New Characteristics in FSQL, a Fuzzy SQL for Fuzzy Databases[*]

JOSÉ GALINDO
Dpto. Lenguajes y Ciencias de la Computación,
University of Málaga, Spain

Abstract: - The FSQL language is an extension of the SQL language which permits us to handle fuzzy information in fuzzy or crisp databases. The first version of FSQL was implemented for Oracle Databases in PL/SQL. This first version defined basic fuzzy (or flexible) queries for fuzzy or traditional databases. In this work we present some new characteristics for FSQL. These new definitions include four new fuzzy comparators, five new fuzzy attributes, six new fuzzy constant types, new characteristics in the fulfillment thresholds, dynamic change of functions in logic operations, fuzzy set operators, the `ALTER FSQL` statement, and some useful functions to handle fuzzy attributes and fuzzy values. Besides, we outline here the definition of some DDL statements of FSQL and eighteen fuzzy comparators for fuzzy time in FSQL. The advantages of these new characteristics are obvious: basically they allow a greater expressiveness.

Key-Words: - Fuzzy relational databases, Fuzzy SQL, FSQL, Fuzzy queries, Fuzzy comparators.

1 Introduction

At a theoretical level, there exists many Fuzzy Relational Database (FRDB) models that, based on the relational model, they extend it in order to allow storing and/or treating vague and uncertain information [1][8]. One of them, the GEFRED model [4][7], is an eclectic synthesis of the different models.

On the other hand, the FSQL or Fuzzy SQL language [3][4][5] 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. This language allows us to express sentences taking into account the characteristics of imprecise information: fuzzy conditions, fuzzy values, computing fulfillment degrees, establishing fulfillment thresholds...

The SQLf language [2] only study the `SELECT` statement with fuzzy comparisons between crisp values (columns) with linguistic labels and the use of the "approximately equal" between crisp values (the only fuzzy comparator included in SQLf). The best of SQLf is the quantified statements (with fuzzy quantifiers).

More recently, the FuzzyEER model [5][6][9] has been defined as an extension of the EER model to create conceptual schemas with fuzzy semantics and notations. This extension provides new and useful definitions: fuzzy attributes, fuzzy entities, fuzzy relationships, fuzzy specializations… In this paper we propose to incorporate some FuzzyEER concepts in FSQL. Besides, we include in this language some tools, which become very useful in some operations.

The next section defines the fuzzy attributes included in the FuzzyEER model. After, we explain some aspects about FSQL, emphasizing the new characteristics.

2 Fuzzy Attributes

A fuzzy database needs a special data dictionary in order to store the information related to the inexact nature or context of each fuzzy attribute. We call it the Fuzzy Metaknowledge Base (FMB).

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. Each class includes some different fuzzy datatypes:

2.1 Fuzzy Sets as Fuzzy Values

These fuzzy attributes may be classified in four datatypes. 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:

- **Type 1**: These are attributes with *"precise data"*, classic or crisp (traditional, with no imprecision). However, we can define linguistic labels in its domain and we can use them in fuzzy queries. This type is useful for extending traditional databases allowing fuzzy queries to be made about classic data. For example: "Give me

[*] Partially supported by MCYT project TIC2002-00480.

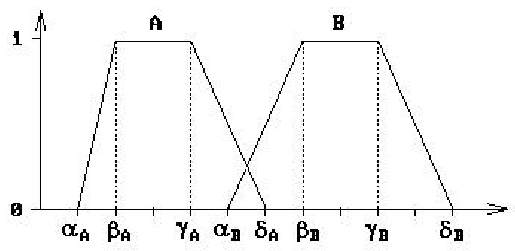

Figure 1. Trapezoidal possibility distributions: A, B.

employees that earn a lot more than the minimum salary" (salary must be a well known attribute).

- **Type 2**: These are attributes that gather "*imprecise data over an ordered referential*". These attributes admit both crisp and fuzzy data, in the form of possibility distributions over an underlying ordered dominion (fuzzy sets). It is an extension of the Type 1 that does, now, allow the storage of imprecise information, such as: "he is approximately 2 metres tall". The most complex of these fuzzy sets are the so-called *extended trapezoidals*. This novel type of fuzzy set is composed by linear functions in some pieces. It allows great flexibility (even for non-convex sets). For the sake of simplicity the simple trapezoidal functions (Figure 1) are more common. Note that this underlying domain is continuous and ordered (usually it will be the real number dominion or the time).
- **Type 3**: They are attributes over "*data of discreet non-ordered dominion with analogy*". In these attributes some labels are defined (e.g. "blond", "ginger", "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 resemble each other. They also allow possibility distributions (or fuzzy sets) over this dominion, like for example, the value {1/dark, 0.4/ginger}, which expresses that a certain person is more likely to be dark than ginger. Note that underlying domain of these fuzzy sets are the set of labels and this set is discreet and non-ordered.
- **Type 4**: These attributes are original and they are defined in the same way as Type 3 attributes, without it being necessary for a similarity relationship to exist between the labels.

2.2 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). In order to keep it simple, we will only use degrees in the interval [0,1], because the other option offers no great advantages.

