FUZZY QUERIES AND RELATIONAL DATABASES

P. BOSC & O. PIVERT

IRISA-LLI-ENSSAT

Keywords : imprecise queries, fuzzy sets, possibility theory, query languages.

Abstract.

In this paper, the interest of fuzzy sets and possibility theory in the context of databases is presented. It is shown that these notions provide an homogeneous framework for both the representation of imprecise/uncertain information and vague queries. A special emphasis is put on flexible queries addressed to regular databases. When comparing various attempts made in this context, the fuzzy set approach turns out to generalize the other solutions. Finally, the principal features of a fuzzy querying language extending SQL are outlined.

INTRODUCTION

If the database domain has tremendously evolved in the last decade, an implicit hypothesis has been maintained nearly all the way long : data are assumed to be precisely known and queries are intended to retrieve elements which qualify for a given crisp condition. This paper investigates some of the issues tied to the relaxation of this hypothesis, since it will turn out to be very restrictive for handling new applications and needs.

After a brief overview of the main aspects concerned by imprecision in database management systems, we will focus on the expression of flexible queries (interpreted in the framework of the fuzzy set theory) addressed to regular databases (where data are accurately known). The objective of such queries is to support preferences and to provide users with results which are ranked according to their adequation with respect to the query.

Then, we present different attempts which have been made without fuzzy sets. Basically, the idea is to extend usual Boolean queries in adding preferences and three main approaches have been suggested. In the first one, a query involves two distinct components : one intended for the selection of tuples, another to order them according to preferences. In the second approach, preferences are implicit and hidden behind a similarity operator relaxing the strict equality. A last approach advocates the use of criteria whose results are values of the unit interval (or linguistic values). It is possible to show that these approaches are very specific with respect to a fuzzy set based approach.

In the next section, we give the outline of a query language supporting fuzzy queries in the context of a relational data model. This query language is an extension of SQL which is a standard for database querying. The "where" clause of the multi-relation select block may involve both Boolean and fuzzy predicates combined by several kinds of connectors, thereby achieving a large number of semantic effects.

Partitions issued from a "group by" can be selected by means of fuzzy conditions bearing on the results delivered by set functions, but also using fuzzy quantifiers.

STORAGE OF IMPRECISE DATA

In this section, we consider the representation of imprecise information in the context of the relational model. We are mainly interested in the case where we do not have a complete information on the value of the attributes in the tuples of a relation, because we believe this to be the most common situation.

If we use a classic DBMS, we are limited because we do not know how to represent disjunctive information, i.e. cases where the effective value of an attribute is one of those specified in a set. Practically, therefore, we have a choice between three solutions : i) to choose a precise arbitrary value from the information available ; ii) to use a null value ; iii) to choose a specific representation (intervals, for example) which then supposes that we have adapted tools at query language level. None of these solutions is satisfactory and it seems more appropriate to allow for such information (in fuzzy databases) to be taken into account by the system itself.

We illustrate the ability to represent ill-known values by means of possibility distributions (denoted π in the following) in some characteristic situations relative to John's salary. Of course, we must be able to express the usual situations, knowing the precise value (case a) ($1764), the "unapplicable" null value (case b) (John does not work), and the "unknown" null value (case c) (John works, but the value of his salary is unknown). We can even represent total ignorance where anything is possible (case d) (John may not be working and, even if he is, his salary is unknown). The possible values of his salary are found in the domain S (e.g. the interval [$500, $80000]) to which a particular element denoted {e} is attached to express the non-applicability of the attribute to the element under consideration. The distributions relevant to these different cases are given hereafter : a) $\pi(1764) = 1$; $\forall x \in S - \{1764\}\ \pi(x) = 0$; b) $\pi(e) = 1$; $\forall x \in S - \{e\}\ \pi(x) = 0$; c) $\pi(e) = 0$; $\forall x \in S - \{e\}\ \pi(x) = 1$; d) $\forall x \in S\ \pi(x) = 1$.

