Flexible Query Languages for Relational Databases: An Overview

Antonio Rosado*, Rita A. Ribeiro**, Slawomir Zadrozny***, Janusz Kacprzyk***

*Universidade do Algarve, Campus de Gambelas, 8000 Faro, Portugal, arosado@ualg.pt

**UNINOVA, Campus of New University of Lisbon, Caparica 2929-516 Portugal, rar@uninova.pt

***Systems Research Institute, Polish Academy of Sciences, 01-447 Waszawa, Poland, {kacprzyk,zadrozny}@ibspan.waw.pl

Abstract

We present an overview of the most important proposals for human-oriented query languages for relational databases, based on fuzzy sets theory. To highlight important issues concerning communication with databases, we propose two taxonomies: the first taxonomy deals with flexible query languages in crisp relational databases and the second deals with flexible query languages in fuzzy relational databases. They can help database designers and users understand and select the best approaches to solve their problems.

Key Words: fuzzy querying, flexible querying, relational database management systems, imprecise information, fuzzy logic.

1 Introduction

Managers rely more and more on the use of databases to obtain insights and updated information on activities of their institutions and companies. More and more people, from experts to non-experts, are depending on information from databases, to fulfill everyday tasks, notably those related to decision making. Basi-

cally, the content of a database describes selected aspects of the *real world* relevant for a given company, institution, etc. Often, our knowledge about the entities represented in a database as well as our preferences as to what should be retrieved from a database are imperfect or imprecise. This raises a question of a proper modeling of imperfect information in the context of *database management systems* (DBMSs).

The focus of this paper is on flexible query languages (FQL) for databases that are based on fuzzy sets theory. Since there are many contributions in this field, we propose two taxonomies to help and guide database designers and users. These taxonomies address the FQL in crisp relational databases and in fuzzy relational databases, respectively. Approaches mentioned in these taxonomies are not exhaustive in terms of literature, which is huge, but they are quite representative. We believe that these two taxonomies provide a better understanding of the field and can help select the best approaches to solve specific problems.

Dubois and Prade (Dubois and Prade 1997) enumerate the two reasons for using fuzzy sets theory (Zadeh 1965) to make querying more flexible. First, fuzzy sets provides a better representation of the user's *preferences*. For example, in a query asking for some apartment "not too expensive and not too far from downtown", the user may feel much more comfortable using linguistic terms instead of precisely specified numerical constraints. Moreover, these linguistic terms express exactly what the preferences of a user are; for example when an interval to which the price has to belong is imprecisely specified. The linguistic terms clearly suggest that there is a smooth transition between acceptable and unacceptable prices. Thus, we can have a price definitely *matching* or definitely *not matching* the user's request, but also *matching to a certain degree*. Another important aspect of using fuzzy sets theory is a direct consequence of the previous. Namely, as soon as we have a *matching degree*, answers can be ranked according to the users' requirements (Dubois and Prade 1997).

According to many authors such as Bosc and Pivert (Bosc and Pivert 1992, 1997), Kacprzyk and Zadrozny (Kacprzyk and Zadrozny 1995), Takahashi (Takahashi 1995), Medina et al. (Medina et al. 1994), etc. there are two main lines of research in the use of fuzzy set theory in the DBMS context. The first one assumes a conventional database and, essentially, develops a fuzzy querying interface using fuzzy sets, possibility theory, fuzzy logic, etc. Among authors who have contributed to this research are Bosc and Pivert (Bosc and Pivert 1992, 1995; Bosc et al. 1999), Dubois and Prade (Dubois and Prade 1997), Tahani (Tahani 1977), Takahashi (Takahashi 1991, 1995), Kacprzyk, Zadrozny and Ziolkowski (Kacprzyk and Ziolkowski 1986; Kacprzyk, Zadrozny and Ziolkowski 1989; Kacprzyk and Zadrozny 1995), Ribeiro and Moreira (Ribeiro and Moreira 1999). In Section 4, we will describe their works and propose a taxonomy for these approaches. The second line of research uses fuzzy or possibilistic elements for developing a fuzzy database model that accounts for imprecision and vagueness in data. Here, of course, querying constitutes also an important element of a model. Some relevant concepts are presented, e.g., in (Baldwin et al. 1993; Bosc and

Pivert 1997; Bosc and Pivert 1997; Buckles and Petry 1985; Buckles et al. 1986; Galindo et al. 1998; Galindo et al. 1999; Medina et al. 1994; Prade and Testemale 1984; Prade and Testemale 1987; Shenoi and Melton 1989; Shenoi et al. 1990). These approaches are described in Section 5.

There are also other issues in the use of fuzzy sets theory in relational databases such as efficiency of fuzzy queries execution; fuzzy functional dependencies/constraints, fuzzy logical databases, but they are beyond the scope here.

In Section 2, we review the fundamental concepts of the relational data model which includes its main querying formalisms: the relational algebra, the relational calculus and the SQL language. Next, in Section 3, we review the main concepts of fuzzy sets theory that will be used in this paper. Sections 2 and 3 provide the theoretical base to make the paper self-contained. We examine in detail the main approaches proposed in the literature that are concerned with the first line of research mentioned above, i.e. flexible query languages for the crisp relational data model. We propose a taxonomy to organize these approaches, thus resulting in an overall picture of the main research done. The last part, Section 5, is devoted to the second line of research mentioned above, i.e. the flexible query languages for fuzzy relational databases. Again, a taxonomy for different approaches is proposed. Finally, in Section 6, we present some conclusions about this work.

2 Brief introduction to the relational data model

A relational database is a collection of relations, defined according to the relational database model (Codd 1970). A relation may be understood as the *relation schema* or the *relation instance*. The relation schema has the following form:

$$R(A_1{:}D_1, \ldots, A_n{:}D_n) \tag{1}$$

where R is the name of the relation, A_i ($1 \le i \le n$) is the i-th *attribute* (which also may be called *column* or *field*) and D_i ($1 \le i \le n$) is the *domain* corresponding to the attribute A_i. Each D_i defines a set of values being possible values for an attribute. Often, we refer to a schema by just indicating the set of attribute names $R(X)$, $X = \{A_i, \ldots A_n\}$.

A relation instance of given relation schema is a set of *tuples*, each composed of values of the attributes belonging to the relation schema:

$$\{< d_1, \ldots, d_n> \mid (d_1 \in D_1), \ldots, (d_n \in D_n)\} \tag{2}$$

where d_i ($1 \le i \le n$) is the value of the tuple corresponding to attribute A_i; this value must belong to the set D_i. Usually the term *relation instance* is abbreviated to *relation* whenever there is no confusion with other aspects of the relation.

A relation is here denoted R, its n attributes are denoted $A_1, \ldots, A_n$ and $D_1, \ldots, D_n$ are their domains. The m tuples of a relation are denoted $t_1, \ldots, t_m$ and d_{ij} repre-

sents the value of the *j*-th attribute in tuple t_i. Relations are often referred to as *tables*, with *columns* and *rows* corresponding to attributes and tuples of the relation.

The typical operations on data in a database include *insertion*, *deletion*, *updating* and *retrieval*. The latter is the most important for our considerations. Usually not all data is retrieved but only the data *matching* certain criteria is required. From the operational point of view there are two approaches to devising a *query language*, i.e., a language allowing to express the range of required data. The first one consists in the use of a restricted version of the predicate/relational calculus of mathematical logic to specify the requirements the retrieved data should meet. In this approach the actual way of data retrieval is completely left to the database management system employed. In the second approach, a query is a sequence of operations that should be executed on the database to obtain the required data. These operations correspond to the underlying data model, i.e., correspond to the operations in relational algebra. From the theoretical point of view both approaches are equivalent in the sense of their expressive power. In Section 2, we present a brief overview of both formalisms. In relational database management systems implementations usually a hybrid approach is adopted, notably exemplified by the SQL language.

Whichever querying approach is assumed, the most important part of a query is a set of conditions (criteria) of which rows will be *selected* to be included in an answer to the query. Thus, it is interesting to study the retrieval process from the perspective where a query is meant to define a prototype of data to be retrieved. Then, during the retrieval process for every row a *matching degree* of its content and the prototype is calculated. In the classical crisp approach this matching degree is binary: a row matches the prototype or not. In real situations, the description of the prototype may be imprecise and this leads to a partial matching degree. This line of reasoning, adopted by many authors, provides an interesting basis for the analysis of flexible (fuzzy) querying languages, which we will explore next.

To clarify the understanding of some proposals in the literature, we will use, whenever necessary, the following relational scheme example:

Employees(#emp, name, #dep, age, job, salary, commission, town) (3)

Departments(#dep, budget, size, city) (4)

with the instances of relations *Employees* and *Departments* given in Fig. 1:

This example will also make possible to explain, in a simple way, the main concepts of relational algebra and relational calculus.

2.1 Relational Algebra

Relational algebra defines a set of operations to manipulate data in relations (Ramakrishnan and Gehrke 2000). The list of basic operations includes:

- the usual set theoretic operations including the *union*, *difference* and *Cartesian product*. The Cartesian product is slightly modified to fit the database context and produces a relation whose scheme is a union of the schemes of the argument relations.
- the *selection*, $\sigma_P(R)$, gives the tuples of relation R that satisfy a Boolean expression P, which is defined over the scheme of R.
- the *projection*, $\pi_Y(R)$, gives a relation obtained when all attributes from the set X-Y are removed, where $R(X)$ is the scheme of the relation R. Thus, the scheme of the resulting relation comprises only a subset Y of the set of attributes of R and some tuples of the original relation are also removed (those with identical values of the attributes belonging to Y).

#dep	budget	size	city
1	$130000	30	S. Francisco
7	$90000	20	New York
9	$110000	11	Washington
8	$50000	7	S. Francisco
3	$40000	11	New York

#emp	name	#dep	age	job	sal	commission	town
22	Arthur	1	30	programmer	$1500	$500	New York
29	John	3	35	accounter	$1800	$400	Boston
31	Mary	7	40	sales manager	$2300	$200	San Francisco
32	Peter	1	39	systems analyst	$2000	$300	New York
58	Barbara	7	39	marketing manager	$2500	$500	Los Angeles
64	Mary	7	27	product manager	$2000	$400	New York
71	Michael	8	30	research assistant	$1500	$100	S. Diego
74	Jude	9	35	secretary	$1600	$300	New York
85	Horatio	9	50	technical assistant	$2400	$400	New York
95	Ken	3	55	controller	$2800	$500	Boston

Fig. 1. Example of a crisp relational database

Other operations can also be used but they could be expressed in terms of the five operations mentioned above. For example, the popular *join* operation, $join_{A\theta_B}(R,S)$, is a combination of the Cartesian product and the selection operation. Another operation, often discussed within the subject of "fuzzification", is the *division* $R \div S$. If X and Y are the schemes of R and S, respectively, with $Y \subset X$, then $R \div S$ gives the maximal (in the sense of "$\subset$") relation T, having the scheme X-Y, such that $T \times S \subset R$, i.e.,:

$$R \div S = \{t : \forall u \in S, (t,u) \in R\} \quad (5)$$

The division operation may also be expressed as a combination of the projection, Cartesian product and difference operations.

We complete this section by providing an example of a query expressed in the relational algebra. The query (Bosc and Pivert 1995):

Q1 - *Find the employees younger than 35 who work in a department whose budget is higher than $100000*

may be expressed in relational algebra as:

$$\pi_{\#emp}(join_{Emp.\#dep=Dep.\#dep}(\sigma_{age<35}(Emp), \sigma_{budget>100000}(Dep))) \quad (6)$$

2.2 Relational Calculus

Relational calculus comprises domain relational calculus (DRC) and tuple relational calculus (TRC) (Ramakrishnan and Gehrke 2000). DRC and TRC are declarative query languages based on the first-order predicate logic. The main idea is to *describe* what is sought rather than to define how to get it, in a similar fashion as in the relational algebra.

Next, we briefly review the DRC language because it is the base for several flexible query languages (FQLs) described in Sections 4 and 5, respectively.

A DRC query is an expression based on the first order predicate calculus language. The general form of such a query is as follows:

$$\{(x_1,\ldots,x_n) \mid \phi(x_1,\ldots,x_n)\} \quad (7)$$

where $\phi(x_1,\ldots,x_n)$ is a formula of the language. An answer to such a query is a set (possibly empty) of tuples $(a_1,\ldots,a_n)$ such that when substituting a_i's for x_i's in ϕ a true formula is obtained. The building blocks of formulae are *atomic formulae*. In the DRC two classes of atomic formulae are usually distinguished:

$$\begin{gathered} R(x_1, x_2, \ldots, x_n) \\ x_1 \,\theta\, x_2 \end{gathered} \quad (8)$$

where x_i $(1 \le i \le n)$ is either a domain variable or a constant; R is an n-ary predicate ; θ is a comparison operator from the set $\{<, >, =, \le, \ge, \ne\}$.

Finally, a formula is:

- an atomic formula, (9)
- $\neg\psi$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \Rightarrow \psi_2$
- $\exists x\ (\psi(x))$
- $\forall x\ (\psi(x))$

where ψ, ψ_1, ψ_2 are formulas and $\psi(x)$ is a formula containing a domain variable x. An example of a DRC query equivalent to the example of the relational algebra query of Section 2.1 (see Eq. 6) is the following:

$$\left\{(x_1) \mid \exists x_2, x_3, x_4, x_5, x_6 \left(\begin{array}{l} EMP(x_1, x_2, x_3, x_4, ...) \ \wedge \ x_4 < 35 \ \wedge \\ DEP(x_5, x_6, ...) \ \wedge \ x_6 > 100000 \ \wedge \ x_5 = x_3 \end{array} \right) \right\} \quad (10)$$

In the tuple relational calculus (TRC) we use the syntax similar to Eqs. (8)-(9) but replacing the domain variables with tuple variables. For example, (10) may be expressed in TRC as:

$$\{t \mid t \in \text{Employees} \wedge t.\text{age} < 35 \wedge \exists\, s \in \text{Departments}\ (s.\#\text{dep} = t.\#\text{dep} \wedge s.\text{budget} > 100000)\} \quad (11)$$

2.3 SQL

SQL is a de facto industry standard command language for the relational database management systems. SQL has commands to deal with all aspects concerning the creation, maintenance and use of a database such as the creation of tables, insertion of rows, querying the database, security issues, etc. In this section, we are only interested in what SQL provides for querying a database.

The syntax of a basic query in SQL is (Ramakrishnan and Gehrke 2000):

```
SELECT  select_list                                    (12)
FROM from_list
WHERE conditions
```

Such a query retrieves required data from some tables and builds a new table. The *select_list* specifies the expressions (often, just column names) whose values are to populate columns of the new table. The columns used in these expressions have to be listed in the *from_list*. The expressions of the SELECT clause are calculated only for the rows, from the FROM clause, that meet the *conditions* specified in the WHERE clause. The condition is a Boolean combination of atomic conditions created using the logical connectives AND, OR, NOT. Each atomic condition has the following syntax: “expression op expression”, where “op” is one of the comparison operators (<, <=, =, <>, >=, >) and “expression” is a column name, a constant or an expression (numeric expression or string expression).