The meaning of these degrees is varied and depends on their use. The processing of the data will be different depending on the meaning. The most important possible meanings of the degrees used by some authors are [4][5]: Fulfilment degree, Uncertainty degree, Possibility degree and Importance degree. Of course, we can define and use other meanings. In this paper we do not aim to demonstrate the usefulness of these degrees and their different meanings. Several authors who have used these degrees have already done so.

The ways of using these fuzzy degrees are classified in two families: Associated and non-associated degrees.

Associated degrees are associated to a specific value to which they incorporate imprecision. These degrees may be associated to different concepts [5]:

- **Degree in each value of an attribute (**we will call it as **Type 5):** Some attributes may have a fuzzy degree associated to them. This implies that each value of this attribute (in every tuple or instance) has an associated degree, measuring the level of fuzziness in that value. In order to interpret it, we need to know the meaning of the degree and the meaning of the associated attribute.
- **Degree in a set of values of different attributes (Type 6):** Here, the degree is associated to some attributes. It joins the fuzziness of some attributes in only one degree.
- **Degree in the whole instance of the relation (Type 7):** This degree is associated to the whole tuple of the relation and not exclusively to the value of a specific attribute of the tuple (or instance). Usually, it can represent something like the "membership degree" of this tuple (or instance) to the relation (or table) of the database. This degree represents the fuzzy degree of a fuzzy relation (for example, a fuzzy entity in the FuzzyEER model).

Non-associated degrees (Type 8): There are cases in which the imprecise information, which we wish to express, can be represented by using only the degree, without associating this degree to another specific value or values. For example, the dangerousness of a medicine may be expressed by a fuzzy degree.

The first version of FSQL only includes the fuzzy attributes Type 1, 2 and 3.

Fuzzy Constant	Significance	Store in	Use in
`UNKNOWN`	Unknown value but the attribute is applicable.	2, 3, 4	2, 3, 4
`UNDEFINED`	The attribute is not applicable or it is meaningless.	2, 3, 4	2, 3, 4
`NULL`	Total ignorance: we know nothing about it.	All	All
$\$[\alpha,\beta,\gamma,\delta]$	Fuzzy trapezoid with $\alpha \le \beta \le \gamma \le \delta$ (Figure 1).	2	1, 2
$\$[\alpha,\beta,\gamma,\delta,$ `P1/N1,…,Pn/Nn]`	Extended fuzzy trapezoid (with some points `Pi/Ni` where all the `Ni` are between α and β or between γ and δ). Values β and γ are both optional. If they do not exist, then this constant is a fuzzy value without kernel.	2	1, 2
`[n,m]`	Interval "Between `n` and `m`".	2	1, 2
`n+-m`	Fuzzy value "Approximately `n`": triangle `n` ± `m`.	2	1, 2
`#n`	Fuzzy value "Approximately `n`": triangle `n` ± *margin*, where *margin* is stored in the FMB for each attribute.	2	1, 2
`$label`	Linguistic Label: it may be a trapezoid or a scalar (defined in FMB).	2, 3, 4	1, 2, 3, 4
`{P1/L1, P2/L2, …, Pn/Ln}`	Non-continuous possibility distribution on labels, where `P1`, `P2`, …, `Pn` are the possibility values and `L1`, `L2`, …, `Ln` are the labels.	3, 4	3, 4
`{L1, L2,…, Ln}`	Non-continuous possibility distribution on labels, where `L1`, `L2`, …, `Ln` are the labels, with possibility degrees 1 for all of them: `{1/L1, …, 1/Ln}`.	3, 4	3, 4
`{P1/N1, P2/N2, …, Pn/Nn}`	Non-continuous possibility distribution on numbers, where `P1`, `P2`, …, `Pn` are the possibility values and `N1`, `N2`, …, `Nn` are the numbers.	2	1, 2
`{N1, N2,…, Nn}`	Non-continuous possibility distribution on numbers, where `N1`, `N2`, …, `Nn` are the numbers, with possibility 1 for all of them: `{1/N1, …, 1/Nn}`.	2	1, 2

Table 1: Fuzzy constants that may be used in FSQL statements and the fuzzy datatypes which can store and use them.

3 Some Aspects of FSQL Language

The FSQL language extends SQL in order to handle fuzzy information and to express fuzzy sentences. The main extensions in FSQL are the followings. We will expose them indicating the new characteristics, which we present in this paper, and the old characteristics.

3.1 Fuzzy Constants

In FSQL, we can use the fuzzy constants as detailed and explained in Table 1. 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 distinguish them easily. Labels for attributes with an ordered underlined fuzzy domain (Type 1 and 2) have an associated possibility distribution (usually with a trapezoidal form like Figure 1).

It should be noted that the non-continuous possibility distributions (the last four constant types in Table 1) are also disjunctive values. The new constants in FSQL are these four types, the extended trapezoid and the explicit approximate value (`n+-m`), which is represented with a triangular function centered in `n` and with base equal to 2`m`. The previous definition of FSQL included the implicit approximate value (`#n`) where the only possible margin is the value stored in the FMB.

