World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:2, 2007

Towards an Extended SQLf: Bipolar Query Language with Preferences

L. Ludovic, R. Daniel, and S-E Tbahriti

***Abstract*—**Database management systems that integrate user preferences promise better solution for personalization, greater flexibility and higher quality of query responses. This paper presents a tentative work that studies and investigates approaches to express user preferences in queries. We sketch an extend capabilities of *SQLf* language that uses the fuzzy set theory in order to define the user preferences. For that, two essential points are considered: the first concerns the expression of user preferences in *SQLf* by so-called fuzzy commensurable predicates set. The second concerns the bipolar way in which these user preferences are expressed on mandatory and/or optional preferences.

***Keywords*—**Flexible query language, relational database, user preference.

I. Introduction

THE amount of information managed by the database management systems (DBMS) becomes increasingly important. As a consequence the interrogation of database should be more and more efficient. This performance can be measured in terms of query time response or a delivered information quality. In particular, taking into account user preferences in query is a key element of relevance. The problem of expressing and managing user preferences has received more and more attention in the last few years [1, 5, 21, 23]. It was shown [7, 13] that the fuzzy set theory provides efficient tools to incorporate user preferences in queries. *SQLf* language is an example that illustrates this idea. More precisely, "vague conditions" with preferences allow to describe the personalized needs of each user. This paper aims to introduce an extension of *SQLf* Language to integrate optional user preferences in a bipolar form (tanks to a new ''THEN'' clause) and to present a brief overview of the recent approaches in order to express user preferences in queries.

In relational mode of database, preferences are mainly employed to filter and personalize the information sought by users. Two general approaches are distinguished in the literature to express preferences. The *implicit* approach in which, each value of attribute is associated with a score. The value is preferred to another if it has a better score. The *explicit* approach in which the user directly expresses his preferences on the various attribute values. That means the preferences are defined by comparing the attribute values. In addition, these preferences can be seen in a bipolar way, i.e., mandatory preferences (viewed as constraint) and optional preferences –viewed as wishes). In this context, the answers of a query must satisfy absolutely all mandatory preferences and satisfy as possible the optional preferences.

The purpose of this paper is to specify on the one hand, the features of the implicit and explicit approaches like that of bipolarity. The emphasis is put on the consequences of the obtained results, when these techniques are applied, in particular if a total or partial order is obtained. On the other hand, we sketch the extended *SQLf* language to integrate optional preferences we are currently working on.

In the following Section we study the features of the two approaches to express preferences and a special importance is put on the commensurability assumption and the impact it has on the query responses. In Section 3 we describe bipolar concept of preferences. Section 4 is dedicated to a presentation of the main extension of *SQLf* language. In Section 5 the main interrogation systems with preferences are investigated and positioned with two axes (*preference expression,* and the *bipolarity*). Conclusion summarizes the principal contribution of the paper.

II. Preferences and Order Relations

In the context of relational database, elementary preferences are defined on attribute values then composed to define more sophistical preferences. Each attribute A_j has associated a domain values D_j. A tuple t_i associates to each A_j a value taken from its domain. For a given attribute, two general ways are used to express user preferences (the *implicit* and *explicit* way).

In the *implicit* approach a scoring function is associated to each attributes [1]. An attribute value is preferred to another if it has obtained a better score. As an example, a score can be a distance from an optimal value. The element having the least distance is preferred.

Example 1. Consider the relation *Car* (cf. Table I). Numerical scores are assigned to the values of "Make" attribute as follows: (BMW = 3, Audi = 2, VW = 1). More the score is high more the make is preferred. Thus, a preference for the "famous makes" car is expressed. ◆

L. Lietard is with IRISA/IUT rue Edouard Branly BP 30219 LANNION, France (e-mail: ludovic.lietardr@univ-rennes1.fr).
D. Rocacher and S-E. Tbahriti with IRISA/ENSSAT BP 447 22305 LANNION, France (e-mail: rocacher@enssat.fr, tbahriti@enssat.fr).