A basic query (12) corresponds to an expression in relational algebra involving the operations of projection, selection and Cartesian product. For example, the following query:

Q2 - *Find the names and age of all employees*, is expressed in SQL as:

```
SELECT name, age                                    (13)
FROM Employees
```

and the result is a table with two columns (name,age), and as many rows as in the table *Employees*.

A way to extend the basic form of a query is to use *nested queries* (subqueries). Usually, the nested queries are used along with the set operators IN, NOT IN, EXISTS, NOT EXISTS, op ANY, op ALL as exemplified by the query:

Q3 - *Find the names of employees who work in New York* may be expressed as:

```
SELECT name                                         (14)
FROM Employees
WHERE #dep IN (SELECT #dep
                    FROM Departments
                    WHERE city = 'New York')
```

The subquery retrieves the set of departments that are located in New York. The main query retrieves the names of employees such that their department is in this set. The set operator IN allows to test whether a value belongs to a set or not.

3 A brief introduction to fuzzy sets theory

Fuzzy sets theory (Zadeh 1965) is an attempt to model an inherent vagueness of natural language. Almost any concept expressed in natural language, like young people, implies that elements of the universe of discourse (the particular people) are young to a *certain degree*. Note that the concept of *young* is context dependent. Such a graduality is modeled by a membership function μ_A that for each person x assigns a value $\mu_A(x)$ from the interval [0,1], representing the degree to which person x belongs to set A. In our example, $\mu_{YOUNG}(x)$ represents the degree to which x is considered young.

The cardinality $|A|$ of a fuzzy set A, defined on a finite universe set X, is given by the sum of the membership values of all elements of X in A, and is sometimes called scalar cardinality to distinguish from other types of cardinality (see e.g. (Liu and Kerre 1998)):

$$|A| = \sum_{x \in X} \mu_A(x) \tag{15}$$

The relative cardinality ||*A*|| of a fuzzy set *A*, in a finite universe set *X*, is defined by:

$$\|A\| = \frac{|A|}{|X|} \tag{16}$$

where |*X*| is the cardinality of the universal set *X*.

The counterparts of the classic operations of the complement, union and intersection for fuzzy sets *A* and *B*, are defined as follows:

$$\mu_{\neg_A}(x) = 1 - \mu_A(x) \tag{17}$$

$$\mu_{A \cup B}(x) = \max [\mu_A(x), \mu_B(x)] \tag{18}$$

$$\mu_{A \cap B}(x) = \min [\mu_A(x), \mu_B(x)] \tag{19}$$

The classic correspondence of set theoretical operations and logical connectives is preserved. Thus, (17)-(19) provide also interpretation for the connectives of negation, disjunction and conjunction.

Several fuzzy implication operators have also been proposed in the literature (Fodor and Yager 2000). The most commonly used are:

$$\text{Kleene-Dienes:} \quad I(x,y) = \max(1 - x, y) \tag{20}$$

$$\text{Lukasiewicz:} \quad I(x,y) = \min(1,(1-x+ y)) \tag{21}$$

$$\text{Gödel:} \quad I(x,y) = \begin{cases} 1 & \text{if } x \le y \\ y & \text{otherwise} \end{cases} \tag{22}$$

$$\text{Goguen:} \quad I(x,y) = \min(y/x,1) \tag{23}$$

Fuzzy sets theory was conceived primarily as a formalism to represent the meaning of natural language expressions. In the following subsections we will briefly review some concepts relevant for this topic.

3.1 Linguistic variable

Basically, a linguistic variable is a variable assuming linguistic values instead of numerical values. Formally, a linguistic variable (Zadeh 1987) is a quintuple (*H, T, U, G, M*) where *H* is the name of the variable, *T* is the set of linguistic names

(called *terms*) that can be assigned to the variable; U is the universe of values that are used to define the meaning M of each linguistic value in T; and G is a grammar that is used to specify the values allowed in T. The meaning $M(X)$ of a term $X \in T$, is specified as a fuzzy subset in U. The terms may be *atomic terms* such as "*young*" or *composite terms*, which result, for example, from applying *modifiers* (see next subsection) and logical connectives to atomic terms. For example (Zadeh 1987), a linguistic variable called *age* (H = age) may have the term set T = {*old*, *very old*, *not old*, *more or less young*, *quite young*, *not very old and not very young*, …}, which, for simplicity reasons, is defined here in an informal way, without defining the grammar G. In this example, *young* and *old* are the atomic terms. The universe of discourse might be U = [0,100] and the meaning of the term *young*, M(*young*), could be given by a fuzzy set, such that:

$$\mu_{M(young)}(u) = \begin{cases} 1, & u \in [0,25] \\ \left[1+\left(\frac{u-25}{5}\right)^2\right]^{-1}, & u \in]25,100] \end{cases}, u \in U \tag{24}$$

3.2 Modifiers

A linguistic modifier can be modeled by using an operator that acts on the fuzzy set corresponding to the linguistic term to which the modifier is applied. For example (Schmucker 1984), the linguistic modifier *very* in the linguistic expression "very young" intensifies the meaning expressed by the fuzzy term *young*. Hence, the effect of *very* is to decrease the membership of the values belonging to the fuzzy set YOUNG. The concentration operator can produce this effect:

$$\mu_{CON(A)}(x) = \mu_A^2(x) \tag{25}$$

Conversely, the dilation operator can be used for defining modifiers, as for example *slightly* and it is modeled as:

$$\mu_{DIL(A)}(x) = \mu_A^{1/2}(x) \tag{26}$$

Other usual modifier is *not* that is modeled by the complement operator. There are many more operators used to model linguistic modifiers, cf., e.g., (Kerre and De Cock 1999).

3.3 Fuzzy (Linguistic) Quantifiers

Classical logic recognizes two quantifiers expressing that all objects posses certain property (*general* quantifier) or that at least one object possesses certain property (existential quantifier), respectively. However, natural languages offer many more forms of quantifiers. For example, quite often one says that *most* of the objects possess certain property. Basically, there are two types o fuzzy (linguistic) quantifiers (Zadeh 1983; Yager 1994): *absolute* - such as "approximately 3" and "several" and *proportional* such as "most" and "a few". There are also two general types of propositions referring to linguistic quantifiers:

1. *Q X's* are *A*'s (type I)
2. *Q B's* are *A*'s (type II)

where *Q* is a linguistic quantifier, and *A* and *B* are fuzzy sets modeling certain fuzzy properties of the objects of the universe *X*. In what follows we briefly discuss how linguistic quantifiers may be formalized.

Zadeh's calculus of linguistically quantified propositions

Zadeh (Zadeh 1983) proposed an interpretation for fuzzy quantified statements of both types I and II based on the concepts of cardinality (Eq. (15)) and relative cardinality (Eq. (16)) of a fuzzy set. A fuzzy proposition *Q X's* are *A's* has the truth degree *T* that is computed using the following equations (Zadeh 1983):

$$T = Q_{absolute}(|A|) = Q_{absolute}(\sum_i \mu_A(x_i)) \tag{27}$$

where *Q* is respectively an absolute and a relative quantifier. For fuzzy propositions *Q B's* are *A's*, where both *A* and *B* are fuzzy sets, we have (Zadeh 1983):

$$T = Q_{absolute}(|A \cap B|) = Q_{absolute}(\sum_i \mu_A(x_i) \wedge \mu_B(x_i)) \tag{28}$$

$$T = Q_{relative}\left(\frac{|A \cap B|}{|B|}\right) = Q_{relative}\left(\frac{\sum_i \mu_A(x_i) \wedge \mu_B(x_i)}{\sum_i \mu_B(x_i)}\right) \tag{29}$$

OWA operators and Yager's calculus of linguistically quantified propositions

Yager (Yager 1994) proposed the use of Ordered Weighted Averaging (OWA) operators for the evaluation of linguistically quantified propositions to overcomee some problems of the Zadeh's proposal (see in (Bosc and Pivert 1995) an example comparing Zadeh's approach with Yager's approach). An OWA operator of dimension *n* is a mapping *f* that performs an aggregation of its *n* arguments $a_1, \ldots, a_n$ (Yager 1994), such that:

$$f(a_1,...,a_n) = \sum_{j=1}^{n} b_j w_j \tag{30}$$

where $a_i \in [0,1]$, b_j is the j-th largest from among a_i, and w_j ($w_j \in [0,1]$) are weights such that $\sum_i w_i = 1$. The classical "AND" and "OR" may be expressed as special OWA operators:

$$\begin{aligned} &\text{for } w_n = 1 \text{ and } w_j = 0\ (\forall j < n) \text{ we obtain } f_1(a_1, \ldots, a_n) = b_n = \min_i a_i \\ &\text{for } w_1 = 1 \text{ and } w_j = 0\ (\forall j > 1) \text{ we obtain } f_2(a_1, \ldots, a_n) = b_1 = \max_i a_i \end{aligned} \tag{31}$$

Moreover, any OWA operator lies somewhere between the "OR" operator and the "AND" operator (Yager 1994) in the sense that:

$$\min_i a_i \le f(a_1, \ldots, a_n) \le \max_i a_i \tag{32}$$

It is also possible to define an OWA operator which may be interpreted as a linguistic quantifier. Yager (Yager 1994) proposed a scheme of defining an OWA operator corresponding to a linguistic quantifier in the sense of Zadeh. Namely, starting with a linguistic quantifier Q that is monotone and regular ($Q(0) = 0$; $Q(1) = 1$) we set the weights of corresponding OWA operator as follows:

$$w_i = Q_{relative}(i/n) - Q_{relative}((i\text{-}1)/n) \quad \forall\ i=1,\ldots,n \tag{33}$$

Then, the degree of truth T of a quantified proposition "Q X's are A" is computed using this OWA operator (Yager 1994):

$$T = f(\mu_A(x_1), \ldots, \mu_A(x_n)) \tag{34}$$

3.4 Possibility distributions

Consider a vague proposition "X is young". This is an imprecise proposition because it does not assign a particular value for X (more precisely, for X's attribute *age*). Instead, it associates, with each possible value of X its possibility degree, a number in the interval [0,1] (Klir and Folger 1988). We can say that proposition p = "X is young" induces a *possibility distribution* π (the notation π_X is often used to indicate what variable is considered) on the domain of the attribute *age*:

$$X \text{ is young} \rightarrow \pi = \text{YOUNG} \tag{35}$$

or, equivalently:

$$\forall u \in U \qquad \pi(u) = \mu_{YOUNG}(u) \tag{36}$$

that is, the possibility that a certain $u \in U$ is an actual value of X is equal to the u's membership degree to the fuzzy set YOUNG, which models the linguistic term *young*. Knowing the possibility distribution π_X we may be also interested in determining what is the possibility that X's value belongs to a set $A \subseteq U$. This leads to the concept of the *possibility measure*, i.e., a function Π such that:

$$\Pi: 2^U \rightarrow [0,1] \tag{37}$$

From the postulated properties of possibility measures it is assumed that (in fact, usually we start with the concept and properties of the possibility measure and only then the notion of the possibility distribution is introduced):

$$\Pi(A) = \sup_{u \in A} \pi(u) \tag{38}$$

The possibility measure alone does not tell us enough about the location of the actual value of X: outside or inside A. Thus, it is usually argued that it should be accompanied by the possibility measure of the complement of A. More precisely, the *necessity measure*, N, is defined as, expressing the "impossibility" of the set $\overline{A}$:

$$N(A) = 1 - \Pi(\overline{A}) = \inf_{u \in A} \pi(u) \tag{39}$$

The formulae of Eqs. (38)-(39) are extended to the case where A is a fuzzy set in the following way:

$$Poss(X \text{ is } A) = \Pi(A) = \sup_{u \in U} \min(\pi(u), \mu_A(u)) \tag{40}$$

and:

$$Nec(X \text{ is } A) = N(A) = \inf_{u \in U} \max(1 - \pi(u), \mu_A(u)) \tag{41}$$

Now, if we know that the possibility distribution of the X's value is π then the degree to which the actual value of X belongs to A (often denoted as "X is A") belongs to the interval $[N(A), \Pi(A)]$.

Eqs. (40)-(41) for the possibility and necessity measures are directly employed when the matching degree is computed in the context of querying possibilistic fuzzy databases - see Section 5. Actually, the interpretation of more advanced queries calls for more sophisticated formulae. Namely, let us assume that we have two variables X and Y, defined on the same universe U, and we know the possibility distributions of their values, π_X and π_Y, respectively. The question is: what is

the possibility that the actual values of these variables are equal. In order to answer this question consistently we proceed as follows. First, we observe that π_X and π_Y jointly represent a possibility distribution π_{XY} on U x U:

$$\pi_{XY}(u,w) = \min(\pi_X(u), \pi_Y(w)) \tag{42}$$

Second, the possibility (necessity) measure associated with π_{XY} will be denoted Π_{XY} (N_{XY}). Thirdly, the answers to our question are the values of the possibility and necessity measures, for the set of pairs of identical elements from U, i.e.,

$$Poss(X = Y) = \Pi_{XY}(\{(u,u): u \in U\}) = \sup_{u \in U} \min(\pi_X(u), \pi_Y(u)) \tag{43}$$

$$Nec(X = Y) = N_{XY}(\{(u,u): u \in U\}) = \inf_{u \in U} \max(1 - \pi_X(u), 1 - \pi_Y(u)) \tag{44}$$

In Section 5, we discuss the Eqs. (40)-(41) for a more general case, when, roughly speaking, the "is" and "=" operators are replaced with other crisp or fuzzy relational operators.

Now, we would like to distinguish another concept, useful in the context of a fuzzy database querying: the *possibility distributions' similarity*. If we know the possibility distributions for two variables, X and Y, we can be interested in finding out how similar both distributions are. Similarity is meant here in a very broad sense. Obviously, Eq. (43) provide an assessment of this similarity, but other measures are also applicable. In Section 5 we discuss this concept in more detail.

3.5 Fuzzy Relations

A crisp relation $R(x_1, x_2, ..., x_n)$ defined on crisp sets $X_1, X_2, \ldots, X_n$ is a subset of the Cartesian product $X_1 \times X_2 \times \ldots \times X_n$. Similarly, a *fuzzy relation* R is a fuzzy set defined on the Cartesian product $X_1 \times X_2 \times \ldots \times X_n$. The membership degrees $\mu_R(x_1, x_2, \ldots, x_n)$ represent the strength of the relation between the elements of the tuples $(x_1, x_2, \ldots, x_n)$. In the relational database terminology we will say that R is defined over schema $(X_1, X_2, \ldots, X_n)$.

