International Journal of Computational Intelligence Systems, Vol. 5, No. 4 (August, 2012), 789-804

An Approach to Database Preference Queries Based on an Outranking Relation

Patrick Bosc [1] , Olivier Pivert [1] , Grégory Smits [2]

[1] *IRISA-ENSSAT, University of Rennes 1*
6, rue de Kerampont, BP 80518
22305 Lannion Cedex, France
E-mail: {bosc, pivert}@enssat.fr

[2] *IRISA-IUT Lannion, University of Rennes 1*
rue Edouard Branly
22305 Lannion Cedex, France
E-mail: gregory.smits@univ-rennes1.fr

Received 9 July 2012
Accepted 18 July 2012

Abstract

In this paper, we describe an approach to database preference queries based on the notion of outranking, suited to the situation where preferences on different attributes are not commensurable. This model constitutes an alternative to the use of Pareto order whose main drawback is to leave many tuples incomparable in general. Even though outranking does not define an order in the mathematical sense of the term, we describe a technique which yields a complete pre-order, based on a global aggregation of the outranking degrees computed for each pair of tuples, which reflects the global "quality" of a tuple w.r.t. the others.

Keywords: Databases, preference queries, outranking.

1. Introduction

The last decade has witnessed an increasing interest in expressing preferences inside database queries. Motivations for such a concern are manifold[15]. First, it has appeared to be desirable to offer more expressive query languages that can be more faithful to what a user intends to say. Second, the introduction of preferences in queries provides a basis for rank-ordering the retrieved items, which is especially valuable in case of large sets of items satisfying a query. Third, on the contrary, a classical query may also have an empty set of answers, while a relaxed (and thus less restrictive) version of the query might be matched by some items in the database. This research trend has motivated several distinct lines of research, in particular fuzzy-set-based approaches and Pareto-order-based ones.

Fuzzy set-based approaches[3,13,14] use fuzzy set membership functions that describe the preference profiles of the user on each attribute domain involved in the query. This is especially convenient and suitable when dealing with numerical domains, where a continuum of values is to be interfaced for each domain with satisfaction degrees in the unit interval scale. Then individual satisfaction degrees associated with elementary conditions are combined

using a panoply of fuzzy set connectives, which may go beyond conjunctive and disjunctive aggregations. It must be emphasized that fuzzy set-based approaches rely on a *commensurability* hypothesis between the satisfaction degrees pertaining to the different attributes taking part to a query.

Approaches based on Pareto order aim at computing non Pareto-dominated answers (viewed as points in a multidimensional space, their set constitutes a so-called skyline), starting with the pioneering works of Bőrzsőnyi *et al.*[2]. Clearly, the skyline computation approach does not require any commensurability hypothesis between satisfaction degrees pertaining to elementary requirements that refer to different attribute domains, as needed in the fuzzy set-based approach. Thus, some skyline points may represent very poor answers with respect to some elementary requirements (while they are excellent w.r.t. others, and Pareto order yields a strict partial order only, while fuzzy set-based approaches lead to complete pre-orders). Kießling[16,17] has provided foundations for a Pareto-based preference model for database systems. A preference algebra including an operator called *winnow* has also been proposed by Chomicki[12] for an embedding of preference formulas into a relational setting (and SQL). See also Torlone and Ciaccia[22], who have focused on the so-called *Best* operator aiming at returning the non-dominated tuples of a relation.

The present paper proposes an alternative to the use of Pareto order in the case where preferences are not commensurable. Our goal is not to show that this approach is "better" than those based on Pareto order, but that it constitutes a different way to deal with preferences inside database queries, that some users may find more suitable and intuitive (at least in some given contexts). The semantics of the two preference models that we propose, a strict and a broad one, is less restricted than the Pareto order model. Indeed, according to the selected model, the result obtained can take into account only discriminative preferences, as for Pareto order, or integrate some compromise by considering indifference situations. Moreover, through the use of indifference thresholds, the model inherently introduces some uncertainty when comparing two tuples. Even though the model we define is not a fuzzy-set-based approach in the sense of the description above, it uses some fuzzy features to compare tuples with one another. The situation considered is that of queries involving preferences on several attributes, which use different ordinal scales and/or different scoring measures (expressed either by fuzzy set membership functions or by *ad hoc* functions as in (Agrawal and Wimmers, 2000)[1]). Then, for a given tuple, each atomic preference leads to compute a score, which may correspond to a level on an ordinal scale. It is assumed, however, that the user does not authorize any trade-off between the atomic preference criteria. In other terms, contrary to the assumption underlying fuzzy-set-based approaches, the scores associated with the different partial preferences *cannot be aggregated*. The approach we advocate rests on the concept of outranking, which was introduced in the context of decision making in (Roy, 1991)[21] but has never been used so far in a database context, to the best of our knowledge. However, the way we define outranking in this paper is a bit different from the definition given in (Roy, 1991)[21], even though the general idea is similar. It is important to emphasize that the mechanism we propose to order the tuples on the basis of their global "quality" yields a *total order* whereas no such mechanism exists in [21] where only a partial order is obtained.

The remainder of the paper is organized as follows. Section 2 presents some related work, in particular the Pareto-based approach to preference handling in databases. In Section 3, we present a preference model based on the notion of outranking and give two versions of it (strict vs. broad preferences). In each case, we define an extension of the model which takes into account smooth transitions between the concepts of concordance, indifference and discordance. In Section 4, we describe the way such preference queries can be expressed by means of an SQL-like language and we briefly deal with query evaluation. Finally, Section 5 concludes the paper and outlines some perspectives for future work. The proofs of all the theorems are in the appendix at the end of the paper.

2. Related work

Let us first recall the general principle of the approaches based on Pareto order. Let $\{G_1, G_2, ..., G_n\}$ be a set of the atomic preferences. We denote by $t \succ_{G_i} t'$ (resp. $t \succeq_{G_i} t'$) the statement "tuple t satisfies preference G_i better than (resp. as least as well as) tuple t'". Using Pareto order, a tuple t dominates another tuple t' iff

$$\forall i \in \{1, \ldots, n\}, t \succeq_{G_i} t' \text{ and } \exists k \in \{1, \ldots, n\}, t \succ_{G_k} t'.$$

In other words, t dominates t' if it is at least as good as t' regarding every preference, and is strictly better than t' regarding at least one preference. The following example uses the syntax of the language *Preference SQL*[17], which is a typical representative of a Pareto-based approach.

Table 1. An extension of relation *car*.

	make	category	price	color	mileage
t_1	Opel	roadster	4500	blue	20,000
t_2	Ford	SUV	4000	red	20,000
t_3	VW	roadster	5000	red	10,000
t_4	Opel	roadster	5000	red	8,000
t_5	Fiat	roadster	4500	red	16,000
t_6	Renault	sedan	5500	blue	24,000
t_7	Seat	sedan	4000	green	12,000

Example 1a. Let us consider a relation *car* of schema (make, category, price, color, mileage) whose extension is given in Table 1, and the query:

select * **from** car
where mileage $\leqslant$ 20,000
preferring (category = 'SUV' **else** category = 'roadster') **and** (make = 'VW' **else** make = 'Ford' **else** make = 'Opel');

The idea is to retain the tuples which are not dominated in the sense of the "preferring" clause. Tuples t_1, t_4, t_5, t_6 and t_7 from Table 1 are discarded since they are Pareto-dominated by t_2 and t_3. On the other hand, t_2 and t_3 are incomparable and the final answer is $\{t_2, t_3\}$.$\diamond$

Consider now the following example.