The fuzzy set [24] constitutes a more specific support in which this idea can be instantiated. The following fuzzy set expresses thus the preference for the "famous make":
Famous-makes= {1/BMW, 0.7/Audi, 0.3/VW}. Each element of this set is associated with a degree. More the degree is high, more the element is preferred. An important effect of the use of score functions if that they induce a *total order* on the values of each attribute.

TABLE I
CAR RELATION

Tuple-id	Make	Price	Cons.	Hp.	Color
t_1	BMW	30000	9/100	90	Green
t_2	Audi	15100	8/100	91	Green
t_3	VW	15000	7/100	91	Black

In the *explicit* expression approach, the preferences are explicitly defined on the attribute values (e.g., someone prefers "green" car to "red" car), which is quite natural to the user's viewpoint. In this case, these attribute values ca, be *totally* or *partially* ordered.

Generally, the aim is to classify the tuples from several elementary preferences. Two directions can be followed; the first point is based on the comparison of the scores, the second based on explicit comparison inter-tuples. We present in the two following sub-sections these directions for preferences expression.

A. Elementary Preferences Defined by a Scoring Function

In the case where elementary preferences expressed by scoring function [17], each tuple t_i is associated with a scores vector $(s_1^i,...,s_n^i)$ Please submit your manuscript electronically for review as e-mail attachments. Each s_j^i corresponds to the evaluation of a preference on attribute A_j of tuple t_i. Two assumptions are considered according to the commensurability of these scores.

If the *commensurability* assumption holds, all scores of a vector are based on the same scale of satisfaction in order to be compared. Thus, scores can be combined by means of an aggregation function *f* (*average, weighted average, min,* etc.) to give a global evaluation to the vector. Consequently, a *total order* relation is established between the score vectors. In this case, t_i is preferable to t_j if and only if: $f(s_1^i,...,s_n^i) \geq f(s_1^j,...,s_n^j)$. Generally *f* is an "ad-hoc" numerical function which can be defined by a user. Obviously *f* must take into account the relative senses of the scores which it combines so that the aggregation mechanism is meaningful. It means that *f* should make, in somehow, these scores commensurable.

When fuzzy predicates are used, the scores are satisfaction degree in [0, 1] and a logical meaning is allotted to them [4]. The functions to aggregate them use logical extended operators [4, 18] (*triangular norm, co-norm, fuzzy implication*, etc.).

Example 2. Consider the following query which has to find the "famous make" and "not expensive" cars. Each degree represents a satisfaction with regard to respective conditions, "famous make" and "not expensive" (cf. Table II). These degrees are aggregated according to a triangular norm "min" (expressing a conjunction), which gives the final degrees: 0.2, 0.7 and 0.5. The tuple are classified as follows: t_2, t_3 then t_1, which mean that t_2 is preferable to t_3 and the last one is preferable to t_1. ◆

TABLE II
ASSIGNEMENT OF DEGREES TO THE ATTRIBUTE VALUES

Tuple-id	Make	Price	$Degree_{make}$	$Degree_{price}$
t_1	BMW	30000	*0.9*	*0.2*
t_2	Audi	15100	*0.7*	*0.7*
t_3	VW	15000	*0.5*	*0.9*

It is also possible within the fuzzy set field to apply other mechanisms like the *Leximin* or *Discrimin* operators [14].

The *Leximin* operator is based on a permutation of the scores of each vector in order ti be able to compare them. It is defined as follows: if $t*$ and $s*$ are two permutations of t respectively s so that $t^*_1 \leq \ldots \leq t^*_n$ and $s^*_1 \leq \ldots \leq s^*_n$ then $t <_{Leximin} s \Leftrightarrow \exists\, k \leq n, \forall i < k, t_i = s_i$ and $t^*_k > s^*_k$.