The composition of two binary crisp relations $P(X,Y)$ and $Q(Y,Z)$, denoted $P(X,Y) \circ Q(Y,Z)$, is a crisp binary relation $R(X,Z)$ such that:

$$R(X,Z) = \{(x,z) \in X \times Z \mid \exists\, y \in Y \ (x,y) \in P \wedge (y,z) \in Q\} \tag{45}$$

The *composition of two fuzzy binary relations* P(X,Y) and Q(Y,Z) may be defined in several ways. The most commonly used definitions are the following (Klir and Folger 1988):

$$\mu_{P \circ Q}(x,z) = \max_{y \in Y} \min[\mu_P(x,y), \mu_Q(y,z)] \tag{46}$$

$$\mu_{P \bullet Q}(x,z) = \max_{y \in Y} \left[\mu_P(x,y) \times \mu_Q(y,z) \right] \quad (47)$$

respectively for max-min composition and max-product composition.

A fuzzy relation R defined on $X \times X$, verifying the following properties: (Klir and Folger 1988):

1. *reflexivity*: $\forall x \in X, \ \mu_R(x,x) = 1$ (48)

2. *symmetry*: $\forall x,y \in X, \ \mu_R(x,y) = \mu_R(y,x)$ (49)

3. *max-min transitivity*: $\forall x,z \in X, \ \mu_R(x,z) \geq \bullet \max_{y \in Y} \min[\mu_R(x,y), \mu_R(y,z)]$ (50)

is called a *similarity relation* $R(X,X)$. If $R(X,X)$ is only reflexive and symmetric then R is called a *proximity relation* (Klir and Folger 1988).

An algebra for fuzzy relations

Relational algebra (see Section 2) may be extended in order to provide the same type of operations for fuzzy relations. Next, we present the definitions for the operations of such an extended algebra (Bosc et al. 1999). The set operations *union*, *intersection*, *difference* and *Cartesian product* of two fuzzy relations R and S are direct applications of operations on fuzzy sets, i.e.:

$$\mu_{R \cup S}(x) = \max(\mu_R(x), \mu_S(x)) \quad (51)$$

$$\mu_{R \cap S}(x) = \min(\mu_R(x), \mu_S(x)) \quad (52)$$

$$\mu_{R - S}(x) = \mu_{R \cap \bar{S}}(x) = \min(\mu_R(x), 1 - \mu_S(x)) \quad (53)$$

$$\mu_{R \times S}(xy) = \min(\mu_R(x), \mu_S(y)) \quad (54)$$

where x and y are tuples of relations R and S. Moreover, the operations *selection*, *projection* and *join* are defined as:

$$\mu_{\sigma_P(R)}(x) = \min(\mu_R(x), \mu_P(x)) \quad (55)$$

$$\mu_{\pi_Z(R)}(z) = \max_v \mu_R(zv) \quad (56)$$

where P is a fuzzy condition, z and v are tuples defined over schema Z and V, respectively, such that $Z \cup V = X$ and $Z \cap V = \varnothing$; A and B are subsets of X and Y respectively, and θ is a fuzzy comparator, i.e. it is a fuzzy relation, defined on sets A and B, such as *approximately equal*.

Let us briefly explain the three formulas above. First, for selecting the tuples x in a fuzzy relation R that satisfy a fuzzy condition P, we compute the matching degree of each tuple against P, that is, $\mu_P(x)$. The membership degree of each x in the resulting relation must take into account two values: $\mu_P(x)$ and its original membership degree $\mu_R(x)$. Projecting a fuzzy relation R through one of its sub-schema Z, that is, $Z \subset X$, requires forming the sub-tuples z corresponding to Z, and considering for their membership degrees the highest membership degree of the tuples x that include z as a sub-tuple. Finally, joining two fuzzy relations R and S using a fuzzy condition $A\ \theta\ B$ requires concatenation of pairs of tuples x and y. The membership degree of each new tuple xy is the minimum of its matching degree related to $A\ \theta\ B$ and the original membership values of tuples x and y.

The relational algebra operation of division (see Eq. (5)) can also be extended using the fuzzy sets (fuzzy relations) instead of the crisp sets (crisp relations). Dubois and Prade (Dubois and Prade 1997) proposed the following extension:

$$\mu_{R \div S}(t) = \min_u \mu_S(u) \rightarrow \mu_R(t,u) \tag{57}$$

where the symbol $\rightarrow$ denotes a fuzzy logic implication.

3.6 PRUF

Above we summarized how the semantics of propositions may be expressed in terms of possibility distributions (more generally, possibility theory). So far, we only discussed the simple propositions exemplified by Eq. (35), however, natural languages are syntactically much richer. Zadeh (Zadeh 1978) classified fuzzy propositions of natural languages in five types: simple fuzzy propositions (type I), modified fuzzy propositions (type II), composed fuzzy propositions (type III), quantified fuzzy propositions (type IV), and qualified fuzzy propositions (type V). He proposed a semantic interpretation for each type in terms of the possibility distribution in a way analogous to Eq. (36). In Fig. 2 below we collected examples of propositions belonging to each type and the corresponding induced possibility distribution formulation.

Type	Example	Induced possibility distribution
I	Maria is young	π_{age} = YOUNG
II	Bob is very tall	π_{height} = VERY_TALL
III	Maria is young *and* Bob is very tall	$\pi_{(age,\ height)}$ = YOUNG × VERY_TALL
IV	*Most* students are young	$\pi_{count(age)}$ = MOST, YOUNG
V	It is *quite true* (*probable* or *impossible*) that Maria is young	π_{age} = QUITE_TRUE, YOUNG

Fig. 2. Classification of fuzzy propositions (Zadeh 1978)

We may observe that: QUITE_TRUE, YOUNG denotes the function $\mu_{QUITE\_TRUE}(\mu_{YOUNG}(x))$; YOUNG × VERY_TALL is a Cartesian product of fuzzy set YOUNG and the *modified* fuzzy set VERY_TALL; and MOST is a quantifier.

4 Taxonomy of flexible query languages (FQLs) for crisp relational databases

In this section we propose a taxonomy for the main proposals found in the literature related to FQLs for crisp relational databases. The purpose of proposing this taxonomy is to offer a structured view about the topic and to highlight main differences and similarities between various approaches. Further, we believe that this taxonomy can offer some guidance and clarification about the most relevant proposals in this research area.

4.1 Taxonomy of FQLs for crisp databases

In Fig. 3 we present the main approaches in this topic grouped in four categories. Group 1, denoted *basic fuzzy predicates*, includes the first approach that used fuzzy predicates in queries. Group 2, denoted *flexible aggregation operators*, presents the proposals that study flexible aggregation of partial matching degrees via linguistic quantifiers and importance weights. Group 3, *denoted SQL extensions*, presents proposals introducing elements of fuzzy the querying paradigm in SQL. Finally, Group 4, denoted *PRUF and flexible querying*, describes proposals that focus on the interpretation of natural language expressions for the purposes of querying.

FQL Taxonomy	Proposals of FQLs for crisp relational databases
G1. Basic fuzzy predicates	P1. Tahani (Tahani 1977)
G2. Flexible aggregation operators	P2. Kacprzyk, Zadrozny and Ziolkowski (Kacprzyk and Ziolkowski 1986; Kacprzyk, Zadrozny and Ziolkowski 1989) P3. Bosc and Pivert (Bosc and Pivert 1993) P4. Dubois and Prade (Dubois and Prade 1997)
G3. SQL extensions	P5. (SQLf) Bosc and Pivert (Bosc and Pivert 1992; Bosc and Pivert 1995; Bosc et al. 1999) P6. (FQUERY) Kacprzyk and Zadrozny (Kacprzyk and Zadrozny 1995)
G4. PRUF and flexible querying	P7. Takahashi (Takahashi 1991; Takahashi 1995)

Fig. 3. Main approaches to Flexible Query Languages (FQLs) for crisp relational databases

Since the SQL's SELECT command is the standard for the querying of crisp relational databases, we will also use it for illustrative purposes and to clearly distinguish the taxonomy groups. The first two groups (G1-G2) of proposals, *basic fuzzy predicates* and *flexible aggregation operators*, have a theoretical character and essentially extend the WHERE clause's condition of the SELECT command, by incorporating linguistic (fuzzy) terms and using flexible aggregation operators (connectives).

The third group (G3) of proposals, *SQL extensions*, comprises more practical approaches that embed the fuzzy predicates in the syntax of the standard SQL. For example, one of the proposals (FQUERY for Access (Kacprzyk and Zadrozny 1995)) implements fuzzy predicates in the WHERE clause and another proposal (SQLf (Bosc and Pivert 1995)) specifies a new language that extends SQL by incorporating fuzzy predicates not only in the WHERE clause but wherever it makes sense. The fourth group (G4) of proposals, *natural language querying languages*, instead of modifying the SQL's SELECT command proposes a set of natural language query types that includes an adequate subset of fuzzy propositions. The proposals included in G4 are based on the PRUF language (Zadeh 1978), and the mismatch between the natural language queries and information stored in the crisp relational database is bridged by *translation rules* that perform the respective conversion.

Next, we will present a summary of the proposals by the authors listed under each group of the taxonomy. We will use Px to refer to each author's proposal, where x is the sequential number of the author in the taxonomy. The description by thematic group will highlight the resemblances of approaches in terms of the topic of study and will simplify the readers' understanding of such a vast literature. In spite of the fact that many other authors had also contributed to the advancement of flexible querying in crisp databases, in this work we selected the most representative for each topic.

G1 Basic fuzzy predicates

P1. Tahani (Tahani 1977) was the first author to propose the use of fuzzy sets theory to improve the flexibility of crisp database queries. He proposed a formal approach and architecture to deal with simple fuzzy queries for crisp relational databases. The queries are based on the SEQUEL language. The idea may be best illustrated by the query:

Q4 - *Find the names and department numbers of employees who are young and have a high salary OR those who are young and have low commission*

Using our database scheme defined in Eqs. (3)-(4) and Tahani's proposal such a query may be expressed as:

SELECT name, # dep (58)
FROM Employees
WHERE age=*YOUNG* AND (sal=*HIGH* OR commission=*LOW*)

Thus, Tahani proposed to use in the query condition vague terms typical for natural language. Syntactically, they are represented as *fuzzy predicates*. Their semantics is provided by appropriate fuzzy sets. Then, the main question is how the matching degree for each particular row is computed. For that purpose Tahani defines the matching function γ. For a tuple t_i and a simple query P of type $A = v$, where A is an attribute (e.g., *age*) and v is a vague (fuzzy) term (e.g., *young*), the value of the function γ is:

$$\gamma(P,t_i) = \mu_v(u) \tag{59}$$

where u is the value of the attribute A in tuple t_i. For example:

$$\gamma(AGE = young, <22, Arthur, 1, 30, programmer, 1500, 50, New\ York>) = 0.5 \tag{60}$$

if $\mu_{young}(30)=0.5$. The matching function γ for complex queries involving logical connectives like *age=YOUNG* AND *(sal=HIGH* OR *commission=LOW)* is:

$$\gamma(P_1 \text{ AND } P_2, t_i) = \min(\gamma(P_1, t_i), \gamma(P_2, t_i)) \tag{61}$$

$$\gamma(P_1 \text{ OR } P_2, t_i) = \max(\gamma(P_1, t_i), \gamma(P_2, t_i)) \tag{62}$$

$$\gamma(\neg P, t_i) = 1-\gamma(P, t_i) \tag{63}$$

where P, P_1, P_2 are queries. Thus, the logical connectives in the queries are interpreted as the original Zadeh's fuzzy connectives.

G2 Flexible aggregation operators

P2. Kacprzyk, Zadrozny and Ziolkowski (Kacprzyk and Ziolkowski 1986; Kacprzyk, Zadrozny and Ziolkowski 1989) were the first to propose the aggregation of partial queries (predicates, conditions) to be guided by a linguistic quantifier (see Section 3). Thus, they proposed to extend the querying language so as the selection operator's condition may be expressed as:

$$P = Q \text{ out of } \{P_1, \ldots, P_n\} \tag{64}$$

where Q is a linguistic (fuzzy) quantifier and P_i are partial conditions to be aggregated. Thus, the overall matching degree may be computed using any of the approaches discussed in Section 3. In (Kacprzyk and Ziolkowski 1986) and Kacprzyk, Zadrozny and Ziolkowski (1989) the original Zadeh's approach has been adopted, but later in (Kacprzyk and Zadrozny 1997) the authors used the OWA operators as the model for the linguistic quantifier. Both type I and type II linguistically quantified propositions (see Section 3) were studied in this context by the authors. In the latter case query (64) may be extended to:

$$P = Q \text{ important out of } \{P_1, \ldots, P_n\}, \tag{65}$$

where the importance is represented by a fuzzy set of P_i*'s* in the sense that the value of the membership function of given P_i is equal to its importance weight. Thus, in (65) importance corresponds to *B* in "*Q B's are A's*" of Section 3.

P3. Another scheme for the aggregation of fuzzy conditions of varying importance has been studied by Bosc and Pivert (Bosc and Pivert 1993). They proposed a fuzzy operator for the hierarchical aggregation of fuzzy conditions, which extends the concept of hierarchical aggregation given by Lacroix and Lavency for crisp conditions (Lacroix and Lavency 1987).

Lacroix and Lavency proposed to extend the concept of classical crisp queries in the following way. A query has two parts: a selection part, *S*, and a set of crisp conditions, *PRF*, called preferences. The semantic of this query is the following: select the tuples satisfying S and rank them according to PRF. More precisely, if there are no tuples satisfying condition *S* then the answer to the query is empty. Otherwise, the answer comprises the tuples that verify *S* and at the same time best satisfy *PRF*. In the latter case, various assumptions on the interrelation of the conditions belonging to the *PRF* may be made. Two cases are considered: (a) the conditions are equally important, (b) there is a (linear) hierarchy of conditions - those higher in hierarchy are more important. Thus, in the second case we have the importance of conditions imposed not by numerical weights, but by their position in the hierarchy. The ranking of the tuples depends on what assumption is made: (a) or (b). In the first case, the count of the conditions in PRF that are satisfied by a tuple is taken into account. In the second case, the lexicographic ordering of the tuples according to their fulfillment of particular conditions belonging to *PRF* (taken in order imposed by the hierarchy) is employed.

Bosc and Pivert (Bosc and Pivert 1993) proposed a fuzzy operator *N* to model the hierarchical aggregation described above, in which the contribution of a condition $P_i \in PRF$ is less or equal than the contribution of conditions higher in the hierarchy. Let us assume that the conditions are ordered according to the hierarchy, i.e., if $i<j$ then P_i is higher in the hierarchy (is more important) than P_j.

chy, i.e., if $i<j$ then P_i is higher in the hierarchy (is more important) than P_j. The fuzzy operator proposed is defined as a combination of two operators. The first, denoted below with μ'_{P_i}, limits the contribution of condition P_j relatively to the contributions of all preceding conditions P_i ($i<j$), while the second combines all contributions to obtain the final value for the aggregation of the fuzzy conditions:

$$\mu_{N(P_1 \ldots P_n)}(t) = \frac{\sum_{i=1}^{n} \mu'_{P_i}(t)}{n} \tag{66}$$

where $\mu'_{P_i}(t) = \min_{(j \le i)}(\mu_{P_j}(t))$. This operator expresses the degree to which a tuple t satisfies the hierarchical aggregation of the fuzzy conditions. The authors also considered another version of their operator replacing the arithmetic mean by the weighted average.