Example 1b. Let us consider again the extension from Table 1, and the query:

select * **from** car
preferring
(color = 'blue' **else** color = 'red' **else** color = 'green') **and**
(make = 'VW' **else** make = 'Seat' **else** make = 'Opel' **else** make = 'Ford') **and**
(category = 'sedan' **else** category = 'roadster' **else** category = 'coupe' **else** category = 'SUV') **and**
(**least** price) **and** (**least** mileage);

Here, all the tuples are pairwise incomparable. So the final answer is the "flat" set $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, i.e., the entire relation *car*.$\diamond$

When the number of dimensions on which preferences are expressed gets high, many tuples may become incomparable. Several approaches have been proposed to define an order for two incomparable tuples in the context of skylines, based on:

- the number of other tuples that each of the two tuples dominates (notion of k-representative dominance proposed by Lin *et al.* [18]) or
- some preference order of the attributes; see for instance the notions of k-dominance and k-frequency introduced by Chan *et al.* [10,11].

Even if these approaches make it possible to some extent to avoid incomparable elements, they are all based on a *Boolean* notion, namely that of dominance. What we propose is an alternative semantics to the modeling of preference queries involving incommensurable criteria, which takes into account the *extent* to which an element is better than another for a given atomic preference. In other words, the approach we propose is fundamentally gradual, unlike those based on Pareto order such as Skyline and its different variants.

Unlike the family of approaches based on Pareto order, score-based approaches (including those based on fuzzy set theory[3,13,14] as well as the quantitative approach proposed by Agrawal and Wimmers[1] and top-k queries[9]) do not deal with incommensurable preferences. Besides Pareto-order-based approaches, only CP-nets[7,8] handle incommensurable preferences, but they do so only within a restrictive interpretation setting.

The use of CP-Nets in database preference queries has been advocated by Brafman and

Domshlak[8] — whereas this approach was initially developed in artificial intelligence[7]. CP-nets are a graphical representation of conditional *ceteris paribus* preference statements. The underlying idea is that users' preferences generally express that, in a given context, a partially described state of affairs is strictly preferred to another mutually exclusive partially described state of affairs, in a *ceteris paribus* way, i.e., everything else being equal in the description of the two compared states of affairs. With CP-nets, the user describes how his preferences over the values of one variable depend on the value of other variables. For instance, a user may express a query involving the following preference statements:

- s_1: the user prefers minivan cars to sedan cars;
- s_2: for minivans, he prefers Chrysler to Ford;
- s_3: for sedans, he prefers Ford to Chrysler;
- s_4: in Ford cars, he prefers the black ones to the white ones;
- s_5: in Chrysler cars, he prefers the white ones to the black ones.

To sum up, CP-nets deal with *conditional* preference statements and use the *ceteris paribus* semantics, whereas we deal with *non-conditional* preference statements and consider the *totalitarian* semantics (i.e., when evaluating the preference clause of a query, one *ignores* the values of the attributes which are not involved in the preference statement). Let us mention that the totalitarian semantics is implicitly favored by most of the authors in the database community, including those who advocate a Pareto-based type of approach.

3. Principle of the approach

In this section, we first present the basic concepts used further. We then describe two versions of the outranking-based preference query model (strict vs. broad preferences). In each case, we define an extension of the model which takes into account smooth transitions between the concepts of concordance, indifference and discordance.

3.1. Basic notions

3.1.1. Atomic preference modeling

Inside their queries, users can integrate atomic preferences represented by means of an ordinal scale, especially for categorical attributes, or by means of explicit scoring function, mainly for numeric attributes.

An ordinal scale is specified as:

$$S_1 > S_2 > ... > S_m$$

such that the elements from S_1 get score m while those from S_m get score 1. In other words, an ordinal scale involving m levels is associated with a mapping: $level \rightarrow \{1, , \ldots, m\}$ such that the preferred level corresponds to score m and the less preferred one to score 1. A value absent from the scale gets score zero. The scale may include the special element *other* as a bottom value so as to express that any value non explicitly specified in the list is an acceptable choice but is worse than the explicitly specified ones: it then corresponds to score 1. Notice that this way of doing "freezes" the distance between the elements of the list. For instance, with the ordered list $\{$VW, Audi$\}$ $\succ$ $\{$BMW$\}$ $\succ$ $\{$Seat, Opel$\}$ $\succ$ $\{$Ford$\}$, the distance between, e.g., VW and BMW is assumed to be the same as, e.g., that between Opel and Ford. If the user wants to avoid this phenomenon, he/she can elicitate the scores in an explicit manner, specifying for instance: $\{1/\{$VW, Audi$\}$, 0.8/$\{$BMW$\}$, 0.5/$\{$Seat, Opel$\}$, 0.3/$\{$Ford$\}\}$. This has no impact on the interpretation model described further.

As to explicitly defined scoring functions (which concern numerical attributes), they model flexible conditions of the form $attribute \leqslant \alpha$, $attribute \approx \alpha$ and $attribute \geqslant \alpha$ where α is a constant. In the following examples, it will be assumed that they take their values in the unit interval $[0, 1]$ but this is not mandatory.

3.1.2. Concordance, indifference, discordance

The outranking relation relies on two basic notions, concordance and discordance. Concordance represents the proportion of preferences which validate

the assertion "t is preferred to t'", denoted by $t \succ t'$, whereas discordance represents the proportion of preferences which contradict this assertion.

Let A_1, A_2,..., A_n be the attributes concerned respectively by the set of preferences $G = \{G_1, G_2, ..., G_n\}$. Let g_1, g_2, ..., g_n be the scoring functions associated respectively with preferences G_1, G_2, ..., G_n.

Indifferent preferences: Each preference G_j may be associated with a threshold q_j. Preference G_j is indifferent with the statement "t is preferred to t'" iff $|g_j(t.A_j) - g_j(t'.A_j)| \leqslant q_j$. This notion makes it possible to take into account some uncertainty or some tolerance on the definition of the elementary preferences.

Concordant preferences: G_j is concordant with the statement "t is preferred to t'" iff $g_j(t.A_j) > g_j(t'.A_j) + q_j$.

Discordant preferences: Preference G_j is discordant with the statement "t is preferred to t'" iff $g_j(t'.A_j) > g_j(t.A_j) + q_j$.

In the following, we denote by $C(t, t')$ (resp. $I(t, t')$, resp. $D(t, t')$) the set of concordant (resp. indifferent, discordant) preferences from G with respect to $t \succ t'$.

3.2. Strict preference model ($\succ$)

In this first model[5], only the discriminating preferences (i.e., the concordant and discordant criteria) are used to compare two tuples, while the indifferent ones are ignored. We first present a version of the approach where the q_j's are interpreted as crisp thresholds, then we extend it in order to handle fuzzy thresholds.

3.2.1. Basic version

In this crisp version, the degree of outranking attached to $t \succ t'$, denoted by $out_1(t, t')$, reflects the truth of the statement: most of the important criteria are concordant with $t \succ t'$ and few of the important criteria are discordant with $t \succ t'$. It is assumed that a weight w_j is attached to each preference G_j in order to express its importance, and that the sum of the weights equals 1. The outranking degree can be evaluated by the following formula:

$$out_1(t, t') = \top(conc(t, t'), 1 - disc(t, t')) \qquad (1)$$

where $\top$ is a triangular norm and

$$conc(t, t') = \sum_{G_j \in C(t,t')} w_j,$$

$$disc(t, t') = \sum_{G_j \in D(t,t')} w_j.$$

In the following, we use Łukasiewicz' t-norm defined as: $\top_{Lu}(x, y) = max(0, x + y - 1)$ since it has interesting properties (cf. Theorem 3a below).

