A Knowledge Representation Example of a Fuzzy Database Implemented in PostgreSQL, with FIRST-2 and FSQL

Angélica Urrutia[1], José Galindo[2], and Alejandro Sepúlveda[1]

[1]Universidad Católica del Maule, Chile, aurrutia@ucm.cl
[2]Universidad de Málaga, España, jgg@lcc.uma.es

Abstract. In this article we present how to implement fuzzy databases based on the relational model. This approach includes many fuzzy attribute types, which can express the most of fuzzy knowledge types. These fuzzy attribute types include imprecise attributes, fuzzy attributes associated with one or more attributes, or with an independent meaning. In order to represent such fuzzy information we must study two aspects of fuzzy information: how to represent fuzzy data and how to represent fuzzy metaknowledge data. This second information is very important and it must be considered in any fuzzy database. This article studies the fuzzy metaknowledge data for any fuzzy attribute and how to represent both in a relational database. Finally, we apply all of this in a real example in the context of medical appointments.

Keywords: Fuzzy relational databases, Fuzzy attributes, Fuzzy degrees, Fuzzy metaknowledge, Representation of Fuzzy Knowledge.

1. Introduction

The relational model was developed by E.F. Codd of IBM and published in 1970. This model is the most used at present. In fact, actually the 90% of dabatases are relational. At a theoretical level, there exists many Fuzzy Relational Database models [1][6][11][14][15][16] that, based on the relational model, they extend it in order to allow storing and/or treating vague and uncertain information.

On the other hand, the FuzzyEER model [7][8][12][15] is an extension of the EER (Enhanced Entity-Relationship) model to create conceptual schemas with fuzzy semantics and notations. This extension is a good eclectic synthesis among the different models and it provides new and useful definitions: fuzzy attributes, fuzzy entities, fuzzy relationships, fuzzy specializations, etc.

In this paper we propose to incorporate the FuzzyEER concepts in a relational DBMS (DataBase Management System). Our aim is to present this extension as simple and useful as possible. Then, we have implemented the FIRST-2 definitions [7], which is based on [9][10]. FIRST-2 is the structure to represent fuzzy data and fuzzy metaknowledge data, and it has been used in some applications [2][4][5][13]. Some other approaches with this objective, like [3], are focused on queries, instead of in general representation issues.

The next section defines the fuzzy attributes included in the FuzzyEER model [7]. After, we define how to represent fuzzy data and how to represent fuzzy metaknowledge data, following the FIRST-2 definitions. The following section show a practical example implemented using PostgreSQL (www.postgresql.org), a powerful, open source object-relational database system, running on all major operating systems. Finally, concluding remarks and future developments are discussed.

2. Fuzzy Attributes

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.

2.1. Fuzzy Sets as Fuzzy Values

These fuzzy attributes may be classified in four types, based on the definitions of [7]. 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, they can have linguistic labels defined over them and we can use them in fuzzy queries. This type of attribute is represented in the same way as precise data, but can be transformed or manipulated using fuzzy conditions. This type is useful for extending traditional databases allowing fuzzy queries to be made about classic data. For example, enquiries of the kind "Give me employees that earn a lot more than the minimum salary".
- **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 extended Type 1 attribute allowing the storage of imprecise information, such as: "he is approximately 2 metres tall". For the sake of simplicity the most complex of these fuzzy sets are supposed to be a trapezoidal function (Figure 1).
- **Type 3**: They are attributes over "*data of discreet non-ordered dominion with analogy*". In these attributes some labels are defined ("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/brown), which expresses that a certain person is more likely to be dark than brown-haired. Note that underlying domain of these fuzzy sets are the set of labels and this set is non-ordered.
- **Type 4**: These attributes are defined in a similar way like Type 3 attributes, without it being necessary for a similarity relationship to exist between the labels. In this case, we suppose that we do not need the similarity relationship (or it does not exist).

