SEMANTICS AND PRAGMATICS OF PREFERENCE QUERIES IN DIGITAL LIBRARIES

SPYRATOS N / CHRISTOPHIDES V / GEORGIADIS P / NGUER M

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

11/2007

Rapport de Recherche N° 1478

Semantics and Pragmatics of Preference Queries in Digital Libraries

N. Spyratos[1], V. Christophides[2], P. Georgiadis[2], M. Nguer[1]

[1] LRI, Université Paris Sud, {spyratos, nguer}@lri.fr
[2] ICS-FORTH, {christop, perge}@ics.forth.gr

Abstract. As information becomes available in increasing amounts, and to growing numbers of users, the shift towards a more user-centered, or personalized access to information becomes crucial. In this paper we consider the semantics and pragmatics of preference queries over tables containing information objects described through a set of attributes. In particular, we address two basic issues:

– how to define a preference query and its answer (semantics)
– how to evaluate a preference query (pragmatics)

With respect to existing work, our main contributions are (a) the proposal of an expressive language for declaring qualitative preferences, (b) a unified framework for expressing and evaluating both quantitative and qualitative preference queries and (c) rewriting algorithms for processing such queries. Although our main motivation originates in digital libraries, our proposal is quite general and can be used in several application contexts.

1 Introduction

As information becomes available in increasing amounts, and to growing numbers of users, the shift towards a more user-centered, or personalized access to information becomes crucial. Personalized access can involve customization of the user interface or adaptation of the content to meet user preferences. This paper addresses the latter issue, and more precisely adaptation of the answer returned by a query to user preferences.

We call *preference query*, a standard query together with a set of user preferences. Such queries are useful in several application contexts where users browsing extremely large data collections don't have a clear view of the information objects that these collections contain nor do they have a particular object in mind. Rather, they are attempting to discover objects that are potentially useful to them, or in other words, objects that suit their preferences best. The main objective of this paper is to introduce a unified formal framework for specifying and evaluating various kinds of preference queries. Although our motivation originates in digital libraries [49], the results presented in this paper apply to other application contexts as well (e.g. searching for gifts in the electronic catalogues of online merchants).

We view a digital library catalogue as a table describing digital documents, as

shown in the example of Figure 1 - that we shall use as our running example. In that catalogue, each document is considered as an information object, described by an identifier, denoted Oid (e.g. the document's URI), and a number of attributes: its year of publication, the (first) author's name, the subject category treated by the document (e.g. Poetry, Fiction, etc.), the language in which the document is written and the electronic format in which the document is available through the library (such as Word, Pdf, and so on). In other words, we view the catalogue just like a table of a relational database, whose schema is C(Oid, Year, Author, Category, Language, Format), and in which each column is associated with a set of values (i.e. a domain). For simplicity, in Figure 1, we denote the document identifiers by integers.
A standard query expressed by a user against the catalogue is a Boolean combination of elementary conditions of the form A=a, where 'A' is an attribute and 'a' is a value in the domain of A. For example, consider the following query:
$\mathsf{Q_1} = [(\mathsf{Category} = \mathsf{Poetry}) \vee (\mathsf{Category} = \mathsf{Fiction})] \wedge (\mathsf{Language} = \mathsf{English})$
To answer this query we must compute the set of documents having Poetry or Fiction as their Category attribute, then the set of documents having English as their Language attribute, and finally take the intersection of these two sets:
$\mathsf{ans}(\mathsf{Q_1}) = (\{1,3,5\} \cup \{2,4,6\}) \cap \{2,3,5,6,8\} = \{2,3,5,6\}$
As the size of the answer set is not known in advance, and as it might be too large to exploit by a casual user, it would be interesting to present the set of documents returned in a decreasing order with respect to user preferences. The user can then inspect the most interesting documents first, and stop inspection when the documents become less and less interesting. However, in order to produce such an ordering of the answer set, the system must have access to user preferences, and this can be done in one of three ways: - The user declares *offline* a set of preferences, and these preferences are stored by the system. - The system elicits user preferences by monitoring and analyzing previous queries by the user. - The user declares *online* a set of preferences, together with the query. As mentioned earlier, a preference query is a standard query together with a set of user preferences. This paper is concerned with preference queries, *independently* of how user preferences are made available to the system. The important issue addressed in this paper is how user preferences influence the answer set. To see this, consider the following statement of preferences over the attribute Category:
$\mathsf{P_1} : (\mathsf{Category} : \mathsf{Poetry} \rightarrow \mathsf{Fiction})$ [meaning that poetry is preferred to fiction]
We would like the previous query Q_1, together with the statement P_1, to return a result showing the documents about poetry before documents about fiction. In other words, we would like the answer to be presented as follows:
$\mathsf{ans}(\mathsf{Q_1}, \mathsf{P_1}) = \{3,5\} \rightarrow \{2,6\}$
It is important to note that the answer set of $\mathsf{Q_1}$, processed alone, and the answer set of $\mathsf{Q_1}$ processed together with the statement $\mathsf{P_1}$ contain the *same* set of documents. The difference lies in the fact that, in presence of $\mathsf{P_1}$, the answer set of $\mathsf{Q_1}$ is partitioned into two subsets *ordered* so that the first subset contains documents about poetry and the second about fiction.
In the previous example a preference was expressed in the form of a pair of

attribute values (namely, Poetry and Fiction) with the understanding that the first value in the pair is preferred to the second. Expressing preferences in the form of pairs of attribute values is referred to in the literature as the *qualitative approach* [7, 15–19, 31, 35–37]. However, there is also a more widely spread approach, whereby preferences are expressed by associating each attribute value of interest with a numerical value or *score*, and documents described by attribute values with higher scores are presented first in the answer set. This approach is referred to as the *quantitative approach* [1, 4, 6, 11, 13, 20–23, 38, 39]. Traditionally the quantitative and the qualitative approach to expressing preferences have been treated with quite different machinery in the literature, and little or no attention has been devoted to their unification.
The main contributions of this paper are (a) the proposal of an expressive language for declaring qualitative preferences, (b) a unified framework for expressing and evaluating both quantitative and qualitative preference queries and (c) rewriting algorithms for processing such queries.