Possibility distributions also allows to represent imprecise or uncertain information, including variations in the possible values (some are more possible than others). Some examples are :

a) the exact salary is unknown but in the interval [$1000, $2000] : usual interval; b) the exact salary is unknown but is around $1500 : the possibility decreases as the considered value moves away from $1500; c) it is known only that the salary is high : distribution corresponding to the restriction of salary by the fuzzy set "high"; d) it is certain to a degree α that the salary is around $1500 : the distribution $\pi(x)$ is $\max(\mu_{\text{around } \$1500}(x), 1-\alpha)$ for all $x \neq e$ and $\pi(e) = 0$ ($\mu_{\text{around } \$1500}$ is the characteristic function for the set around $1500).

These different situations involve a single-valued attribute and it is clear that if one limits oneself to characteristic functions with a trapezoid form, the inner representation of such values amounts to five values at the most (example d). Equally, one can represent values relative to a multi-valued

attribute. It is clear, then, that its storage requires more space, since we must consider a priori a fuzzy set over the powerset of the universe.

HANDLING IMPRECISE DATA AND VAGUE CRITERIA

Introduction to vague criteria

When querying a database, one sometimes does not wish to define precise limits of acceptance or rejection for a condition due to a feeling of graduality. This also occurs when one wishes to express preferences and thus to distinguish between the elements in a finer way than a Boolean filter would. This attitude has the effect, among others, firstly of being able to provide a reply where a classic request would have produced an empty response, and secondly of returning to the user the n best replies rather than a long list of undifferentiated replies. As a result, such an approach seems adapted to avoid the need for the user to formulate a succession of requests. Here, vague criteria are assumed to be represented by fuzzy sets. A vague predicate can be made up of several types of elements : atomic predicates, modifiers , connectors and quantifiers. An atomic predicate is an application from a set of domains $D_1, .., D_n$ to [0,1]; it often corresponds to an adjective such as "tall", "young", "important", etc. A modifier is an application from [0,1] to [0,1] corresponding to an adverb such as "very", "really", "more or less", "rather", etc. A connector is an application from $[0,1]^n$ to [0,1], such as for example the conjunction and the disjunction (weighted where necessary), and the mean operators [13]. A quantifier is an application from [0,1] or the real line to [0,1]. It is used in clauses like "most of the employees which satisfy A satisfy also B" or "a dozen departments satisfy B". Several interpretations of quantifiers have been suggested [10, 14, 16].

Matching vague criteria against imprecise data

We now move on to the general case of filtering imprecise/uncertain data by a vague criterion which will be illustrated by the extension of relational selection. The result is a pair of fuzzy sets $t\Pi$ and tN where each tuple of a relation R is linked to the degree of possibility (resp. necessity or certainty) that it satisfies the criterion (denoted F hereafter). It should be noted that a relation is not obtained and that the usual principle of compositionality is not preserved. Because of complementation, we have :

$$t\Pi(R; \bar{F}) = 1 - tN(R; F) \text{ and } tN(R; \bar{F}) = 1 - t\Pi(R; F)$$

We can approach compound predicates by conjunction and disjunction using formulae :

$t\Pi(R; F_1 \textbf{ or } F_2) = t\Pi(R; F_1) \cup t\Pi(R; F_2)$;
$tN(R; F_1 \textbf{ or } F_2) = tN(R; F_1) \cup N(R; F_2)$;
$t\Pi(R; F_1 \textbf{ and } F_2) = t\Pi(R; F_1) \cap t\Pi(R; F_2)$;
$tN(R; F_1 \textbf{ and } F_2) = tN(R; F_1) \cap tNR(R, F_2)$.