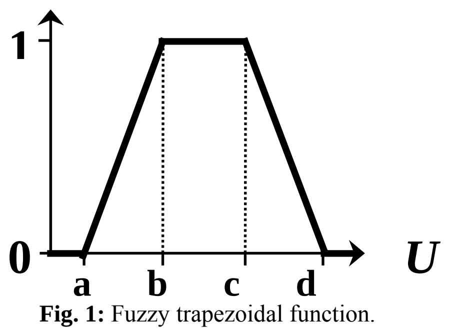


Fig. 1: Fuzzy trapezoidal function.

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 [6][7][16]: Fulfilment degree, Uncertainty degree, Possibility degree and Importance degree. FIRST-2 allows the user to define and use other meanings.

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

Associated degrees are associated with a specific value to which they incorporate imprecision. These degrees may be associated with different concepts [7]:

- **Degree in each value of an attribute (**we will call it as **Type 5):** Some attributes may have a fuzzy degree associated with them. This implies that each value of this attribute (in every tuple or instance) has an associated degree, that measures the level of fuzziness of 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 with some attributes. Whilst this is an unusual case, it can sometimes be very useful. 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 with 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.

Non-associated degrees (Type 8) are useful when 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.

In this paper we do not aim to demonstrate the usefulness of these degrees with their different meanings. Several authors who have used these degrees have already done so [1][6][14][16].

3. Representation of Fuzzy Attributes

This representation is different according to the fuzzy attribute Type. Fuzzy attributes Type 1 are represented as usual attributes, because they do not allow fuzzy values. Fuzzy attributes Type 2 need five classic attributes: One stores the kind of value (Table 1) and the others four store the crisp values representing the fuzzy value. Note, in Table 1, that trapezoidal fuzzy values (Figure 1) need the others four values. An approximate value (approximately *d*, *d*±margin) is represented with a triangular function centered in *d* (degree 1) and with degree 0 in *d*–margin and *d*+margin, where value margin depend on the context (Figure 1 with b = c and b–a = d–c = margin).
Fuzzy attributes Type 3 need a variable number of attributes: One stores the kind of value (Table 2). Note, in Table 2, that number 3 needs only two values (the degree and the label identifier), but number 4 needs 2*n* values, where *n* is the maximum length for possibility distributions for each fuzzy attribute. Value *n* must be defined for each fuzzy attribute Type 3, and it must be stored in the FMB (see following section).

Table 1: Kind of values of fuzzy attributes Type 2.

Number	Kind of values
0, 1, 2	**UNKNOWN, UNDEFINED, NULL**
3	**CRISP:** *d*
4	**LABEL:** label_identifier
5	**INTERVAL:** [*n*,*m*]
6	**APPROXIMATE VALUE:** *d*
7	**TRAPEZOIDAL:** [a,b,c,d]

Table 2: Kind of values in Type 3 and 4.

Number	Kind of values
0, 1, 2	**UNKNOWN, UNDEFINED, NULL**
3	**SIMPLE:** Degree/Label
4	**POSSIBILITY DISTRIBUTION:** $\text{Degree}_1/\text{label}_1 + ... + \text{Degree}_n/\text{Label}_n$

Fuzzy attributes Type 4 are represented just like Type 3. The differences between them are 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].

4. Representation of Fuzzy Metaknowledge Data: The FMB

Fuzzy metaknowledge data are the necessary knowledge about the fuzzy database (fuzzy attributes specially). This information is stored in relational format in the so-called FMB (Fuzzy Metaknowledge Base). First, we define the information stored in the FMB, and then we explain the structure of it (i.e., the relations of the FMB).

1. Attributes with fuzzy capabilities: fuzzy attributes and degrees (Types 1 to 8).
2. The metaknowledge of each attribute is different according to its type: Types 1 and 2 store in the FMB the definition (fuzzy set) of each linguistic label, the "margin" for approximate values, and the minimum distance to consider two values as very separated (so-called "much" and used in comparisons like "much greater than"). Types 3 and 4 need the value *n* (explained above), name of linguistic labels and, only for Type 3, the similarity relationship between whatever two labels. Types 5 and 6 store the meaning of the degree and attribute (Type 5) or attributes (Type 6) to which the degree is associated. Types 7 and 8 only store the meaning.
3. Other objects: These objects include fuzzy qualifiers (associated with an attribute and used to set the threshold in queries) and fuzzy quantifiers (associated with a relation or to an attribute). Fuzzy quantifiers are used in queries (for example "Give me employees who belong to *most* of projects"), and also in fuzzy constraints (for example "An employee must work in *many* projects").