Oid	**Author**	**Year**	**Category**	**Language**	**Format**
1	A1	2001	Poetry	French	Word
2	A1	1998	Fiction	English	Pdf
3	A2	2000	Poetry	English	Pdf
4	A3	2001	Fiction	German	Pdf
5	A1	2002	Poetry	English	Word
6	A2	2000	Fiction	English	Word
7	A4	1998	Drama	German	Pdf
8	A2	2002	Comedy	English	Pdf
9	A3	2007	Comedy	French	Pdf

Table 1. A Digital Library Catalogue

We would like to emphasize here that there is an important difference between the preference queries studied in this paper and the Order-by queries of SQL. Indeed, using Order-by one can ask the system to return the results of a query in an ascending or descending order, following the *predefined* order of some attribute domain (e.g. the domain of Year in our running example). In the preference queries considered here, it is the user that *inputs* an order for the attribute domain - an order expressing the user's preferences (and, in fact, the order input by the user might contradict the predefined order of the attribute domain). Additionally, some attribute domains have no predefined order (e.g. Category or Format in our running example) so Order-by simply doesn't apply to such attributes, whereas preference queries of our approach apply to *any* attribute.

2 Expressing Preferences

We have just seen that there exist two approaches for expressing preferences, namely the quantitative approach and the qualitative approach. In this section we detail these two approaches, and we introduce a language for expressing qualitative preferences, strictly more expressive than existing languages.
In the quantitative approach each object of interest from a set X is examined in isolation (i.e. independently of other objects) and a numerical value is associated with it; this numerical value is called the *score* of the object, and it is usually from the closed interval $[0, 1]$. The association of objects with scores creates a function, called a *scoring function*. A scoring function can either be provided by the user or computed by the system (based on previous user queries), and might be a partial function if not all objects of X are associated with a score.
Anyhow, the score of an object implies both, a rank for the object and an intensity by which the object is desired. For example, referring to Figure 1, consider the following scoring functions on the domain of Category:
$\mathsf{Scoring\ function\ S_1 : Poetry : 0.90, Fiction : 0.89}$
$\mathsf{Scoring\ function\ S_2 : Poetry : 0.90, Fiction : 0.01}$
Both scoring functions imply the same ranking (Poetry ranked higher than Fiction), but the intensities are quite different. According to S_1, Fiction is a very close second to Poetry, whereas according to S_2 Fiction is a very distant second to Poetry.
The use of scoring functions has a long history in the area of economics, social choice theory and decision support systems, and a large body of literature has developed over the years [2, 3, 30, 32, 40, 42]. Completeness and transitivity which have been essential preconditions to these fields were too strict, thus various attempts were made in the last thirty years to relax them, mostly through the proposal of specific order structures [5, 9, 10, 14, 24–29, 41, 43–47]. A brief overview of basic problem setup and various approaches to it from an operational research viewpoint can be found in [8]. Various metrics attempting to quantify the fairness of a resultant ranking against its constituent ones have also been proposed [20–22, 33, 34].
In the qualitative approach, attribute values are examined in pairs and if an attribute value x is preferred to an attribute value y then the pair (x, y) is declared by the user; such a pair is called a *preference*, and the set of all user preferences is called the user's *preference relation* [17, 37].
Clearly the user should be free to express any desired preference; hence the preference relation could be, in principle, *any* set of pairs.
However, in the literature, the preference relation is usually modeled as a strict partial order [35–37, 31] that is a binary relation "<" satisfying the following properties:
$x \not< x$ for all x in X (non reflexivity)
$x < y$ implies $y \not< x$ for all x, y in X such that $x \neq y$ (asymmetry)
$x < y$ and $y < z$ implies $x < z$ (transitivity)
We consider this modeling choice as unnecessarily restrictive. In particular, we consider that transitivity should *not* be imposed as a constraint for the prefer-

ence relation to be acceptable.
Indeed, a preference represents a decision made by the user when comparing two objects x and y *in isolation* that is independently of other objects. As a consequence, if the user has expressed only two preferences, say (x, y) and (y, z), it is unrealistic to *infer* a third preference (x, z) that the user has *not* expressed. For example, if the user has expressed preference of Poetry over Drama and of Drama over Comedy, there is no reason to believe that the user will prefer Poetry to Comedy if confronted with the latter two in isolation. Of course, it is conceivable that transitivity might be reasonable to assume in some applications, however, it is certainly not realistic to assume transitivity in *every* application.
In our approach, *we impose no constraint whatsoever on the preference relation.* In other words, the user can declare *any* set of pairs of attribute values as a preference relation. In addition, we allow the user to declare that two attribute values are equally preferred. Formally, we treat the declaration "x and y are equally preferred" as a declation of two pairs, namely (x, y) and (y, x), that is as a cycle between x and y. In reality, while declaring preferences, a user may create cycles consciously or unconsciously. The presence of a cycle in the preference relation might either mean that all values in the cycle are equally preferred or that the user has inadvertently created a cycle.
In the prototype that we are currently developping, if the system detects cycles in the preference relation, these cycles are handled in one of two ways, depending on the application context:

– *Dialogue with the user:* Each cycle is presented to the user, and the user is asked to either confirm the cycle or "break" it (by modifying the declared preferences).