Besides, Table 1 shows the fuzzy datatypes that can store and use these fuzzy constants (in queries, fuzzy conditions, etc.). Some implementations may limit the number of elements in the stored non-continuous possibility distributions or in the extended trapezoid.

3.2 Fuzzy Comparators

In addition to the typical comparators (=, >, >=...), FSQL includes the eighteen fuzzy comparators in Table 2. The first version of FSQL only included fourteen. As in SQL, fuzzy comparators compare one column with one constant or two columns of the same (or compatible) type.

Possibility	Necessity	Significance
FEQ or **F=**	**NFEQ** or **NF=**	Possibly/Necessarily Fuzzy Equal than…
FDIF, **F!=** or **F<>**	**NFDIF**, **NF!=** or **NF<>**	Possibly/Necessarily Fuzzy Different to…
FGT or **F>**	**NFGT** or **NF>**	Possibly/Necessarily Fuzzy Greater Than…
FGEQ or **F>=**	**NFGEQ** or **NF>=**	Possibly/Necessarily Fuzzy Greater or Equal than…
FLT or **F<**	**NFLT** or **NF<**	Possibly/Necessarily Fuzzy Less Than…
FLEQ or **F<=**	**NFLEQ** or **NF<=**	Possibly/Necessarily Fuzzy Less or Equal than…
MGT or **F>>**	**NMGT** or **NF>>**	Possibly/Necessarily Much Greater Than…
MLT or **F<<**	**NMLT** or **NF<<**	Possibly/Necessarily Much Less Than…
FINCL	**INCL**	Fuzzy Included in… / Included in…

Table 2: The 18 fuzzy comparators for FSQL (Fuzzy SQL):
16 in the Possibility/Necessity Family, and 2 in the Inclusion Family.

As possibility comparators are more general (less restrictive) than necessity comparators, necessity comparators retrieve fewer tuples, and these tuples *necessarily* comply with the conditions (whereas with possibility comparators, the tuples only possibly comply with the condition, without any absolute certainty). Table 3 shows the definition for all the fuzzy comparators in the Possibility/Necessity family for fuzzy attributes Type 1 and 2, with respect to trapezoidal functions (Figure 1).

In attributes with a non-ordered underlying domain (Fuzzy Type 3 or 4) only the fuzzy comparators FEQ, FDIF, INCL and FINCL can be used, since they lack order.

Comparator INC and FINCL do not use possibility and necessity measures, and INCL is more restrictive than FINCL (INCL retrieves less rows than FINCL). Comparator INC examines if one fuzzy value is included in other, returning a crisp value of the tri-valued logic:

$$\mathtt{A\ INCL\ B} = \begin{cases} \mathtt{NULL} & \text{if } \mathtt{A} \text{ or } \mathtt{B} \text{ are } \mathtt{NULL} \\ \mathtt{TRUE} & \text{if } \mathtt{A}(x) \le \mathtt{B}(x),\ \forall x \\ \mathtt{FALSE} & \text{otherwise} \end{cases}$$

FINCL defines a degree of subsethood. This degree is computed by

$$\mathtt{CDEG(A\ FINCL\ B)} = \frac{Card\,(\mathtt{A}) - Card\,(\mathtt{A} \cap \mathtt{B})}{Card\,(\mathtt{A})}$$

where *Card* is the cardinality of the membership function. The intersection of A and B may use the minimum t-norm. Thus, if CDEG(A FINCL B) = 1, this means that A is totally included in B. In the other extreme, if CDEG(A FINCL B) = 0, then this means that A is not included in B at all.

Operator NOT can precede to every condition. The fuzzy comparator of “inequality” or “possibly different” may be modeled denying (with NOT) a comparison with FEQ or NFEQ, using the following format: NOT A FEQ B. However, this method obtains different results to when FDIF and NFDIF are used, because the NFDIF comparator must be more restrictive than FDIF. Then, by definition NFDIF denies the comparison with FEQ, and FDIF denies the comparison with NFEQ.

Furthermore, the behavior of the NOT operator may be changed (see Section 3.4 and 3.6).

In order to define the comparators FEQ/FDIF for Fuzzy Attributes Type 3 and 4, let us suppose that we want to compare two possibility distributions, F and X, on the linguistic labels of a fuzzy attribute Type 3: F FEQ X, where

$$\mathtt{F} = \{\mathtt{FP}_i / \text{label}\mathtt{F}_i\} \quad \text{with} \quad i = 1,2,\ldots,\mathtt{LEN}_\mathtt{F}$$

$$\mathtt{X} = \{\mathtt{XP}_j / \text{label}\mathtt{X}_j\} \quad \text{with} \quad j = 1,2,\ldots,\mathtt{LEN}_\mathtt{X}$$

label$\mathtt{F}_i$ and label$\mathtt{X}_j$ being linguistic labels, which belong to the same attribute and therefore they can be compared with its similarity relation. The values $\mathtt{FP}_i$ and $\mathtt{XP}_j$ are the possibility degrees, in [0,1], associated to these labels respectively. $\mathtt{LEN}_\mathtt{F}$ and $\mathtt{LEN}_\mathtt{X}$ indicate the number of pairs {degree/label} of the possibility distributions F and X, respectively, with $\mathtt{LEN}_\mathtt{F} \ge 1$ and $\mathtt{LEN}_\mathtt{X} \ge 1$.