Theorem 1a. Let us define:

$$ind(t, t') = \sum_{G_j \in I(t,t')} w_j.$$

One has:

$$\forall (t, t'), conc(t, t') + ind(t, t') + disc(t, t') = 1.\square$$

Lemma 1a. One has:

$$conc(t, t') = 1 \Rightarrow disc(t, t') = 0$$

and

$$disc(t, t') = 1 \Rightarrow conc(t, t') = 0.\square$$

Theorem 2a. $\forall (t, t'), conc(t, t') = disc(t', t).\square$

Theorem 3a. (Antisymmetry) If $conc(t, t') \geqslant disc(t, t')$ then:

$$out_1(t, t') = conc(t, t') - disc(t, t') \text{ and}$$
$$out_1(t', t) = 0.$$

Otherwise, one has:

$$out_1(t, t') = 0 \text{ and}$$
$$out_1(t', t) = disc(t, t') - conc(t, t').$$

In other words, if t somewhat outranks t', then t' does not outrank t at all, and reciprocally.$\square$

Theorem 4a. $\forall t, out_1(t, t) = 0.\square$

Example 2a. Let us consider the extension of the relation *car* from Table 1 and the preferences:

for color:
blue $\succ$ black $\succ$ red $\succ$ yellow $\succ$ green $\succ$ white $\succ$ *other*; q_{color} = 1; w_{color} = 0.1;

for make:
VW $\succ$ Audi $\succ$ BMW $\succ$ Seat $\succ$ Opel $\succ$ Ford $\succ$ *other*; q_{make} = 1; w_{make} = 0.2;

for category:
sedan $\succ$ roadster $\succ$ coupe $\succ$ SUV $\succ$ *other*; $q_{category}$ = 1; $w_{category}$ = 0.3;

for price:
score(price) = 1 if price $\leqslant$ 4000, 0 if price $\geqslant$ 6000, linear in-between; q_{price} = 0.2; w_{price} = 0.2;

for mileage:
score(mileage) = 1 if mileage $\leqslant$ 15,000, 0 if mileage $\geqslant$ 20,000, linear in-between; $q_{mileage}$ = 0.2; $w_{mileage}$ = 0.2.

Table 2. Scores obtained by the values from *car* (Example 2a).

	make	category	price	color	mileage
t_1	3	4	0.75	7	0
t_2	2	2	1	5	0
t_3	7	4	0.5	5	1
t_4	3	4	0.5	5	1
t_5	1	4	0.75	5	0.8
t_6	1	5	0.25	7	0
t_7	4	5	1	3	1

Table 2 gives the scores obtained by each tuple for every preference. Notice that in the sense of Pareto order, t_3 dominates t_4 and tuples $t_1, t_2, t_3, t_5, t_6, t_7$ are incomparable. Thus the result of a Pareto-based system such as *Preference SQL*[17] would be the "flat" set $\{t_1, t_2, t_3, t_5, t_6, t_7\}$, whereas the approach we propose yields a much more discriminated result, as we will see below.
Let us compute the outranking degree $out_1(t_1, t_2)$. The concordant criteria are category and color; the indifferent ones are make and mileage; the only discordant one is price. We get:

$conc(t_1, t_2) = w_{category} + w_{color} = 0.4$,
$ind(t_1, t_2) = w_{make} + w_{mileage} = 0.4$,
$disc(t_1, t_2) = w_{price} = 0.2$, hence:
$out_1(t_1, t_2) = max(0, 0.4 + (1 - 0.2) - 1) = 0.2$.$\diamond$

Table 3 gives the outranking degree obtained for every pair of tuples (t, t') from relation *car*. A value in the table corresponds to the outranking degree of $t \succ t'$ where t is the tuple corresponding to the line and t' that corresponding to the column. From such a table, one can build a directed graph such that a vertex corresponds to a tuple, and there is an edge from t to t' iff $out_1(t, t') > 0$. The potential use of such a graph-based modeling is discussed at the end of this subsection.

Table 3. Outranking degrees (Example 2a).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	μ_1
t_1	0	0.2	0	0.1	0.1	0.4	0	0.13
t_2	0	0	0	0	0	0	0	0
t_3	0.1	0.5	0	0.2	0	0.5	0.1	0.23
t_4	0	0.3	0	0	0	0.5	0	0.13
t_5	0	0.3	0	0	0	0.3	0	0.1
t_6	0	0.2	0	0	0	0	0	0.03
t_7	0.3	0.6	0	0.1	0.3	0.5	0	0.3
max	0.3	0.6	0	0.2	0.3	0.5	0.1	
μ_2	0.07	0.35	0	0.07	0.07	0.37	0.02	

Notice that due to the use of indifference thresholds, the degree of outranking does not define an order since the notion of outranking is not transitive (there may exist cycles in the graph). It is worth recalling that the outranking degree computed during the comparison of two tuples does not define a score and does not characterize one of the two compared tuples. This degree represents the trust in the outranking relation between these two tuples.

For instance, consider the three tuples from Table 4. We have:

- t'_1 somewhat outranks t'_2: $out_1(t'_1, t'_2) = 0.2$,
- t'_2 somewhat outranks t'_3: $out_1(t'_2, t'_3) = 0.1$,
- t'_3 somewhat outranks t'_1: $out_1(t'_3, t'_1) = 0.3$.

Table 4. Non-transitivity of outranking.

	make	category	price	color	mileage
t'_1	4	4	0.5	1	0
t'_2	1	2	0.5	7	1
t'_3	5	3	0.5	4	0.5

However, several ways can be envisaged to rank the tuples, based on different aggregations of the

outranking degrees, thus on a *global* evaluation of each tuple:

1. for every t, one computes the degree

$$\mu_1(t) = \frac{\Sigma_{t' \in r \setminus \{t\}} out_1(t, t')}{|r| - 1}$$

where $|r|$ denotes the cardinality of r. Then, one ranks the tuples in decreasing order of μ_1. Then, a tuple t is all the more preferred as it outranks most of the other tuples (where the fuzzy quantifier *most* is assumed to be defined as $\mu_{most}(x) = x, \forall x \in [0, 1]$). With the data from Table 1, one gets the order: $t_7 > t_3 > \{t_1, t_4\} > t_5 > t_6 > t_2$.

2. for every t, one computes the degree

$$\mu_2(t) = \frac{\Sigma_{t' \in r \setminus \{t\}} out_1(t', t)}{|r| - 1}$$

Then, one ranks the tuples in increasing order of μ_2. A tuple t is all the less preferred as most of the other tuples somewhat outrank it. With the data from Table 1, one gets the order: $t_3 > t_7 > \{t_1, t_4, t_5\} > t_2 > t_6$.

3. for each tuple t, one computes the degree:

$$\mu_3(t) = \top(\mu_1(t), 1 - \mu_2(t))$$

where $\top$ is a triangular norm expressing a fuzzy conjunction and one ranks the tuples in increasing order of μ_3. This degree reflects the extent to which t outranks many tuples *and* is not outranked by many. Using the t-norm minimum, the data from Table 1 leads to: $t_7 > t_3 > \{t_1, t_4\} > t_5 > t_6 > t_2$, i.e., the same order as with μ_1 alone. The t-norm product also yields the same order, whereas with Łukasiewicz' t-norm one gets: $t_7 > t_3 > \{t_1, t_4\} > t_5 > \{t_2, t_6\}$. The triangular norm is a parameter of the model and its choice depends on the way one wants to combine the two aspects "outranking many" and "not being outranked by many". A reasonable default choice may be considered to be the minimum (as in a fuzzy querying setting).

