representation and treatment of fuzzy information by using relational DB.

2.2. The GEFRED Model

The GEneralised model Fuzzy heart Relational Database (GEFRED) has been proposed in 1994 by Medina *et al*. [14]. One of the major advantages of this model is that it consists of a general abstraction that allows for the use of various approaches, regardless of how different they might look. In fact, it is based on the generalized fuzzy domain and the generalized fuzzy relation, which include respectively classic domains and classic relations. The data types supported by this model are showed in the Table 1.

Table 1. Data types in the GEFRED model.

1. A single scalar (e.g., Behavior=good, represented by the possibility of distribution 1/good).
2. A single number (e.g., Age=28, represented by the possibility of distribution 1/28).
3. A set of mutually exclusive possible scalar assignations (e.g., Behavior={Bad, Good}, represented by {1=Bad, 1=Good}).
4. A set of mutually exclusive possible numeric assignations (e.g., Age={20, 21}, represented by {1/20, 1/21}).
5. A possibility distribution in a scalar domain (e.g., Behavior={0.6/Bad, 1.0/Regular}).
6. A possibility distribution in a numeric domain (e.g. Age={0.4/23, 1.0/24, 0.8/25}, fuzzy numbers or linguistic labels).
7. A real number belonging to [0, 1], referring to the degree of matching (e.g., Quality=0.9).
8. An Unknown value with possibility distribution Unknown={1/u: u ∈U} on domain U, considered.
9. An Undefined value with possibility distribution Undefined={0/u: u ∈U} on domain U, considered.
10. A NULL value given by NULL={1/Unknown, 1/Undefined}.

2.2.1. Fuzzy Attributes in GEFRED Model

In order to model fuzzy attributes we distinguish between two classes of fuzzy attributes: Fuzzy attributes whose fuzzy values are fuzzy sets and fuzzy attributes whose values are fuzzy degrees [9, 11].

A. Fuzzy Sets as Fuzzy Values

These fuzzy attributes may be classified in four data types. This classification is performed taking into account the type of referential or underlying domain. In all of them the values unknown, undefined, and null are included:

- Fuzzy Attributes Type 1 (FTYPE1): these are attributes with "precise data", classic or crisp (traditional, with no imprecision). However, they can have linguistic labels defined over them, which allow us to make the query conditions for these attributes more flexible.
- Fuzzy Attributes Type 2 (FTYPE2): these attributes admit both crisp and fuzzy data, in the form of possibility distributions over an underlying ordered domain (fuzzy sets). It is an extension of the Type1 that does, now, allow the storage of imprecise information, such as: "he is approximately 2 meters tall". For the sake of simplicity the most complex of these fuzzy sets are supposed to be a trapezoidal function Figure 1. Table 2 shows the kinds of values defined in these attributes.
- Fuzzy Attributes Type 3 (FTYPE3): they are attributes over "data of discreet non-ordered dominion with analogy". In these attributes some labels are defined ("blond", "red", "brown", *etc*.) that are scalars with a similarity (or proximity) relationship defined over them, so that this relationship indicates to what extent each pair of labels be similar to each other.
- Fuzzy Attributes Type 4 (FTYPE4): these attributes are defined in the same way as Type 3 attributes, without it being necessary for a similarity relationship to exist between the labels.

B. Fuzzy Degrees as Fuzzy Values

The domain of these degrees can be found in the interval [0, 1], although other values are also permitted, such as a possibility distribution (usually over this unit interval) [11]. The meaning of these degrees is varied and depends on their use. The most important possible meanings of the degrees used by some authors are: fulfillment degree, Uncertainty degree, Possibility degree and Importance degree. The ways of using these fuzzy degrees are classified in two families: associated degrees (type 5, type 6, type 7) and non-associated degrees (type 8) [9].

2.2.2. Representation of Fuzzy Attributes

This representation is different according to the fuzzy attributet [8, 11]. Fuzzy attributes type 1 are represented as usual attributes, because they do not allow fuzzy values. Fuzzy attributes type 2 need five classic attributes Table 2.