– *Automatic Processing:* The system processes the cycles without help from the user, by considering all objects on a cycle as being equivalent [3] .

In any case, the resulting preference relation is an acyclic relation. To simplify matters, in the remaining of the paper, we assume that the preference relation is indeed acyclic; and in order to differentiate from the preference relations used in the literature, we shall call it a *precedence relation.*

Definition 1 (Precedence Relation).
Given a set X *of objects, a precedence relation on* X *is an acyclic binary relation on* X.

We shall denote a precedence relation by $\rightarrow$, read as "precedes". For example, in the table of Figure 1, assuming that X is the domain of the attribute Category,

[3] More formally, cycles can be removed from a cyclic relation P if one defines an equivalence relation as follows: (a) $x \equiv x$, for all objects appearing in P and (b) $x \equiv y$, if x and y are on the same cycle. Then instead of P one works with the quotient relation $\mathsf{P}/\equiv$.

the following is a precedence relation on X:
$\mathsf{Poetry} \rightarrow \mathsf{Fiction}, \mathsf{Drama} \rightarrow \mathsf{Fiction}, \mathsf{Fiction} \rightarrow \mathsf{Comedy}$
These declarations are interpreted as follows: Poetry precedes Fiction, Drama precedes Fiction and Fiction precedes Comedy.
One important property of precedence relations is that they are *strictly* more expressive than strict partial orders. Indeed, each strict partial order on a set X is a precedence relation on X (because a strict partial order is always acyclic); in the opposite direction, a precedence relation on X is non reflexive and asymmetric (because of acyclicity), but not necessarily transitive. It follows that, given a set X, the set of all strict partial orders on X is strictly included in the set of all precedence relations on X. Actually, the relationship between precedence relations and strict partial orders is more intricate, as stated in the following proposition.

Proposition 1 (A Basic Property of Precedence Relations).
If P *is a precedence relation on* X *then the following relation "*$<_{\mathsf{P}}$*" is a strict partial order on* X*:* $x <_P y$ *if there is a path from* x *to* y *in* P*, for all* x*,* y *in* X*. Conversely, if "*$<$*" is a strict partial order on* X *then its transitive reduction* [4] *is a precedence relation on* X*.*

Clearly the relation "$<_{\mathsf{P}}$" is the transitive closure of P. We shall refer to "$<_{\mathsf{P}}$" as the strict partial order induced by P (and we shall usually omit the subscript P, whenever no confusion is possible).
Summarizng our discussion in this section, languages for expressing preferences based on precedence relations are more expressive than those based on strict partial orders.

3 Preference Queries

As mentioned in the introduction, we consider preference queries over tables of the form $\mathsf{C}(\mathsf{Oid}, \mathsf{A}_1, \cdots, \mathsf{A}_\mathsf{n})$, where Oid denotes the identifier of an information object and $\mathsf{A}_1, \cdots, \mathsf{A}_\mathsf{n}$ denote attributes of that object. Henceforth, we shall refer to a table of this form as a *catalogue*. For the purposes of this paper, we follow [49] and consider each attribute A_i as a function from the domain of Oid to the domain of attribute A_i, that is, $\mathsf{A}_\mathsf{i} : \mathsf{dom}(\mathsf{Oid}) \rightarrow \mathsf{dom}(\mathsf{A}_\mathsf{i})$. For example, in Figure 1, we have $\mathsf{Category}(1) = \mathsf{Poetry}, \mathsf{Format}(2) = \mathsf{Pdf}$, and so on. Under this view, we define a query over the table C as follows:

Definition 2 (Query Over a Table C). *Call elementary condition over a catalogue* C *any expression of the form* $\mathsf{A} = \mathsf{a}$*, where* A *is an attribute and* a *is in the domain of* A*. A query* Q *over* C *is defined as follows, where* Q_1*,* Q_2 *are queries:*

$$\mathsf{Q} ::= \mathsf{A} = \mathsf{a} | \mathsf{Q1} \wedge \mathsf{Q2} | \mathsf{Q1} \vee \mathsf{Q2} | \mathsf{Q1} \wedge \neg \mathsf{Q2} | (\mathsf{Q})$$

[4] A strict partial order is an acyclic relation, and the transitive reduction of an acyclic relation is unique [50]. Uniqueness of the transitive reduction is not guaranteed, in general, for cyclic relations.

In other words, a query over C is either an elementary condition or a Boolean combination of elementary conditions.
The answer of Q*, denoted* $\mathsf{ans(Q)}$*, is defined in three steps as follows :*

1. *replace each elementary condition of the form* $\mathsf{A} = \mathsf{a}$ *appearing in* Q *by the inverse image of* a *under* A *(i.e. by the set of objects* $\mathsf{A}^{-1}(\mathsf{a})$*)*
2. *replace each Boolean connective by the corresponding set theoretic operation (i.e. replace* $\wedge$ *by* $\cap$*,* $\vee$ *by* $\cup$ *and* $\wedge\neg$ *by* $\setminus$*)*
3. *Perform the set theoretic operations*

As an example, consider the following query over the table C of Figure 1:
Q= ((Category= Poetry)$\vee$(Category= Fiction))$\wedge\neg$(Language= French)
Its answer is computed as follows:

$$\begin{aligned}\mathsf{ans}(Q) &= (\mathsf{Category}^{-1}(\mathsf{Poetry}) \cup \mathsf{Category}^{-1}(\mathsf{Fiction}))\setminus(\mathsf{Language}^{-1}(\mathsf{French})) \\ &= (\{1,3,5\} \cup \{2,4,6\})\setminus\{1,9\} = \{2,3,4,5,6\}\end{aligned}$$

To simplify the presentation, we shall "factor out" attributes when they are repeated. For example, the previous query Q will be written as follows:
$\mathsf{Q} = (\mathsf{Category} = \mathsf{Poetry} \vee \mathsf{Fiction}) \wedge \neg(\mathsf{Language} = \mathsf{French})$
As mentioned in the introduction, a preference query is a standard query together with a set of user preferences. However, as we saw in the previous section, user preferences can be expressed either in the form of a scoring function or in the form of a precedence relation. Hence the following definition of a preference query.