If two fuzzy attributes (Types 1, 2, 3 or 4) need the same definitions we can register these two attributes as compatibles. This simplifies data in the FMB.

Figure 2 shows the FMB relations (or tables), their attributes, their primary keys (underlined) and their foreign keys (with arrows). We use OBJ# as the relation identifier, and COL# as the column or attribute identifier (just like Oracle). We cannot explain all attributes of all FMB relations for lack of space. Then we only try to give an idea about the usefulness of each relation:

- **FUZZY_COL_LIST:** It describes fuzzy attributes identified by (OBJ#,COL#). F_TYPE set the fuzzy type (from 1 to 8). LEN is the value *n*. CODE_SIG indicates the degree meaning when F_TYPE$\in$[5,8].
- **FUZZY_DEGREE_SIG:** It stores all the degree meanings of our database.
- **FUZZY_OBJECT_LIST:** This relation contains declarations of fuzzy objects related with fuzzy attributes. These fuzzy objects are: linguistic labels, qualifiers and fuzzy quantifiers. Fuzzy quantifier may be absolute or relative, and may have one or two arguments.
- **FUZZY_LABEL_DEF:** It defines the linguistic labels using trapezoidal functions (Figure 1).
- **FUZZY_APPROX_MUCH:** Values "margin" and "much" for Types 1 and 2.
- **FUZZY_NEARNESS_DEF:** Similarity relation-ships for Type 3.
- **FUZZY_COMPATIBLE_COL:** Compatible fuzzy attributes, i.e., attributes which use the same linguistic labels.
- **FUZZY_QUALIFIERS_DEF:** It defines fuzzy qualifiers.
- **FUZZY_DEGREE_COLS:** This relation sets the attributes (or columns) associated with fuzzy degrees (only for Type 5 and 6). Note that a Type 5 degree has only one associated attribute, a Type 6 degree has some attributes and an attribute may have many degrees associated with it (but these degrees must be Type 5 or 6). Of course Type 7 and 8 degrees do not use this table.

- **FUZZY_ER_LIST:** Using FuzzyEER words, this relation stores fuzzy entities and fuzzy relationships. DEGREE_TYPE take ‘M’ for fuzzy entities, ‘C’ for fuzzy entities with degrees computed automatically, ‘E’ and ‘I’ for fuzzy weak entities (dependency on existence or dependency on identification) and, finally, ‘R’ for fuzzy relationships represented by the table OBJ#.
- **FUZZY_TABLE_QUANTIFIERS:** Definition of quantifiers associated with a relation or table (not to an attribute). These quantifiers are used in fuzzy constraints and they may be absolute or relative.

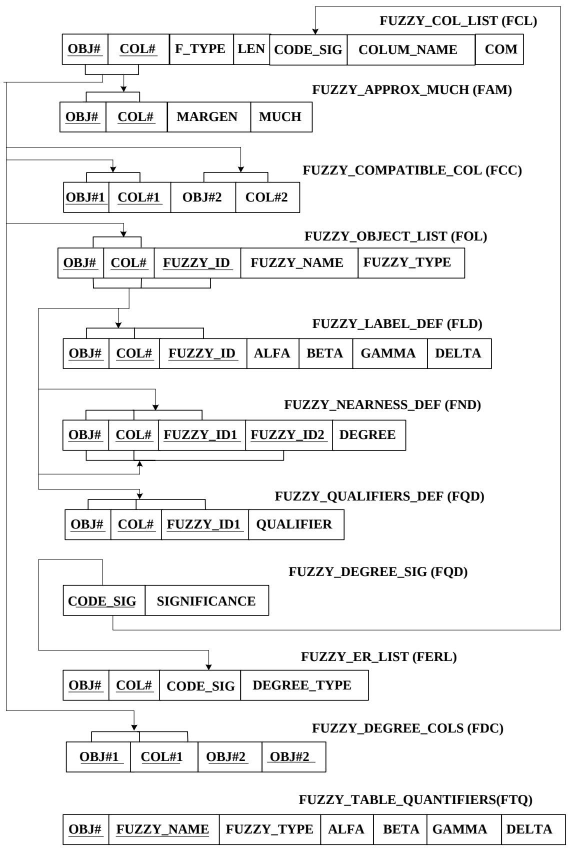
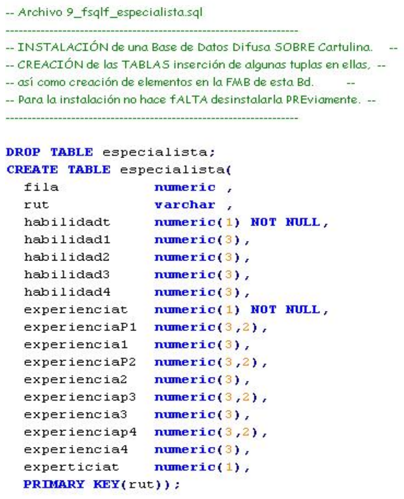