- FT: stores the kind of value which the attribute in question can take (0 for UNKNOWN, 1 for UNDEFINED, *etc*). The letter T is concatenated the name of the attribute.
- F1, F2, F3 et F4 : stores the description of the parameters which define the data and which depend on the type of value (FT), the name of these attributes is formed by the concatenation of numbers 1, 2, 3 and 4 in the name of the attribute to which they belong.

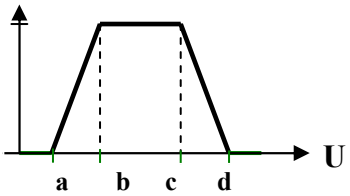

Figure 1. Trapezoidal linear and normalized distribution function.

Table 2. Kind of values of fuzzy attributes type 2.

Kind of values	Attributes in the DB for each fuzzy attribute Type 2				
	FT	F1	F2	F3	F4
UNKNOWN	0	NULL	NULL	NULL	NULL
UNDEFINED	1	NULL	NULL	NULL	NULL
NULL	2	NULL	NULL	NULL	NULL
CRISP	3	d	NULL	NULL	NULL
LABEL	4	FUZZY_ID	NULL	NULL	NULL
INTERVAL	5	n	NULL	NULL	m
APPROXIMATELY	6	d	d-marge	d+marge	marge
TRAPEZE [α,β,γ,δ]	7	α	β	γ	δ
APPROXIMATE VALUE $d \pm m$	8	d	$d - m$	$d + m$	m
POSSIBILITY DISTRIBUTION-2	9	p1	d1	p2	d2
POSSIBILITY DISTRIBUTION-4	10	d1	d2	d3	d4

The fuzzy attributes type 3 is represented by a variable number of traditional attributes according to the form described in Table 3.

- FT: is similar to FT used in FTYPE2 attribute.
- (FP_1, F_1),…, (FP_n,F_n): in these attributes, we store data of the distribution of possibility. For example, in a value of the SIMPLE type, only first couple is used and value of possibility will be 1 (to be standardized).

Table 3. Kind of values of fuzzy attributes type 3.

Kind of Values	Attributes of the DB for Every Attribute Type 3					
	FT	FP1	F1		FPn	Fn
UNKNOWN	0	NULL	NULL	…	NULL	NULL
UNDEFINED	1	NULL	NULL	…	NULL	NULL
NULL	2	NULL	NULL	…	NULL	NULL
SIMPLE	3	p	d	…	NULL	NULL
POSS. DISTR.	4	p1	d1	…	pn	dn

Fuzzy attributes type 4 is represented just like type 3. The different between them is shown in the next section. Fuzzy degrees (types 5, 6, 7 and 8) are represented using a classic numeric attribute, because their domain is the interval [0, 1].

2.3. The FSQL Language

The FSQL language is an authentic extension of SQL language to model fuzzy queries. It means that all the valid statements in SQL are also valid in FSQL [5, 11, 13]. The SELECT command is extended to express flexible queries and, due to its complex format [9], we only show an abstract with the main extensions added to this command:

- Linguistic Labels: if an attribute is able of fuzzy treatment then linguistic labels can be defined on it. These labels will be preceded with the symbol $ to distinguish them easily. Every label has an associated trapezoidal possibility distribution as shown in Figure 1 (for fuzzy attributes type 1 and 2) or a scalar (for fuzzy attributes type 3 and 4).
- Fuzzy Comparators: besides the typical comparators (=,>, *etc*.), FSQL includes fuzzy comparators. The definition of all these comparators is presented in [11].
- Function CDEG: the function CDEG (compatibility degree) may be used with an attribute in the argument to compute. It computes the fulfillment degree of the condition of the query for the attribute mentioned in its argument.
- Fulfillment Thresholds: for each simple condition, a fulfillment threshold τ may be established (default is 1) with the format: <condition> THOLD τ indicating that the condition must be satisfied with minimum degree $\tau \in [0,1]$ to be considered.
- Fuzzy constants: besides the typical constants (numbers, NULL…), FSQL included many constants such as fuzzy trapezoidal $[a,b,c,d], approximate values #n, $label, [n,m], UNKNOWN, UNDEFINED, *etc*.
- Fuzzy Quantifiers: there are of two types: absolute and relative. They allow us to use expressions like "most", "almost all", "many", "very few", *etc*.