Definition 3 (Preference Query). *Let* $\mathsf{C}(\mathsf{Oid}, \mathsf{A}_1, \cdots, \mathsf{A}_n)$ *be a catalogue. A preference query over* C *has one of two forms:*
Scoring Query: $\mathsf{S_q} = (\mathsf{Q}, \mathsf{S})$ *where* Q *is a query over* C *and* S *is a set of scoring functions over attribute domains in* C
Precedence Query: $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$ *where* Q *is a query over* C *and* P *is a set of precedence relations over attribute domains in* C*.*

We note that, in a precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, it is possible to have attribute values appearing in Q but not in P, and vice versa. For example, the following query Q and precedence relation P are perfectly compatible:
$\mathsf{Q} : (\mathsf{Category} = \mathsf{Poetry} \vee \mathsf{Fiction} \vee \mathsf{Drama})$
$\mathsf{P} : (\mathsf{Category} : \mathsf{Comedy} \rightarrow \mathsf{Poetry}, \mathsf{Poetry} \rightarrow \mathsf{Fiction})$
The value Drama appears in Q but not in P whereas the value Comedy appears in P and not in Q. Nevertheless, the pair (Q, P) qualifies as a precedence query. Additionally, it is possible to have attributes appearing in the usual query Q but not in the preferences, and vice versa. For example, the following query Q and precedence relation P are perfectly compatible:
$\mathsf{Q} : (\mathsf{Category} = \mathsf{Poetry} \vee \mathsf{Fiction} \vee \mathsf{Drama})$
$\mathsf{P} : (\mathsf{Language} : \mathsf{Greek} \rightarrow \mathsf{English}, \mathsf{French} \rightarrow \mathsf{English})$
The attribute $\mathsf{Category}$ appears in Q but not in P whereas the attribute $\mathsf{Language}$ appears in P but not in Q. Nevertheless, again, the pair (Q, P) qualifies as a precedence query.

The key question now is how to define a common formal framework in which to evaluate both forms of preference query. To do this we exploit the fact that both, scoring functions and precedence relations, each induce a ranking of objects. This is rather obvious in the case of scoring functions but less so in the case of precedence relations. It is precisely this common property (i.e. the induced ranking of objects) that we use in this paper as the basis for unifying the two approaches. First, we introduce a special kind of precedence relation that we call a ranking domain.

4 Ranking Functions

Intuitively, a *ranking domain* is a set of labels that one attaches to objects in order to denote their rank with respect to other objects in a set. More formally, a ranking domain is just a special case of precedence relation, as stated in the following definition.

Definition 4 (Ranking Domain). *Given a set* X *of objects, a ranking domain on* X *is a finite precedence relation* R *satisfying the following properties:*
there is exactly one minimal element, called the first element
there is exactly one maximal element, called the last element
each element other than the first has exactly one predecessor
each element other than the last has exactly one successor
We shall refer to the elements of a ranking domain as ranks.

Since a ranking domain R is also a precedence relation, it follows that R induces a strict partial order "$<_\mathsf{R}$". It is not difficult to see that the induced strict partial order "$<_\mathsf{R}$" is complete, that is for any two ranks r, r', either $\mathsf{r} <_\mathsf{R} \mathsf{r}'$ or $\mathsf{r}' <_\mathsf{R} \mathsf{r}$. In other words, a ranking domain is a finite precedence relation, all elements of which lie on a single path. In our examples we shall use a finite set of integers as a ranking domain.
The following definition introduces the notion of ranking function, which is necessary for the evaluation of preference queries in the next section.

Definition 5 (Ranking Function). *Given a set* X *of objects, we call ranking function on* X*, or simply ranking on* X *any (possibly partial) function* $\mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Y}$*, such that* $\mathsf{range}(\mathsf{f})$ *is a ranking domain.*

We recall now two facts about functions in general, that we shall use later on for ranking functions, in particular. First a function $\mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Y}$ can be represented unambiguously by the set of pairs $\{\langle \mathsf{r}, \mathsf{f}^{-1}(\mathsf{r})\rangle / \mathsf{r} \in \mathsf{range}(\mathsf{f})\}$, or equivalently, by $\{\langle \mathsf{f}(\mathsf{x}), \mathsf{f}^{-1}(\mathsf{f}(\mathsf{x}))\rangle / \mathsf{x} \in \mathsf{def}(\mathsf{f})\}$. We call this representation the *partition representation* of f, and denote it by $\mathsf{pr}(\mathsf{f})$. For example, the partition representation of the function Category contains the following pairs:
$\langle \mathsf{Poetry}, \{1,3,5\}\rangle, \langle \mathsf{Fiction}, \{2,4,6\}\rangle, \langle \mathsf{Drama}, \{7\}\rangle, \langle \mathsf{Comedy}, \{8,9\}\rangle$
Second, a function can be restricted to a subset of its domain of definition. Let $\mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Y}$ be a function and let E be a subset of $\mathsf{def}(\mathsf{f})$. The restriction of f to E, denoted f/E is a function from E to Y defined as follows: $(\mathsf{f}/\mathsf{E})(\mathsf{e}) = \mathsf{f}(\mathsf{e})$, for

all e in E. The partition representation of $\mathsf{f/E}$ can be computed from that of f as follows:
$\mathsf{pr(f/E)} = \{\mathsf{B} \cap \mathsf{E}/\mathsf{B} \in \mathsf{pr(f)}, \mathsf{B} \cap \mathsf{E} \neq \varnothing\}$
In the rest of this paper we adopt the convention that if E is not a subset of $\mathsf{def(f)}$ then $\mathsf{f/E}$ stands for $\mathsf{f/E'}$, where $\mathsf{E'} = \mathsf{E} \cap \mathsf{def(f)}$
The evaluation of preference queries that we shall see shortly relies on the following two facts concerning ranking functions:

- Let $\mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Y}$ be a ranking function and let R be the range of f. As R is a ranking domain, it follows that the induced order $<_{\mathsf{R}}$ is complete. Therefore we can sort the partition representation of f in an ascending or in a descending order of rank.