FUZZY_COL_LIST (FCL)
OBJ# COL# F_TYPE LEN CODE_SIG COLUM_NAME COM
FUZZY_APPROX_MUCH (FAM)
OBJ# COL# MARGEN MUCH
FUZZY_COMPATIBLE_COL (FCC)
OBJ#1 COL#1 OBJ#2 COL#2
FUZZY_OBJECT_LIST (FOL)
OBJ# COL# FUZZY_ID FUZZY_NAME FUZZY_TYPE
FUZZY_LABEL_DEF (FLD)
OBJ# COL# FUZZY_ID ALFA BETA GAMMA DELTA
FUZZY_NEARNESS_DEF (FND)
OBJ# COL# FUZZY_ID1 FUZZY_ID2 DEGREE
FUZZY_QUALIFIERS_DEF (FQD)
OBJ# COL# FUZZY_ID1 QUALIFIER
FUZZY_DEGREE_SIG (FQD)
CODE_SIG SIGNIFICANCE
FUZZY_ER_LIST (FERL)
OBJ# COL# CODE_SIG DEGREE_TYPE
FUZZY_DEGREE_COLS (FDC)
OBJ#1 COL#1 OBJ#2 OBJ#2
FUZZY_TABLE_QUANTIFIERS(FTQ)
OBJ# FUZZY_NAME FUZZY_TYPE ALFA BETA GAMMA DELTA

Fig. 2: FMB tables in FIRST-2.