The compatibility degree of F and X is subsequently computed by:

$$\mathtt{CDEG(F\ FEQ\ X)} = \max_{\substack{i=1,2,\ldots,\mathtt{LEN}_\mathtt{F} \\ j=1,2,\ldots,\mathtt{LEN}_\mathtt{X}}} \{m_R(\text{label}\mathtt{F}_i, \text{label}\mathtt{X}_j) * \mathtt{FP}_i * \mathtt{XP}_j\}$$

where $m_R(\text{label}\mathtt{F}_i, \text{label}\mathtt{X}_j)$ express the similarity degree between both labels.

The previous equation is simplified when the comparison is performed directly on a label ($\mathtt{label}$=label$\mathtt{X}_1$ with $\mathtt{XP}_1$=1):

$$\mathtt{CDEG(F\ FEQ\ \$label)} = \max_{i=1,2,\ldots,\mathtt{LEN}_\mathtt{F}} \{m_R(\text{label}\mathtt{F}_i, \mathtt{label}) * \mathtt{FP}_i\}$$

Similarly, the compatibility degree of a comparison with a fuzzy attribute Type 4, F FEQ X, is computed. Now, $m_R(\text{label}\mathtt{F}_i, \text{label}\mathtt{X}_j)$ is 1 if label$\mathtt{F}_i$ = label$\mathtt{X}_j$, and 0 if label$\mathtt{F}_i \ne$ label$\mathtt{X}_j$.

F_Comp	Possibility operators `CDEG(A F_Comp B)=`	Necessity operators `CDEG(A F_Comp B)=`
`FEQ` `NFEQ`	$= \sup_{d \in U} \min(A(d), B(d))$ where U is the domain of A and B. A(d) is the possibility degree for $d \in U$ in the distribution A.	$= \inf_{d \in U} \max(1 - A(d), B(d))$ where U is the domain of A and B. A(d) is the possibility degree for $d \in U$ in the distribution A
`FDIF` `NFDIF`	= 1 – `CDEG(A NFEQ B)`	= 1 – `CDEG(A FEQ B)`
`FGT` `NFGT`	$= \begin{cases} 1 & \text{if } g_A \geq d_B \\ \frac{d_A - g_B}{(d_B - g_B) - (g_A - d_A)} & \text{if } g_A < d_B \,\&\, d_A > g_B \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } a_A \geq d_B \\ \frac{b_A - g_B}{(d_B - g_B) - (a_A - b_A)} & \text{if } a_A < d_B \,\&\, b_A > g_B \\ 0 & \text{otherwise} \end{cases}$
`FGEQ` `NFGEQ`	$= \begin{cases} 1 & \text{if } g_A \geq b_B \\ \frac{d_A - a_B}{(b_B - a_B) - (g_A - d_A)} & \text{if } g_A < b_B \,\&\, d_A > a_B \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } a_A \geq b_B \\ \frac{b_A - a_B}{(b_B - a_B) - (a_A - b_A)} & \text{if } a_A < b_B \,\&\, b_A > a_B \\ 0 & \text{otherwise} \end{cases}$
`FLT` `NFLT`	$= \begin{cases} 1 & \text{if } b_A \leq a_B \\ \frac{a_A - b_B}{(a_B - b_B) - (b_A - a_A)} & \text{if } b_A > a_B \,\&\, a_A b_B \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } d_A \leq a_B \\ \frac{g_A - b_B}{(a_B - b_B) - (d_A - g_A)} & \text{if } d_A > a_B \,\&\, g_A < b_B \\ 0 & \text{otherwise} \end{cases}$
`FLEQ` `NFLEQ`	$= \begin{cases} 1 & \text{if } b_A \leq g_B \\ \frac{d_B - a_A}{(b_A - a_A) - (g_B - d_B)} & \text{if } b_A > g_B \,\&\, a_A < d_B \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } a_A \leq g_B \\ \frac{g_A - d_B}{(g_B - d_B) - (d_A - g_A)} & \text{if } d_A > g_B \,\&\, g_A < d_B \\ 0 & \text{otherwise} \end{cases}$
`MGT` `NMGT`	$= \begin{cases} 1 & \text{if } g_A \geq d_B + M \\ \frac{g_B + M - d_A}{(b_A - a_A) - (g_B - d_B)} & \text{if } g_A < d_B + M \,\&\, d_A > g_B + M \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } a_A \geq d_B + M \\ \frac{g_B + M - b_A}{(a_A - b_A) - (d_B - g_B)} & \text{if } a_A < d_B + M \,\&\, b_A > g_B + M \\ 0 & \text{otherwise} \end{cases}$
`MLT` `NMLT`	$= \begin{cases} 1 & \text{if } b_A \leq a_B - M \\ \frac{b_B - M - a_A}{(b_A - a_A) - (a_B - b_B)} & \text{if } b_A > a_B - M \,\&\, a_A < b_B - M \\ 0 & \text{otherwise} \end{cases}$	$= \begin{cases} 1 & \text{if } d_A \leq a_B - M \\ \frac{b_B - M - g_A}{(d_A - g_A) - (a_B - b_B)} & \text{if } d_A > a_B - M \,\&\, g_A < b_B - M \\ 0 & \text{otherwise} \end{cases}$

M is the minimum distance to consider two attributes as very separate. M is defined in FMB for each attribute.