In the following, we limit ourselves to the case of an atomic predicate in order to explain the basis of the calculation of degrees of possibility and necessity. Considering a relation $R(A_1,...,A_n)$, we take $A_i(t)$ to be the possibility distribution representing the value of attribute A_i in the tuple t and the fuzzy set F representing a criterion applicable to A_i. We have :

$$\mu_{t\Pi(R; F)}(t) = \Pi(F \mid A_i(t)) = \sup_{u \in dom(A_i)} \min(\mu_F(u), \pi_{A_i}(u))$$

$$\mu_{tN(R; F)}(t) = N(F \mid A_i(t)) = 1 - \Pi(\bar{F} \mid A_i(t))$$
$$= \min(\inf_{u \in dom(A_i)} \max(\mu_F(u), 1 - \pi_{A_i}(u)), 1 - \pi(e)).$$

The degree of possibility is 1 if the cores of the fuzzy sets representing the datum and the criterion, have an intersection which is not empty. Similarly, the degree of possibility is strictly positive if their supports overlap. One can observe four situations : i) the general case (imprecise data - vague criterion), ii) the data is imprecise and the predicates are boolean, iii) the data is precise and the predicates are vague; in the particular case where the condition has the form X = a, the necessity degree is null except if the data is represented by $\pi(a) = 1$, $\pi(x \neq a) = 0$, iv) both data and criterion are precise (the regular case). In the last two cases, we notice that the degrees of possibility and necessity become equal since there is no uncertainty on the result of the query.

The particular case mentioned above where the condition is : "attribute = value" can be extended to the two more general cases : i) "attribute θ value" or ii) "$attribute_1$ θ $attribute_2$" where θ is any comparison operator. The former is typically a predicate involved in a relational selection and the latter a joining predicate. These cases will not be detailed here for the sake of conciseness.

We have presented the basis for the evaluation of a vague condition applying to imprecise data and it is then possible to define an extended relational algebra (see [9]), where the usual operations (selection, projection, join, Cartesian product) are generalized.

FLEXIBLE QUERYING OF REGULAR DATABASES

In this section, we focus on an approach of great interest, since it concerns existing bases (without modification) and provides discriminated answers. The introduction of preferences over the elements which satisfy a constraint has been the object of a limited number of proposals which can be divided into two categories : i) those using fuzzy predicates and ii) those based on an ad'hoc extension of the querying capabilities offered by certain relational systems. The principles of these two approaches are successively described hereafter. Moreover, we show how the allowed queries of type ii) can be expressed in a fuzzy set framework.

Flexible querying with fuzzy sets

In the context of regular databases, the principle of composition is maintained, i.e. if we combine a set of relations, we still get a relation. From an algebraic point of view, we can consider that the relational operations (selection, projection and joins) and set operations are extended to fuzzy relations. One of the first to advocate the use of fuzzy sets for querying conventional databases was V. Tahani [12]. His idea consisted of allowing the expression of imprecise conditions inside queries seen as fuzzy sets. V. Tahani suggested the extension of the SEQUEL base block in order to support the imprecise comparison between an attribute value and a constant or between two attribute values (joins). These elementary predicates can be combined using the connectors AND and OR working as intersection and union of fuzzy sets (min/max). In so far that each tuple receives a degree of membership with respect to the query, discriminated answers are produced "naturally". Beyond the straightforward extension of usual predicates and connectors, one of the interests of the fuzzy sets is to allow for sophisticated operations on predicates, especially using fuzzy connectors. We describe in the last part of the paper how various fuzzy querying features have been integrated in the unified framework of an extended SQL language in order to provide users with a wide range of imprecise queries [1].

About the expression of the other approaches in the fuzzy sets framework

In the next subsections, we will show how the queries allowed in ad'hoc extensions of relational systems can be expressed in terms of fuzzy sets. Basically, any such query involves two

aspects : a selection part (S) and an ordering part (O) and can be seen as : "select the tuples satisfying S, then rank them according to O". In the framework of fuzzy sets, we will have one component for S (S') valued over {0,1} (generally S itself) and one for O (O') expressing the ordering behaviour of the system as a membership degree over [0,1]. It is of course of importance that the effect of O' is exactly the same as that of the initial mechanism expressed by O. One major problem is then to find out an appropriate fuzzy set expression in each case. In the following subsections, we will point out this expression for various kinds of systems.