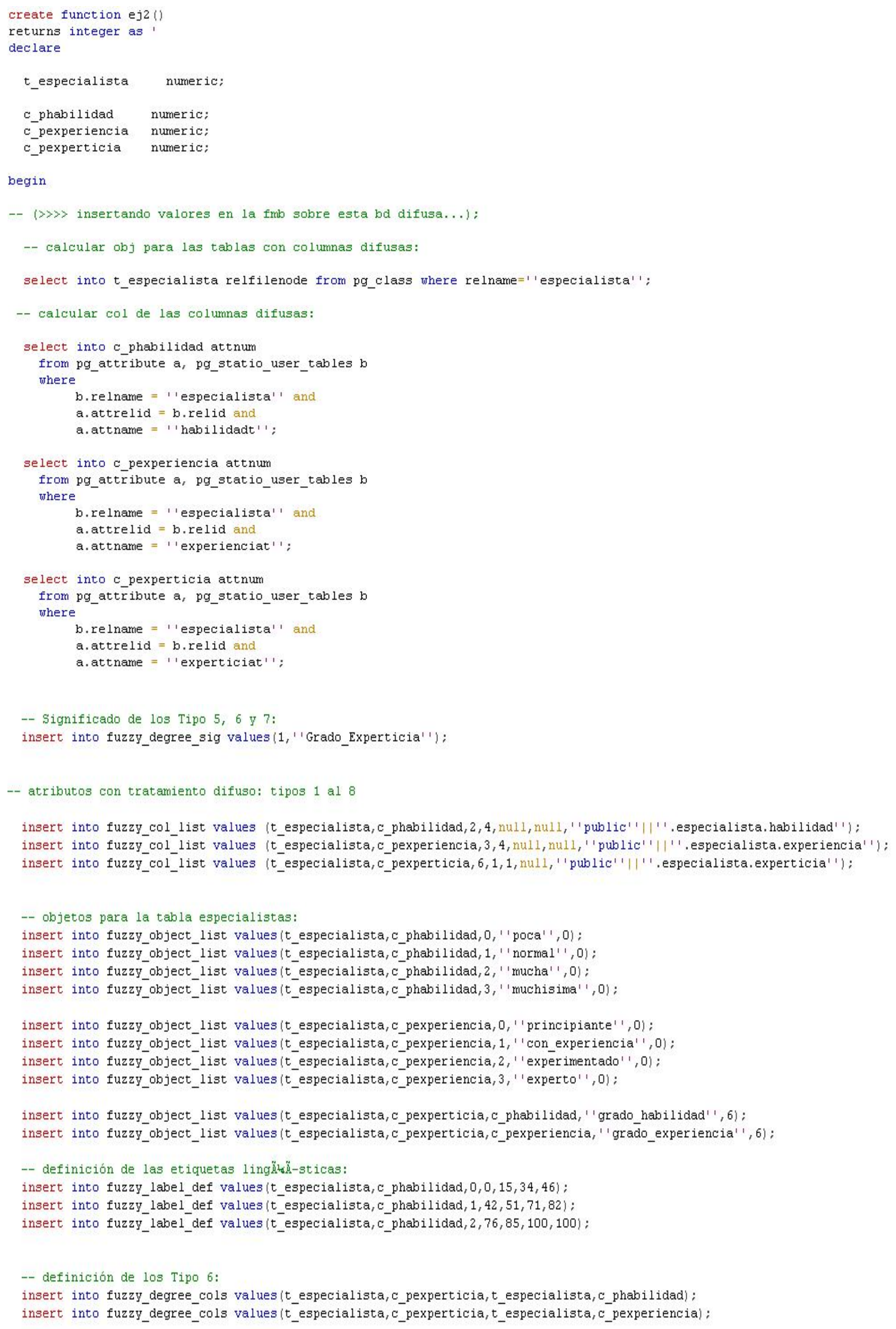
5. Entity Example Specialist

First, we designed the FuzzyEER model [7][15] for this application [13]. The Specialist entity stores information for professionals caring for patients in a health case (medical appointments). It is composed of two non fuzzy attributes (RUT and Name), a fuzzy datatype Type 2 "Ability" (skill), a Type 3 "Experience" attribute whose domain data are discrete and there is a relationship of similarity between them, and finally a Type 6 degree "Expertise", associated with "Experience" and "Ability". Then, we can use Fuzzy SQL, FSQL [6][7][14][16] in this application. An example of query could be: "Get the RUT, and the degree of specialist expertise of the entity whose degree is less than 0.2". In [13] a more complete explanation about this and other entities is presented.