Table 3: Definition for the fuzzy comparators in the Possibility/Necessity Family, using two trapezoidal possibility distributions: `A` and `B` (like Figure 1).

The "fuzzy different" comparator, `FDIF`, is defined by denying the comparator `FEQ`:

`CDEG(A FDIF B)` = 1 – `CDEG(A FEQ B)`

3.3 Fulfillment Thresholds and Qualifiers

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. The reserved word `THOLD` (threshold) is optional and may be substituted by a traditional crisp comparator (`=`, `<`, `>=`, ...), modifying the query meaning. The word `THOLD` is equivalent to using the crisp comparator `>=`.

In the new FSQL rather than a number, τ may be a qualifier (defined in the FMB associated to each attribute), i.e. an identifier or label that should be defined in the FMB. Qualifiers are also preceded by the symbol `$`.

Example: "Give me all persons with fair hair (in minimum degree 0.5) that are possibly taller than label `$Tall` (with a high degree)":

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

This new FSQL admits thresholds in compound conditions (with logical operators). In general, it is preferable to use parentheses to clarify the influence of the threshold. For example, `(<condition1> AND <condition2>) THOLD τ`.

Operator	Returns
FUNION	All rows selected by either query (`R` or `T`). If there are duplicated tuples (in `R` and `T`), it uses, by default, the maximum s-norm: max (`R.CDEGROW`, `T.CDEGROW`)
FINTERSECT	All rows selected by both queries (`R` and `T`). If there are duplicated tuples (in `R` and `T`), it uses, by default, the minimum t-norm: min (`R.CDEGROW`, `T.CDEGROW`)
FMINUS	All distinct rows selected by the first query (`R`) but not the second (`T`). If there are duplicated tuples (in `R` and `T`), it uses, by default, the function: max (`0`, `R.CDEGROW – T.CDEGROW`)

Table 4: Fuzzy Set Operators in FSQL, applied to queries `R` and `T`.

3.4 Function `CDEG()` and Logic Operators

The function `CDEG` (Compatibility Degree) may be used with an attribute in the argument. Thus, it computes the fulfillment degree of the condition of the query, for the specific attribute.

We can use `CDEG(*)` to obtain the fulfillment degree of each tuple (with all of its attributes, not just one of them) in the condition.

If logic operators (`NOT`, `AND`, `OR`) appear in the condition, the calculation of this compatibility degree is carried out, by default, using the classic negation, the minimum t-norm and the maximum s-norm respectively. The user may change these default values with the `ALTER FSQL` statement. In order to change these functions dynamically for a specific logic operation the FSQL user may use the following:

a) `NOT (negation)`
b) `AND (t-norm)`
c) `OR (s-norm)`

where `negation`, `t-norm` and `s-norm` are alphanumeric values like, for example, “minimum”, “product”, “drastic product”, “bounded product *p*”, “Einstein product”, “Hamacher product *p*”, etc. for `t-norm`, and “maximum”, “sum-product”, “drastic sum”, “bounded sum *p*”, “Einstein sum”, etc. for `s-norm`. It should be noted that for the sake of simplicity, if the norm needs some argument *p*, it is included after the name.

3.5 Fuzzy Set Operators

In SQL, you can combine multiple queries using the set operators `UNION`, `UNION ALL`, `INTERSECT`, and `MINUS` (or `EXCEPT`). In FSQL, if these queries include some fuzzy degree associated to the whole tuple (or row), you can use an extended version of these set operators, which are listed and explained in Table 4 together with the default values. This degree is the fuzzy degree of a fuzzy entity or a fuzzy relationship, or the compatibility degree, `CDEG(*)`, of a fuzzy subquery. We can change the default functions of these fuzzy set operators, using the `ALTER FSQL` statement. However, we can change these functions dynamically for a specific operation:

a) `FUNION (s-norm)`
b) `FINTERSECT (t-norm)`
c) `FMINUS (s-norm)`

where `t-norm` and `s-norm` are alphanumeric values just like we saw in Section 3.4.

3.6 Modifying FSQL Options

The `ALTER FSQL` and `ALTER SESSION` statements specify or modify certain parameters that affect the behavior of some aspects in FSQL statements. `ALTER FSQL` affects all personal connections to the database (definitively), whereas `ALTER SESSION` only affects the current session (or connection). Both statements have the same syntax and the `ALTER FSQL` statement syntax is represented in Figure 2. This statement has three clauses.