Example:

"Give me all persons with fair hair (in minimum degree 0.5) that are possibly taller than label $Tall (with a high degree)". This query is modeled in FSQL language as follows:

```
SELECT * FROM Person
    WHERE Hair FEQ $Fair THOLD 0.5
AND Height FGT $Tall THOLD $High;
```

3. Proposed Architectures for the Flexible Querying Modeling

In this section, we present the different architectures proposed to model the flexible queries.

3.1. The Architecture Proposed by Bosc for the Flexible Querying Modeling

The approach proposed by Bosc Figure 2 consists in using the capacities of the commercial DBMS (in particular their mechanisms of optimization) while adding a supplementary layer assuring the interface between flexible queries and boolean queries [2, 3]. As Figure 2 shows the fuzzy query process is done by a transformation procedure located on top of the existing DBMS. The translation mechanism generates a procedural evaluation program and determines the expressions which are used to compute the membership degrees. The program processes the SQL queries which are derived from the SQLf query [1], computes the degrees and separate if necessary the n-uplets whose degree is lower to the fixed λ threshold.

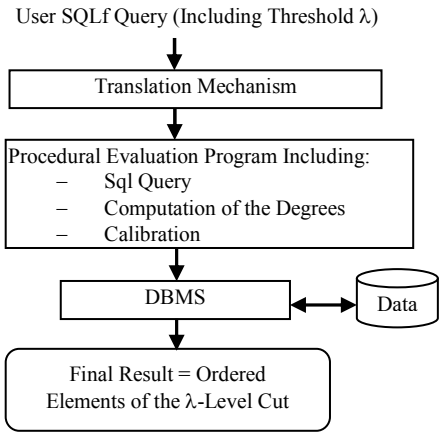

Figure 2. The architecture proposed by Bosc.

3.2. The Architecture Proposed by Medina

In order to implement a system which represent and manipulate "imprecise" information, Medina *et al.* have developed FIRST architecture (a fuzzy Interface for relational systems) [15] which have been enhanced with FIRST-2 [9]. It is built on RDBMS Client-Server architecture provided by Oracle. It extends the existing structure and adds new components to handle fuzzy information. The main important component added to this architecture is the FSQL Server which assures the translation of flexible queries written in FSQL in a comprehensible language by the DBMS (SQL). The schema of the architecture is showed in Figure 3.

In this architecture, Medina uses also the DBMS while adding a server assuring the translation of flexible queries in a comprehensible language by this DBMS (SQL). This server, named FSQL server [5], is based on a theoretical model named GEFRED. It has been developed in PL/SQL.

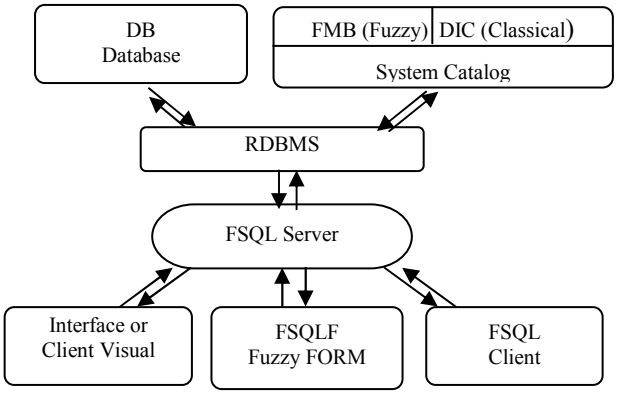

Figure 3. General architecture of FSQL server.

To model the flexible and the concept of fuzzy attributes, an extension of the SQL language named FSQL has been defined. Since the server is implemented for the Oracle DBMS, which only supports SQL and PL/SQL languages, it is natural that all extensions made in FSQL language must be supported directly by Oracle. For this reason, Medina *et al.* defined a meta base named Fuzzy Meta knowledge Base (FMB) [15] formed by a set of tables which extend the RDBMS dictionary or catalog in order to store all necessary information to describe and to manipulate fuzzy attributes. Figure 4 shows the relations in the FMB, its attributes, its primary keys (underlined) and its foreign keys (with the arrows).