The *Discrimin* ordering between two score vectors. It is defined as follows: if $D(u, v) = \{i, u_i \neq v_i\}$ is the set of index for which the corresponding values in the scores vectors u and v are different, $u >_{\text{Discrimin}} v$, $\min_{i \in D(u,v)} u_i > \min_{i \in D(u,v)} v_i$.

In the no commensurability assumption the scores allotted to the various attributes of a tuple are not comparable. Consequently these scores can not be aggregated and only a partial order can be defined on the tuples. In particular, the Pareto order can be used.

Pareto Order. We want to compare two tuples, *v* and *u* such as $v = (v_i \ldots v_n)$, $u = (u_1 \ldots u_n)$. It is defined as follows [14, 15]: $v >_{\text{Pareto}} u \Leftrightarrow (\forall i, v_i \geq u_i, \exists j, v_j > u_j)$.

Example 3. Let us consider the following query which has to find the cars of price around 15.000, with a consumption (Cons.) around 7/100 and a horse power (Hp.) around 90. The preferences on these three criteria are qualified by distances (cf. Table III). Since the commensurability assumption is not considered here, the scores on the values of various attributes can not be compared and combined. Consequently, the score vector of v(t_1) = (15000, 2, 0) can not be compared with the other vectors of $v(t_2)$ = (100, 1, 1) and $v(t_3)$ = (0, 0, 1). By using the Pareto Order, the result of example 3 is given as follows: $v(t_3) >_{\text{Pareto}} v(t_2)$ but $v(t_1)$ can not be compared neither with $v(t_2)$ nor with $v(t_3)$. Thus, t_3 is thus preferred to t_2 and the t_1 can not be compared neither with t_3 nor with t_2 . ◆

TABLE III
ASSIGNMENT OF THE DEGREES TO THE ATTRIBUTE VALUES

Id	Price	Conso.	Hp	$Dist_{price}$	$Dist_{conso}$	$Dist_{Hp}$
t_1	30000	9/100	90	*15000*	*2*	*0*
t_2	15100	8/100	91	*100*	*1*	*1*
t_3	15000	7/100	91	*0*	*0*	*1*

B. *Elementary Preferences Defined Explicitly*

It has been observed at a glance that implicit approach to express preferences, has a limited expressive power, since they can not be used to model more complex patterns of preferences. For example, if a user wants to indicate his preferences over paint, it is easier to compare them one by one.

Preference over a relation database is expressed by a collection of pairs of tuples [12]. Each pair specifies the preference of one tuple over another one. In this case, a preference relation $\succ$ is defined over R and it is a binary relation such that $t_i \succ t_j$. Thus t_i is preferable to t_j. In explicit preferences, we just assume that tuples can be compared using some logical expressions that, in real case, defines a *partial order* over the tuples. In this context, the commensurability is not necessary; it prohibits taking into account compensation phenomena between various preferences (contrary to the score approach). When several elementary preferences are considered, the preference relation between two tuples consists in comparing directly the values of each attribute.

Example 4. The preference a car with a green color and low consume is defined on the both attributes "color" and "consumption". Each tuple is associated with a vector composed of a color and consumption. For example, v_{t1} is associated with (green, 9), v_{t2} is associated with (green, 8) and v_{t3} is associated with (black, 7). For the comparison of vectors, one can use a partial order relation defined by the Pareto order. In the previous example $v_{t1} \succ v_{t2}$ and v_{t3} can not be compared to v_{t1} or v_{t2}. ◆

III. BIPOLARITY OF PREFERENCES

User preferences are not considered at all time mandatory. This idea has been illustrated by the concept of bipolar information proposed by Dubois and Prade [16]. The bipolarity concept distinguishes, on the one hand, mandatory preferences, called *constraints*, from optional preferences, called *wishes*: Wishes are free, but there is no guarantee that they can all be satisfied at all times. Constraints and wishes are respectively defined by acceptable values set, noted ***A***, and a desired values set, noted ***D***. For the constraints, queries are exact-match with hard selection criteria, delivering exactly the desired tuple if it is there and otherwise reject the user's query.