Complementary ordering criterion

In the PREFERENCES system [7], a query is composed of a principal condition C and a complementary part P that is relative to the description of preferences, both of which are based on boolean expressions. The meaning of this type of query is : "find the tuples which satisfy C and rank them according to their satisfaction of P." This system allows for the combination of preference clauses (P) by means of two constructors : nesting and juxtaposition . Following on from R_C, subset of the tuples of a relation R satisfying condition C, the nesting (resp. juxtaposition) of preference clauses P_1, ... , P_n leads to the sets : S_1 the subset of R_C satisfying P_1 and not P_2 (resp. one single clause); S_2 the subset of R_C satisfying P_1 and P_2 but not P_3 (resp. exactly two clauses); ... ; S_n the subset of R_C satisfying P_1 and ... and P_n. The user receives as an initial response the set S_i, not empty and with the highest index, and he can go back to the previous sets. One of the significant advantages of this system is the avoidance of successive formulations in reaching a desired set of responses. However, it must be noticed that the discrimination capacity remains limited, since it directly depends on the number of preference predicates given by the user. It is possible to show that the ordering produced by the initial condition nest(P_1, ... , P_n) (resp. juxt(P_1, ... , P_n)) can be equivalently expressed in the fuzzy framework as : $AG_1(P_1, \dots, P_n)$ (resp. $AG_2(P_1, \dots, P_n)$) defined as : $\mu_{AG_1(P_1, \dots, P_n)}(x) = \Sigma_i \mu'_{P_i}(x) / n$, $\mu_{AG_2(P_1, \dots, P_n)}(x) = \Sigma_i \mu_{P_i}(x) / n$, where $\mu'_{P_i}(x) = \min_{j \leq i} \mu_{P_j}(x)$ and $\mu_{P_i}(x) = 1$ if $P_i(x)$ is true, 0 otherwise.

Another attempt based on a complementary criterion has been suggested by Chang with the system called DEDUCE2 [4]. However, we have shown that the composition of predicates, which is based on ranks issued from sorts, was not really meaningful in this approach [2].

Distance to an ideal object

A second idea relies upon questions which include conditions resting on the notion of similarity (≈) rather than strict equality. Here we use conditions of the type "X ≈ v" where "v" represents an ideal value, but where other values are nevertheless acceptable (for instance salary ≈ $2000 means that $2000 is excellent but values around (the interval [1950-2050] for instance) can also be accepted). The evaluation of such a condition on an element t is aimed at defining a distance d(t) and obeys the following principle : if X(t) is somewhat similar to the value "v", then the value for d(t) is the fixed distance between X(t) and v, otherwise d(t) is infinite. In the presence of connectors such as conjunction and disjunction, an overall distance must then be calculated, thus allowing the elements concerned to be ordered. Several systems based on an operator of explicit (ARES [6], VAGUE [8]) or implicit (the "nearest neighbor" technique [5]) similarity, have been suggested and we point out the corresponding expression of their ordering semantics in the context of fuzzy sets.

In ARES, elementary distances are attached to a given domain and are given by means of a relation expressing the distance between any two values. In a given query (which involves both boolean (B_i) and predicates involving similarity (S_i) that can only be ANDed), the user chooses a threshold (t_i) for each predicate involving a similarity. The global distance is defined as the sum of the elementary distances tied to the similarity predicates involved in the query. It is possible to show that the initial similarity condition "S_1 **and** ... **and** S_n" and the expression in the framework of fuzzy sets $AG_3(S_1, \dots, S_n)$ lead to the same ranking if AG_3 is defined as : $\mu_{AG_3(S_1, \dots, S_n)}(x) = (\Sigma_i \mu_{S_i}(x) * t_i) / (\Sigma_i t_i)$ with $\mu_{S_i}(x) = (t_i - dist_{S_i}(x))/t_i$ if $dist_{S_i} \leq t_i$, 0 otherwise. Here, the ordering mechanism is obtained through a weighted mean.