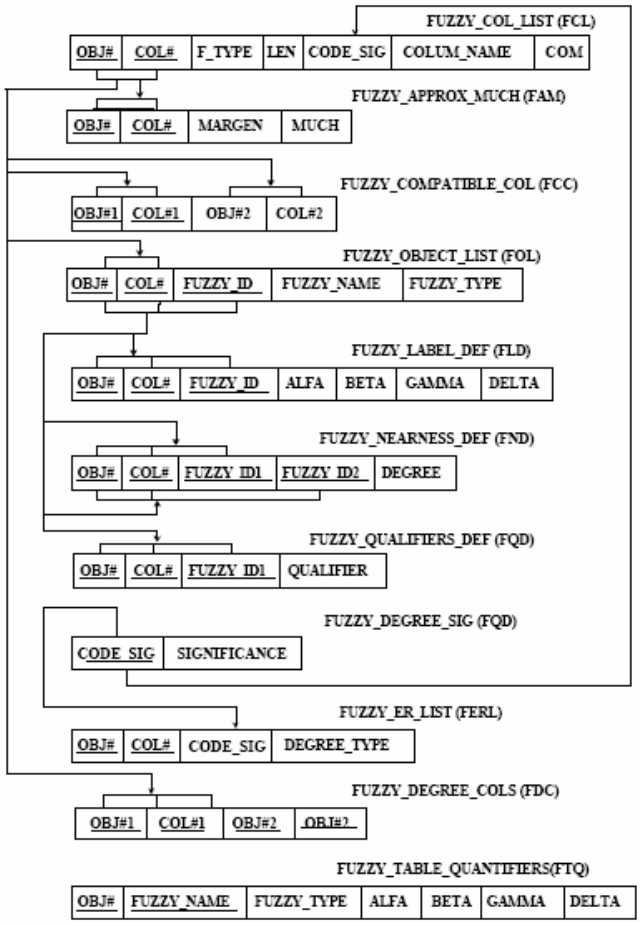

Figure 4. Relations in the FMB.

3.3. Limits of the Existing Works

As we presented in the previous section, the solutions currently proposed are restricted to the modeling of the flexible queries in the RDB. The FRDB is already supposed manually implemented by the user. It returns these solutions limited to some simple examples of FRDB and to academic uses. We propose in the continuation a new approach that allows the user to descript and manipulate the FRDB directly with FSQL language.

4. Architecture Type of an FRDBMS

An FRDBMS is considered first of all a DBMS. It must assure the following functions: (1) the description of the data, assured through the intermediary of a Data Description Language (DDL), (2) The manipulation of the data, assured through the intermediary of a Data Manipulation Language (DML), (3) The integrity maintenance of the FRDB,

assured by the definition of integrity rules, (4) The confidentiality, assured by the verification of the access rights, (5) The management of the competition of access, (6) The security of working in case of breakdown, and (7) the use help.

On another side, an FRDMS must (1) be capable to represent the fuzzy information in all its shapes, (2) offer an adequate setting to store and to represent the significance of this information, and (3) provide a minimum set of operators to recover and to treat the fuzzy data. An FRDMS must be made up of the core of DBMS permitting to store, manipulate fuzzy attributes and execute the classic operations of the DBMS. This architecture is illustrated in the Figure 5.

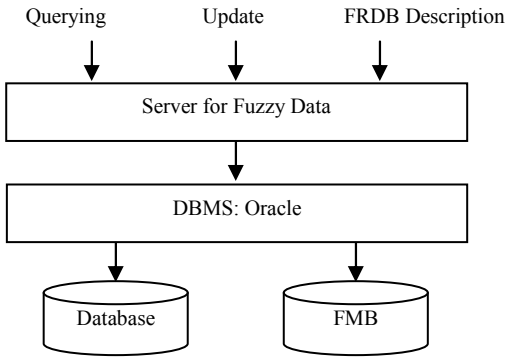

Figure 5. FRDBMS architecture.

Two possible solutions to implement an FRDMS: (1) develop a specific FRDMS to evaluate the queries written in FSQL, by analogy with the strategy put in work in the usual DBMS, but the development cost would risk to be prohibitive, (2) use the capacities of the commercial DBMS (in particular their mechanisms of optimization) while attaching a software layer that permits to support the fuzzy concept.