- Let $\mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Y}$ be any function and let $\mathsf{g} : \mathsf{Y} \longrightarrow \mathsf{Z}$ be a ranking on Y. Then the composition $\mathsf{g} \circ \mathsf{f} : \mathsf{X} \longrightarrow \mathsf{Z}$ is a ranking on X.

The above two facts are the basic building blocks of our formal framework for the uniform evaluation of both types of preference queries. Indeed, given a scoring query $\mathsf{S_q} = (\mathsf{Q}, \mathsf{S})$ or a precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, the approach that we follow in the rest of the paper can be outlined as follows:

1. First we show how $\mathsf{S_q}$ and $\mathsf{P_q}$ each induces a ranking over the set of objects in the catalogue C; call these rankings $\mathsf{f_S}$ and $\mathsf{f_P}$, respectively.
2. Then we define the partition representation of the induced ranking, restricted to the answer of Q, to be the answer of $\mathsf{S_q}$ or $\mathsf{P_q}$, respectively; in other words, $\mathsf{ans(S_q)} = \mathsf{pr(f_S/ans(Q))}$ and $\mathsf{ans(P_q)} = \mathsf{pr(f_P/ans(Q))}$
   We note that as $\mathsf{f_S}$ and $\mathsf{f_P}$ are rankings, the answers just defined can be sorted and presented to the user in either ascending or descending order of rank.

5 Induced Rankings

Let $\mathsf{S_q} = (\mathsf{Q}, \mathsf{S})$ be a scoring query where S is a scoring function over a single attribute, say A. Then $\mathsf{S_q}$ induces a ranking over the set of objects of C in a straightforward manner. Indeed, recall that S is a function from the domain of A to a set of numerical values, therefore the range of S is a ranking domain.
A precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, where P is a precedece relation over a single attribute, also induces a ranking over the set of objects of C but in an indirect manner. The following definition introduces our proposal for the ranking induced by the precedence relation P.

Definition 6 (Induced Ranking). *Let* X *be any set, let* P *be a precedence relation on* X. *Let* m *be the largest length of path over all paths having as source a minimal element of* P. *For each element x appearing in* P, *define the rank of x, denoted as* $\mathsf{Rank_P}(x)$ *as follows:*
if x is a minimal element of P *then* $\mathsf{Rank_P}(x) = \mathsf{m}$
else $\mathsf{Rank_P}(x) = \mathsf{m} - \mathsf{i}$, *where* i *is the largest length of path from a minimal element of* P *to x*

Intuitively, the ranking induced by P assigns the highest rank to each minimal element, as minimal elements are the most preferred (no element precedes a minimal element in P); it assigns the lowest rank to each maximal element, as maximal elements are the least preferred (a maximal element precedes no other element in P); and it assigns a rank to an intermediate element such that the further the element it is from the minimal elements, the lower its rank. These observations are better understood in the following example.

Example 1. Consider the following precedence relation on the domain of Category:
$\mathsf{Poetry} \rightarrow \mathsf{Fiction}, \mathsf{Drama} \rightarrow \mathsf{Fiction}, \mathsf{Drama} \rightarrow \mathsf{Comedy},\ \mathsf{Fiction} \rightarrow \mathsf{Comedy}$
Applying Definition 6 we find the following induced ranking:
$\mathsf{Rank_P(Poetry)} = \mathsf{Rank_P(Drama)} = 2, \mathsf{Rank_P(Fiction)} = 1, \mathsf{Rank_P(Comedy)} = 0$

The ranking induced by P has two interesting properties, as stated in the following proposition.

Proposition 2 (Properties of the Induced Ranking). *Let* X *be any set, let* P *be a precedence relation on* X*, and let* $\mathsf{Rank_P}$ *be the ranking induced by* P *on* X*. Then the following hold:*

1. $\mathsf{x} <_\mathsf{P} \mathsf{y}$ *implies* $\mathsf{Rank_P(y)} < \mathsf{Rank_P(x)}$
2. $\mathsf{Rank_P(x)} = \mathsf{Rank_P(y)}$ *implies* $(\mathsf{x}, \mathsf{y}) \notin \mathsf{P}$ *and* $(\mathsf{y}, \mathsf{x}) \notin \mathsf{P}$ *(in this case we call* x *and* y *non comparable in* P*)*

6 Answering Preference Queries

In this section we present our approach to evaluating a preference query over a catalogue C. Recall that a preference query is a standard query together with a set of preferences either in the form of scoring functions or in the form of precedence relations.

6.1 Answering a Precedence Query

We begin with the evaluation of a precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$ and we distinguish two cases for the precedence relation P: precedence over a single attribute, and precedence over two or more attributes.