In VAGUE, we have three main differences with respect to ARES : the disjunction of predicates is allowed, similarity predicates can be explicitly weighted and the global distance mechanism for a conjunction is based on the euclidian distance. In the framework of fuzzy sets, the disjonction in VAGUE can be expressed by a minimum and the conjunction by a quadratic mean.

In the approach known as nearest neighbors, a query involves a set of values which characterize an ideal tuple M. Each concerned tuple is then compared with M by means of a global function which gathers the results of local distance functions applied to some attributes. One of the most used global functions is the L_p-norm defined as : $(\Sigma_i dist_i(x)^p)^{1/p}$ with $dist_i(x) = |x_i - M_i| / (max_i - min_i)$ where x_i and M_i stand for the values of the i^{th} attribute of the current tuple and the model which can vary between min_i and max_i. As opposed to ARES and VAGUE, no element is discarded. In this context, the counterpart of the query : "P_1 **and** ... **and** P_n" is written : $AG_4(P_1, \dots, P_n)$ where AG_4 is defined by its characteristic function : $\mu_{AG_4(P_1, \dots, P_n)}(x) = \Sigma_i \mu_{P_i}(x) / n$ with $\mu_{P_i}(x) = 1 - (dist_{P_i}(x))^p$.

Criteria with preferences and weighting

In the framework of information retrieval, the flexible retrieval system called MULTOS [11] has been proposed. Its principle consists in replacing a traditional criterion with a set of criteria to which an explicit preference (value between 0 and 1, or linguistic term) is attached. Thus, if we are interested in the year of publication, we might write : {year ∈ [1978, 1982] **preferred** , year ∈ [1983, 1988] **accepted**}. Furthermore, one can weight each set of criteria (e.g. the subject matter is more important (**high**) than the price of the document (**medium**) which in turn is more important than the year of publication (**low**)). The connectors, conjunction and disjunction, allow the combination of several criteria. We have shown that the expression of the ordering behaviour of MULTOS in the framework of fuzzy sets is based on a weighted averaging operation.

This brief overview of non fuzzy set based approaches aiming at discriminated answers shows that in any case : i) queries are expressible in the context of fuzzy sets, ii) the ordering mechanism is basically a mean, which is not surprising since means are intended to express compromises between several criteria, iii) the allowed queries have a very special typology and iv) each system proposes only one (or two) aggregation mechanism(s) and it is then clear that fuzzy sets provide a much more general framework where the user can choose the appropriate aggregation mechanism.

OVERVIEW OF SQLf

Introduction : a look at SQL

In this section, we present an overview of an extension of a database query language, namely SQL. SQL has the property that a same need can be expressed through several queries,

which gives rise to an equivalence phenomenon. Our two main objectives were to introduce fuzzy predicates into the language wherever possible and so that the equivalences remain valid. First, we recall the principal features of SQL and then we present the extensions.

An SQL query is made of one or several base blocks and is founded on the tuple relational calculus. The fundamental construct is the base block that specifies the structure of the resulting relation by means of the *select clause*, the concerned relation(s) of the database in the *from clause* and the condition to satisfy in the *where clause*. When several relations are involved, one can consider that they are mixed into a single relation (using a Cartesian product) to which the condition applies. This construct has thus at least the power of selection, projection and join operations of the relational algebra.