The last solution, characterized by its easiness realization, consists in cooperating the FSQL server and the DBMS. The FSQL Server translates the fuzzy queries written in FSQL language while looking in the information stored in the FMB. Once this phase is finished, the DBMS manages the crisp data translated by the FSQL Server with a transparent way. We speak then about weak "Coupling" Figure 6 or strong Coupling Figure 7.

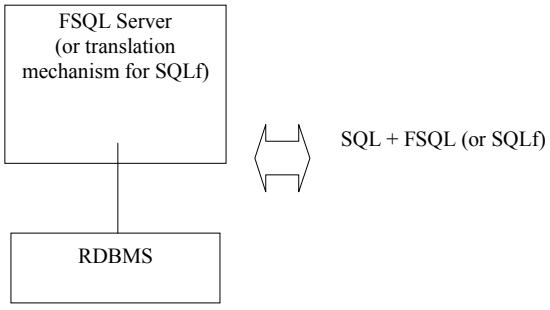

Figure 6. Weak coupling.

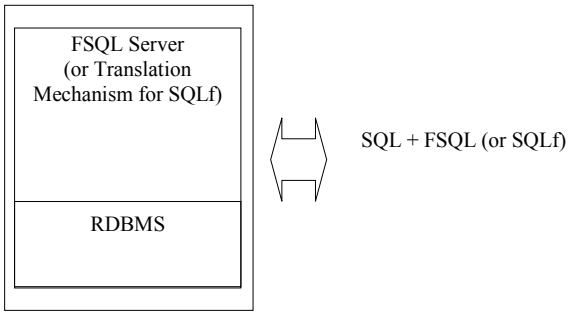

Figure 7. Strong coupling.

5. New Architecture of the FRDBMS

We propose the weak coupling approach with Oracle DBMS. The FRDBMS proposed respects the GEFERD model. The language of description and manipulation of the data is therefore FSQL. Seen that the FSQL language is an extension of the SQL language, a FRDBMS can model a RDB (described in SQL language) or a FRDB (described in FSQL language). The principle of this coupling is the definition of a software layer that allows the transformation of the command written by the user in FSQL language in their equivalent written in SQL. This principle is illustrated in the Figure 8.

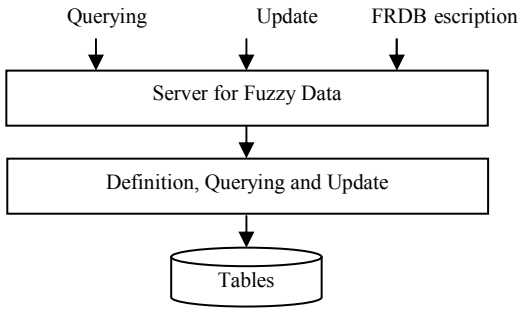

Figure 8. Illustration of the FRDBMS architecture.

5.1. The FRDMS Architecture

The installation of this architecture is described in Figure 9.

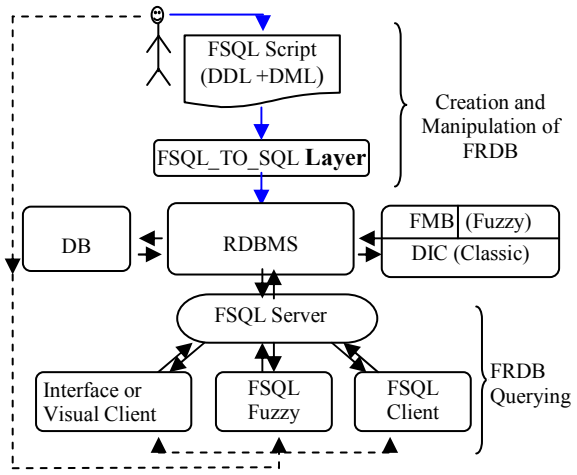

Figure 9. Architecture of extended FIRST.

5.2. Presentation of the FSQL to SQL Layer

This layer presents a tool, called FSQL_TO_SQL that permits the implementation of a FRDB described in FSQL under DBMS Oracle 8i. This tool gives the automatically transformations of the FSQL script in a script equivalent SQL while specifying the modifications that made to the level of the FMB.