Bosc and Pivert adopt a different interpretation of hierarchy of conditions than originally assumed by Lacroix and Lavency. Namely, in the latter case, if no tuple satisfies a condition from, e.g., the k-th level of the hierarchy, then the conditions of the lower levels do play a role in the ranking of tuples. In the former approach, all these lower levels are neglected.

P4. Dubois and Prade (cf. (Dubois and Prade 1997)) studied the question of conditions P_i with varying degrees of importance forming together a compound condition P via the conjunction. The first model considers some importance weight w_i for each elementary condition P_i and the matching degree of the weighted condition P_i against a value u of the corresponding attribute is given by the following equation:

$$\mu_{P_i^*}(u) = \max(\mu_{P_i}(u), 1 - w_i) \tag{67}$$

where P_i^* denotes the condition P_i with an importance associated to it. Then, the matching degree of the condition P is calculated using the standard min operator:

$$\mu_P(u) = \min_{i=1,\ldots,n} \mu_{P_i^*}(u) = \min_{i=1,\ldots,n} \max(\mu_{P_i}(u), 1 - w_i) \tag{68}$$

When the importance is minimal ($w_i = 0$), the condition P_i is not considered in the evaluation. On the other hand, with $w_i = 1$, the evaluation of condition P_i is vital for the evaluation of condition P. This model has been refined (see in (Dubois and Prade 1997)) to deal with a variable importance w_i – depending on the matching degree of the associated elementary condition. For example, in a specific context it may be useful to assume w_i constant for relatively high satisfaction of the elementary condition but for extremely low satisfaction it should be stronger re-

flected in the overall matching degree by automatically increasing w_i. For example, if we want to buy a car and one of our criterion is to have a moderate price, but it is not our primary criterion (condition), then we assume an importance weight smaller than 1.0. However, if a particular car has a very high price, the price criterion becomes more important ($w_i = 1$) and the car is rejected by having a very low satisfaction membership value.

The second model, originally proposed by Yager in 1984 (see in (Dubois and Prade 1997)) considers a threshold α_i for each elementary condition P_i. If condition P_i is satisfied to a degree above threshold α_i, that is, $\mu_{P_i}(u) \geq \alpha_i$, the resulting partial matching degree becomes 1, i.e., $\mu_{P_i^*}(u) = 1$. On the other hand, if the threshold is not reached, i.e., $\mu_{P_i}(u) < \alpha_i$, then we may consider two ways for the evaluation of the condition: (a) $\mu_{P_i^*}(u) = \mu_{P_i}(u)$ or (b) $\mu_{P_i^*}(u) = \frac{\mu_{P_i}(u)}{\alpha_i}$. It turns out that both ways may be expressed by a formula:

$$\mu_P(u) = \min_{i=1,\ldots,n} \mu_{P_i^*}(u) = \min_{i=1,\ldots,n} \alpha_i \rightarrow \mu_{P_i}(u) \tag{69}$$

where $\rightarrow$ is the implication logical operation. Then, using the Gödel implication (see Eq. (22)) and the Goguen implication (see Eq. (23)) we can obtain (a) and (b), respectively. Note that the first model of importance proposed by Dubois and Prade (see in (Dubois and Prade 1997)) and formalized by Eq. (67) is also covered by the general formula of (69) when the Kleene-Dienes implication (see Eq. (20)) is assumed.

Still another model of importance applicable to the aggregation of partial matching has been proposed by Dubois and Prade (Dubois and Prade 1997) in which they use conditional requirements $P_1 \rightarrow P_2$ to provide an interpretation for the hierarchical aggregation of fuzzy predicates. The authors consider a similar context to that of the paper by Lacroix and Lavency (Lacroix and Lavency 1987). An overall condition P is considered to be a sequence of elementary conditions $P_{i=1,n}$ accompanied by importance weights (called here *priorities*). It is interpreted in such a way that "P_1 should be satisfied (with priority 1) and among the solutions meeting P_1 (if any) the ones satisfying P_2 are preferred (with priority α_2), and among those satisfying both P_1 and P_2, those satisfying P_3 are preferred with priority α_3 ($\alpha_3 < \alpha_2 < 1$) and so on". This may be interpreted as nested implication operators: $P_1 \rightarrow (P_2 \rightarrow (P_3 \rightarrow \ldots$. The overall matching degree (the results of the aggregation) may be thus represented by the following membership function defining a fuzzy set of elements (rows) satisfying P (when P consists of 3 partial predicates):

$$\mu_P(u) = \min\begin{pmatrix} \mu_{P_1}(u), \max(\mu_{P_2}(u), 1 - \min(\mu_{P_1}(u), \alpha_2)), \\ \max(\mu_{P_3}(u), 1 - \min(\mu_{P_1}(u), \mu_{P_2}(u), \alpha_3)) \end{pmatrix} \tag{70}$$

where $\min(\mu_{P_1}(u),\alpha_2)$ and $\min(\mu_{P_1}(u),\mu_{P_2}(u),\alpha_3)$ are priority levels (corresponding to w_i in Eq. (67)) of fuzzy conditions P_2 and P_3, respectively. Hence, concerning the predicate P_2, its priority is α_2 if P_1 is fully satisfied and is zero if P_1 is not at all satisfied, which reflects the fact that P_2 is conditioned by P_1. This is another example of the variable importance weight, but this time depending on the satisfaction of the "preceding" partial condition. Notice, that the hierarchy (nesting) of the conditions is here used in the same sense as in Bosc and Pivert's approach rather than in Lacroix and Lavency's sense.

G3. SQL extensions

In G1 and G2 we reviewed proposals for the application of fuzzy conditions, in the context of crisp relational database querying. In terms of relational algebra we considered them as extensions to the *selection* operation making it possible to use linguistic terms and flexible aggregation operators in conditions.

Here, we discuss two proposals of the most popular extensions of a de facto standard querying language of relational databases, i.e., SQL. Notice that already in Section G1 we discussed Tahani's approach directly referring to SQL (or SEQUEL). However, both approaches discussed here, the SQL*f* and FQUERY for Access, differ from Tahani's approach. The first one is an extension to SQL syntax introducing linguistic (fuzzy) terms, wherever it makes sense, and the second is an example of the implementation of a specific "fuzzy extension" to SQL for Microsoft Access®, a popular desktop DBMS.

P5. The previously discussed approaches concentrated on the *fuzzification* of conditions appearing in the WHERE clause of the SQL's SELECT instruction. Bosc and Pivert (Bosc and Pivert 1992; Bosc and Pivert 1995; Bosc and Pivert 1997) proposed a new language, called SQL*f*, that is a much more comprehensive and complete *fuzzy* extension to the crisp SQL language. In SQL*f* linguistic terms may appear as:

1. Fuzzy values, relations and quantifiers (as aggregation operators) in the WHERE and HAVING clauses;
2. The linguistic quantifiers in addition to the classical EXISTS and ALL quantifiers used together with subqueries.

Moreover, the authors observe that in case of complex SQL queries involving linguistic (fuzzy) terms the partial results are fuzzy relations. Thus, all operations of relational algebra (implicitly or explicitly used in SQL's SELECT instruction) have to be redefined to properly process fuzzy relations. Hence, the union, intersection and difference operations are considered. Special attention is also paid to the division operation which may be interpreted in a different way due to many possible versions of the implication available in fuzzy logic. Other typical operations for SQL require a redefinition, including the partition of relations (fuzzy re-

lations) with the operator (clause) GROUP BY and the operators IN and NOT IN used together with subqueries.

All the features of SQL just mentioned were extended in such a way so as to preserve the equivalences that occur in the *crisp* SQL. To illustrate this work, we describe below an extension of the nesting operator IN and how the partition of relations is adapted to the case of a fuzzy relation in SQL*f*.

SQL*f* allows fuzzy conditions as described in G2. For example, the query (Bosc and Pivert 1995):

Q5 - *Find the young employees who work in a high-budget department*

can be expressed in SQL*f* as (Bosc and Pivert 1995):

```
SELECT #emp                                                    (71)
FROM Employees
WHERE age = 'young' AND #dep IN (SELECT #dep
                                 FROM Departments
                                 WHERE budget = 'high')
```

where the result of the subquery is a fuzzy relation. Consequently, the meaning of the condition *a* IN *E*, where *a* is an element and *E* is a crisp relation, must be extended to deal with fuzzy relations. Fuzzy sets theory suggests the following definition for the predicate IN (Bosc and Pivert 1995):

$$\mu_{IN}(a,A) = \sup_{b \in \text{support}(A)} \min(\mu_{=}(a,b), \mu_A(b)) \tag{72}$$

where *a* is an element and *A* is a fuzzy set. In case of the classical identity relation "=" this boils down to $\mu_{IN}(a,A)=\mu_A(a)$. Bosc and Pivert (Bosc and Pivert 1995) propose to obtain more flexibility by replacing "=" with another operator referring to the similarity between elements. This leads to a concept of a *fuzzy membership*, IN_F. For example, the query (Bosc and Pivert 1995):

Q6 - *Find the employees who work in a department whose budget is about 1000 times their own salary*

can be expressed in SQL*f* as (Bosc and Pivert 1995):

```
SELECT #emp                                                    (73)
FROM Employees
WHERE sal * 1000 IN_F (SELECT budget
                       FROM Departments
                       WHERE #dep = Employees.#dep)
```

In SQL*f*, the HAVING clause is extended in two ways: with a fuzzy condition and/or fuzzy quantified proposition. As an example of a fuzzy condition, we can replace the identity operator "=" with the similarity operator $\approx$. As an example of using fuzzy quantified propositions, the query (Bosc and Pivert 1995):

Q7 - *Find the departments where most of the young employees are well paid*

may be expressed in SQL*f* as (Bosc and Pivert 1995):

```
SELECT #dep                                                    (74)
FROM Employees
GROUP BY #dep
HAVING most (age = 'young') ARE (sal = 'well paid')
```

Recently, the authors of SQL*f* are working on the interpretation of SQL's aggregate functions such as MAX, AVG etc. for the case of fuzzy relations. For example, it is not clear what should be the answer to the query: "*Find maximum salary of young employees"*. For a discussion of this topic see, e.g., (Dubois and Prade 1990; Bosc et al. 2001; Bosc et al. 2002).

P6. Kacprzyk and Zadrozny (Kacprzyk and Zadrozny 1995) started with the syntax of SQL language as it is implemented in the Microsoft Access® DBMS. The authors proposed to include in the language linguistic (fuzzy) terms (predicates) in the spirit of the approaches discussed in Section G1. More specifically, they proposed to take into account the following types of linguistic terms:

- fuzzy values (e.g. YOUNG)
- fuzzy comparators (e.g. MUCH GREATER THAN)
- fuzzy quantifiers (e.g. MOST)

The matching degree of relevant rows is calculated according to the previously discussed semantics of fuzzy conditions and linguistically quantified propositions.

Kacprzyk and Zadrozny (Kacprzyk and Zadrozny 1995) have implemented this extension to SQL as an add-in, called FQUERY for Access, in the Microsoft Access package, thus extending the native Access's querying interface with a capability of manipulating linguistic terms.

In FQUERY for Access, the user composes a query using a Query By Example type user interface provided by the host environment, i.e., Microsoft Access. It is executed more or less as a regular Access's query while FQUERY is responsible for the calculation of matching degrees of the rows, interpreting linguistic terms in an appropriate way. The resulting rows of the answer relation are ordered non-increasingly with respect to the matching degree. FQUERY for Access was one of the first implementations demonstrating the usefulness of fuzzy querying features for a crisp database. Besides the syntax and semantics of the extended SQL,

the authors proposed also a scheme for the elicitation and manipulation of linguistic terms to be used in queries. The problem has been solved in accordance with the relational data model paradigm. Linguistic terms are maintained by FQUERY in a dictionary, “de facto” as another system table storing metadata in regular relational database management systems.

The concept of FQUERY for Access has been later developed in two directions. In (Zadrozny and Kacprzyk 1998) and (Kacprzyk and Zadrozny 1999) the very same concept has been applied in the environment of the Internet WWW service. Another interesting line of development (Kacprzyk and Zadrozny 2000; Kacprzyk and Zadrozny 2000) boils down to the addition of some data mining capabilities to the existing fuzzy querying interface. Such a combined interface partially exploits the same modules and data structures and seems to be a promising direction for the development of advanced data analysis tools.

G4. PRUF and flexible querying

P7. Takahashi (Takahashi 1991; Takahashi 1995) proposed a flexible query language (FQL) that is an extension to the domain relational calculus (DRC) Thus, in fact he proposed to use fuzzy logic language instead of classical logic language to express conditions that requested data should meet. The author follows to some extent the idea of Zadeh’s PRUF (see Section 3.6.). However, it seems that the grammar proposed by the author is unnecessarily complicated.

As in case of the crisp DRC, the result of a query, expressed as in (7), is a set of rows satisfying a formula. Note that these rows may come directly from the relations defined in a database or can be “constructed” from existing relations (e.g., as in algebraic join operation).

There are two problems with the Takahashi’s language. First, he does not discusses any extension of the concept of *safe formula* (Ullman 1982) for his language so that some queries (formulas) may produce an infinite number of rows as an answer. Second, Takahashi refers to Zadeh’s PRUF that is not fully sound in the context of a querying language. Basically, PRUF provides semantics for a subset of natural language propositions. This semantics is based on possibility theory and thus is more appropriate for the representation of imprecise facts in the database rather than for the interpretation of meaning of a query. Anyway, the calculations overlap to some extent and those proposed by Takahashi for matching degree assessment are still valid.

An experimental application for querying a business database using the PRUF translation rules (section 3) that showed the flexibility of the PRUF rules was developed by (Ribeiro and Moreira 2003).

5 Flexible query languages (FQLs) for fuzzy relational databases

Several fuzzy querying models for modeling incomplete information in fuzzy relational databases have been proposed in the literature (see for instance (Bosc et al. 1999)). Usually, each model requires a specialised querying formalism. In this section we propose a taxonomy for the main proposals found in the literature related to fuzzy relational databases. The objective of proposing this taxonomy is to offer a structured view about the topic. We hope to shed some light on main differences and similarities between the particular approaches. Further, we hope the taxonomy can offer some guidance and clarification about the most relevant proposals in the area.

5.1 Taxonomy of FQLs for fuzzy databases

In Fig. 4 we propose a classification for different FQLs proposed in the literature for fuzzy relational databases.