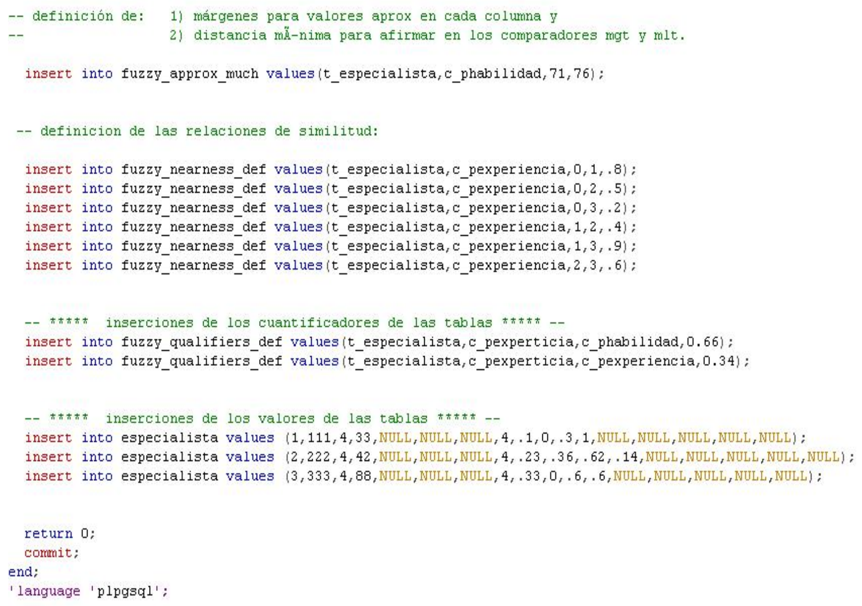
In order to define each entity, we must create the table and insert the corresponding values in the FMB tables. For example, the Specialist entity needs the following statements:



```
-- Archivo 9_fsqlf_especialista.sql
------------------------------------------------------------------
-- INSTALACIÓN de una Base de Datos Difusa SOBRE Cartulina.    --
-- CREACIÓN de las TABLAS inserción de algunas tuplas en ellas, --
-- así como creación de elementos en la FMB de esta Bd.        --
-- Para la instalación no hace fALTA desinstalarla PREviamente. --
------------------------------------------------------------------

DROP TABLE especialista;
CREATE TABLE especialista(
  fila           numeric ,
  rut            varchar ,
  habilidadt     numeric(1) NOT NULL,
  habilidad1     numeric(3),
  habilidad2     numeric(3),
  habilidad3     numeric(3),
  habilidad4     numeric(3),
  experienciat   numeric(1) NOT NULL,
  experienciaP1  numeric(3,2),
  experiencia1   numeric(3),
  experienciaP2  numeric(3,2),
  experiencia2   numeric(3),
  experienciap3  numeric(3,2),
  experiencia3   numeric(3),
  experienciap4  numeric(3,2),
  experiencia4   numeric(3),
  experticiat    numeric(1),
  PRIMARY KEY(rut));
```



```
create function ej2()
returns integer as '
declare

  t_especialista     numeric;

  c_phabilidad     numeric;
  c_pexperiencia   numeric;
  c_pexperticia    numeric;

begin

-- (>>>> insertando valores en la fmb sobre esta bd difusa...);

  -- calcular obj para las tablas con columnas difusas:

  select into t_especialista relfilenode from pg_class where relname=''especialista'';

 -- calcular col de las columnas difusas:

  select into c_phabilidad attnum
    from pg_attribute a, pg_statio_user_tables b
    where
         b.relname = ''especialista'' and
         a.attrelid = b.relid and
         a.attname = ''habilidadt'';

  select into c_pexperiencia attnum
    from pg_attribute a, pg_statio_user_tables b
    where
         b.relname = ''especialista'' and
         a.attrelid = b.relid and
         a.attname = ''experienciat'';

  select into c_pexperticia attnum
    from pg_attribute a, pg_statio_user_tables b
    where
         b.relname = ''especialista'' and
         a.attrelid = b.relid and
         a.attname = ''experticiat'';


  -- Significado de los Tipo 5, 6 y 7:
  insert into fuzzy_degree_sig values(1,''Grado_Experticia'');


-- atributos con tratamiento difuso: tipos 1 al 8

  insert into fuzzy_col_list values (t_especialista,c_phabilidad,2,4,null,null,''public''||''.especialista.habilidad'');
  insert into fuzzy_col_list values (t_especialista,c_pexperiencia,3,4,null,null,''public''||''.especialista.experiencia'');
  insert into fuzzy_col_list values (t_especialista,c_pexperticia,6,1,1,null,''public''||''.especialista.experticia'');


  -- objetos para la tabla especialistas:
  insert into fuzzy_object_list values(t_especialista,c_phabilidad,0,''poca'',0);
  insert into fuzzy_object_list values(t_especialista,c_phabilidad,1,''normal'',0);
  insert into fuzzy_object_list values(t_especialista,c_phabilidad,2,''mucha'',0);
  insert into fuzzy_object_list values(t_especialista,c_phabilidad,3,''muchisima'',0);

  insert into fuzzy_object_list values(t_especialista,c_pexperiencia,0,''principiante'',0);
  insert into fuzzy_object_list values(t_especialista,c_pexperiencia,1,''con_experiencia'',0);
  insert into fuzzy_object_list values(t_especialista,c_pexperiencia,2,''experimentado'',0);
  insert into fuzzy_object_list values(t_especialista,c_pexperiencia,3,''experto'',0);

  insert into fuzzy_object_list values(t_especialista,c_pexperticia,c_phabilidad,''grado_habilidad'',6);
  insert into fuzzy_object_list values(t_especialista,c_pexperticia,c_pexperiencia,''grado_experiencia'',6);

  -- definición de las etiquetas lingÃ¼Ãsticas:
  insert into fuzzy_label_def values(t_especialista,c_phabilidad,0,0,15,34,46);
  insert into fuzzy_label_def values(t_especialista,c_phabilidad,1,42,51,71,82);
  insert into fuzzy_label_def values(t_especialista,c_phabilidad,2,76,85,100,100);


  -- definición de los Tipo 6:
  insert into fuzzy_degree_cols values(t_especialista,c_pexperticia,t_especialista,c_phabilidad);
  insert into fuzzy_degree_cols values(t_especialista,c_pexperticia,t_especialista,c_pexperiencia);
```



```
-- definición de:   1) márgenes para valores aprox en cada columna y
--                  2) distancia mÃnima para afirmar en los comparadores mgt y mlt.

  insert into fuzzy_approx_much values(t_especialista,c_phabilidad,71,76);


 -- definicion de las relaciones de similitud:

  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,0,1,.8);
  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,0,2,.5);
  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,0,3,.2);
  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,1,2,.4);
  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,1,3,.9);
  insert into fuzzy_nearness_def values(t_especialista,c_pexperiencia,2,3,.6);


  -- *****  inserciones de los cuantificadores de las tablas ***** --
  insert into fuzzy_qualifiers_def values(t_especialista,c_pexperticia,c_phabilidad,0.66);
  insert into fuzzy_qualifiers_def values(t_especialista,c_pexperticia,c_pexperiencia,0.34);


  -- *****  inserciones de los valores de las tablas ***** --
  insert into especialista values (1,111,4,33,NULL,NULL,NULL,4,.1,0,.3,1,NULL,NULL,NULL,NULL,NULL);
  insert into especialista values (2,222,4,42,NULL,NULL,NULL,4,.23,.36,.62,.14,NULL,NULL,NULL,NULL,NULL);
  insert into especialista values (3,333,4,88,NULL,NULL,NULL,4,.33,0,.6,.6,NULL,NULL,NULL,NULL,NULL);


  return 0;
  commit;
end;
'language 'plpgsql';
```