5.2.1. Principle of Functioning

The principle of functioning is as follows:

Begin
Cut a DDL instruction in several lines containing each one an attribute.
Study every DDL line:
If it contains a classic attribute then
copy this line in a first file (result1.sql)
Else
Make a very specific treatment to every fuzzy attribute. This treatment divides in two under-treatments:
a) Treatment in DB: translate the command that concerns the DB and copy it in the file (result1.sql)
b) Treatment in FMB: provide a script writes in the file (result2.sql), containing the formation in the FMB concerning the fuzzy attributes and the objects defined on them.

End

As result of these two treatments, we get two files. a file containing the DDL part of the DB and a second one containing the modification be done in the FMB. We can also regroup the two treatments in a same file since they will be executed in the DBMS. After a detailed study of the GEFRED model and the FQ software, we defined a set of rules, described in [12], and which we must apply to achieve the transformation according to the fuzzy attributes types. In fact, the treatment in the DB and in the FMB depends to the FSQL command and to the fuzzy attributes type.

5.2.2. Example of Translation of Some Commands

In this example, we present the translation of the principle commands:

- Translation of the command CREATE TABLE, the modeling of this command will be effectuated while respecting the following steps:

1. Call the classic command CREATE TABLE with a modification of the fields containing the fuzzy attributes.
2. Insert in the FMB the tuples containing the information about the fuzzy attributes defined on this table.
3. Execute the different commands relative to the type of the fuzzy attribute (CREATE LABEL for FTYPE1 and FTYPE2, CREATE NEARNESS for FTYPE3, *etc.*)

- Translation of the command CREATE LABEL.

1. Insert in the FOL table the identifier of the linguistic label, its name and its type.
2. Insert in the FLD table the parameters of linguistics label (a, b, c, d), *etc*.

- Translation of the command CREATE NEARNESS

1. Insert in the FOL table the identifier of the linguistic label, its name and its type.
2. Insert in the FND table the list it of the linguistic labels with their similarity degrees.

We present now a simplified Algorithm of translation of script from FSQL to SQL:

Input: Source FSQL Script (SFS)

Output: Target SQL Script (TSS)

Begin
To create the FMB tables.
Foreach attribute A of SFS do
If A remains classic then
no modification in its definition;
Else / This treatment is divided in two under-treatments: in the DB and in the FMB */*
Modify the tables structure of the DB according to the fuzzy attribute representation.
Switch (type of A)
Case FTYPE1:
A remains unchanged.
Case FTYPE2:
Create 5 attributes with the same name of A;
Concatenate the first one with the letter 'T' (AT);
Concatenate the others ones respectively with 1, 2, 3 and 4 (A1, ... , A4);
Case FTYPE3 and FTYPE4:
Create 2n+1 attributes with the same name of A ;
/ n=maximum number of data for the values of A, (by default n=1) */*
Concatenate the first one with the letter 'T' (AT);
Foreach pair of the remaining attributes (2n) do
Cconcatenate the first attribute with Pi;
Concatenate the other with i; /(1 ≤ i ≤ n)*/*
/ (AP1, A1, ... , APn, An) */*
Update the FMB tables with the attribute information, including its fuzzy objectslike linguistic labels, similarity relations (only for FTYPE3), fuzzy quantifiers, etc.
End

5.2.3. FSQL_TO_SQL layer Interface

FSQL_TO_SQL offers a convivial interface that, on the one hand, presents an editor of description FSQL script, and on the other hand, give automatically its equivalent in SQL with the update of the corresponding FMB. It is developed in C++ language. Figures 10 and 11 shows an example of translation of

FSQL script to SQL script. FSQL_TO_SQL can moreover detect and signal the syntactic errors in a FSQL script. If the error is to the level of the new objects of FSQL, our tool displays this error, the most possible cause of its apparition and the line where it is Figure 12. Otherwise the error occurred to the level of the SQL commands; a report concerning this error will be generated by the Oracle DBMS.