Taxonomy	Proposals of FQLs for fuzzy databases
G1. Possibilistic model	P1. Prade and Testemale (Prade and Testemale 1984; Prade and Testemale 1987) P2. Baldwin, Coyne and Martin (Baldwin et al. 1993) P3. Bosc and Pivert (Bosc and Pivert 1997)
G2. Similarity-based model	P4. Buckles and Petry (Buckles and Petry 1985) P5. Buckles, Petry and Sachar (Buckles et al. 1986) P6. Shenoi, Melton and Fan (Shenoi and Melton 1989; Shenoi et al. 1990)
G3. Hybrid models	P7. Medina, Pons and Vila (Medina et al. 1994) P8. Galindo, Medina and Aranda (Galindo et al. 1999) P9. Galindo, Medina, Pons and Cubero (Galindo et al. 1998)

Fig. 4. Taxonomy for Flexible Query Languages (FQLs) for fuzzy databases

The proposed taxonomy includes three main groups. Group 1 (G1) is devoted to proposals related to possibilistic fuzzy databases. Group 2 (G2) includes proposals relevant for similarity- based models. Group 3 (G3) presents proposals for hybrid models, i.e. combined possibilistic and similarity-based models.

The literature on fuzzy databases is much richer and includes among others: (Umano 1982; Umano and Fukami 1994; Zemankova-Leech and Kandel 1984). We selected some representative proposals as listed in Fig. 4 and now we will examine them in a more detailed way.

G1 Possibilistic model

P1. Prade and Testemale (Prade and Testemale 1984) generalize the concept of a relational database in such a way that value d_{ij} of the j-th attribute in tuple t_i may be given as a possibility distribution. This makes it possible to store incomplete or imprecise information. The idea may be best illustrated with an example (Prade and Testemale 1984). Let us consider the PERSON relation (Fig. 5) storing information about students where M1 corresponds to the grade in mathematics during the first quarter and NAME and AGE represent name and age of a student.

All numerical and non-numerical values in the columns AGE and M1 may be easily represented using appropriate possibility distributions. For instance, the value used for Jill's AGE attribute means nothing more than that our knowledge about her age is drawn from the proposition: "Jill is young". This proposition induces the possibility distribution on the domain of the AGE attribute. Because we do not know what is Jill's exact age we can only assign a possibility degree to all potential numbers representing her age. This distinction between a fuzzy set and an induced possibility distribution is important for approaches dealing with relational databases.

Person:	Name	Age	M1
	Tom	Young	15
	David	20	Rather_bad
	Bob	22	bad_to_very_bad
	Jane	about_21	Rather_good
	Jill	Young	around_10
	Joe	about_23	[14,16]
	Jack	[22,25]	Unknown

Fig. 5. Example from (Prade and Testemale 1984)

Prade and Testemale (Prade and Testemale 1984) proposed to adapt the classical relational algebra to the case of the possibilistic database. Thus, all standard operations of selection, Cartesian product, join (see (Bosc et al. 2000) for some problems concerning the possibilistic join), projection, union and intersection are extended. In order to illustrate the algebra, we discuss the selection operation and give a relevant example of this type of query. The selection, $\sigma_P(R)$, of a relation R upon the condition P may refer to two types of atomic conditions for P_i:

1. $A_j \, \theta \, a$, where A_j is the name of an attribute, θ is a comparison operator (fuzzy or not) and a is a constant (fuzzy or not);
2. $A_j \, \theta \, A_k$, where A_k is also an attribute name.

More complex conditions can be built from the above atomic conditions and the logical connectives of negation, disjunction and conjunction.

The matching degree of an atomic condition and a tuple t_i is computed as a pair: possibility and necessity measure (with respect to the possibility distributions d_{ij} and d_{ik}) of relevant sets. In case (a) it is the set, crisp or fuzzy, of the elements belonging to the domain of A_j and being in relation θ (crisp or fuzzy) with the constant a. In the second case (b) it is the subset of the Cartesian product of domains of A_j and A_k containing only the pairs of elements being in relation θ. In this case a joint possibility distribution over the Cartesian product of the domains of A_j and A_k is used.

Formally, the matching degree for case (a) is computed as follows. Let us denote by F a set (in general fuzzy) whose possibility and necessity measures have to be computed. Its membership function for the elements of the domain of attribute A_j is:

$$\mu_F(d) = \sup_{d' \in D} \min(\mu_\theta(d, d'), \mu_a(d')) \quad d \in Dom(A_j) \tag{75}$$

Now, the possibility and necessity measures of set F with respect to the possibility distribution $\pi_{d_{ij}}$ being the value of d_{ij} are computed as:

$$\Pi_{d_{ij}}(F) = \sup_{d \in D} \min(\pi_{d_{ij}}(d), \mu_F(d)) \tag{76}$$

$$\mathrm{N}_{d_{ij}}(F) = \inf_{d \in D} \max(1 - \pi_{d_{ij}}(d), \mu_F(d)) \tag{77}$$

In case (b) set F comprises the pairs of elements (d, d'), $d \in$ Dom(A_j), $d' \in$ Dom(A_k) such that $d\ \theta\ d'$ is satisfied. Thus, its membership function is identical to that of θ:

$$\mu_F(d, d') = \mu_\theta(d, d') \tag{78}$$

Now we compute the possibility and necessity measures with respect to a joint possibility distribution $\pi_{(d_{ij}, d_{ik})}$:

$$\pi_{(dij, dik)}(d, d') = \min(\pi_{dij}(d), \pi_{dik}(d')) \qquad \forall (d, d') \in D \times D \tag{79}$$

Then, the possibility and necessity measures are computed as previously:

$$\Pi_{(d_{ij}, d_{ik})}(F) = \sup_{d \in D} \min(\pi_{(d_{ij}, d_{ik})}(d, d'), \mu_F(d, d')) \tag{80}$$

$$\mathrm{N}_{(d_{ij}, d_{ik})}(F) = \inf_{d \in D} \max(1 - \pi_{(d_{ij}, d_{ik})}(d, d'), \mu_F(d, d')) \tag{81}$$

P2. Baldwin and his collaborators (Baldwin et al. 1993) implemented a system for querying a fuzzy relational database that uses semantic unification and the evidential logic rule. The value of an attribute in the database may be either a crisp value or a possibility distribution of values. The queries are composed of one or more conditions (corresponding to attributes of the database), with an importance for each condition and they are represented by a *filtering* function (similar to the notion of quantifier) and a threshold.

The specific feature of their work (Baldwin et al. 1993) is the process of *semantic unification* used for matching the fuzzy values of the conditions with the possibility distributions of different attributes of a tuple in a database. That process is based on the mass assignment theory developed by Baldwin (Baldwin et al. 1995) which gives for the matching of two fuzzy sets, an interval [n, p], where *n* (necessary) is the certain degree of matching and *p* (possibility) is the maximum possible degree of matching. Next, it is described how that interval is computed.

The mass assignments theory (Baldwin et al. 1995) provides a bridge between the two forms of uncertainty: probability and fuzziness. A fuzzy set induces a family of probability distributions that can be represented by a function called a *mass assignment*. The interpretation of this translation (fuzzy set to a mass assignment) may be briefly explained as follows (Ribeiro 1993).

A *mass assignment* is a function m: $2^X \rightarrow [0,1]$, such that:

$$\begin{aligned} & m(A_i) \geq 0 \\ & \sum_i m(A_i) = 1 \end{aligned} \tag{82}$$

where A_i are subsets of a set $X = \{x_1, x_2, \ldots, x_n\}$, such that $A_i = \{x_1, \ldots, x_i\}$. Hence, $A_1 \subset A_2 \ldots \subset A_n$. Note that these expressions mean that m represents a family of probability distributions. A fuzzy set is converted to a mass assignment in the following way. The normalized fuzzy set $A = x_1/\mu(x_1) + x_2/\mu(x_2) + \ldots + x_n/\mu(x_n)$, where $\mu(x_1) = 1$ and $\mu(x_1) \geq \mu(x_2) \geq \ldots \geq \mu(x_n)$, induces a possibility distribution π such that $\pi_x(x_i) = \mu_A(x_i)$ (see Eq. (36)). Then, the mass assignment m associated with π_x is defined over the subsets A_i as:

$$m(A_i) = m_i \tag{83}$$

where $m_i = \pi_x(x_i) - \pi_x(x_{i+1})$, $\pi_x(x_1) = 1$ and $\pi_x(x_{n+1}) = 0$.

Semantic unification computes the matching of two fuzzy sets calculating the mass assignment $m(A|A')$, which represents a conditional probability distribution, over the truth set {t, f, u}, of A given A'. The resulting support pair (S_n, S_p) is the sum of the truth- values (t), in the case of S_n, and the sum of the truth-values t with the uncertain values u, for the case of S_p. For example (Baldwin et al. 1993), the support pair for matching the two fuzzy sets:

$$\begin{aligned} \text{cheap} &= 10/1 + 20/0.5 + 25/0.25 + 30/0.01 \\ \text{average} &= 20/0.01 + 25/0.5 + 30/1 \end{aligned} \tag{84}$$

is computed calculating the following matrix of mass assignments *m*(*cheap* | *average*):

$\{M_i\}$: m_i	{30}: 0.5	{30, 25}: 0.49	{30, 25, 20}: 0.01
{10}: 0.5	f	f	F
{10, 20}: 0.25	f	f	u: 0.0025
{10, 20, 25}: 0.24	f	u: 0.1176	u: 0.0024
{10,20,25,30}: 0.01	t: 0.005	t: 0.0049	t: 0.0001

Fig. 6. Example from (Baldwin et al. 1993)

to which corresponds the following support pair:

$$(S_n,S_p)=([0.005+0.0049+0.0001],[0.0025+0.1176+0.0024+S_n])=(0.01,0.1325)$$

Afterwards, they combine different matching degrees of conditions with their importances using a process called an *evidential support logic rule*. This rule uses a function called an *S function* that acts as an aggregation operator OR or AND or even something between these two functions (similar to quantifiers). Formally, the support pair (S_n, S_p) for a tuple is computed in the following way:

$$(S_n,S_p)=\left(S\left(\sum_{j=1}^{n} w_j\alpha_j\right),S\left(\sum_{j=1}^{n} w_j\beta_j\right)\right) \tag{85}$$

where (α_j β_j) are the supports for the *n* conditions, w_j are their importances and $\sum_{j=1}^{n} w_j = 1$.

Next, we present an example of using semantic unification on querying a fuzzy relational database. The following relation is a simplification of that presented in (Baldwin et al. 1993) with some attributes removed.

Common_name	Upper_fur	Body_length
Pine_marten	({brown:1,black:0.4}(very_dark))	(average pine_marten)
polecat	({brown:0.7,black:0.6}(very_dark))	(average polecat)
ferret	({brown:0.5,black:0.7}(dark))	(average polecat)
mink	({brown:1,black:0.7,chocolate:0.8}(very_dark))	(average mink)

Fig. 7. Example from (Baldwin et al. 1993)

The attribute *common_name* is crisp whereas the attributes *upper_fur* and *body_length* are fuzzy. The attribute *upper_fur* is a compound fuzzy attribute,

which means it is composed of two possibility distributions: the first is discrete (e.g., {brown: 1, black: 0.4}) and the second is continuous (e.g., very_dark). *Body_length* is defined using continuous possibility distributions such as *average pine_marten*, which specifies that the length is average in the context of *pine_marten* mammals. The answer for the query (Baldwin et al. 1993):

Selection criteria. Threshold = 0.

Body_length: (average polecat), Importance: *high*
Upper_fur: ({brown: 1, black: 0.7}(very_dark)), Importance: *low*
is:

mink has support (0.40175 1)
ferret has support (0.3 0.728)
polecat has support (0.426 0.88)
pine_marten has support (0.116 0.953846)

The above answer shows that there are two mammals competing for the best solution: the mink and the polecat; the mink has a higher possible support and the polecat has a higher necessary support. Note that the threshold supplied by the user is a necessary threshold, which means that all mammals with a necessary support greater or equal to zero appear in the answer.

P3. Bosc and Pivert (Bosc and Pivert 1997) propose a new type of query for possibilistic databases that do not rely on the fuzzy pattern matching (see **P1**). Instead, the queries of this new type refer to the representation of the data (possibility distributions). In this case, an answer is either a crisp or fuzzy set depending if the queries are crisp or fuzzy. Moreover, these new queries improve the expression power of the associated FQL: besides the atomic conditions of types $A_j\ \theta\ a$ and $A_j\ \theta\ A_k$, FQLs are enriched with new conditions that we will describe next.

First, let us consider such new queries referring to only one possibility distribution. In order to express their conditions Bosc and Pivert (Bosc and Pivert 1997) define the following three functions:

$$\text{Poss}(A,\{d_1, \ldots,d_n\}) = \min(\pi_A(d_1), \ldots, (\pi_A(d_n)) \tag{86}$$

$$\text{Card\_cut}(A,\lambda) = |\{d \in D: \pi_A(d) \geq \lambda\}| \tag{87}$$

$$\text{Card\_supp}(A) = |\{d \in D: \pi_A(d) > 0\}| \tag{88}$$

where A is an attribute in the possibilistic database; $d_1, \ldots,d_n$ are values in the domain, D, of attribute A; π_A is a possibility distribution representing the value of A, and λ is a number from the interval [0,1]. Function Poss($A,\{d_1,\ldots,d_n\}$) supplies the truth degree of the statement "all the values $d_1, \ldots, d_n$ are possible for A". Func-

tions *Card_cut*(*A*, λ) and *Card_supp*(*A*) supply the number of values that are possible for *A* with possibility degrees above or equal to, respectively, λ and 0.

Therefore, we can easily express queries of the following type (Bosc and Pivert 1997): "find the houses for which the price value \$100.000 is considered more possible than the value \$80.000" or "find the houses for which \$100.000 is the only price value which is completely possible", as Poss(PRICE,{100.000}) ≥ Poss (PRICE, {80.000}) and Poss(PRICE,{100.000}) =1 and Card_cut(PRICE,1) = 1, respectively.

These conditions are Boolean and, hence, the respective answers are crisp relations. However, these conditions can be fuzzified and then the respective answers are fuzzy relations (this corresponds to the case of fuzzy queries against crisp data, which was detailed in Section 4.).

In order to perform a syntactical comparison between two possibility distributions various comparison techniques may be employed, cf. (Raju and Majumdar 1988). Bosc and Pivert (Bosc and Pivert 1997) use an extended resemblance relation defined on the interval [0,1], called a *fuzzy equality measure*, which is defined as follows:

$$\mu_{EQ}(\pi,\pi') = \min_{u \in D} \psi(\pi(u), \pi'(u)) \tag{89}$$

where π, π' are two possibility distributions to be compared and ψ is a resemblance relation, i.e., a reflexive and symmetric relation defined on [0,1]. They also propose an extended formula based both on the resemblance relation over *D*, denoted *RES*, and the resemblance relation over [0,1] (*a proximity measure*, *pr*,), defined as:

$$\mu_{s(\pi,\pi')}(u) = \sup_{v \in \sup p(\pi')} \min\left(\mu_{RES}(u,v), \mu_{pr}(\pi(u), \pi'(v))\right) \tag{90}$$

The equation given above measures the degree to which the possibility distribution π can be replaced by the possibility distribution π' with respect to an element *u* belonging to the support of π. Such a replacement is acceptable (the computed degree is high) if there exists *v* belonging to the support of π' such that *u* and *v* are similar (in the sense of *RES*) and the values $\pi(u)$ and $\pi'(v)$ are similar (in the sense of *pr*). Then, the degree to which we can replace a possibility distribution π with a possibility distribution π' with respect to the whole support of π is given by the following equation (Bosc and Pivert 1997):

$$\mu_{repl}(\pi,\pi') = \inf_{u \in \sup(\pi)} \max\left(1 - \pi(u), \mu_{S(\pi,\pi')}(u)\right) \tag{91}$$

where the resulting degree is the weighted combination of the degrees $\mu_{s(\pi,\pi_j)}(u)$ for all the elements u in the support of π and the values $\pi(u)$ are weights (see Eq. (67)).