4. for each t, one computes the maximum *max* over the outranking degrees $t' \succ t$ for every t' of r. Then, one ranks the tuples in increasing order of *max*. A tuple is all the more preferred as it is not highly outranked by any other tuple. With the data from Table 1, one gets the order: $t_3 > t_7 > t_4 > \{t_1, t_5\} > t_6 > t_2$.

Notice that since the outranking graph built from Table 3 does not induce any outranking cycle, there exists a straightforward method to order the tuples. The idea is to partition the relation the following way:

$class_1$: the tuples from r which are outranked by no other; $r_1 = r - class_1$;
$class_2$: the tuples from r_1 which are outranked by no other from r_1; $r_2 = r_1 - class_2$;
...

Here, we get the order: $t_3 > t_7 > t_1 > t_4 > t_5 > \{t_2, t_6\}$. However, a drawback of this method is that it does not take into account the *degrees* of outranking (it views outranking as a Boolean notion).

Considering that:

- μ_1 and μ_2 only capture one side of the problem (i.e., either how much a tuple outranks the others or how much it is outranked by the others), whereas μ_3 captures both,
- the *max* method is purely qualitative and reflects only the "worst" outranking situation for each tuple,

it can be argued that the most suitable ranking method is that based on μ_3. In order to relativize the impact of small differences between the outranking degrees, it is useful to make these degrees appear in the ranked result returned to the user. In the case of Example 2a, the result would then be presented in the form:

$$0.3/t_7 > 0.23/t_3 > 0.13/\{t_1, t_4\} > 0.1/t_5 > 0.03/t_6 > 0/t_2.$$

3.2.2. Fuzzy version

The idea is to make gradual the transition related to the threshold q_j and consequently the notions of concordance and discordance, as suggested in (Perny and Roy, 1992)[20] in a decision-making context. One introduces a second positive threshold δ_j attached to a preference G_j.

Let $\mu_{ind_j(a)}$ be the trapezoidal membership function of support $[g_j(a)-q_j-\delta_j, g_j(a)+q_j+\delta_j]$ and of core $[g_j(a)-q_j, g_j(a)+q_j]$ (cf. Figure 1).

Let $\mu_{conc_j(a)}$ be the trapezoidal membership function of support $]-\infty, g_j(a)-q_j]$ and of core $]-\infty, g_j(a)-q_j-\delta_j]$ (cf. Figure 1).

Let $\mu_{disc_j(a)}$ be the trapezoidal membership function of support $[g_j(a)+q_j, +\infty[$ and of core $[g_j(a)+q_j+\delta_j, +\infty[$ (cf. Figure 1).

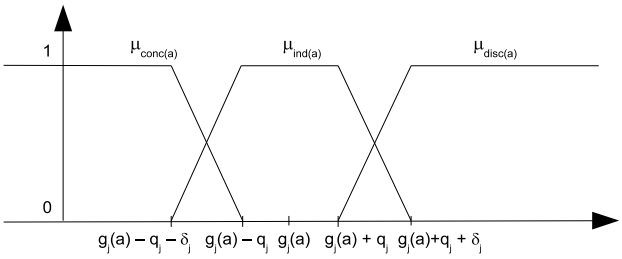

Fig. 1. Concordance, indifference, discordance.

One can define:

- the degree of concordance of G_j with $t \succ t'$ as:

$$c_j(t,t') = \mu_{conc_j(t.A_j)}(g_j(t'.A_j)).$$

- the degree of indifference of G_j w.r.t. $a \succ t'$ as:

$$ind_j(t,t') = \mu_{ind_j(t.A_j)}(g_j(t'.A_j)).$$

- the degree of discordance of G_j with $t \succ t'$ as:

$$d_j(t,t') = \mu_{disc_j(t.A_j)}(g_j(t'.A_j)).$$

Theorem 5a.

$$\forall (t,t'),\ c_j(t,t') + ind_j(t,t') + d_j(t,t') = 1.\square$$

Theorem 6a.

$$\forall (t,t'),\ c_j(t,t') > 0 \Rightarrow d_j(t,t') = 0.\square$$

Theorem 7a.

$$\forall (t,t'),\ c_j(t,t') = d_j(t',t).\square$$

As to the overall degree of concordance (resp. discordance), one defines:

$$conc(t,t') = \sum_{i=1..n} w_i \cdot c_i(t,t').$$

and:

$$disc(t,t') = \sum_{i=1..n} w_i \cdot d_i(t,t').$$

Theorem 8a. Theorem 1a still holds, i.e., one has:

$$\forall (t,t'),\ conc(t,t') + ind(t,t') + disc(t,t') = 1.\square$$

The overall degree of outranking of $t \succ t'$ is expressed as before as:

$$out_1(t,t') = \top_{Lu}(conc(t,t'), 1 - disc(t,t'))$$

Theorem 9a. Theorem 2a still holds, i.e., one has:

$$\forall (t,t'),\ conc(t,t') = disc(t',t).\square$$

Theorem 10a. Theorem 3a still holds.□

Example 3a. Let us use again the data from Table 1 with the same weights w_i and the same thresholds q_i as in Example 2a, and the following thresholds δ_i:

$\delta_{color} = 2$, $\delta_{make} = 2$, $\delta_{category} = 2$,
$\delta_{price} = 0.2$, $\delta_{mileage} = 0.2$

Notice that (q_i = 1, δ_i = 2) means that $ind_i(t,t')$ = 1 if $|t.A_i - t'.A_i| \leqslant 1$, 0.5 if $|t.A_i - t'.A_i| = 2$, 0 if $|t.A_i - t'.A_i| \geqslant 3$. Let us compute the degree of outranking $out_1(t_1,t_2)$. We have:

$c_{color}(t_1,t_2)$ = 0.5, $ind_{color}(t_1,t_2)$ = 0.5,
$c_{category}(t_1,t_2)$ = 0.5, $ind_{category}(t_1,t_2)$ = 0.5,
$d_{price}(t_1,t_2)$ = 0.25, $ind_{price}(t_1,t_2)$ = 0.75,
$ind_{make}(t_1,t_2)$ = 1,
$ind_{mileage}(t_1,t_2)$ = 1.

We get:

$conc(t_1,t_2) = 0.1 \times 0.5 + 0.3 \times 0.5 = 0.2$
$ind(t_1,t_2)$ = 0.75,

$disc(t_1, t_2) = 0.2 \times 0.25 = 0.05$.

Finally:

$out_1(t_1, t_2) = max(0, 0.2 + (1 - 0.05) - 1) = 0.15$.

Using the individual scores from Table 2, we get the outranking degrees represented in Table 5.

Table 5. Outranking degrees (Example 3a).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	μ_1
t_1	0	0.15	0	0	0	0.3	0	0.07
t_2	0	0	0	0	0	0	0	0
t_3	0.3	0.35	0	0.2	0.15	0.4	0.05	0.24
t_4	0.1	0.15	0	0	0.05	0.3	0	0.1
t_5	0.05	0.3	0	0	0	0.35	0	0.12
t_6	0	0.15	0	0	0	0	0	0.02
t_7	0.15	0.55	0	0.15	0.2	0.5	0	0.26
μ_2	0.1	0.27	0	0.06	0.07	0.31	0.008	

For the same reasons as before, we choose the ranking based on μ_3 and the result returned to the user is:

$0.26/t_7 > 0.24/t_3 > 0.12/t_5 > 0.1/t_4 > 0.07/t_1 > 0.02/t_6 > 0/t_2$.◇

3.3. Broad preference model ($\succeq$)

In this second model[6], we take into account all of the criteria (including the indifferent ones) in the definition of outranking.