`LOGIC` clause specifies the function to use (in the `CDEG` function) when logic operators are used:

- **Logic_Operator** may be one of the following words: {**`NOT`**, **`AND`**, **`OR`**, **`ALL`**}. The word `ALL` refers to all the three basic logic operators (`NOT`, `AND`, `OR`).
- **Function_ts_norm** is the function to use in the previously specified logic operator. Besides, the `NOT` function must only have one argument and the `AND`/`OR` functions must have two arguments, and they represent a particular t-norm and s-norm, respectively. If we use the word **`DEFAULT`**, then the statement sets the default functions (`GREATEST` function for the `OR` operator, `LEAST` function for the `AND` operator, and negation `1–X` function for the `NOT` operator).

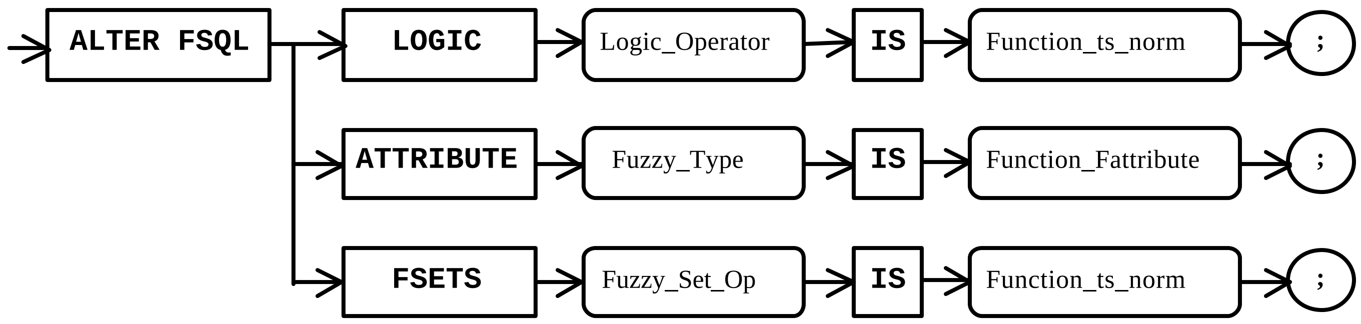

Figure 2: `ALTER FSQL` statement.

`ATTRIBUTE` clause specifies what FSQL does by default when it finds fuzzy attributes in unusual or special positions. For example, imagine that a fuzzy attribute appears in an `ORDER BY` clause, or like an argument in a function (different than `CDEG`).

- **Fuzzy_Type** may be one of the following words: {**`FTYPE2`**, **`FTYPE3`**, **`FTYPE4`**, **`ALL`**}. The word `ALL` refers to all the three fuzzy attributes Type 2, 3 and 4.
- **Function_Fattribute** is the function to use when the previously specified fuzzy attribute type appears in a special position. In addition, we can use the following predefined options: **`ERROR`** (FSQL gives an error), **`FTYPE`** (FSQL uses the numeric type of the value in that fuzzy attribute), and **`TO_CHAR`** (FSQL uses the text which represents each value in the fuzzy attribute and, for example, an approximate value is represented with '7±2').

`FSET` clause is useful for specifying the function to use when fuzzy set operators are used:

- **Logic_Set_Op** may be one of the following words: {**`FUNION`**, **`FINTERSECT`**, **`FMINUS`**, **`ALL`**}. The word `ALL` refers to all the three fuzzy set operators.
- **Function_ts_norm** is the function to use in the previously specified fuzzy set operator. In addition, this function must have two arguments. If we use the word **`DEFAULT`**, then the statement sets the default functions (see Table 4).

Of course, all functions must be defined in the DBMS, and the user must be permitted to execute it.

3.7 Some DDL Statements

The **DDL** (Data Definition Language) of FSQL includes the modification of certain statements and some new statements of three families: `CREATE`, `DROP` and `ALTER`. These statements are applied to the following objects of an FRDB.

Object `TABLE`: Fuzzy relations or fuzzy tables (with or without fuzzy attributes). The group of statements formed by `CREATE TABLE`, `ALTER TABLE` and `DROP TABLE` already exists in SQL standard. FSQL expands their syntax so that they enable the fuzzy characteristics. For example FSQL predefines some new datatypes: `FTYPE1` or `CRISP`, `FTYPE2` or `POSSIBILISTIC`, `FTYPE3` or `SCALAR`, `FTYPE4` or `NONSIMILAR`, `FUZZY` or `FUZZY DEGREE`, fuzzy time datatypes (`FUZZY_DATE`, `FUZZY_TIME` and `FUZZY_TIMESTAMP`) and the associated degrees (see Section 2.2). The user may create his/her own fuzzy datatypes with the following object.

Object `FDATATYPE`: This object allows the definition of specific fuzzy datatypes identified with one name. These names may be used wherever a fuzzy datatype may be used.

Object `VIEW` or `MATERIALIZED VIEW` (`SNAPSHOP`): the statements `CREATE`, `ALTER` and `DROP`, applied to these objects, already exist in SQL, but in FSQL fuzzy queries are allowed with the `SELECT` of FSQL.