Rather than putting all relations into a single block, a user can often express his query by means of several nested blocks (also called subqueries). The connection between two blocks can be achieved by several operators : i) set membership ([NOT] IN), ii) set existence ([NOT] EXISTS), iii) existential or universal quantification (ANY, ALL), iv) scalar comparison if the inner block results in a single value using aggregates (MIN, SUM, ...). If we consider a base consisting of the relations EMPLOYEE (num, name, salary, job, age, city, depart), DEPART (nd, manager, budget, location), the query "find the number and name of the employees who work in a department located in their own city" can be expressed :

a) single block : **select** num, name **from** EMPLOYEE, DEPART **where** depart = nd **and** city = location
b) nesting (i) : **select** num, name **from** EMPLOYEE E **where** depart **IN** (**select** nd **from** DEPART **where** location = E.city)
c) nesting (ii) : **select** num, name **from** EMPLOYEE E **where** **EXISTS** (**select** * **from** DEPART **where** nd = E.depart **and** location = E.city).

It must be noticed that queries b) and c) are such that the condition appearing in the subquery refers to the current (and is evaluated for each) tuple of the outer block.

The last important feature of SQL concerns the operations allowed on sets of tuples. As a matter of fact, it is possible to partition using a *group by* clause a relation into subsets, mainly in order to select some subsets using a *having* clause made of set-oriented predicates usually calling on aggregate functions (MIN, AVG : average, ...). The query : "find the departments in which the mean salary of clerks is over 1400" would be stated:

select depart **from** EMPLOYEE **where** job = "clerk"
group by depart **having** AVG(salary) > 1400.

In the next subsections, we will review the various constructs and present ways in which they can be extended to support fuzzy querying capabilities.

Single block queries in SQLf

The objective is to introduce some fuzziness in the base block of SQL. This can be achieved at two principal levels : in the predicates and in the way they are combined (connectors). First of all, we assume that a fuzzy condition fc delivers a fuzzy relation fr (where each tuple has a degree expressing its membership to the relation) and that the result of a query must be a usual relation, more precisely the "best" elements of fr. So, it becomes necessary to provide the user with an output regulation mechanism that can be either the number n of desired responses or $t \in [0,1]$ for the t-cut of fr. In so doing, the new formulation for a simple base block is :
select <n/t> <attr> **from** <relations> **where** <fuzzy cond>.
Sometimes, in the forthcoming examples, we omit this element of a query without loss of generality. Basically, a fuzzy condition applying to individual tuples is composed of Boolean and fuzzy predicates and connectors (and, or, means, etc). Just like in an ordinary query a predicate can express a join between two relations, it is possible to connect two relations by means of a fuzzy predicate, like in :

select ... **from** R, S **where** ... more or less equal (R.A, S.B).

It is possible that several tuples selected by the condition have the same value on the specified attributes but have different grades of membership. We shall assume that only the one with the highest grade is retained. The semantics of a query is mainly based on the following calculus. Let R be a relation (possibly fuzzy) defined on a set of domains denoted $X = \{X1, ..., Xn\}$. The result, denoted Res, of the restriction of R by the predicate P is defined as :

$$\forall x \in X,\ \mu_{Res}(x) = \min(\mu_R(x), \mu_P(x)).$$

Using subqueries

The objective is to define the semantics of operators like IN, etc, when fuzzy relations are involved and to extend them if necessary. Concerning the connector IN, we want that, if fc1 (resp. fc2) stands for a fuzzy condition applying to R (resp. S), it remains valid to use equally :
select R.* **from** R, S **where** fc1 **and** fc2 **and** R.A = S.B or
select * **from** R **where** fc1 **and** A **IN** (**select** B **from** S **where** fc2).

It is possible to show that the equivalence is obtained if the IN predicate is defined as :

$$\mu_{IN}(a, SQ) = \sup_{b \in support(SQ)} (\min(\mu_{=}(a,b), \mu_{SQ}(b)))$$

where SQ denotes the result of the subquery. In fact, it is not compulsory to retrieve all the attributes of R (specified by *). In the following examples and formulae, we will assume this fact only for the convenience of notation. In general, the query **select** A **from** R **where** fc results in a set of A values and we have to define the grade of membership of any a in dom(A) that is :