The query conditions involving two representations can be expressed as:

$$\text{REP}(A_1) \approx \text{REP}(A_2) \tag{92}$$

where $\text{REP}(A_1)$ and $\text{REP}(A_2)$ are the "shapes" of the possibility distributions to be compared. For example (Bosc and Pivert 1997), the condition REP(AGE) ≈ REP(*middle_aged*) will evaluate the extent to which a value of attribute AGE (possibility distribution) is syntactically similar to the possibility distribution induced by the fuzzy set corresponding to the *middle_aged* concept. Notice that in the case of fuzzy pattern matching a similar query may be used. However, it would produce a possibility/necessity measures of the event that the value of the AGE attribute belongs to the fuzzy set *middle_aged* provided that all we know about the age is a possibility distribution.

G2 Similarity-based model

P4. Buckles and Petry (Buckles and Petry 1985; Buckles et al. 1986; Petry 1996) introduced a similarity based model for a fuzzy database in which:

1. Each domain D_j is equipped with a similarity relation S_j (see Fig. 8b)), which extends the identity relation used in the crisp relational model. It means that two values of an attribute match not only when they are identical but also if they are similar enough. Similarity relations support basic features of fuzzy querying: a query requesting a given attribute value will also be satisfied by other similar attribute values.
2. The value of tuple i for attribute j, d_{ij}, may be any valid (that is, verifying the semantics of the relation) subset of its domain D_j except for the null set. That is: $d_{ij} \in 2^{D_j} \quad d_{ij} \neq \varnothing$. This definition helps represent uncertainty within the tuples as well as it is the consequence of the similarity relations introduced.

Fig. 8c) illustrates a relation in this model; the domain of each attribute and the corresponding similarity relations are shown in Fig. 8a) and Fig. 8b), respectively.

The authors also proposed an extension to the relational algebra. The idea of this extension is illustrated on the example of the selection operation (see Fig. 8d)). The syntax of the selection operation (Buckles et al. 1986) is slightly richer in comparison with the one introduced in Section 2 (note that here we use our notation, not the original one):

$$\sigma_p(R) \text{ with } \langle \textit{level condition} \rangle \tag{93}$$

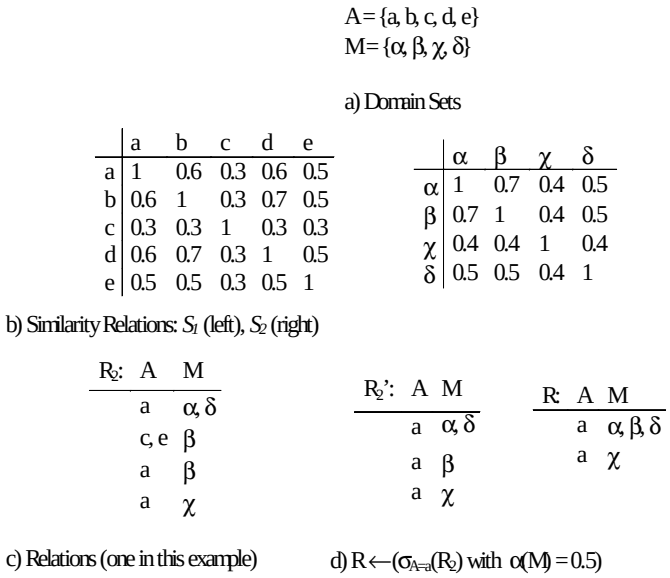


Fig. 8. Example from (Buckles and Petry 1985): components of a fuzzy relational database (a-c); a fuzzy relational algebra operation: selection (d).

where *R* and *P* denote the relation and a Boolean expression as previously, while the *level condition* specifies a similarity threshold, a number belonging to [0,1], for the domain D_j of an attribute A_j appearing in *P*. This operation selects the tuples from *R* that satisfy condition *P* but if a subcondition of *P* is of the form A_j=*a*, where A_j is an attribute and *a* is constant, then it is interpreted as "*a* is similar to an element of the value of A_j at least to the degree expressed with the *level condition*" (remember that the values of attributes are, in general, sets). See Fig. 8d) where R_2' is the intermediate result for the query obtained from the tuples satisfying condition *A* = *a*, where *A* is an attribute name and *a* belongs to *A*'s domain. Then, the result *R* is obtained by removing redundant tuples that are identified using the following definition. Two tuples t_i and t_k are redundant if (Buckles and Petry 1985):

$$I(d_{ij} \cup d_{kj}) \geq \alpha(D_j), \qquad j = 1,2,\ldots,m \tag{94}$$

where d_{ij} is the value of tuple *i* for attribute *j*; D_j is the domain of attribute *j*; $\alpha(D_j)$ is the similarity threshold and *I* is a similarity index defined as:

$$I(H) = \min_{x,y \in H} S_j(x,y), \quad H \subseteq D_j \tag{95}$$

where $S_j(\cdot,\cdot)$ is a similarity relation associated with the domain D_j.

In summary, two tuples are redundant if the values of all corresponding attributes are similar. Two values, d_{ij} and d_{kj} of an attribute A_j are *similar* if the minimum similarity degree between a pair of elements in $d_{ij} \cup d_{kj}$, $I(d_{ij} \cup d_{kj})$, is greater than a pre-specified one for this attribute level, $\alpha(D_j)$. All queries should pre-specify these levels for all attributes involved (by default it is assumed to be 1.0). In the example of Fig. 2b), the tuples (a,{α, δ}) and (a,β) in relation R_2' are redundant because *a* is obviously similar to *a* (the similarity relation is reflexive) and β is similar to both α and δ, that is:

$$\min \{S_2(\alpha,\beta), S_2(\delta,\beta)\} \geq \alpha(\mathrm{M}) = 0.5 \tag{96}$$

Hence, the tuples (a,{α,δ}) and (a,β) are merged, producing the tuple (a,{α,β,δ}) in the resulting relation *R*.

P5. Buckles et al (Buckles et al. 1986) adapted DRC (Section 2) to the similarity based model. Syntactically, this adaptation manifests itself with the addition of the *with clause* following each formula. This clause is to be interpreted in a way analogous to the case of their extended relational algebra. Thus, starting with the standard DRC query equivalent formula $\{(X_1,\ldots,X_n) | \Psi(X_1,\ldots,X_n)\}$, Buckles and Petry's fuzzy domain calculus comprises the following atomic formulae (Buckles et al. 1986):

$$R(X_1,X_2,\ldots,X_n) \text{ with } < \textit{domain level conditions} > \tag{97}$$

$$X_1\ \theta\ X_2 \text{ with } <\textit{operator level condition} > \tag{98}$$

where *R* is a database relation, X_i is a constant or a domain variable and the *domain level conditions* are level conditions for all attributes in *R*; θ is a comparison operator and the *operator level condition* is a level condition applying to θ (when omitted its value is 1 by default).

In case of Eq. (97) the variables $X_1, X_2, \ldots, X_n$ are instantiated from the values of a tuple $t_i = (d_{i1},\ldots,d_{in})$ producing the following vector of matching degrees (Buckles et al. 1986):

$$<S(d_{i1},X_1), S(d_{i2},X_2), \ldots, S(d_{in},X_n)> \tag{99}$$

that is, the matching degree, $\gamma(R(.),t_i)$, of $R(X_1,X_2,\ldots,X_n)$ against tuple t_i. The values $S(d_{ij},X_j)$ are defined as follows (provided all variables X_i are instantiated from the same tuple - otherwise $S(d_{ij},X_j) = 0\ \forall j$):

$$S(d_{ij},X_j)=\begin{cases}\min\limits_{u\in d_{ij}\wedge v=X_j} S_j(u,v), & \text{if } X_j \text{ is a constant}\\ \min\limits_{u\in d_{ij}\wedge v\in x_j} S_j(u,v), & \text{if } x_j \text{ is an instantiation for } X_j\end{cases} \quad (100)$$

Tuple t_i satisfies Eq. (97) above if for each j, $S(d_{ij},X_j)$ is greater or equal to the similarity threshold corresponding to the attribute A_j.

In case of the Eq. (98) the matching degree, $\gamma(X_1\ \theta\ X_2,t_i)$, of formula $X_1\ \theta\ X_2$ against tuple t_i is the minimum value $\theta(x_1,x_2)$ over the pairs (x_1,x_2) where $x_1 \in d_{i1}$, $x_2 \in d_{i2}$ and d_{ij} correspond to a constant or an instantiation of the variable. Tuple t_i satisfies Eq. (98) above if this matching degree is greater or equal to the threshold specified by the *operator level condition*.

Further, a formula in Buckles and Petry' fuzzy domain calculus is defined in a similar way to the definition of a domain calculus formula (Section 2). Specifically, it is one of the following expressions (Buckles et al. 1986):

1. An atomic formula
2. $\psi_1 \wedge \psi_2$ with < *level condition* >, $\psi_1 \vee \psi_2$ with < *level condition* >, $\neg\psi$ with < *level condition* >
3. $\exists X(A)\ \psi$ with < *level condition* >, $\forall X(A)\ \psi$ with < *level condition* >
4. (ψ), $[\psi]$

where ψ, ψ_1, ψ_2 are formulas, X is a domain variable associated to attribute A, and *level condition* applies to any free or bound variable in the formulas ψ_i. The authors introduce also the concept of the *safe formula* in a way analogous to the crisp case. Then, they prove that the expressive power of their DRC is at least the same as the previously discussed fuzzy relational algebra (see Proposal **P4**).

P6. Shenoi and Melton (Shenoi and Melton 1989) extended the fuzzy relational database model of Buckles and Petry (see Proposals **P4** and **P5**) by relaxing the *transitivity* property required for the similarity relation. They replaced the similarity relation with a proximity relation and showed that general properties of the model are preserved. This provides the user with the freedom to define the closeness among the different elements of the domain. For example (Shenoi and Melton 1989), if *one* is close to *two* with a degree of 0.8 and *two* is close to *three* also with a degree of 0.8, then a degree of 0.6 between *one* and *three* would not be allowed if we used a similarity relation: the max-min transitivity would impose a value equal or higher to 0.8 which would contradict common sense.

Further, Shenoi et al (Shenoi et al. 1990) proposed a more general model, an *equivalence classes model*. The model is based on the following two assumptions (Shenoi et al. 1990):

1. the existence of partitions at the desired level of precision for each non-empty subset of a domain
2. the total ordering of partitions related to the same domain but for different precision levels is provided by the concepts of a *finer* and *coarser* partition.

Assumption 1 is motivated by the postulate that a partitioning of a domain should be defined separately for each subset of elements currently appearing in a database. The authors call such a subset a *temporal domain* (it is also called an *active* domain).

Assumption 2 expresses an intuitive requirement that elements indistinguishable (i.e., falling into the same equivalence class) at a given level of precision should be still indistinguishable at any lower level of precision. Thus, any two partitions of the same temporal domain taken at different precision levels should be in a coarser/finer relation.

This is an abstract model (Shenoi et al. 1990) because it does not specify how to partition the domain: the users have a freedom to choose how to define the equivalence classes (*information chunks*). For instance, Buckles and Petry (Buckles and Petry 1985) used similarity relations on scalar domains while Shenoi and Melton (Shenoi and Melton 1989) used proximity relations on scalar domains. In fact, Shenoi et al. (Shenoi et al. 1990) show that any partitions satisfying assumptions 1 and 2 guarantee that the fundamental properties of the relational model are preserved.

They also proposed a fuzzy relational algebra for this model. For example (Shenoi et al. 1990), Fig. 9 shows a fuzzy relation of candidates to a political election storing the name, age and political view of the candidates and the associated algebra.

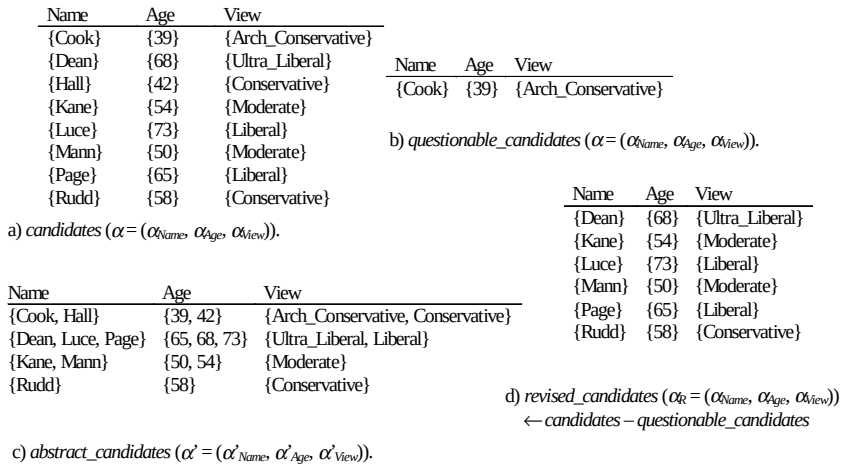


Fig. 9. Example from (Shenoi et al. 1990): algebra for the equivalence classes' model

The scheme of a fuzzy relation is defined by adding the set of precision levels α_R used for each attribute:

$$\alpha_R = (\alpha_{Name}, \alpha_{Age}, \alpha_{View}) \quad (101)$$

In Fig. 9(a-b), the partition of temporal domains into equivalence classes of *identical elements* results in partitions whose equivalence classes are sets with only one element, corresponding to the special case of the relational model. In fact, since each component of a tuple is a subset of an equivalence class in the partition of its respective attribute, each attribute value in the relation *candidates* must have only one element. For example, for attribute *Name* we can have the following partition:

$$P_{Name}: \{\{Cook\},\{Dean\},\{Hall\},\{Kane\},\{Luce\},\{Mann\},\{Page\},\{Rudd\}\} \quad (102)$$

Partitions P_{Name}, P_{Age}, P_{View} have implicit precisions, respectively, α_{Name}, α_{Age}, α_{View} (see relation *candidates* in Fig. 9a)). On the other hand, the following partitions:

$$P'_{Name}: \{\{Cook, Dean, Hall, Kane, Luce, Mann, Page, Rudd\}\} \quad (103)$$

$$P'_{Age}: \{\{39,42\}, \{50,54,58\}, \{65,68,73\}\} \quad (104)$$

$$P'_{View}: \{\{Ultra\_Liberal, Liberal\},\{Moderate\},\{Conservative, Arch\_Conservative\}\} \quad (105)$$

create equivalence classes of *more or less identical* elements. We can see that the precision of P'_{Name}, α'_{Name}, makes all elements of the temporal domain D_{Name} to be indistinguishable whereas the precision of partitions P'_{Age} and P'_{View}, α'_{Age} and α'_{View}, split temporal domains D_{Age} and D_{View} into three equivalent classes. It is obvious that $\alpha' \leq \alpha$ means that the corresponding partition P' is coarser than the corresponding partition P.