Object `LABEL`: This object includes fuzzy labels of attributes Types 1, 2 and 4. If it belongs to Type 1 or 2, then it is associated to one possibility distribution. This object is exclusive of FSQL.

Object `NEARNESS`: This object represents the similarity relationships of fuzzy attributes Type 3. The `CREATE NEARNESS` statement implies the definition of labels for fuzzy attributes Type 3 and the similarity relation between them. This object is exclusive of FSQL.

Object `QUALIFIER`: This object represents a constant inside the context of the degrees of an attribute (Section 3.3). This object is exclusive of FSQL.

Object `QUANTIFIER`: This exclusive object of FSQL represents fuzzy quantifiers inside the context of attributes, tables or the system.

Object `MEANING`: This object represents the meanings or significances for some of the degrees with which the FRDB works (Section 2.2). This object is exclusive of FSQL.

3.8 Some Useful Functions

The new FSQL presented here includes the definition of some useful functions. The more interesting functions are the followings.

`FDEGREE(attribute_list)` returns the fuzzy degree associated to the attribute or attributes given in its arguments (see Type 5 and 6 in Section 2.2).

`FDEGROW(table)` returns the fuzzy degree associated to the row, i.e. to the whole tuple (Type 7). The argument of this function is the table name.

`CARD(fuzzy_value)` returns the cardinality of a fuzzy value. For example, in order to know the rows with less fuzziness in an attribute than a fuzzy constant, a `SELECT` statement may include the next condition: `CARD(Quality) < CARD(3+-2)`. It should be noted that `CARD(X +-` *m*`) =` *m*.

`NORM(fuzzy_value)` normalizes the fuzzy value, dividing the original membership function by the height of the fuzzy value.

`CONC_DILAT (fuzzy_value, p)` returns the membership values raised to power **p**:

- If **p** > 1, it returns a *concentrated* version of the fuzzy value. The membership function of this version takes on relatively smaller values. Usually, **p** = 2.
- If **p** ∈ (0,1), this function returns a *dilated* version of the fuzzy value. The new membership function takes on relatively greater values. Usually, **p** = 0.5 (the square root).

`MORE_CONTRAST(fuzzy_value, p)` is the contrast *intensification* function and it returns the fuzzy value with the most contrast. The membership values lower than 0.5 are diminished while the grades of membership above 0.5 are elevated. The operation is defined by (usually **p** = 2):

$$\mathtt{MORE\_CONTRAST(A,\ p)}(x) = \begin{cases} 2^{p-1}\mathrm{A}^{p}(x) & \text{if } \mathrm{A}(x) \le 0.5 \\ 1-2^{p-1}(1-\mathrm{A}(x))^{p} & \text{otherwise} \end{cases}$$

`FUZZIFICATION(fuzzy_value, p)` has a complementary effect to that of intensification. The operation is defined by (usually **p** = 2):

$$\mathtt{FUZZIFICATION(A,\ p)}(x) = \begin{cases} \sqrt[p]{\mathrm{A}(x)/2} & \text{if } \mathrm{A}(x) \le 0.5 \\ 1-\sqrt[p]{(1-\mathrm{A}(x))/2} & \text{otherwise} \end{cases}$$

`INTERSECTION(fuzzy_values,t)` returns the intersection of the fuzzy values, with the t-norm indicated in the last argument.

`UNION(fuzzy_values, s)` returns the union of the fuzzy values, with the indicated s-norm.

Functions **`CONC_DILAT`**, **`MORE_CONTRAST`** and **`FUZZIFICATION`** are useful for implementing linguistic hedges such as *specially*, *very*, *slightly* and *more or less*.

3.9 Other Characteristics

The first definition of FSQL has some other characteristics. For example, the wild card `%` or the comparison with `IS` or `IS NOT`, followed by one constant type: `UNKNOWN`, `UNDEFINED` or `NULL`.

However, the most important new characteristics related to queries which are not defined in this paper are:

- New syntax for quantified queries using fuzzy quantifiers
- Fuzzy division queries.
- Fuzzy time, extending the comparators of TSQL2 for fuzzy temporal querying.

These and others themes will be published in [5]. Table 5 summarizes the new fuzzy comparators for fuzzy time in FSQL defining them using existing fuzzy comparators (Table 2). We only study valid time databases because transaction time databases need the exact system time. However, valid time databases need the time in which the fact was considered to be true in the real world. Sometimes, this time is not exactly known or it is a vague time period.

In the same way as usual temporal RDB, valid time relations have two additional attributes whose data type is one of the previously defined fuzzy time types: `VST` (Valid Start Time) and `VET` (Valid End Time). These attributes in tuple `t` represent the fact that its information is only valid in the real world during the time period `[t.VST, t.VET]`.