$$\mu(a) = \sup_{x \in support(R)} (\min(\mu_R(x), \mu_{fc}(x)) \mid x.A = a).$$

According to the example given in introduction, we have to consider (when it is meaningful) the case of a fuzzy query involving the EXISTS predicate equivalent to a query expressed using a single block. Two kinds of interpretations are a priori possible for the predicate EXISTS (select ...) : a quantitative one based on the cardinality of the considered fuzzy set (resulting from the select) and a qualitative one based on the determination of the extent to which at least one element belongs to this set. This second interpretation has been retained since it preserves the equivalence between :
select R.* **from** R, S **where** fc1 **and** fc2 **and** R.A ≈ S.B and
select * **from** R **where** fc1 **and** EXISTS (**select** * **from** S **where** fc2 **and** B ≈ R.A)

as long as we have for any subquery SQ :

$$\mu_{EXISTS}(SQ) = \sup_{x \in support(SQ)} (\mu_{SQ}(x)).$$

The last nesting mechanism concerns the quantifiers (ALL, ANY). Here again, the objectives were to allow the use of these quantifiers together with fuzzy comparisons and to preserve the equivalences. The semantics of these extended operators can be found in [1].

Partitioning and quantification

We saw in SQL that it is possible to apply conditions to sets of tuples issued from a given relation. In SQLf, our intention is to extend this capability in allowing fuzzy conditions for sets of tuples in a having clause. The first extension is derived directly from SQL using aggregates whereas the second relies on fuzzy quantifiers. These two mechanisms can obviously be mixed in a same fuzzy query.

In SQL, the selection of a partition is obtained by using a predicate involving one or several aggregate functions. This kind of feature has been slightly adapted in the context of SQLf according to two directions. The aggregates are still used but their result can be a parameter of a fuzzy predicate. Moreover, the various conditions can be linked by fuzzy connectors. The following example searching for the 10 best departments with respect to the condition "the mean salary of clerks is around 1600" illustrates this possibility :

select 10 depart **from** EMPLOYEE **where** job = "clerk"
group by depart **having** AVG(salary) = "around 1600"

A second way of qualifying partitions relies on the use of fuzzy quantifiers and has no counterpart in SQL. These quantifiers allow the expression of fuzzy constraints on the sum or the proportion that characterizes the absolute or relative cardinality of a fuzzy set [14, 16]. Let us recall that absolute (several, about 5,...) and relative (none, a few of, most of,...) quantifiers can be used. In the context of SQLf, such quantifiers are used to determine the extent to which the different partitions of a relation satisfy a proposition. The general syntax is : **select** ... **from** ... **where** ... **group by** ... **having** ... <quantified proposition> ... Two kinds of basic predicates are possible : i) Qf **are** fc, where Qf is an absolute quantifier applying to the number of tuples of a partition that satisfy the fuzzy condition fc, ii) Qf [fc1] **are** fc2, where Qf is a relative quantifier that applies to the proportion of tuples of a given partition that satisfy fc2 with respect to those that satisfy fc1 (all if fc1 is omitted). If we want to retrieve the 10 best departments with respect to the condition "most of the young employees are well-paid", we can write : **select** 10 depart **from** EMPLOYEE **group by** depart **having** most-of (age = "young") **are** "well-paid".

Different interpretations are possible, notably : one based on the crisp cardinality of a fuzzy set [16], and one based on the use of a specific mean (OWA) [13]. It must be noted that fuzzy quantified propositions can be used to express a generalized quotient operator . This point is not detailed here and any interested reader can refer to [15].

CONCLUSION

This paper is concerned with fuzzy sets and their contribution to database management systems which concerns two levels : the representation and storage of imprecise or uncertain data and its handling through the introduction of vague criteria inside queries.

We have shown how the values of an imprecise information could be represented by an appropriate distribution of possibility. We have also described the principle of selecting this data by means of conditions which may themselves be imprecise or vague. In the general case, each element receives two satisfaction degrees : one expresses the possibility, the other the certainty that the datum satisfies the criterion.