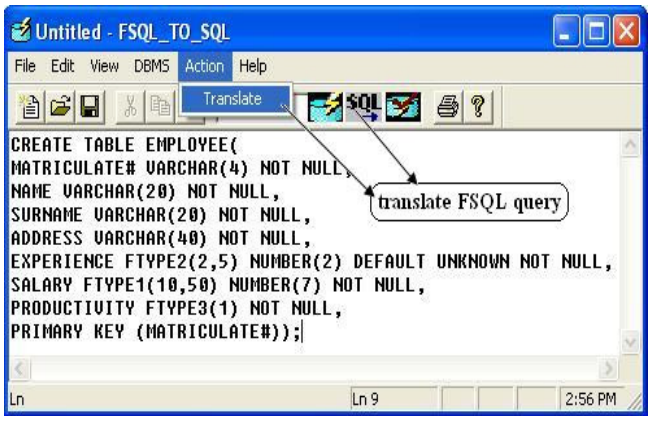

Figure 10. FSQL script.

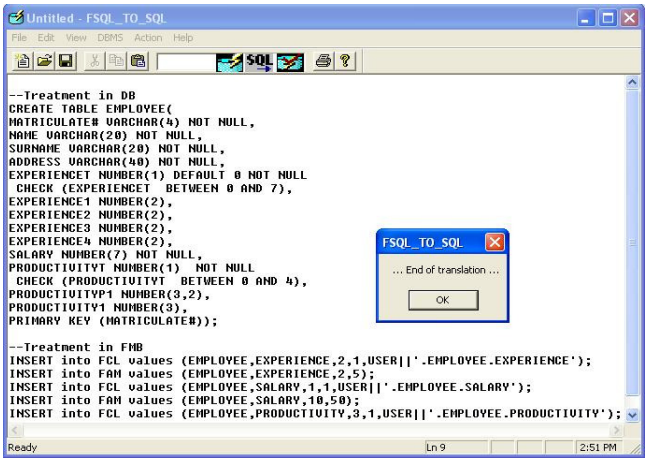

Figure 11. Translation FSQL script to SQL script.

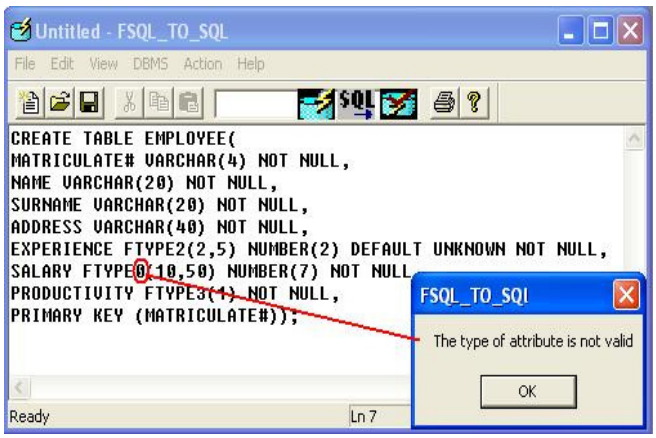

Figure 12. detection of errors.

6. Conclusion

Several real applications need to manage fuzzy information and to make benefit their users from flexible queries. Several theoretical solutions have been proposed. We are interested to the works of Medina et al. whose proposed FSQL server to treat flexible query with FSQL language. This prototype is constructed by the addition of a layer around a classic RDBMS while supposing that the user already implements the FRDB manually.

We presented in this paper a new architecture of FRDBMS based on the GEFRED model. This architecture is based on the weak coupling concept with the Oracle DBMS. This FRDBMS offers all functionalities of a classic DBMS. It permits, in particular, the description, the manipulation and the querying of FRDB in FSQL language. As futures perspectives of this work, we mention the automatic mapping of existing relational DB to FRDB. This point is theoretically done but not implemented yet, so we think that it will contribute to make easier the use of the FRDB in real applications.

References

[1] Bosc P. and Pivert O., "SQLf: A Relational Database Language for Fuzzy Querying," *Computer Journal of IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 80-89, 1995.

[2] Bosc P., Liétard L., and Pivert O., "Bases de Données et Flexibilité: Les Requêtes Graduelles," *Computer Journal of Techniques et Sciences Informatiques*, vol. 7, no. 3, pp. 355-378, 1998.