3.3.1. Basic version

The outranking degree attached to the statement $t \succeq t'$ (meaning "t is at least as good as t'"), denoted by $out_2(t, t')$, now reflects the truth of the statement: most of the important criteria are concordant *or indifferent* with $t \succeq t'$ and few of the important criteria are discordant with $t \succeq t'$. It is evaluated by the following formula:

$$\begin{aligned} out_2(t, t') &= conc(t, t') + ind(t, t') \\ &= 1 - disc(t, t'). \end{aligned} \quad (2)$$

Theorems 1a and 2a straightforwardly hold since they do not depend on the definition of outranking. On the other hand, Theorem 3a does not hold anymore and must be replaced by the following.

Theorem 3b. $\forall (t, t'), out_2(t, t') \geqslant 1 - out_2(t', t)$.□

Theorem 4b. $\forall t, out_2(t, t) = 1$.□

From Equation 2 and Theorem 2a, one gets:

$$out_2(t, t') = 1 - conc(t', t). \quad (3)$$

Table 6. Concordance degrees (Example 2b).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_1	0	0.4	0.3	0.3	0.3	0.4	0.1
t_2	0.2	0	0.2	0.2	0.2	0.2	0.1
t_3	0.4	0.7	0	0.2	0.2	0.6	0.3
t_4	0.2	0.5	0	0	0.2	0.6	0.1
t_5	0.2	0.5	0.2	0.2	0	0.4	0.1
t_6	0	0.4	0.1	0.1	0.1	0	0.1
t_7	0.4	0.7	0.2	0.2	0.4	0.6	0

Table 7. Outranking degrees (Example 2b).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	μ_1
t_1	1	0.8	0.6	0.8	0.8	1	0.6	0.77
t_2	0.6	1	0.3	0.5	0.5	0.6	0.3	0.47
t_3	0.7	0.8	1	1	0.8	0.9	0.8	0.83
t_4	0.7	0.8	0.8	1	0.8	0.9	0.8	0.8
t_5	0.7	0.8	0.8	0.8	1	0.9	0.6	0.77
t_6	0.6	0.8	0.4	0.4	0.6	1	0.4	0.53
t_7	0.9	0.9	0.7	0.9	0.9	0.9	1	0.87

Example 2b. Let us use the data from Table 2 and compute the degree $out_2(t_1, t_2)$. The concordant criteria are category and color; the indifferent ones are make and mileage; the only discordant one is price. We get:

$conc(t_1, t_2) = w_{category} + w_{color} = 0.4$,
$ind(t_1, t_2) = w_{make} + w_{mileage} = 0.4$,
$disc(t_1, t_2) = w_{price} = 0.2$, hence:
$out_2(t_1, t_2) = 0.4 + 0.4 = 0.8$.

Table 6 gives the concordance degree obtained for every pair of tuples (t, t') from relation *car*. Table 7 — which can be straightforwardly computed from Table 6 thanks to Equation 3 — gives the outranking degree of $t \succeq t'$ for every pair of tuples (t, t') from

relation *car*. Table 7 includes an extra column μ_1 whose meaning is given hereafter.◇

This time, the global evaluation of each tuple may be based on the following process:

1. for every tuple t, one computes the degree:

$$\mu_1(t) = \frac{\Sigma_{t' \in r \setminus \{t\}} out_2(t, t')}{|r| - 1}$$

where $|r|$ denotes the cardinality of r. Degree $\mu_1(t)$ expresses the extent to which t is better to (or as good as) most of the other tuples from r (where the fuzzy quantifier *most* is assumed to be defined as $\mu_{most}(x) = x$, $\forall x \in [0, 1]$). These degrees appear in the last column of Table 7.

2. one ranks the tuples in increasing order of $\mu_1(t)$.

The data from Table 1 leads to:

$$0.87/t_7 > 0.83/t_3 > 0.8/t_4 > 0.77/\{t_1, t_5\} > 0.53/t_6 > 0.47/t_2.$$

It is interesting to notice that $\mu_1(t)$ also captures the extent to which t is not worse than most of the other tuples. Indeed, let us consider

$$\mu_2(t) = \frac{\Sigma_{t' \in r \setminus \{t\}} conc(t', t)}{|r| - 1}$$

Degree $\mu_2(t)$ expresses the extent to which t is worse than most of the other tuples from r. Due to Equation 3, one has: $\forall t$, $\mu_1(t) = 1 - \mu_2(t)$. Thus, ranking the tuples according to μ_1 or to $1 - \mu_2$ leads to the same ordering.

3.3.2. *Fuzzy version*

As before, the transition between concordance and indifference on the one hand, indifference and discordance on the other hand, are made gradual. c_j, ind_j and d_j are defined as in Subsection 3.2.2 but outranking degrees are now computed using out_2 (cf. Formula 3).

Theorems 5a-9a straightforwardly hold since they do not depend on the definition of outranking.

Theorem 10b. Theorem 3b still holds.□

Table 8. Concordance degrees (Example 3b).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_1	0	0.2	0.1	0.1	0.15	0.3	0.1
t_2	0.25	0	0.2	0.2	0.05	0.2	0.05
t_3	0.4	0.55	0	0.2	0.2	0.45	0.25
t_4	0.2	0.35	0	0	0.1	0.35	0.05
t_5	0.2	0.35	0.05	0.05	0	0.05	0.25
t_6	0	0.35	0.05	0.05	0.05	0	0.6
t_7	0.25	0.6	0.2	0.2	0.25	0.1	0

Example 3b. Let us compute the degree $out_2(t_1, t_2)$. We have:

$c_{color}(t_1, t_2) = 0.5$, $ind_{color}(t_1, t_2) = 0.5$,
$c_{category}(t_1, t_2) = 0.5$, $ind_{category}(t_1, t_2) = 0.5$,
$d_{price}(t_1, t_2) = 0.25$, $ind_{price}(t_1, t_2) = 0.75$,
$ind_{make}(t_1, t_2) = 1$,
$ind_{mileage}(t_1, t_2) = 1$.

We get:

$conc(t_1, t_2) = 0.1 \times 0.5 + 0.3 \times 0.5 = 0.2$
$ind(t_1, t_2) = 0.75$,
$disc(t_1, t_2) = 0.2 \times 0.25 = 0.05$.

Finally: $out_2(t_1, t_2) = 0.2 + 0.75 = 0.95$.

Using the individual scores from Table 2, we get the concordance degrees represented in Table 8. Table 9 gives the outranking degree of $t \succeq t'$ for every pair of tuples (t, t') from relation *car*.

Table 9. Outranking degrees (Example 3b).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	μ_1
t_1	1	0.95	0.6	0.8	0.8	1	0.75	0.82
t_2	1	1	0.45	0.65	0.65	0.65	0.4	0.63
t_3	0.9	0.8	1	1	0.95	0.95	0.8	0.9
t_4	0.9	0.8	0.8	1	0.95	0.95	0.8	0.87
t_5	0.85	0.95	0.8	0.9	1	0.6	0.95	0.84
t_6	0.7	0.8	0.55	0.65	0.6	1	0.9	0.7
t_7	0.9	0.95	0.75	0.95	0.95	0.4	1	0.82

The ranking method based on μ_1 yields:

$$0.9/t_3 > 0.87/t_4 > 0.84/t_5 > 0.78/\{t_1, t_7\} > 0.7/t_6 > 0.6/t_2.\diamond$$

3.4. Relation with Pareto order

Let us now examine the type of relationship that exists between the outranking-based approach(es) described in the previous subsections and the preference query model based on Pareto order.