Definition 7 (Precedence Over a Single Attribute). *Let* $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$ *be a precedence query over a catalogue* C*, where* P *is a precedence over a single attribute* A*. The answer to* $\mathsf{P_q}$*, denoted* $\mathsf{ans(P_q)}$*, is defined as follows:*
$\mathsf{ans(P_q)} = \mathsf{pr}\big[(\mathsf{Rank_P} \circ \mathsf{A})/\mathsf{ans(Q)}\big]$

If we recall the definition of partition representation from the previous section, and assume that $0, \cdots, \mathsf{m}$ are the ranks produced by the ranking $\mathsf{Rank_P}$ on the domain of Ref (see Definition 6), then we have:
$\mathsf{Ans(P_q)} = \big\{\langle 0, ((\mathsf{Rank_P} \circ \mathsf{A})^{-1}(0)) \cap \mathsf{ans(Q)}\rangle, \cdots, \langle \mathsf{m}, ((\mathsf{Rank_P} \circ \mathsf{A})^{-1}(\mathsf{m})) \cap \mathsf{ans(Q)}\rangle\big\}$
The key observation here is that the set of objects $(\mathsf{Rank_P} \circ \mathsf{A})^{-1}(\mathsf{i})$, $\mathsf{i} = 0, \cdots,$

m, can be expressed as follows:

$(\mathsf{Rank_P} \circ \mathsf{A})^{-1}(\mathsf{i}) = \cup \Big\{ \mathsf{A}^{-1}(\mathsf{a})/\mathsf{a} \in \mathsf{Rank_P^{-1}}(\mathsf{i}) \Big\}$ (1)

Now, suppose that $\mathsf{Rank_P^{-1}}(\mathsf{i}) = \{\mathsf{a_1}, \cdots, \mathsf{a_p}\}$, and define $\mathsf{Q_i} = \vee \mathsf{Rank_P^{-1}}(\mathsf{i}) = \mathsf{a_1} \vee \cdots \vee \mathsf{a_p}$. Then the right-hand side of expression (1) above is precisely the answer to the standard query $\mathsf{Q_i}$. In other words we have: $(\mathsf{Rank_P} \circ \mathsf{A})^{-1}(\mathsf{i}) = \mathsf{ans}(\mathsf{Q_i})$, therefore the answer to $\mathsf{P_q}$ can be rewritten as follows:

$\mathsf{ans}(\mathsf{P_q}) = \Big\{ \langle 0, \mathsf{ans}(\mathsf{Q_0}) \cap \mathsf{ans}(\mathsf{Q}) \rangle, \cdots, \langle \mathsf{m}, \mathsf{ans}(\mathsf{Q_m}) \cap \mathsf{ans}(\mathsf{Q}) \rangle \Big\}$

Example 2. Consider the precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, where

$\mathsf{Q} : (\mathsf{Language} = \mathsf{English} \vee \mathsf{German})$

$\mathsf{P} : \mathsf{Poetry} \rightarrow \mathsf{Fiction}, \mathsf{Drama} \rightarrow \mathsf{Fiction}, \mathsf{Drama} \rightarrow \mathsf{Comedy}, \mathsf{Fiction} \rightarrow \mathsf{Comedy}$

The induced ranking $\mathsf{Rank_P}$ is as in Examle 1. To compute the answer of $\mathsf{P_q}$ we proceed as follows:

Anwer to $\mathsf{Q} : \mathsf{ans}(\mathsf{Q}) = \{2, 3, 4, 5, 6, 7, 8\}$

Answer to $\mathsf{Q_i}$, for $\mathsf{i} = 2, 1, 0$:

$\mathsf{ans}(\mathsf{Q_2}) = (\mathsf{Rank_P} \circ \mathsf{A})^{-1}(2) = \mathsf{ans}(\mathsf{Category} = \mathsf{Poetry} \vee \mathsf{Drama}) = \{1, 3, 5, 7\}$

$\mathsf{ans}(\mathsf{Q_1}) = (\mathsf{Rank_P} \circ \mathsf{A})^{-1}(1) = \mathsf{ans}(\mathsf{Category} = \mathsf{Fiction}) = \{2, 4, 6\}$

$\mathsf{ans}(\mathsf{Q_0}) = (\mathsf{Rank_P} \circ \mathsf{A})^{-1}(0) = \mathsf{ans}(\mathsf{Category} = \mathsf{Comedy}) = \{8, 9\}$

Answer to Pq:

$\mathsf{ans}(\mathsf{P_q}) = \big\{ \langle 2, \{3, 5, 7\} \rangle, \langle 1, \{2, 4, 6\} \rangle, \langle 0, \{8\} \rangle \big\}$

In the previous example, we first computed the answers to the queries $\mathsf{Q}, \mathsf{Q_0}, \cdots$, $\mathsf{Q_m}$ and then we computed the answer to $\mathsf{P_q}$ by taking the intersection of each $\mathsf{ans}(\mathsf{Q_i})$ with the set $\mathsf{ans}(\mathsf{Q})$. However, it is possible to rewrite the answer to $\mathsf{P_q}$, in a way that the intersections are computed *within* the query answering process. Indeed, we can rewrite the answer to $\mathsf{P_q}$ as follows:

$\mathsf{ans}(\mathsf{Pq}) = \big\{ \langle 0, \mathsf{ans}(\mathsf{Q_0} \wedge \mathsf{Q}) \rangle, \cdots, \langle \mathsf{m}, \mathsf{ans}(\mathsf{Q_m} \wedge \mathsf{Q}) \rangle \big\}$

If we set $\mathsf{Q'_i} = \mathsf{Q_i} \wedge \mathsf{Q}, \mathsf{i} = 1, \cdots, \mathsf{m}$, then we can write:

$\mathsf{ans}(\mathsf{Pq}) = \big\{ \langle 0, \mathsf{ans}(\mathsf{Q_i}) \rangle, \cdots, \langle \mathsf{m}, \mathsf{ans}(\mathsf{Q'_m}) \rangle \big\}$

Determining when this last form is preferable to the previous one is an optimization issue that lies outside the scope of this paper.