Example 5. A user wants to buy a not expensive (<15000) car and wishes it has a "green" color. The constraint "not expensive" allows determining acceptable cars set (whose price is <15000), then the condition on the color expresses a wish, which if it is satisfied supports the associated answer.

The fundamental property of the constraints and the wishes is that the set of desired values is a subset of acceptable values ($D \subseteq A$). Indeed, it is incoherent to wish non-acceptable values (a constraint defined by "an European car" is incoherent with the whish: "a Japanese car").

It is also possible to consider constraints and wishes are defined by fuzzy set. In this context, the condition of inclusion $D \subseteq A$ is rewritten: For each tuple t_i, $\mu_D(t_i) \leq \mu_A(t_i)$, such as $\mu_D(t_i)$ (respectively $\mu_A(t_i)$) is a satisfaction degree of t_i on D (respectively on A).

Constraints and wishes are different in nature (for example, a non-satisfied wish, does not reject a tuple, unlike a non-satisfied constraint). Consequently, the degrees expressing constraints and wishes are *non commensurable* and can not be combined in a logical expression. They must be treated *independently*. The constraint being imperative, it is possible to order the tuples by using a *lexicographical* order [2] on the constraints and the wishes. Thus, the wishes allow to differentiate between the tuples who are equal with respect to the constraints and a *total order* can be obtained on A and D. For the example 5, a lexicographical method classifies tuples satisfying the constraint "not expensive", and favoured among of them, those, which satisfy the wish "green color". So, t_i is classified before another t_j if $\mu_A(t_i) \geq \mu_A(t_j)$ or $(\mu_A(t_i) = \mu_A(t_j) \wedge \mu_D(t_i) \geq \mu_D(t_j))$.

IV. TOWARDS AN EXTEND OF *SQLF*

This section is mainly concerned with the extension of language *SQLf* capabilities. First of all, we briefly set out the base structure of *SQLf* language, then we present some main extensions for this language along two axes:

1. In order to envisage a greater flexibility and delivered response quality, we define user preferences via the set of fuzzy predicates $P = P_A \cup P_D$ where P_A expresses mandatory preferences and P_D expresses optional preferences.

2. Concerning the optional and mandatory preferences, the aim is that *SQLf* be able to distinguish these two types of preferences. These two axes are orthogonal and can be treated independently.

A. *SQLf*

The *SQLf* language extends the *SQL* language in order to allow the user to formulate queries on atomic conditions defined by fuzzy sets [6, 7]. Each attribute of a tuple is associated with a satisfaction degree μ in [0, 1]. The semantic of degrees is the same, what implies that the criteria are *commensurable*.

A query in *SQLf* language has the following syntax:

```
SELECT [distinct][n|t|n,t] <attributes>
FROM   <crisp relation>
WHERE  <fuzzy condition>
```

Where *fuzzy condition* contains mandatory preferences expressed by a set of commensurable fuzzy predicates P_A. The parameters *n* and *t* of the *select* block limit the number of the answers by using a quantitative condition (*the best answers*) or a qualitative condition (data which satisfy the query according to a level higher than *t*).

B. Extended SQLf

As we have seen in Section 3, constraint and wish can not be combined in a same logical expression. In addition, the processing of constraint must be independent of that of the wish. To guarantee this effect in *SQLf*, we consider a bipolar query in *SQLf* as ***A then D*** where the satisfaction of the wishes in *D* is only used for ordering tuples. For this matter, we extend *SQLf* by introducing a new clause "THEN" to express optional preferences in query, as follows:

```
SELECT [distinct][n|t|n,t] <attributes>
FROM   <crisp relations>
WHERE  <fuzzy condition>
THEN  <optional preferences>
```

The "THEN" clause may involve both Boolean and fuzzy predicates in PD to express optional preferences (wish) combined by several kinds of connectors. Two cases are considered to process optional preferences:

1. PD is a set of commensurable predicates of the same importance level. In this case, we can utilise the Leximin or Discrimin operator to keep, in each class (which corresponds to a certain value of degree relative with A), only the undominated tuples. These operators provide an order on a response set but they do not provide scores.