Let t and t' be two tuples such that t is better than t' in the sense of Pareto order (denoted by $t >_P t'$). It is straightforward to prove that in the case where $\forall j, q_j = 0$ (case of usual Skyline or Preference SQL queries), one has

$t >_P t'$
$\Rightarrow conc(t,t') > 0$ and $disc(t,t') = 0$.
$\Rightarrow \top_{Lu}(conc(t,t'), 1 - disc(t,t')) = conc(t,t') > 0$
$\Rightarrow out_1(t,t') > 0$ and $out_1(t',t) = 0$ (Theorem 3a).

Besides:

$t >_P t' \Rightarrow conc(t,t') > 0$ and $disc(t,t') = 0$.
$conc(t,t') > 0 \Leftrightarrow disc(t',t) > 0$ (Theorem 2a)
$disc(t,t') = 0 \Rightarrow out_2(t,t') = 1$ and
$disc(t',t) > 0 \Rightarrow out_2(t',t) < 1$.

Hence:

$$t >_P t' \Rightarrow out_2(t,t') > out_2(t',t).$$

Moreover, we have:

$$t >_P t' \Rightarrow (\forall t'', out(t,t'') \geqslant out(t',t'') \text{ and } out(t'',t) \leqslant out(t'',t'))$$

where out is either out_1 or out_2. Notice that this latter result is valid even when the indifference threshold q_j's are nonzero. Indeed, $t >_P t'$ means that for every preference G_j, one has $g_j(t) \geqslant g_j(t')$, which implies:

$$\forall t'', conc(t,t'') \geqslant conc(t',t'') \text{ and } disc(t,t'') \leqslant disc(t',t'').$$

Thus,

$$\forall t'', out(t,t'') \geqslant out(t',t'').$$

This result guarantees that t will be ranked before t' in the final result, whatever the preference model used (strict or broad).

3.5. On the originality of the approach

The aim of this subsection is to point out what, in the approach we propose, comes from (Roy, 1991)[21] and what is original. First, let us recall that the outranking approach described in (Roy, 1991)[21] is situated in a decision-making context, not a database querying one. The author considers a set of potential actions, a decision-maker who assesses each potential action according to a given set of criteria, and the objective is to determine which are the "best" actions. In our approach, a potential action corresponds to a tuple, and a criterion corresponds to a preference concerning a certain attribute.

As mentioned before, the notions of indifference, concordance and discordance are defined in (Roy, 1991)[21], but we revisited their definition so as to make them more intuitive and more simple:

- we made indifference a symmetrical concept, whereas it is defined in (Roy, 1991)[21] as:

$$a I_j a' \Leftrightarrow g_j(a) - g_j(a') \leqslant q_j;$$

- in our approach, the notions of indifference and strict preference are based on the same threshold q_j; on the other hand, (Roy, 1991)[21] uses a threshold q_j to define indifference and another one p_j ($> q_j$) to define strict preference:
  - $a I_j a' \Leftrightarrow g_j(a) - g_j(a') \leqslant q_j$,
  - $a P_j a' \Leftrightarrow g_j(a) - g_j(a') > p_j$.

  The author also defines an extra relation S_j the following way: $a S_j a'$ holds iff a is at least as good as a' w.r.t. G_j, i.e., iff $g_j(a) - g_j(a') \geqslant q_j$. Notice that this corresponds rather to our definition of strict preference, if one replaces $\geqslant$ by $>$.
- (Roy, 1991)[21] considers that a criterion G_j is:

- concordant with the statement "t outranks t'", denoted by $t\,S\,t'$, iff $t\,S_j\,t'$, i.e., iff:

$$g_j(t.A_j) - g_j(t'.A_j) \geqslant q_j,$$

- discordant with $t\,S\,t'$ iff $t'\,P_j\,t$, i.e., iff:

$$g_j(t'.A_j) - g_j(t.A_j) > p_j,$$

- neither one nor the other iff:

$$g_j(t'.A_j) - p_j \leqslant g_j(t.A_j) < g_j(t'.A_j) - q_j.$$

We rather chose to have a strict symmetry between the concepts of concordance and discordance, and we introduced the additional notion of an indifferent criterion (which is absent from (Roy, 1991)[21] where indifference is only considered a binary relation between values). Notice that our notion of an indifferent criterion is not equivalent to that of a criterion which is "neither concordant nor discordant" from (Roy, 1991)[21]. Notice also that our partitioning of the criteria into the three classes {concordant, indifferent, discordant} is what makes it possible to define both a strict preference model and a broad preference one.

- (Roy, 1991)[21] also introduces a so-called "veto threshold" v_j defined as follows: $g_j(a') - g_j(a) > v_j$ is incompatible with the assertion $a\,S\,a'$ whatever the scores related to the other criteria are. Again, our aim was to propose a simple and intuitive model, and we decided to keep the veto aspect for a future extension.
- (Roy, 1991)[21] computes a so-called credibility degree $\sigma(t, t')$ attached to the assertion "t outranks t'": it is the result of the aggregation of a concordance index $c(t, t')$ and a discordance index $d(t, t')$. Since concordance and discordance are defined differently in our approach and in (Roy, 1991)[21], the formulas are necessarily different too, but the general principle is similar in both cases and consists in computing a sum (or a weighted sum) of importance degrees.

Above all, the major originality of our approach with respect to (Roy, 1991)[21] lies in the mechanism we propose to order the tuples on the basis of their global "quality".

4. About query expression and processing

In this section, we first give some elements concerning the syntax of queries in the outranking-based model. Then, we tackle implementation issues.

4.1. Syntactical aspects

Let us consider the SQL language as a framework. We introduce a new clause aimed at expressing preferences, which will be identified by the keyword *preferring* as in the *Preference SQL* approach. This clause can come as a complement to a *where* clause, and then only the tuples which satisfy the condition from the *where* clause are concerned by the preference clause.

The preference clause specifies a list of preferences, and each element of the list includes:

- the name of the attribute concerned,
- an ordered scale or the definition of a scoring function,
- the optional weight associated with the preference,
- the optional thresholds q and δ.

We assume that scoring functions take their values in [0, 1]. A simple way to define them is to specify their core (ideal values) and support (acceptable values) and to assume that the functions are trapezoidal:

- *attribute* $\leqslant \alpha$: ideal: $\leqslant \alpha$, acceptable: $< \alpha + \beta$,
- *attribute* $\approx \alpha$:
  ideal: $\in [\alpha - \beta, \alpha + \beta]$,
  acceptable: $\in]\alpha - \beta - \lambda, \alpha + \beta + \lambda[$
- *attribute* $\geqslant \alpha$: ideal: $\geqslant \alpha$, acceptable: $> \alpha - \beta$.

When scoring functions concern categorical attributes (case where the user wants to avoid the "distance freezing" phenomenon induced by an ordinal scale, cf. Subsection 3.1.1), they have to be given in extension, as in: {1/{VW, Audi}, 0.8/{BMW}, 0.5/{Seat, Opel}, 0.3/{Ford}}.