Note that TSQL2 only includes five comparators (`INCLUDES`, `INCLUDED_IN`, `OVERLAPS`, `BEFORE` and `AFTER`). FSQL extends these five comparators with possibility and necessity versions (with prefixes `F_` and `NF_` respectively). Besides, FSQL includes totally new fuzzy comparators (`XBEFORE`, `XAFTER`,

Expression with Temporal Fuzzy Comparator	Equivalence
`[t.VST,t.VET] F_INCLUDES [T1,T2]`	`T1 FGEQ t.VST AND T2 FLEQ t.VET`
`[t.VST,t.VET] F_INCLUDED_IN [T1,T2]`	`T1 FLEQ t.VST AND T2 FGEQ t.VET`
`[t.VST,t.VET] F_OVERLAPS [T1,T2]`	`T1 FLEQ t.VET AND T2 FGEQ t.VST`
`[t.VST,t.VET] F_BEFORE [T1,T2]`	`T1 FGEQ t.VET`
`[t.VST,t.VET] F_AFTER [T1,T2]`	`T2 FLEQ t.VST`
`[t.VST,t.VET] NF_INCLUDES [T1,T2]`	`T1 NFGEQ t.VST AND T2 NFLEQ t.VET`
`[t.VST,t.VET] NF_INCLUDED_IN [T1,T2]`	`T1 NFLEQ t.VST AND T2 NFGEQ t.VET`
`[t.VST,t.VET] NF_OVERLAPS [T1,T2]`	`T1 NFLEQ t.VET AND T2 NFGEQ t.VST`
`[t.VST,t.VET] NF_BEFORE [T1,T2]`	`T1 NFGEQ t.VET`
`[t.VST,t.VET] NF_AFTER [T1,T2]`	`T2 NFLEQ t.VST`
`[t.VST,t.VET] F_XBEFORE [T1,T2]`	`T1 FGT  t.VET`
`[t.VST,t.VET] F_XAFTER [T1,T2]`	`T2 FLT  t.VST`
`[t.VST,t.VET] F_MUCH_BEFORE [T1,T2]`	`T1 MGT  t.VET`
`[t.VST,t.VET] F_MUCH_AFTER [T1,T2]`	`T2 MLT  t.VST`
`[t.VST,t.VET] NF_XBEFORE [T1,T2]`	`T1 NFGT  t.VET`
`[t.VST,t.VET] NF_XAFTER [T1,T2]`	`T2 NFLT  t.VST`
`[t.VST,t.VET] NF_MUCH_BEFORE [T1,T2]`	`T1 NMGT  t.VET`
`[t.VST,t.VET] NF_MUCH_AFTER [T1,T2]`	`T2 NMLT  t.VST`

**Table 5: The 18 Fuzzy Comparators for Fuzzy Time in FSQL.
Extending 5 TSQL2 comparators and 8 new ones
(possibility and necessity versions): X means "eXclusively".**

MUCH_BEFORE and MUCH AFTER) also with possibility and necessity versions (with prefixes F_ and NF_ respectively).

4 Conclusions

The FSQL language is an extension of the SQL language which permits us to handle fuzzy information in fuzzy or crisp databases. The first version of FSQL was implemented for Oracle databases [3][4]. Actually, the FuzzyEER model [5][6][9] has been defined as an extension of the EER model to create conceptual schemas with fuzzy semantics and notations, and some FuzzyEER concepts may be incorporated in FSQL, in order to enrich it. Thus, we have included in this language some tools, which become very useful.

These new definitions have not been implemented still, and in the current DBMS some of them are not easy. Let us hope the DBMS incorporate soon new types of internal data, which allow storing fuzzy values and fuzzy processing of these values. We think that the FSQL definition will be a useful start point in order to generalize the fuzzy databases in the real database world.

References:

[1] Bosc P., Galibourg M, "Indexing principles for a fuzzy data base". Inf. Systems, Vol. 14-6, pp. 493-499, 1989.

[2] Bosc P., Pivert O., "SQLf: A Relational Database Language for Fuzzy Querying". IEEE Transactions on Fuzzy Systems, 3, pp. 1-17, 1995.

[3] Galindo J., Medina M., Pons O., Cubero J. C., "A Server for Fuzzy SQL Queries". In "Flexible Query Answering Systems". Lecture Notes in Artificial Intelligence 1495, pp. 164-174. Ed. Springer, 1998.

[4] Galindo J., "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales". Ph. Doctoral Thesis Universidad de Granada, Spain, 1999 (www.lcc.uma.es).

[5] Galindo J., Urrutia A., Piattini M., "Fuzzy Databases: Modeling, Design and Implementation". To publish by Idea Group Publishing Hershey, USA, 2005.

[6] Galindo J., Urrutia A., Carrasco R.A., Piattini M., "Relaxing Constraints in Enhanced Entity-Relationship Models using Fuzzy Quantifiers". IEEE Trans. on Fuzzy Systems 12-6, pp. 780-796, 2004.

[7] 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.

[8] Petry F.E., "Fuzzy Databases: Principles and Applications". *International Series in Intelligent Technologies*. Ed. H.J. Zimmermann. Kluwer Academic Publ. (KAP), 1996.

[9] Urrutia A., Galindo J., Piattini M., "Modeling Data Using Fuzzy Attributes". Proc. IEEE Computer Soc. Press XXII Int. Conf. of the Chilean Computer Science Soc. (SCCC 2002), pp. 117-123. Chile, 2002.