In order to define the answer to the query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, where P is a set of precedence relations over multiple attributes, we need some auxiliary definitions and notations. Let us denote by $\mathsf{P/A}$ a precedence relation over attribute A, and let us denote by values($\mathsf{P/A}$), the set of values of $\mathsf{dom}(\mathsf{A})$ that appear in $\mathsf{P/A}$. Given a set of such precedence relations, say $\mathsf{P} = \{\mathsf{P/A_1}, \cdots, \mathsf{P/A_k}\}$, let us denote by tuples(P) the cartesian product of the sets values($\mathsf{P/A_i}$), $\mathsf{i} = 1, \cdots, \mathsf{k}$, that is:

$\mathsf{tuples}(\mathsf{P}) = \mathsf{values}(\mathsf{P/A1}) \times \cdots \times \mathsf{values}(\mathsf{P/Ak})$

There are two ways to define a precedence on the set $\mathsf{tuples}(\mathsf{P})$, based on the precedence relations $\mathsf{P/A_1}, \cdots, \mathsf{P/A_k}$. The first way, called *Pareto precedence*, considers that no $\mathsf{A_i}$ has precedence over any other attribute $\mathsf{A_j}$; the second, called *Lexicographic* precedence considers that the $\mathsf{A_i}$'s form a ranking domain. The following two propositions describe how the set P of precedences induces a Pareto or Lexicographic precedence over the set $\mathsf{tuples}(\mathsf{P})$.

Proposition 3 (Pareto Precedence). *The following binary relation "$\rightarrow_{\mathsf{P}}$" on the set* $\mathsf{tuples(P)}$ *is a precedence relation : for all tuples* s, t *in* $\mathsf{tuples(P)}$, $\mathsf{s} \rightarrow_{\mathsf{P}} \mathsf{t}$ *iff* $\mathsf{s} \neq \mathsf{t}$ *and, for* $\mathsf{i} = 1, \cdots, \mathsf{k}$, *either* $\mathsf{s.A_i} = \mathsf{t.A_i}$ *or* $\mathsf{s.A_i} \rightarrow_{\mathsf{i}} \mathsf{t.A_i}$ *(where* $\rightarrow_{\mathsf{i}}$ *denotes the precedence over* $\mathsf{A_i}$*).*

Proposition 4 (Lexicographic Precedence). *Let the attributes* $\mathsf{A_1}, \cdots, \mathsf{A_k}$ *form a ranking domain with first element* $\mathsf{A_1}$*, last element* $\mathsf{A_k}$ *and succ(*$\mathsf{A_i}$*)* $= \mathsf{A_{i+1}}$*, for* $\mathsf{i} = 1, \cdots, \mathsf{k}-1$*. The following binary relation "*$\rightarrow_{\mathsf{L}}$*" on the set* $\mathsf{tuples(P)}$ *is a precedence relation: for all tuples* s, t *in* $\mathsf{tuples(P)}$, $\mathsf{s} \rightarrow_{\mathsf{L}} \mathsf{t}$ *iff* $\mathsf{s} \neq \mathsf{t}$ *and [either* $\mathsf{s.A_1} \rightarrow_1 \mathsf{t.A_1}$ *or (*$\mathsf{s.A_1} = \mathsf{t.A_1}$ *and* $\mathsf{s.A_2 \cdots A_k} \rightarrow_{\mathsf{L}} \mathsf{t.A_2 \cdots A_k}$*)]*
Here $\mathsf{s.X}$ *denotes the restriction of tuple s to the attributes of the set X.*

With the above notations and definitions at hand, we can now give the definition of answer when the query contains precedences over more than one attribute.

Definition 8 (Precedence Over Multiple Attributes). *Let* $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$ *be a precedence query, where* $\mathsf{P} = \{\mathsf{P/A_1}, \cdots, \mathsf{P/A_k}\}$*. The answer to* $\mathsf{P_q}$*, denoted by* $\mathsf{ans(P_q)}$*, is defined in three steps as follows:*

1. *Use the precedences* $\mathsf{P/A_1}, \cdots, \mathsf{P/A_k}$ *to define a precedence on the set tuples(P); call the resulting precedence* $\mathsf{P}^{\times}$ *where* $\mathsf{P}^{\times}$ *stands for either "Pareto" or "lexicographic" precedence (a choice made by the user).*
2. *Define the function* $\mathsf{A_1} \times \cdots \times \mathsf{A_k} : \mathsf{Ref} \rightarrow \mathsf{dom(A_1)} \times \cdots \times \mathsf{dom(A_k)}$ *such that* $(\mathsf{A_1} \times \cdots \times \mathsf{A_k})(\mathsf{i}) = (\mathsf{A_1(i)}, \cdots, \mathsf{A_k(i)})$
3. *Define the answer to* $\mathsf{P_q}$ *as follows:*
   $\mathsf{ans(P_q)} = \mathsf{pr}\big[(\mathsf{Rank_{P^\times}} \circ (\mathsf{A_1} \times \cdots \times \mathsf{A_k}))/\mathsf{ans(Q)}\big]$

Note that when P consists of a precedence over a single attribute, then we find again the answer defined in Definition 7. In fact, all we have said in the case of precedence over a single attribute carry over for precedence over multiple attributes, if we replace $\mathsf{Rank_P}$ by $\mathsf{Rank_{P^\times}}$, and the function A by the function $\mathsf{A_1} \times \cdots \times \mathsf{A_k}$ (see Definition 8). The only difference is that now, instead of p attribute values $\mathsf{a_1}, \cdots, \mathsf{a_p}$ we have p tuples $\mathsf{t_1}, \cdots, \mathsf{t_p}$, so the query $\mathsf{Q_i}$ is now defined as follows:
$\mathsf{Q_i} = \vee(\mathsf{Rank_{P^\times}})^{-1}(\mathsf{i}) = \mathsf{t_1} \vee \cdots \vee \mathsf{t_p}$,
Here, each $\mathsf{t_i}$ is seen as the conjunction of its attribute values. Let us see an example.