2. PD is a set of hierarchical commensurable predicates. We can use an operator of semantics "if possible then" level of importance.

Example 6. Let us consider again an instance of the relation car (cf. Table IV) and the following query: Find preferably, among the "*not expensive*" car, those which are "*famous make*". The predicate "*not expensive*" allows to express mandatory preferences *A* defined by fuzzy set. While predicate "*famous make*" permit to describe an optional preference *D*, which if it is satisfied, supports the associated results of query. In particular, for the tuples of the employee relation we have:

not-expensive= {0.2/30000.7/15100, 0.8/15000, 0.9/9000}
famous make= {1/BMW, 0.8/Audi0, 8/VW, 0.3/Fiat}

TABLE IV
ASSIGNMENT OF DEGREES

Tuple-id	Make	Price	Degree $_{make}$	Degree $_{price}$
t_1	BMW	30000	*1*	*0.2*
t_2	Audi	15100	*0.8*	*0.7*
t_3	VW	15000	*0.8*	*0.8*
t_4	Fiat	9000	*0.3*	*0.9*

Each tuple ti is associated with a vector which represents its situation with respect to the atomic conditions (see section 2.1). The result is evaluated on two steps:

First, we selected tuples satisfying the mandatory preferences *A* ("not expensive").

Then we obtain: $t_4(0.9) > t_3(0.8) > t_2(0.7) > t_1(0.2)$ which means that the tuple t_4 is preferable on *A* to t_3 and last one is preferable to t_2 and so on. We keep tuples having a degree α_A higher than 0.5 (for example). Secondly, among the tuples satisfying the mandatory preferences *A* with respect α_A, we select only those which satisfy the optional preferences with a degree $\alpha_D = 0.7$ (for example). Thus, we obtain two classes in which all results are totally ordered, such as *class1* that contains the tuples satisfying *A* and *D* and a *class2* that contains the tuples satisfying only *A*. The final result is as follow: Since t_4 does not satisfy optional condition "famous make", thus it is not selected, *class1* = $\{t_3 > t_2\}$, that means t_3 is preferred to t_2. *class2* = $\{t_4 > t_3 > t_2\}$.◆

Beyond the extension of *SQLf* optional preferences itself, an important question puts relate to performances of a SGBD accepting such queries with preferences, and thereafter an evaluation mechanism of these queries. To enable query processing and optimization, we present flexible query with preferences by means of a system of transformation algebraic rules. We will present in the next research papers the performance experiments.

In conclusion, the predicate selection of the clause *where* permits to select acceptable results that satisfying the mandatory preferences, while "then" clause allows to express optional preferences (*wishes*) and to order the selected tuples. With this way, extended *SQLf* is be able to process user preferences within a bipolar framework. In this context, such preferences are taken into consideration through the expression of *commensurable* fuzzy predicates, modelled by fuzzy set of more or less satisfactory values and the selection of the results is *totally* ordered.

V. Query Languages with Preferences

User preferences can be embedded into database query languages in several different ways. In this section, we briefly present the principal propositions to integrate user preferences in queries. We situate these propositions with respect to implicit and/or explicit preferences and bipolarity.