6. Conclusions and Future Lines

This article presents how to store fuzzy knowledge of fuzzy databases in a classic relational database. This allows us to implement fuzzy databases [6][11] modeled with the FuzzyEER model [7]. It should be stressed that these fuzzy attributes types can express the most of fuzzy knowledge types. This research studies how to represent fuzzy data, what is the necessary metaknowledge about these fuzzy data, and how to represent this fuzzy metaknowledge data. This second information is very important and must be considered in any fuzzy database.

At present, different fuzzy databases has been developed with some of these characteristics [1][2][4][6][13] and with the main target of fuzzy queries. A very good review about fuzzy queries can be found in [16].

Finally, we apply all of this in a real example in the context of medical appointments. We only define here one entity, the Specialist entity, showing some details about its definition. Besides, FSQL (Fuzzy SQL) language [5][6][7][14] may be used in this database.

This work can be mixed with others in the Data Mining area [4] in order to achieve a more complete implementation. In [17] we can find an useful introduction to fuzzy Data Mining methods that should be right for this purpose.

Acknowledgments. This work has been partially supported by the “Ministry of Education and Science” of Spain (projects TIN2006-14285 and TIN2006-07262) and the Spanish “Consejería de Innovación Ciencia y Empresa de Andalucía” under research project TIC-1570. Thanks to the Catholic University of Maule by internal project support DINP 2008-2010.