Then, if we decrease the precisions α (*identical*) for attributes of relation *candidates* (see Fig. 9a)) to the new precisions α' (*more or less identical*), redundant tuples will appear which must be merged. Fuzzy relation *abstract_candidates* (see Fig. 9c)) shows the result of merging the tuples in relation *candidates* considering the new precisions α'_{Name}, α'_{Age}, α'_{View}. We can see, for example, that candidates *Kane* and *Mann* are equivalent and hence merged because their names, ages and political views are considered equivalent (see partitions P'_{Name}, P'_{Age} and P'_{View}) but *Rudd* is not equivalent to *Kane* and *Mann* because Rudd's political view (*conservative*) is not equivalent to Kane and Mann' political view (*moderate*). Note that *moderate* and *conservative* do not belong to the same equivalence class in parti-

tion P'_{View}. Therefore, *Rudd* is not redundant and therefore is not merged with any other candidate in *candidates*' relation.

Finally, the difference operation between the fuzzy relations *candidates* (see Fig. 9a)) and *questionable_candidates* (see Fig. 9b)), using precision levels α'_{Name}, α'_{Age}, α'_{View}, is shown in Fig. 9d). We can see that the two tuples, corresponding to candidates *Cook* and *Hall*, from fuzzy relation *candidates* are missing in the result, i.e. fuzzy relation *revised_candidates*. Besides the tuple appearing in relation *questionable_candidates* (Cook's tuple), an extra tuple corresponding to candidate *Hall* is also removed from *candidates* because candidate *Hall* is equivalent to candidate *Cook* under the *more or less identical* precision (see P'_{Name}, P'_{Age}, P'_{View}).

G3 Hybrid models

P7. Medina et al (Medina et al. 1994) proposed a fuzzy database model, GEFRED (*generalized fuzzy relational database*) that tries to integrate features of both the possibilistic and similarity based models. The data is represented with *generalized fuzzy relations* that take into account imprecision as well as uncertainty of information. The latter is dealt with via a compatibility degree associated to each attribute value. More precisely, a generalized fuzzy relation R is composed of two sets (Medina et al. 1994): $R = (H,B)$, where H (*Head*) is the set:

$$H = \{(A_1 : D_{G1} [,C_1]), (A_2 : D_{G2} [,C_2]), ..., (A_n : D_{Gn} [,C_n]) \} \quad (106)$$

and B (*Body*) is the set:

$$B = \{(A_1 : d_{i1} [,c_{i1}]), (A_2 : d_{i2} [,c_{i2}]), ..., (A_n : d_{in} [,c_{in}]) \} \quad (107)$$

where A_j ($j = 1…n$) is the j-th attribute, D_{Gj} ($j = 1…n$) is the family of all possibility distributions defined over the domain of attribute A_j, which is called a *generalized fuzzy domain*; C_j is the compatibility attribute of attribute A_j, which may be optional; and d_{ij} is the value of attribute A_j in tuple i and c_{ij} ($j = 1…n$) is the compatibility degree of value d_{ij}, which is a value in the interval [0,1]. For example, Fig. 10b) (adopted from (Medina et al. 1994)) shows a generalized fuzzy relation with attributes NAME, ADDRESS, AGE, PRODUCTIVITY, SALARY and compatibility attribute C_{AGE}. The compatibility attributes associated with other attributes are equal to one and, therefore, they are not shown.

Medina et al (Medina et al. 1994) defined an algebra, a *generalized fuzzy relational algebra*, to manipulate information stored in the fuzzy database. Next, we will illustrate the selection operation, which is called a *generalized fuzzy selection*. It is based on a simple condition $\theta_G(A_j,a) \geq \alpha$, where θ_G is a comparison operator (*generalized fuzzy comparator*); α is a compatibility threshold ($\alpha \in [0,1]$), and a is a constant. Comparison operator θ_G is defined as:

$$\theta_G : D_{Gj} \times D_{Gj} \rightarrow [0,1] \tag{108}$$

$$\theta_G(\pi,\pi') = \sup_{(d_1,d_2)\in D_k\times D_k} \min(\theta(d_1,d_2),\pi(d_1),\pi'(d_2)) \tag{109}$$

where π and π' are possibility distributions and θ can be:

- A classical comparison operator such as =, ≠, >, ≥, <, ≤;
- A fuzzy comparison operator such as *approximately equal*, *much greater than*, etc;
- A similarity comparison operator which is defined using a similarity relation on scalar data.

H	Name	Address	Age	Productivity	Salary	Department
B	Antonio	Reyes Católicos	Middle	Fair	100000	Production
	Francisco	P. A. Alarcón	Old	Excellent	150000	Comercial
	Luis	Recogidas	{.8/30,1/31}	Good	110000	Production
	Juan Carlos	Camino Ronda	Young	Bad	90000	Production
	Julia	Puerta Real	Young	Good	130000	Comercial
	Javier	Gran Vía	[30,35]	Fair	105000	Human Resources

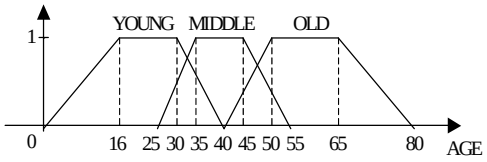
a) The generalized fuzzy relation *Emp*

H	Name	Address	Age	C_{Age}	Productivity	Salary
B	Antonio	Reyes Católicos	Middle	0.75	Fair	100000
	Francisco	P. A. Alarcón	Old	1	Excellent	150000

b) Relation S = $\sigma_{=(AGE,Old)\geq 0.6}$ (*Emp*).

H	Department	$C_{Department}$	
B	Production	1	S_1
	Human resources	0.5	S_2

c) Relation R: departments with at least one bad (≥ 0.5) employee.



d) Labels definitions for attribute *Age*.

Fig. 10. Example from (Medina et al. 1994): algebra for the GEFRED model

In fact, Eq. (109) is similar to Eq. (80) used while querying possibilistic databases. A new element is here an explicitly stated threshold value which resembles queries in the similarity based model. Compatibility degrees of all involved attrib-

utes require a special treatment in the matching degree calculation. In fact, these compatibility degrees correspond to partial matching degrees that are immediately aggregated to an overall matching degree in other models. For example, the selection $\sigma_{=(AGE,old) \geq 0.6}(Emp)$ of relation *Emp* (see Fig. 10a)) using the condition $=(AGE,old) \geq 0.6$ results in relation *S* (see Fig. 10b)). We can see from the definition of label OLD (see Fig. 10d)) that tuples corresponding to employees Luis (his age is 30 or 31), Javier (his age is between 30 and 35) do not belong to *S* because their compatibility degree C_{AGE} is zero and hence the condition is not satisfied. The same is true for tuples corresponding to employees Juan Carlos and Julia, which are YOUNG (see Fig. 10d)). Finally, the compatibility degrees for tuples corresponding to employees Antonio and Francisco are computed from Eq. (109).

P8. Galindo et al (Galindo et al. 1999) extended the GEFRED model with a fuzzy domain relational calculus (FDRC) for querying fuzzy relational databases. The FDRC language is described next.

Galindo et al.'s fuzzy domain calculus comprises the following atomic formulas (Galindo et al. 1999):

1. $R(X_1, X_2, ..., X_n) \geq \alpha$, where *R* is predicate symbol corresponding to a generalized fuzzy relation with n attributes, and each X_i is a constant or a domain variable. This atom requires that the tuple $(X_1, X_2, ..., X_n)$ belongs to a relation corresponding to *R* to a degree higher or equal α. For a given instantiation of variables X_i this degree of membership is computed as follows:

2. $$R(X_1, X_2, ..., X_n) = \max_{r=1,...,m} \min_{c=1,...,n} = (d_{rc}, X_c) \qquad (110)$$

   where *m* is the number of tuples in relation corresponding to *R*, = is a generalized fuzzy comparator defined by Eq. (109). Thus, the membership degree of a given tuple $(X_1, X_2, ..., X_n)$ – note that now X_i denotes a constant originally present in the atomic formula or a value substituting the domain variable – is computed by comparing this tuple with all tuples of relation corresponding to the predicate symbol *R*. This comparison is done using a fuzzy comparator for all attributes separately; d_{rc} denotes a value of the *c*-th attribute in the *r*-th tuple. Then, the total result of comparison is taken as the minimum of these per-attribute comparisons. A tuple of the relation for which this maximum value is attained is referred to as *the most similar tuple*.

   The fulfillment of threshold α is a value in the interval [0,1] that is the minimum value admissible for $R(X_1, X_2, ..., X_n)$ in order to make the atom true.
3. $\theta_G(X,Y) \geq \alpha$, where θ_G is a generalized fuzzy comparator, and *X* and *Y* are constants or domain variables. This atom expresses that the value *X* is related to the value *Y* by the fuzzy comparator θ_G to a certain truth degree, which is greater or equal to α.

Examples of fuzzy atoms are $R(X_1, Good, X_3)$ and $=(X_1,Good) \geq 0.9$. In the first case, the threshold is omitted which means that its value is equal to one.

Further, a formula in Galindo et al.'s fuzzy domain calculus is defined in a similar way to the definition of a domain calculus formula (Section 2). Specifically, it is either an atomic formula or one of the following expressions: $\neg\psi_1$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \Rightarrow \psi_2$,$\exists X\ \psi_1(X)$,$\forall X\ \psi_1(X)$, where ψ_1 and ψ_2 are fuzzy formulas and X is a domain variable. Afterwards, they demonstrate an expressive power of FDRC proving that any expression in the fuzzy relational algebra has an equivalent expression in FDRC.

Observe that so far we are still operating within the framework of classical logic – due to the threshold value α in an atomic formulae, they are true or false. A possible partial matching is preserved using the concept of a compatibility degree. The result of a query is a set of tuples satisfying it – a new generalized relation. Each attribute value of this new relation may be associated with a compatibility degree expressing how well this specific value matches the query. This degree is computed by a matching function γ which takes three arguments: a formula (query) ψ, a tuple $t_i = (d_{i1},d_{i2},\ldots,d_{in})$, and X, a domain variable (attribute) for which a compatibility degree is to be computed. This may be formally described as:

$$\gamma(\psi, t_i, X) \in [0,1] \cup \lambda \tag{111}$$

Value λ is a value not belonging to [0,1] that asserts that degree γ is not applicable or meaningless.

Function $\gamma(\psi,t_i,X)$ is defined depending on the structure of ψ. There are four cases: ψ is an atomic formula, a negation, a disjunction or a formula with an existential quantifier (Galindo et al. 1999).

When ψ is an atomic formula of type $R(X_1, X_2, ..., X_n,K) \geq \alpha$ we have:

$$\gamma\left(R(X_1, X_2, ..., X_n, K) \geq \alpha, t_i, X\right) = \begin{cases} R(K), & \text{if there are no variables in } \psi \\ \min\{c_{ij}, R(t_i)\}, & \text{if } X = A_j \\ \lambda, & \text{otherwise} \end{cases} \tag{112}$$

where K is the list of constants present in ψ, the values $R(K)$ and $R(t_i)$ are computed using Eq. (110); $X = A_j$ indicates that variable X is an attribute A_j in R; c_{ij} is the value of the compatibility attribute C_j for the tuple most similar to t_i; if there is no c_{ij} associated with this most similar tuple, then c_{ij} is assumed to be equal 1.0.

The atomic formula $\theta_G(X_j,Y) \geq \alpha$ is evaluated as:

$$\gamma\left(\theta_G\left(X_j, Y\right) \geq \alpha, t_i, X\right) = \begin{cases} \theta_G(d_{ij}, Y), & \text{if } X_i = X \\ \theta_G(X_j, Y), & \text{if } X_j \text{ is a constant} \\ \lambda, & \text{otherwise} \end{cases} \tag{113}$$

The evaluation of the negation and disjunction expressions is done using the complement and maximum operators, respectively (see (17)-(18)). On the other hand, for formula ψ expressed as $\exists X_{n+1}$ (ψ_1 (X_1, X_2, ..., X_n, X_{n+1})):

$$\gamma(\psi(X_1,...,X_n),t_i,X)=\max_{d_{i(n+1)}\in DOM(\psi)}\gamma(\psi_1(X_1,...,X_n,d_{i(n+1)}),t_i,X) \tag{114}$$

where DOM(ψ) is the set of all symbols that appear in formula ψ or in a tuple of a relation appearing in ψ. The remaining expressions, that is, the conjunction, the implication, and the universal quantifier can be expressed in an equivalent form using the negation, the disjunction and the existential quantifier. Then, the result we obtain for a general query $\{X_1, X_2, \ldots, X_n \mid \psi(X_1, X_2, ..., X_n)\}$ is a generalized relation R (see Eqs. (106)-(107)) that is computed using the following two steps (Galindo et al. 1999):

1. Compute all the tuples ($d_{r1},\ldots,d_{rn}$) that make true the formula $\psi(d_{r1},\ldots,d_{rn})$;
2. The compatibility values c_{rj} (j=1,…,n) for each compatibility attribute C_j, corresponding to the r (r = 1, …, m) tuples of R computed in 1., are computed as:

$$c_{rj} = \gamma(\psi(X_1,\ldots,X_n),t_r,X_j) \tag{115}$$

where t_r = ($d_{r1},\ldots,d_{rn}$) is the r-th tuple of relation R. If $c_{rj} = \lambda$ or c_{rj} = 1 for all r = 1, …, m, the attribute C_j is removed from R.

For example, the query “show the departments with at least one bad employee (with a degree greater than or equal to 0.5) “ may be expressed in FDRC as:

$$\{d \mid \exists n, a, ag, p, s\ (Emp(n,a,ag,p,s,d) \wedge =(p,Bad) \geq 0.5)\} \tag{116}$$

Which, considering the fuzzy relation in Fig. 10a), produces the resulting fuzzy relation in Fig. 10c). Further, the value $C_{1Department}$ is computed in the following way:

$$\begin{aligned} C_{1Department} &= \gamma(\psi,t_1,d) = \exists_{n,\ a,\ ag,\ p,\ s}\ (\gamma(Emp(n,a,ag,p,s,d),t_1,d) \wedge \\ &\quad \wedge \gamma(=(p,\text{Bad}) \geq 0.5,t_1,d)) \\ &= \max\{\gamma(Emp(\text{Antonio, Reyes Católicos, Middle, fair, 100000, } d),t_1,d) \wedge \\ &\quad \wedge \gamma(=(\text{fair,Bad}) \geq 0.5,t_1,d), \\ &\quad \gamma(Emp(\text{Luis, Recogidas, \{.8/30, 1/31\}, good, 110000, } d),t_1,d) \wedge \\ &\quad \wedge \gamma(=(\text{good, Bad}) \geq 0.5,t_1,d), \\ &\quad \gamma(Emp(\text{Juan Carlos, Camino Ronda, Young, Bad, 90000, } d),t_1,d) \wedge \\ &\quad \wedge \gamma(=(\text{Bad, Bad}) \geq 0.5,t_1,d)\} \\ &= \max\{\min(1,0.5), \min(1,0), \min(1,1)\} = \max\{0.5, 0, 1\} = 1 \end{aligned}$$

Note that the existential quantifier (see Eq. (114)) is replaced with the values of tuples such that *department = production* (t_1). Then, there is computed the matching degree for those tuples (three) and finally the maximum of these degrees is considered.