A. Preference SQL

Queries in Preference SQL [19, 20] are mainly made of two parts :

• a WHERE clause aiming at selecting tuples (using Boolean conditions, also called conditions of type must),

• a PREFERRING clause to specify preferences (also called light conditions) in order to make a discrimination between tuples. The typical query block in Preference SQL is then:

```
SELECT *
FROM      <list of relation>
WHERE     <must conditions>
PREFERRING <light conditions>
```

Basically, the preferences appearing in the PREFERRING clause are defined by distances (from optimal values) or scores (level of satisfaction) and a Pareto ordering is used to distinguish between tuples. Preference SQL delivers to the user the tuples satisfying the WHERE clause which are undominated with respect to the preferences (i.e. such that no preferred tuples can be found). If no tuples satisfy the preferences, all tuples satisfying the WHERE clause are delivered to the user.

Example 7. The query "find hotels from Paris with a price *around 100* and a *high category*" is expressed in Preference SQL by:

```
SELECT *
FROM Hotel WHERE City = Paris
PREFERRING around_100(Price)
        and high(Category);
```

where the relation *Hotel* is given by Table V.

TABLE V
HOTEL RELATION

Tuple-id	Price	Category	City
t_1	*200*	***	*Paris*
t_2	*100*	**	*Paris*
t_3	*150*	**	*Paris*
t_4	*50*	*	*Lyon*

The preference price *around 100* is modelized by a distance (between the price and top value 100), while the preference *high category* is represented by a level (from 1 to 3, depending on the category). This query discards hotel t_4 (since not located in Paris) and evaluates the preferences (in the form of a vector (distance, level)) for hotels t_1, t_2 and t_3 respectively associated to (100, 1), (0, 2) and (50, 2). The Pareto ordering gives: t_2 $_{\text{Pareto}}$ t_3 while t_1 is not comparable with t_2 and t_3. As a consequence, the non dominated tuples correspond to t_2 and hotel t_1 which are presented to the user.

Preference SQL follows the bipolar model, since the *must* predicates represent constraints while the *light* predicates represent wishes (a tuple which does not satisfy the must predicates is discarded and, **if possible**, the non dominated tuples with respect to the *light* predicates are returned to the user), but it is limited to Boolean constraints and non commensurable preferences since elementary preferences are distances or levels.

B. *Top k-queries*

In this approach, an ad-hoc ranking function *f* is used in order to classify all tuples according to the preferences [8, 9, 22]. In turn, a top-k query returns *k* tuple with the highest score for the query. Function *f* is computed on numerical attributes values and can incorporate elementary scores (which can be computed on non numerical attributes).

Example 8. Relation *Persons (name, age, weight, height)* gathers information about people. A top-k query on relation *Persons* is "find the 5 best persons according to the preference *to be over-weighted* and *to be young*. The over-weight of a person described by a tuple ti can be calculated by the following function: $f(t_i) = t_i.\text{weight} - (t_i.\text{height} - 100)$, while a fuzzy set *Young* indicates the extent to which it is young. In this case, the ranking function f must make commensurable the overweight and the score μY oung(ti.age and it possible to define: $f(t_i) = (t_i.\text{weight} - (t_i.\text{height} - 100))/(t_i.\text{weight} + \mu_{\text{Young}}(t_i.\text{age}))$, which means that the overweight is all the more important as it is associated to a light weight. Function *f* is evaluated for each person and the 5 best ones are returned. ◆

The approach advocated by top-k queries delivers a total order since elementary preferences are considered commensurable and aggregated in the ranking function. However, some elementary preferences may not be commensurable and the definition of function f may leads, in this case, to a result which can be difficult to justify.

C. *Preference Queries*

This approach provides an algebraic framework to formulate query with preferences and an algebraic operator "winnow" [10, 11]. This one picks the set of tuples which are not dominated, according to a given preference relation $\succ$. It is defined by the following formula: If *R* is a relation schema and $\succ$ preference relation over *R*, then winnow operator is written as $w_{\succ}(r)$ and for every instance *r* of *R*, $w_{\succ}(r) = \{ t_i \in r \mid \nexists\, t_j \in r, t_j \succ t_i \}$. A special case of winnow is called *Skyline* [3] where preferences are predefined and limited to a set of operations.