As to the weights, their sum must be equal to 1, and if none is given by the user, each weight is automatically set to $1/m$ where m is the number of preferences in the list. In order to make the system more user-friendly, one can also think of letting the user specify the weights by means of a linguistic scale such as {*very important*, *rather important*, *medium*, *not very important*, *rather unimportant*}, assuming that the system automatically translates these linguistic terms into numerical weights (for instance 0.1 may be associated with *rather unimportant*, 0.3 with *not very important*, 0.5 with *medium*, 0.7 with *rather important*, 1 with *very important*) and normalizes the set of weights obtained in such a way that their sum equals 1.

The optional thresholds q and δ must be consistent with the ordinal scale used or with the unit interval in the case of a scoring function. If q is not specified, its default value is zero, which means that indifference corresponds to equality. If δ is not specified, its default value is also zero, which corresponds to the "basic version" of the approach (sharp boundary between concordance and indifference, and between indifference and discordance). When both q and δ are zero, one has:

- $a\,I_j\,a' \Leftrightarrow g_j(a) = g_j(a')$
- $a\,P_j\,a' \Leftrightarrow g_j(a) > g_j(a')$

The preference concerning an attribute can be either *strict* (then one uses the keywords *strict*) or *tolerant*. If it is strict, it means that a tuple which gets the score zero for the preference concerned is discarded. If it is tolerant (as in the previous examples), even the tuples which get a zero degree on that preference are ranked. The notion of a strict preference frees the user from the tedious task of specifying an additional condition in the *where* clause.

Example 4. With the specifications above, the query from Examples 3a and 3b can be expressed as:

select * **from** car
preferring
color:
blue $>$ black $>$ red $>$ yellow $>$ green $>$ black $>$ other; w = 0.1; q = 1; $\delta = 2$;
make:
VW $>$ Audi $>$ BMW $>$ Seat $>$ Opel $>$ Ford $>$ other; w = 0.2; q = 1; $\delta = 2$;
category:
sedan $>$ roadster $>$ coupe $>$ SUV $>$ other; w = 0.3; q = 1; $\delta = 2$;
price:
ideal: $\leqslant$ 4000; acceptable: $\leqslant$ 6000; w = 0.2; q = 0.2; $\delta = 0.2$;
mileage:
ideal: $\leqslant$ 15,000; acceptable: $\leqslant$ 20,000; w = 0.2; q = 0.2; $\delta = 0.2$;$\diamond$

The following example gives a more simple case of preference query on the same relation.

Example 5. Hereafter is an example of a query with a crisp interpretation of concordance, indifference and discordance, which in addition illustrates the use of strict preferences:

select * **from** car
preferring
color: blue $>$ black $>$ red $>$ yellow;
make: VW $>$ Audi $>$ BMW $>$ Seat;
category **strict**: sedan $>$ roadster $>$ coupe $>$ SUV;
price **strict**: ideal: $\leqslant$ 4000; acceptable: $\leqslant$ 6000;
mileage: ideal: $\leqslant$ 15,000; acceptable: $\leqslant$ 20,000$\diamond$

4.2. About query evaluation

Let us denote by n the cardinality of the relation concerned. The data complexity of a preference query based on outranking is obviously in $\theta(n^2)$ since all the tuples have to be compared pairwise. This is also the case for Pareto-based preference queries. Notice that on the other hand, fuzzy queries have a linear data complexity (but let us recall that they can be used only when the preferences are commensurable). Even though outranking-based preference queries are significantly more expensive than regular selection queries (n^2 instead of n), they remain perfectly tractable (they belong to the same complexity class as self-join queries in the absence of any index).

Notice that when the result of the SQL query on

which the preferences apply is small enough to fit in main memory, the extra cost will be relatively limited (the data complexity is linear in this case).

The approach has been implemented in a prototype named DISCORD (Database Interrogation System Computing OutRanking Degrees), using PostgreSQL 8.3, Apache 2.2, PHP 5.3 and C. This last language is used to speed up the comparison process through the use of compiled programs. The most sensitive aspect concerns the handling of the degrees involved in the calculus (in particular, the concordance degrees). Different implementations may be envisaged:

1. intermediary results (in particular, the table storing the concordance degrees) are stored in main memory. For n tuples, this implies using $4 \cdot n^2$ bytes, using a 32 bit representation for the type `float`. In practice, only relatively small relations (containing about 25,000 tuples with 3Gb memory available, for instance) can be handled. On the other hand, this implementation is rather fast (less than 10 seconds for a relation containing 10,000 tuples);

2. no intermediary results are stored. Then, the size of the memory is not a problem anymore, but the temporal performances dramatically decrease (several hours for a relation containing 10,000 tuples);

3. parallel calculus, which exploits the fact that pairwise comparisons are independent. Indeed, due to the fact that the outranking relation is not transitive, one can compute outranking degrees in parallel and then merge the results to obtain the global evaluation of each tuple.

So far, the prototype has been implemented using the first strategy, but we intend to apply the third one in a new version.

In order to make the query evaluation process more efficient, another solution could be to use a classification process, as proposed in (Bosc *et al.*, 2009)[4]. Instead of being compared pairwise like in Pareto-order-based approaches, tuples are compared to acceptability profiles that are associated with predefined classes. According to their satisfaction of user preferences, tuples are then assigned to classes with a certain degree. This approach relies on a classification algorithm with a linear complexity, which leads to a data complexity of the preference queries which is linear too. The pairwise comparison can then be performed on the tuples grouped in the class that is associated with the most selective acceptability profile.

5. Conclusion

In this paper, we have proposed an alternative to the use of Pareto order for the modeling of preference queries in the case where preferences on different attributes are not commensurable. The approach we defined is based on the concept of outranking, which was initially introduced in a decision-making context (but its definition was revisited here so as to fit our purpose). Outranking makes it possible to compare tuples pairwise, and even though it does not define an order (it is not transitive), we showed how a complete-preorder could be obtained by aggregating the outranking degrees in such a way that the aggregate characterizes the global "quality" of a tuple (regarding a given set of preferences) with respect to the others. As in the case of Pareto-order-based approaches, the data complexity of queries is quadratic, thus tractable even though signifcantly higher than that of regular selection queries.

As a perspective for future research, we intend to investigate a variant of the outranking-based preference model where the tuples, instead of being ordered by means of the aforementioned aggregation function, are clustered into predefined classes — in the spirit of (Mousseau *et al.*, 2000)[19]. Such a method leads to a linear data complexity (each tuple is compared to a set of profiles, but tuples are not compared pairwise anymore) at the price of a reduced discrimination power.