[3] Bosc P. and Pivert O., "SQLf Query Functionality on Top of a Regular Relational Database Management," *in Proceedings of Knowledge Management in Fuzzy Databases*, Heidelberg, pp. 171-190, 2000.

[4] Buckles P. and Petry E., "A Fuzzy Representation of Data for Relational Databases," *in Proceedings of Fuzzy Sets and Systems*, USA, pp. 213-226, 1982.

[5] Galindo J., Medina M., Pons O., and Cubero J., "A Server for Fuzzy SQL Queries," *in Proceedings of Lecture Notes in Artificial Intelligence (LNAI)*, USA, pp. 165-175, 1998.

[6] Galindo J., "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión Del Modelo y Adaptación de los SGBD Actuales," *Doctoral Thesis*, *Universidad de Granada*, 1999.

[7] Galindo J., Aranda M., Caro L., Guevara A., and Aguayo A., "Applying Fuzzy Databases and FSQL to the Management of Rural Accommodation," *Tourist Management Journal*, vol. 23, no. 6, pp. 623-629, 2002.

[8] Galindo J., Urrutia A., and Piattini M., *Representation of Fuzzy Knowledge in Relational Databases*, IEEE Computer Society, 2004.
[9] Galindo J. "New Characteristics in FSQL: A Fuzzy SQL for Fuzzy Databases," *Computer Journal of WSEAS Transactions on Information Science and Applications*, vol. 2, no. 2, pp. 161-169, 2005.
[10] Galindo J., "Le Serveur FSQL and FQ," http: //www.lcc.uma.es/personal/ppgg/FSQL.html, 2008.
[11] Galindo J., Urrutia A., and Piattini M., *Fuzzy Databases: Modeling, Design and Implementation*, Idea Group Publishing, Hershey, 2006.
[12] Grissa A., Ben Hassine A., and Ounelli H., "Extended_FSQL_Server: A Server for the Description and the Manipulation of FRDB," *in Proceedings of the 4th International Multi Conference on Computer Science and Information Technology (CSIT)*, Jordan, pp. 454-464, 2006.
[13] Medina M., Pons O., and Vila A., "An Elemental Processor of Fuzzy SQL," *Computer Journal of Math Ware and Soft Computing*, vol. 1, no. 3, pp. 285-295, 1994.
[14] Medina M., Pons O., and Vila A., "GEFRED: A Generalized Model of Fuzzy Relational Data Bases", *Computer Journal of Information Sciences*, vol. 76, no. 1, pp. 87-109, 1994.
[15] Medina M., Pons O., and Vila A., "FIRST: A Fuzzy Interface for Relational Systems," *in Proceedings of VI International Fuzzy Systems Association World Congress (IFSA)*, Sao Paulo, pp. 29-31, 1995.
[16] Prade H. and Testemale C., "Fuzzy Relational Databases: Representational Issues and Reduction Using Similarity Measures," *Computer Journal of Information Sciences,* vol. 38, no. 2, pp. 118-126, 1987.
[17] Umano M., Fukami S., Mizumoto M., and Tanaka K., "Retrieval Processing from Fuzzy Databases," *Technical Reports*, University of Maryland, 1980.
[18] Urrutia A., Galindo J., and Piattini M., "Modeling Data Using Fuzzy Attributes," *in Proceedings of the 22nd International Conference of the Chilean Computer Science Society (SCCC'02)*, UK, pp. 117-123, 2002.
[19] Zemankova M. and Kandel A., "Implementing Imprecision in Information Systems," *Computer Journal of Information Sciences,* vol. 37, no. 1, pp. 107-141, 1985.

Amel Grissa Touzi received the diploma of engineering in computer science and PhD in computer science from the Faculty of Sciences of Tunis, Tunisian in 1989 and 1994, respectively.

Mohamed Ali Ben Hassine is a PhD student at the Faculty of Sciences of Tunis, Tunisia, Department of Computer Sciences. He received the BSc degree in computer science from the Faculty of Science of Tunis in 2003, and the Master degree in automatic and signal processing from the National School of Engineer of Tunis in 2006.