The interest of winnow is to be justified when the preference relation delivers a partial order. It allows to select elements undominated and can not be compared between them. However, it was shown [25] that a bipolar query where the constraints *A* and wishes *D* are Boolean conditions can be defined by means of the winnow operator. Indeed, $w_{R'}(\sigma_A(T))$ such as *T* is the tuples set and *R'* is the preference relation defined by: $R'(t_1, t_2)$, $P(t_1) \wedge \rceil P(t_2)$, where σ is a classical selection and *P*(.) is predicates corresponding to the preferred conditions.

VI. Conclusion

In this we have studied and presented two families of approaches to express user preferences: the *implicit* and the *explicit*. In the implicit approach, an elementary preference is defined by a score (provided by a function). The aggregation of several scores is possible only when the commensurability assumption holds and leads to a total order of query answers. On the other hand if the preferences are non-commensurable a partial order is obtained since some tuples may be not comparable. In the explicit approach the preferences are specified by binary relation of preferences and in the majority of the cases, a partial order is obtained on the tuples. In addition, the preferences can be considered as *constraints* (mandatory preferences) and *wishes* (optional preferences).

In this paper, we have presented an extension of *SQLf* language in order to integrate optional preference according to bipolar form. The extended *SQLf* language we are currently working on, uses the fuzzy set theory in order to define the preferences and consider the commensurability assumption. This language provides a founded framework to combine mandatory and optional preferences. We have dealt also with the main interrogation systems which support preferences. The processing of such systems has been discussed and positioned with respect to two aspects (preferences expression and bipolarity). The approaches advocated by the systems *Preference SQL* and *Preference Queries* are based on a partial order, consequently, they release to the user only the undominated tuples. *Preference SQL* incorporates a concept of bipolarity in the Preferring clause. In *top-k queries* system, relatively little attention has been devoted to the design of appropriate scoring functions, a problem of critical importance since the quality and usefulness of the top-k answers for a query are highly dependent upon the underlying quality of the scoring technique.

Further studies can be made on the results provided by this paper. First, it will be of interest to define more sophisticated operators to determine the best answers in case of non commensurable preferences. We have presented a first step to an extended *SQLf* language with preferences.

References

[1] A. Agrawal and E.L. Wimmers, "A Framework for Expressing and Combining Preferences". *In Proc. of the 2000 ACM SIGMOD International Conference on Management of Data,* Dallas, USA, pp. 297–306, 2000.

[2] H. Andreka, M. Ryan, and P-Y. Schobbens, "Operators for Combining Preference Relations". In *Jour. of Logic and Computation*, 12(1):13–53., 2002.

[3] S. Börzsnyi, D. Kossmann and K. Stoker, "The Skyline Operator". In *Proc. of the 17th International Conference on Data Engineering*, (ICDE01), Heidelberg, Germany, pp. 421–430, 2001.

[4] G. Bordogna, G. Pasi, "Linguistic aggregation operators of selection criteria in fuzzy information retrieval" In *Int. Jour. of Intelligent Systems*, vol. 10(2), pp.233–248, 1995.

[5] P. Bosc, O. Pivert, "Some approaches for relational databases flexible querying". In *Jour. of Intelligent Information Systems*, 1, pp. 323–354.

[6] P. Bosc, O. Pivert, "SQLf query functionality on top of a regular relational DBMS". In *Knowledge Management in Fuzzy Databases*, O. Pons, M.A. Vila, and J. Kacprzyk (Eds.) Heidelberg: Physica-Verlag, 2000.

[7] P. Bosc, O. Pivert, "SQLf: a relational database language for fuzzy querying". In *IEEE Transactions on Fuzzy Systems*, vol. (3) pp.1–17, 1995.

[8] N. Bruno, L. Gravano and A. Marian, "Evaluating Top-k Queries over Web-Accessible Databases". In *Proc. of the 18th International Conference on Data Engineering* (ICDE02), San Jose, California, USA, pp. 369–382, 2002.