We devoted the last two sections to a particular case : the querying of classic (relational) databases with the aid of vague conditions. The central interest in this type of query is the introduction of flexibility into the criterion and the ordering of answers according to their degree of satisfaction. In order to meet this need, some solutions based on an ad'hoc extension of boolean systems have been suggested and are briefly described in this paper. It is also possible to base the interpretation of such requests on fuzzy sets and we have shown that this process encompasses the previous ones and allows a wide semantic variety to be achieved.

Finally, we have presented an extension to the relational language SQL, in which fuzzy predicates may be used. One of the aims was to introduce imprecise capabilities wherever possible and moreover to adopt an extension so that most of the usual equivalences in SQL remain valid in SQLf. The extended language has the same structure as SQL and it is possible to apply imprecise conditions to individual tuples as well as to sets of tuples issued from partitioning. In this latter case, conditions involving fuzzy quantifiers are allowed and they have no counterpart in SQL.

An important topic related to the support of additional capabilities concerns the performances. It seems that a reason why some methods do not rely on fuzzy sets is some ease in the implementation that is very close to usual systems and so no additional complexity appears. In the context of a system intended for supporting the querying capabilities of SQLf, specific strategies are necessary. This aspect has not been examined in the paper but we can mention that, for a subset of imprecise queries, a method based on the derivation of a boolean query that is expected to select a small subset comprising all the desired tuples has been proposed [3].

REFERENCES

[1] Bosc P. & Pivert O., About equivalences in SQLf, a relational language supporting imprecise querying, Proc. IFES, Yokohama, November 1991, pp 309-320.

[2] Bosc P. & Pivert O., Some approaches for relational databases flexible querying, Journal of Intelligent Information Systems, 1, 1992, pp 323-354.

[3] Bosc P. & Pivert O., On the evaluation of simple fuzzy relational queries : principles and measures, in Fuzzy Logic : State of the Art (R. Lowen editor), Kluwer Ac. Pub., 1993.

[4] Chang C.L., Decision support in an imperfect world, Research report RJ3421, IBM San José, CA, USA, 1982.

[5] Friedman J.H., Baskett F., Shustek L.J., An algorithm for finding nearest neighbors, IEEE Transactions on Computers, 1975, pp 1001-1006.

[6] Ichikawa T. & Hirakawa M., ARES : a relational database with the capability of performing flexible interpretation of queries, IEEE Transactions on Software Engineering, 12(5), 1986, pp 624-634.

[7] Lacroix M. & Lavency P., Preferences : putting more knowledge into queries, Proc. 13th Conference VLDB, Brighton, GB, September 1987, pp 217-225.

[8] Motro A., VAGUE : a user interface to relational databases that permits vague queries, ACM Transactions on Office Information Systems, 6(3), 1988, pp 187-214.

[9] Prade H. & Testemale C., Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries, Information Sciences, 34, 1984, pp 115-143.

[10] Prade H., A two-layer fuzzy pattern matching procedure for the evaluation of conditions involving vague quantifiers, J. of Intelligent and Robotics Systems, 3, 1990, pp 93-101.

[11] Rabitti F., Retrieval of multimedia documents by imprecise query specification, Lecture Notes on Computer Science, 1990, 416.

[12] Tahani V., A conceptual framework for fuzzy query processing; a step toward very intelligent database systems, Inf. Processing and Management 13, 1977, pp 289-303.

[13] Yager R.R., On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Trans. on Systems, Man and Cybernetics 18, 1988, pp 183-190.

[14] Yager R.R., Connectives and quantifiers in fuzzy sets, Fuzzy Sets and Systems, 40, 1991, pp 39-76.

[15] Yager R.R., Fuzzy quotient operators for fuzzy relational databases, Proc. IFES, Yokohama, November 1991, pp 289-296.

[16] Zadeh L.A., A computational approach to fuzzy quantifiers in natural language, Computer and Mathematics with Applications, 9(1), 1983, pp 149-184.