Galindo et al (Galindo et al. 2000) propose also the inclusion of fuzzy quantifiers in the FDRC language just described.

P9. Galindo et al (Galindo et al. 1998) implemented the FRDB model, *GEFRED*, on the crisp DBMS Oracle and a FQL, *fuzzy SQL* (FSQL). They extended the SELECT command of SQL in order to allow for more flexible conditions by choosing between possibility and necessity within fuzzy comparators; retrieving the most (least) important tuples using fulfillment thresholds or allowing fuzzy constants in the right side of the condition. More precisely, the main functionalities added are (Galindo et al. 1998):

1. *Linguistic labels.* These labels can be defined for two types of attributes: attributes with an ordered domain or attributes with a scalar domain. In the first case, the labels are defined as trapezoidal possibility distributions and, in the second case, a similarity relation between the labels of the attribute is defined.
2. *Fuzzy comparators.* They extend the usual comparators =, >, ≥, <, ≤ providing comparators that have two forms, corresponding to the possibility and the necessity cases. For example, the operator FEQ evaluates the possibility of two attributes (or one attribute and a constant) being equal using Eq. (80) (or Eq. (76)). And the NFEQ operator evaluates the necessity of two attributes (or one attribute and a constant) being equal using Eq. (81) (or Eq. (77)). Note that FEQ and NFEQ are instances of the generalized fuzzy comparator θ_G. Additionally, they provide the fuzzy comparators *much greater than* (MGT, NMGT) and *much less than* (MLT, NMLT).
3. *Fulfillment thresholds* (α). They are specified with the syntax:

$$\text{<}condition\text{>}\ [\text{THOLD}]\ \ \alpha \tag{117}$$

   which is equivalent to having THOLD replaced by ≥, and where the reserved word THOLD may be substituted by a crisp comparator (=, <, …), modifying the meaning of the condition.
4. Function CDEG(<attribute>) shows a column with the *compatibility degree* for the argument attribute, corresponding to the compatibility degree of the conditions in which the attribute appears. Function CDEG(*) shows the compatibility degree for all the fuzzy attributes appearing in the condition. If we want to see the compatibility columns for all attributes and the compatibility degree of the whole tuples, we will use the % character in the SELECT clause (like SQL does for the * character).
5. *Fuzzy constants.* FSQL allows for the use of the following constants: UNKNOWN, UNDEFINED, NULL, $[*a,b,c,d*], $*label*, [*n,m*], #*n*. $[*a,b,c,d*] is

a fuzzy trapezoid function with $a \leq b \leq c \leq d$. $*label* is a linguistic label (as described above); [*n*,*m*] is an interval, for which $a = b = n$ and $c = d = m$; and #*n* means *approximately n*, in which the fuzzy trapezoid is replaced by a triangle with $b = c = n$ and $n - a = d - n$.

6. *Condition with IS.* A condition,

$$\text{<}\textit{Fuzzy\_Attribute}\text{>}\ \text{IS [NOT] (UNKNOWN | UNDEFINED | NULL)} \tag{118}$$

is true (without NOT) when the fuzzy attribute value is equal to the fuzzy constant on the right.

For example, the query:

Q8 - *Find the Spanish cities with more than "around 500 thousands" inhabitants*

may be expressed in FSQL as (Galindo et al. 1998):

```
SELECT city, CDEG(inhabitants)                                    (119)
FROM Population
WHERE country = 'Spain' AND
        inhabitants FGEQ $[200,350,650,800] .75 AND
        inhabitants IS NOT UNKNOWN
```

where FGEQ is the fuzzy comparator extending the crisp ≥ comparator; $[200,350,650,800] is a fuzzy constant, and .75 is a fulfillment threshold requiring that the condition on the number of inhabitants is satisfied at least to the degree 0.75. Note that two columns would be displayed: the name of a city and the degree to which its number of inhabitants is greater or equal to around 500 thousands. The cities with unknown number of inhabitants are excluded.

6 Conclusion

In the last two decades, imprecision (fuzziness) and uncertainty has been studied in the context of relational DBMS, in particular in the area of querying and in the area of modeling and storing imprecise and uncertain data. The first area has led to the appearance of increasingly flexible query languages (FQLs) which provide more human consistent interfaces in comparison to the classical query languages, by using fuzzy sets theory. In the second area, fuzzy sets theory is used to extend

the relational database model, leading to what are usually called fuzzy relational database models, and on the flexible querying of the resulting models.

We introduced two taxonomies for FQLs within the context of relational database models to organize the field and to offer a structured view of the topic. One taxonomy organizes the research on FQLs for crisp relational databases and another organizes the research on fuzzy relational databases. Both taxonomies provide a structured view of the main research topics studied and highlight the main differences and similarities between approaches. We believe that our contribution in organizing the field of flexible query languages in relational databases can shed some light on the most relevant proposals in the area, as well as guide designers and interested users in understanding and selecting the best approaches to suit their aims.

References

Baldwin JF, Coyne MR, Martin TP (1993) Querying a database with fuzzy attribute values by iterative updating of the selection criteria. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'93)

Baldwin JF, Martin TP, Pilsworth BW (1995) FRIL - Fuzzy and evidential reasoning in artificial intelligence. John Wiley & Sons, Inc, New York

Bosc P (1999) Fuzzy Databases. In: Bezdek J, Dubois D, Prade H (eds) Fuzzy sets in approximate reasoning and information systems, The Handbooks of Fuzzy Sets Series. Kluwer Academic Publishers, pp 403-468

Bosc P, Lietard L, Pivert O (2000) About ill-known data and equi-join operations. In: Larsen HL, Kacprzyk J, Zadrozny S, Andreasen T, Christiansen H (eds) Flexible query answering systems. Recent advances. Physica-Verlag, Heidelberg New York, pp 65-74

Bosc P, Pivert O (1992) Fuzzy querying in conventional databases. In: Zadeh LA, Kacprzyk J (eds) Fuzzy logic for the management of uncertainty. John Wiley & Sons, pp 645-671

Bosc P, Pivert O (1993) An approach for a hierarchical aggregation of fuzzy predicates. In: Proceedings of 2nd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE´93), USA, pp 1231-1236

Bosc P, Pivert O (1995) SQLf: A relational database language for fuzzy querying. IEEE Transactions on Fuzzy Systems 3: 1-17

Bosc P, Pivert O (1997) Fuzzy queries against regular and fuzzy databases. In: Andreasen T, Christiansen H, Larsen HL (eds) Flexible query answering systems. Kluwer Academic Publishers, pp 187-208

Bosc P, Pivert O (1997) On representation-based querying of databases containing ill-known values. In: Proceedings of International Symposium on Methodologies for Intelligent Systems (ISMIS' 97), pp 477-486

Bosc P, Pivert O, Lietard L (2001) Aggregate operators in database flexible querying. In: Proceedings of 9th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'2001), Melbourne, Australia, pp 1231-1234

Bosc P, Pivert O, Lietard L (2003) On the comparison of aggregates over fuzzy sets. In: Bouchon-Meunier B, Foulloy L, Yager RR (eds) Intelligent systems for information processing. From representation to applications. Elsevier, pp 141-152

Buckles BP, Petry FE (1985) Query languages for fuzzy databases. In: Kacprzyk J, Yager RR (eds) Management decision support systems using fuzzy sets and possibility theory. Verlag, TUV Rheinland, pp 241-251

Buckles BP, Petry FE, Sachar HS (1986) Design of similarity-based relational databases. In: Prade H, Negoita CV (eds) Fuzzy logic in knowledge engineering. Verlag TUV Rheinland, pp 3-7

Codd EF (1970) A relational model of data for large shared data banks. Communications of the ACM 13(6): 377-387

Dubois D, Prade H (1990) Measuring properties of fuzzy sets: A general technique and its use in fuzzy query evaluation. Fuzzy Sets and Systems 38(2): 137-152

Dubois D, Prade H (1997) Using fuzzy sets in flexible querying: why and how?. In: Andreasen T, Christiansen H, Larsen HL (eds) Flexible query answering systems. Kluwer Academic Publishers, pp 45-60

Fodor J, Yager RR (2000) Fuzzy set-theoretic operators and quantifiers. In: Dubois D, Prade H (eds) Fundamentals of fuzzy sets. Kluwer Academic Publishers, 125-193

Galindo J; Medina JM; Aranda GMC (1999) Querying fuzzy relational databases through fuzzy domain calculus. International Journal of Intelligent Systems 14(4): 375-411

Galindo J, Medina JM, Cubero JC, García MT (2000) Fuzzy quantifiers in fuzzy domain calculus. In: Proceedings of 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU´2000), Spain, pp 1697-1702

Galindo J, Medina JM, Pons O, Cubero JC (1998) A server for fuzzy SQL queries. In: Andreasen T, Christiansen H, Larsen HL (eds) Flexible query answering systems. LNAI: 1495, Springer, pp 164-174

Kacprzyk J, Zadrozny S (1995) FQUERY for Access: fuzzy querying for Windows-based DBMS. In: Bosc P, Kacprzyk J (eds) Fuzziness in database management systems. Physica-Verlag, Heidelberg, pp 415-433

Kacprzyk J, Zadrozny S (1997) Implementation of OWA operators in fuzzy querying for Microsoft Access. In: Yager RR, Kacprzyk J (eds) The ordered weighted averaging operators: theory and applications. Kluwer, Boston, pp 293-306

Kacprzyk J, Zadrozny S (1999) Fuzzy querying via WWW: implementational issues. In: Proceedings of 7th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'1999), Seoul, Korea, pp 603-608

Kacprzyk J, Zadrozny S (2000) On a fuzzy querying and data mining interface. Kybernetika 36: 657-670

Kacprzyk J, Zadrozny S (2000) On combining intelligent querying and data mining using fuzzy logic concepts. In: Bordogna G, Pasi G (eds) Recent research issues on the management of fuzziness in databases. Physica-Verlag, Heidelberg New York, pp 67-81

Kacprzyk J, Zadrozny S, Ziolkowski A (1989) FQUERY III+: a 'human-consistent` database querying system based on fuzzy logic with linguistic quantifiers. Information Systems 6: 443-453

Kacprzyk J, Ziolkowski A (1986) Database queries with fuzzy linguistic quantifiers. IEEE Transactions on Systems, Man and Cybernetics SMC 16: 474-479

Kerre EE, De Cock M (1999) Linguistic modifiers: an overview. In: Chen G, Ying M., Cai K-Y (eds) Fuzzy logic and soft computing. Kluwer Academic Publishers, pp 69-85

Klir GJ, Folger TA (1988) Fuzzy sets, uncertainty and information, Prentice-Hall.

Lacroix M, Lavency P (1987) Preferences: putting more knowledge into queries. In: Proceedings of 13rd International Conference on Very Large Databases (VLDB' 87), Brighton (GB), pp 217-225

Liu Y, Kerre EE (1998) An overview of fuzzy quantifiers (I). Interpretations. Fuzzy Sets and Systems 95: 1-21

Medina JM, Pons O, Vila MA (1994) GEFRED: a generalized model of fuzzy relational databases. Information Sciences 76(1-2): 87-109

Petry FE (1996) Fuzzy databases: principles and applications. Kluwer Academic Publishers

Prade H, Testemale C (1984) Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. Information Sciences 34: 115-143

Prade H, Testemale C (1987) Representation of soft constraints and fuzzy attribute values by means of possibility distributions in databases. In: Bezdek JC (ed) Analysis of fuzzy information, vol. II, CRC Press. pp 213-229

Raju KVSVN, Majumdar AK (1988) Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. ACM Transactions on Database Systems 13: 129-166

Ramakrishnan R, Gehrke J (2000) Database management systems, McGraw-Hill.

Ribeiro RA (1993) Application of support logic theory to fuzzy multiple attribute decision problems, University of Bristol, UK

Ribeiro RA, Moreira AM (1999) Intelligent query model for business characteristcs. In: Proceedings of IEEE/WSES/IMACS CSCC'99 Conference, Greece

Ribeiro RA, Moreira AM (2003) Fuzzy query interface for a business database. International Journal of Human Computer Studies 58(4): 363-391

Schmucker KJ (1984) Fuzzy sets, natural language computations, and risk analysis, Computer Science Press

Shenoi S, Melton A (1989) Proximity relations in the fuzzy relational database model. Fuzzy Sets and Systems 31: 285-296

Shenoi S, Melton A, Fan LT (1990) An equivalence classes model of fuzzy relational databases. Fuzzy Sets and Systems 38: 153-170

Tahani V (1977) A conceptual framework for fuzzy query processing: a step toward very intelligent database systems. Information Processing and Management 13: 289-303

Takahashi Y (1991) A fuzzy query language for relational databases. IEEE Transactions on Systems, Man and Cybernetics SMC 21: 1576-1579

Takahashi Y (1995) A fuzzy query language for relational databases. In: Bosc P, Kacprzyk J (eds) Fuzziness in database management systems. Physica-Verlag, Heidelberg, pp 365-384

Ullman JD (1982) Principles of database systems, Computer Science Press.

Umano M (1982) FREEDOM-0: a fuzzy database system. In: Gupta M, Sanchez E (eds) Fuzzy information and decision processes. North-Holland, Amsterdam, pp 339-347

Umano M, Fukami S (1994) Fuzzy relational algebra for possibility-distribution-fuzzy relational model of fuzzy data. Journal of Intelligent Information Systems 3: 7-27

Yager RR (1994) Interpreting linguistically quantified propositions. International Journal of Intelligent Systems 9: 541-569

Zadeh LA (1965) Fuzzy sets. Information and Control 8: 338-353

Zadeh LA (1975) The concept of a linguistic variable and its application to approximate reasoning - II. Information Sciences, 8: 219-269

Zadeh LA (1978) PRUF -a meaning representation language for natural languages. International Journal of Man-Machine Studies 10: 395-460

Zadeh LA (1983) A computational approach to fuzzy quantifiers in natural languages. Computational Mathematics Applications 9: 149-184

Zadrozny S, Kacprzyk J (1998) Implementing fuzzy querying via the Internet/WWW: Java applets, ActiveX controls and cookies. In: Andreasen T, Christiansen H, Larsen HL (eds) Flexible query answering systems. LNAI: 1495, Springer, Berlin Heidelberg, pp 382-392

Zemankova-Leech M, Kandel A (1984) Fuzzy relational databases - A key to expert systems. Koln, Germany, TUV Rheinland.