References

1. R. Agrawal and E. Wimmers. A framework for expressing and combining preferences. In *Proc. of SIGMOD 2000*, pages 297–306, 2000.
2. S. Bőrzsőnyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. of the 17th IEEE Inter. Conf. on Data Engineering*, pages 421–430, 2001.
3. P. Bosc and O. Pivert. SQLf: A relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, 3(1):1–17, 1995.
4. P. Bosc, O. Pivert, and G. Smits. A flexible querying approach based on outranking and classification. In *Proc. of the 8th International Conference on Flexible Query Answering Systems (FQAS'09)*, pages 1–12, Roskilde, Denmark, 2009.
5. P. Bosc, O. Pivert, and G. Smits. A database preference query model based on a fuzzy outranking relation. In *Proc. of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'10)*, pages 38–43, Barcelona, Spain, 2010.
6. P. Bosc, O. Pivert, and G. Smits. A model based on outranking for database preference queries. In *Proc. of the 13th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'10), CCIS vol. 81*, pages 95–104, Dortmund, Germany, 2010. Springer.
7. C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.
8. R. Brafman and C. Domshlak. Database preference queries revisited. Technical report, TR2004-1934, Cornell University, Computing and Information Science, 2004.
9. N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: mapping strategies and performance evaluation. *ACM Transactions on Database Systems*, 27(2):153–187, 2002.
10. C. Chan, H. Jagadish, K. Tan, A. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *Proc. of SIGMOD 2006*, pages 503–514, 2006.
11. C. Chan, H. Jagadish, K. Tan, A. Tung, and Z. Zhang. On high dimensional skylines. In *Proc. of EDBT 2006, LNCS 3896*, pages 478–495, 2006.
12. J. Chomicki. Preference formulas in relational queries. *ACM Transactions on Database Systems*, 28(4):427–466, 2003.
13. D. Dubois and H. Prade. Using fuzzy sets in flexible querying: Why and how? In *Proc. of the 1996 Workshop on Flexible Query-Answering Systems (FQAS'96)*, pages 89–103, 1996.
14. R. Fagin. Fuzzy queries in multimedia database systems. In *Proc. of PODS 1998*, pages 1–10, 1998.
15. A. Hadjali, S. Kaci, and H. Prade. Database preference queries — A possibilistic logic approach with symbolic priorities. In *Proc. of the 5th Symposium on the Foundations of Information and Knowledge Systems (FoIKS'08)*, pages 291–310, 2008.
16. W. Kießling. Foundations of preferences in database systems. In *Proc. of the 28th VLDB Conference (VLDB'02)*, pages 311–322, 2002.
17. W. Kießling and G. Köstler. Preference SQL — Design, implementation, experiences. In *Proc. of the 28th Conference on Very Large Data Bases (VLDB'02)*, pages 990–1001, 2002.
18. X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: the k most representative skyline operator. In *ICDE 2007*, pages 86–95, 2007.
19. V. Mousseau, R. Slowinski, and P. Zielniewicz. A user-oriented implementation of the ELECTRE-TRI method integrating preference elicitation support. *Computers and Operations Research*, 27:757–777, 2000.
20. P. Perny and B. Roy. The use of fuzzy outranking relations in preference modelling. *Fuzzy Sets and Systems*, 49(1):33–53, 1992.
21. B. Roy. The outranking approach and the foundations of ELECTRE methods. *Theory and Decision*, 31:49–73, 1991.
22. R. Torlone and P. Ciaccia. Finding the best when it's a matter of preference. In *Proc. of the 10th Italian National Conf. on Advanced Data Base Systems (SEBD 2002)*, pages 347–360, 2002.

Appendix A

Proof of Theorem 1a. Straightforward since the sum of the weights equals 1 and $conc(t,t')$, $ind(t,t')$ and $disc(t,t')$ form a partition of r.

Proof of Theorem 2a. One has: $C(t,t') = D(t',t)$ by definition of concordant and discordant criteria. Thus, the definitions of $conc(t,t')$ and $disc(t,t')$ straightforwardly imply Theorem 2.

Proof of Theorem 3a. Let us use the following notations: $a = conc(t,t')$ and $b = disc(t,t')$. Due to the definition of Łukasiewicz' implication, one has: $out_1(t,t') = max(0, a+1-b-1) = max(0, a-b)$. Due to Theorem 2, one has: $conc(t',t) = disc(t,t') = b$ and $disc(t',t) = conc(t,t') = a$. Thus $out_1(t',t) = max(0, b+1-a-1) = max(0, b-a)$. If $conc(t,t') \geqslant disc(t,t')$, i.e., if $a \geqslant b$, then $out_1(t,t') = a - b = conc(t,t') - disc(t,t')$

and $out_1(t', t) = 0$, otherwise $out_1(t, t') = 0$ and $out_1(t', t) = b - a = conc(t', t) - disc(t', t) = disc(t, t') - conc(t, t')$.

Proof of Theorem 4a. Straightforward from Theorem 1 and the fact that $\forall t$, $ind(t, t) = 1$.

Proof of Theorem 5a. Straightforward from the definitions of $\mu_{conc_j(a)}$, $\mu_{ind_j(a)}$ and $\mu_{disc_j(a)}$ (see Figure 1), and those of c_j, ind_j and d_j. Notice in particular that the intersection of $\mu_{conc_j(a)}$ and $\mu_{ind_j(a)}$ is located at level 0.5, as well as the intersection of $\mu_{ind_j(a)}$ and $\mu_{disc_j(a)}$.

Proof of Theorem 6a. Follows straightforwardly from the definitions of $\mu_{conc_j(a)}$ and $\mu_{disc_j(a)}$ (see Figure 1).

Proof of Theorem 7a. $c_j(t, t') = \mu_{conc_j(t.A_j)}(g_j(t'.A_j))$ = 1 if $g_j(t'.A_j) \leqslant g_j(t.A_j) - q_j - \delta_j$, 0 if $g_j(t'.A_j) \geqslant g_j(t.A_j) - q_j$, linear in-between.
$d_j(t', t) = \mu_{disc_j(t'.A_j)}(g_j(t.A_j))$ = 1 if $g_j(t.A_j) \geqslant g_j(t'.A_j) + q_j + \delta_j$, 0 if $g_j(t.A_j) \leqslant g_j(t'.A_j) + q_j$, linear in-between. Let us use the notations $g_j(t.A_j) = a$, $g_j(t'.A_j) = a'$. One has:

- $a' \leqslant a - q_j - \delta_j \Leftrightarrow a \geqslant g_j(t'.A_j) + q_j + \delta_j$ and in this case $c_j(t, t') = 1 = d_j(t', t)$
- $a' \geqslant a - q_j \Leftrightarrow a \leqslant a' + q_j$ and in this case $c_j(t, t') = 0 = d_j(t', t)$
- $a - q_j - \delta_j \leqslant a' \leqslant a - q_j \Leftrightarrow a' + q_j \leqslant a \leqslant a' + q_j + \delta_j$ and in this case $c_j(t, t') = (a - a' - q_j)/\delta_j = d_j(t', t)$.

Proof of Theorem 8a. $\forall(t, t')$, $conc(t, t') + ind(t, t') + disc(t, t') = \sum_{i=1..n} w_i \times c_i(t, t') + \sum_{i=1..n} w_i \times ind_i(t, t') + \sum_{i=1..n} w_i \times d_i(t, t') = \sum_{i=1..n} w_i \times (c_i(t, t') + ind_i(t, t') + d_i(t, t')) = \sum_{i=1..n} w_i$ (due to Theorem 5) = 1 (since the sum of the weights equals 1).

Proof of Theorem 9a. $conc(t, t') = \sum_{i=1..n} w_i \times c_i(t, t') = \sum_{i=1..n} w_i \times d_i(t', t)$ (due to Theorem 7) = $disc(t', t)$.

Proof of Theorem 10a. Straightforwardly follows from the definition of Łukasiewicz' t-norm and the arguments used (cf. the proof of Theorem 3, which remains valid due to Theorem 9).

Proof of Theorem 3b. $\forall(t, t')$, $out_2(t, t') + out_2(t', t) = conc(t, t') + ind(t, t') + conc(t', t) + ind(t', t) = conc(t, t') + ind(t, t') + disc(t, t') + ind(t', t)$ (cf. Theorem 2a) = 1 + $ind(t', t)$ (cf. Theorem 1a) $\geqslant 1$ (since $ind(t', t) = ind(t, t') \geqslant 0$).

Proof of Theorem 10b. The proof of Theorem 3b is also valid in the fuzzy case since Theorems 1a and 2a hold in this case too.