[9] S. Chaudhuri, L. Gravano, "Evaluating top-k selection queries". In *Proc. of the 25th International Conference on Very Large Databases*, (VLDB), Edinburgh, Scotland, pp. 397–410, 1999.

[10] J. Chomicki, "Querying with Intrinsic Preferences". In *Proc. of the 8th International Conference on Extending Database Technology*, (EDBT02), Prague, Czech Republic, pp. 34–51, 2002.

[11] J. Chomicki, "Preference Formulas in Relational Queries". In *ACM Transactions on Database Systems*, (TODS'03), 28(4):1–39, 2003.

[12] P. Ciaccia and R. Torlone, "Finding the Best when it's a Matter of Preference". Technical report available at: *http://www/dia.uniroma3.it/torlone/pubs/pub.htm*, 2002.

[13] D. Dubois and H. Prade, "Using fuzzy sets in database systems: Why and how?". In *Proc. of the 1996 Workshop on Flexible Query-Answering Systems*, (FQAS'96), Roskilde, Denmark, pp. 89–103, 1996.

[14] D. Dubois, H. Fargier and H. Prade, "Beyond min aggregation in multi-criteria decision: (ordered) weighted Min, Discrimin, Leximin". In The Ordered Weighted Averaging Operators - Theory and Applications. R.R. Yager, J. Kacprzyk (Eds.), Kluwer Academic Publ., Boston, pp. 181–192, 1997.

[15] D. Dubois, "Pareto-Optimality and Qualitative Aggregation Structures" 22nd LINZ Seminar on Fuzzy Set Theory, Austria, pages 53–56, 2001.

[16] D. Dubois and H. Prade, "Bipolarity in flexible querying". In *Proc. of the 5th International Conference on Flexible Query Answering Systems*, (FQAS'02), Copenhagen, Denmark, 2002.

[17] P.C. Fishburn, "Preference Structures and Their Numerical representations". Theoretical Computer Science, 217(2): pp 359–383, 1999.

[18] J. Fodor, M. Roubens, "Fuzzy Preference Modelling and Multi-criteria Decision suppot". Kluwer Academic Publisher, 1994.

[19] W. Kießling, "Foundations of Preferences in Database Systems". In *Proc. of the 28th International Conference on Very Large Data bases*, (VLDB), Hong Kong, China, pp. 331–322, 2002.

[20] W. Kießling and G. Köstler, "Preference SQL - Design, Implementation, Experiences". In *Proc. of the 28th International Conference on Very Large Data bases*, (VLDB), Hong Kong, China, pp. 990–1001, 2002.

[21] M. Lacroix, P. Lavency, "Preferences: Putting More Knowledge into Queries". In *Proc. of 13 th International Conference on Very Large Data Bases*, (VLDB), Brighton, England, pp. 217–225, 1987.

[22] C. Li1, M.S. Soliman, C.K. Chang, I.F. Ilyas, "RankSQL: Supporting Ranking Queries in Relational Database Management Systems". In *Proc. of the 31th International Conference on Very Large Databases*, (VLDB), Trondheim, Norway, pp. 1342–1345, 2005.

[23] M. Öztrk, A. Tsoukia`s, P. Vincke, "Preference Modelling". In M. Ehrgott, S. Greco, J. Figueira (eds.), State of the Art in Multiple Criteria Decision Analysis, Springer-Verlag, Berlin, 27–72, 2005.

[24] L. A. Zadeh, "Fuzzy sets". In *Jour. of Information and Control*, vol.8, pp. 338–353, 1965.

[25] S. Zadrozny, J. Kacprzyk, "Bipolar Queries and Queries with Preferences". (Invited Paper). In *Proc. of the 17th International Conference on Database and Expert Systems Applications*, (DEXA'06), Krakow, Poland, pp. 415–419, 2006.