References

1. Barranco, C.D., Campaña, J.R., & Medina, J.M. (2008). Towards a Fuzzy Object-Relational Database Model. In Handbook of Research on Fuzzy Information Processing in Databases, Vol. II, pp. 435-461. Information Science Reference (http://www.info-sci-ref.com).
2. Blanco I., Cubero J.C., Pons O., Vila M.A.: An Implementation for Fuzzy Deductive Relational Databases. In Recent Issues on Fuzzy Databases, Ed. G. Bordogna and G. Pasi. Physica-Verlag (Studies in Fuzziness and Soft Computing), pp. 183-207, 2000.
3. Bosc, P., Galibourg, M.: Indexing principles for a fuzzy data base. Inf. Systems, Vol. 14-6, pp. 493-499, 1989.
4. Carrasco, R.A., Araque, F., Salguero, A., Vila, M.A.: Applying Fuzzy Data Mining to Tourism Area. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases, Vol. II, pp. 563-584. Hershey, PA, USA: Information Science Reference (http://www.info-sci-ref.com), 2008.

5. Galindo, J., Medina, J.M., Pons, O., Cubero, J.C.: A Server for Fuzzy SQL Queries. In Flexible Query Answering Systems, Eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in Artificial Intelligence (LNAI) 1495, pp. 164-174. Ed. Springer, 1998.
6. Galindo, J. (Ed.): Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science Reference (http://www.info-sci-ref.com), 2008.
7. Galindo, J., Urrutia, A., Piattini, M.: Fuzzy Databases: Modeling, Design and Implementation. Idea Group Publishing Hershey, USA, 2006.
8. Galindo, J., Urrutia, A., Carrasco, R.A., Piattini M.: Relaxing Constraints in Enhanced Entity-Relationship Models using Fuzzy Quantifiers. IEEE Transactions on Fuzzy Systems, Vol. 12(6), pp. 780-796, Dec. 2004.
9. Medina J.M.: Bases de datos Relacionales Difusas: Modelo Teórico y Aspectos de su Implementación, Ph. Doctoral Thesis, Universidad de Granada, España, 1994 (www.decsai.ugr.es).
10. Medina J.M., Pons O., Vila A.: FIRST. A Fuzzy Interface for Relational SysTems. VI International Fuzzy Systems Association World Congress (IFSA 1995). Sao Paulo (Brasil), 1995.
11. Petry F.E.: Fuzzy Databases: Principles and Applications. *International Series in Intelligent Technologies*. Ed. H.J. Zimmermann. Kluwer Academic Publ. (KAP), 1996.
12. Urrutia A., Galindo J., Piattini M.: Modeling Data Using Fuzzy Attributes. Proceedings published by IEEE Computer Society Press of the XXII International Conference of the Chilean Computer Science Society (SCCC 2002), pp. 117-123. Copiapo (Chile), 2002. ISBN: O-7695-1867-2.
13. Urrutia, A., Galindo J., Sepúlveda, A.: Implementación de una base de datos difusa con FIRST-2 y PostgreSQL. XV Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2010, Huelva, Spain, 2010.
14. Urrutia, A., Tineo, L., Gonzalez, C.: FSQL and SQLf: Towards a Standard in Fuzzy Databases. In Handbook of Research on Fuzzy Information Processing in Databases, Vol. I, pp. 270-298. Information Science Reference (http://www.info-sci-ref.com), 2008.
15. Urrutia, A., & Galindo, J. Fuzzy Database Modeling: An Overview and New Definitions. In Anbumani, K., & Nedunchezhian, R. (eds.), Soft Computing Applications for Database Technologies: Techniques and Issues, pp. 1–21. Information Science Reference, IGI Global, Hershey, PA, USA, 2010.
16. Zadrożny, S., de Tré, G., de Caluwe, R., Kacprzyk, J.: An Overview of Fuzzy Approaches to Flexible Database Querying. In Handbook of Research on Fuzzy Information Processing in Databases, Vol. I, pp. 34-54. Hershey, PA, USA: Information Science Reference (http://www.info-sci-ref.com), 2008.
17. Feil, B., Abonyi, J.: Introduction to Fuzzy Data Mining Methods. In Handbook of Research on Fuzzy Information Processing in Databases, Vol. I, pp. 55-95. Hershey, PA, USA: Information Science Reference (http://www.info-sci-ref.com), 2008.