Example 3. Consider the precedence query $\mathsf{P_q} = (\mathsf{Q}, \mathsf{P})$, where
$\mathsf{Q} : (\mathsf{Language} = \mathsf{English} \vee \mathsf{German})$
$\mathsf{P/Category} : \mathsf{Poetry} \rightarrow \mathsf{Fiction}$ and $\mathsf{P/Format} : \mathsf{Pdf} \rightarrow \mathsf{Word}$
Precedence over attributes: $\mathsf{Category} \rightarrow \mathsf{Format}$ (actually given by the user)
To compute the answer of $\mathsf{P_q}$ we proceed as follows:
$\mathsf{values(P/Category)} = \{\mathsf{Poetry}, \mathsf{Fiction}\}$, $\mathsf{values(P/Format)} = \{\mathsf{Pdf}, \mathsf{Word}\}$
$\mathsf{tuples(P)}$=$\mathsf{values(P/Category)} \times \mathsf{values(P/Format)}$= {Poetry.Pdf, Poetry.Word, Fiction.Pdf, Fiction.Word}
Note: To simplify matters, we shall use a shorthand notation for the tuples: PP,

PW, FP, FW stand for the tuples Poetry.Pdf, Poetry.Word, Fiction.Pdf, Fiction.Word, respectively.
Induced precedence $\mathsf{P}^{\times}$ (following the given precedence $\mathsf{Category} \rightarrow \mathsf{Format}$ over attributes):
$\mathsf{PP} \rightarrow_{\mathsf{L}} \mathsf{PW}, \mathsf{PP} \rightarrow_{\mathsf{L}} \mathsf{FP}, \mathsf{PP} \rightarrow_{\mathsf{L}} \mathsf{FW}, \mathsf{PW} \rightarrow_{\mathsf{L}} \mathsf{FP}, \mathsf{PW} \rightarrow_{\mathsf{L}} \mathsf{FW}, \mathsf{FP} \rightarrow_{\mathsf{L}} \mathsf{FW}$
Ranking $\mathsf{Rank}_{\mathsf{P}^\times}$: $\mathsf{Rank}_{\mathsf{P}^\times}(\mathsf{PP}) = 3$, $\mathsf{Rank}_{\mathsf{P}^\times}(\mathsf{PW}) = 2$, $\mathsf{Rank}_{\mathsf{P}^\times}(\mathsf{FP}) = 1$,
$\mathsf{Rank}_{\mathsf{P}^\times}(\mathsf{FW}) = 0$
We are now ready to proceed as in the case of precedence over a single attribute, letting f stand for the function $\mathsf{Category} \times \mathsf{Format}$:
Anwer to Q : $\mathsf{ans}(\mathsf{Q}) = \{2, 3, 4, 5, 6, 7, 8\}$
Answer to Q_i, for $\mathsf{i} = 3, 2, 1, 0$
$\mathsf{ans}(\mathsf{Q}_3) = (\mathsf{Rank}_{\mathsf{P}^\times} \circ \mathsf{f})^{-1}(3) = \mathsf{ans}(\mathsf{Category} = \mathsf{Poetry} \wedge \mathsf{Format} = \mathsf{Pdf}) = \{3\}$
$\mathsf{ans}(\mathsf{Q}_2) = (\mathsf{Rank}_{\mathsf{P}^\times} \circ \mathsf{f})^{-1}(2) = \mathsf{ans}(\mathsf{Category} = \mathsf{Poetry} \wedge \mathsf{Format} = \mathsf{Word}) = \{1, 5\}$
$\mathsf{ans}(\mathsf{Q}_1) = (\mathsf{Rank}_{\mathsf{P}^\times} \circ \mathsf{f})^{-1}(1) = \mathsf{ans}(\mathsf{Category} = \mathsf{Fiction} \wedge \mathsf{Format} = \mathsf{Pdf}) = \{2, 4\}$
$\mathsf{ans}(\mathsf{Q}_0) = (\mathsf{Rank}_{\mathsf{P}^\times} \circ \mathsf{f})^{-1}(0) = \mathsf{ans}(\mathsf{Category} = \mathsf{Fiction} \wedge \mathsf{Format} = \mathsf{Word}) = \{6\}$
Answer to P_q:

$$\begin{aligned}\mathsf{ans}(\mathsf{P}_\mathsf{q}) = \Big\{ & \langle 3, \mathsf{ans}(\mathsf{Q}_3) \cap \mathsf{ans}(\mathsf{Q})\rangle, \langle 2, \mathsf{ans}(\mathsf{Q}_2) \cap \mathsf{ans}(\mathsf{Q})\rangle, \langle 1, \mathsf{ans}(\mathsf{Q}_1) \cap \mathsf{ans}(\mathsf{Q})\rangle, \\ & \langle 0, \mathsf{ans}(\mathsf{Q}_0) \cap \mathsf{ans}(\mathsf{Q})\rangle \Big\} \\ = \Big\{ & \langle 3, \{3\}\rangle, \langle 2, \{5\}\rangle, \langle 1, \{2, 4\}\rangle, \langle 0, \{6\}\rangle \Big\}\end{aligned}$$

6.2 Answering a Scoring Query

To answer a scoring query we use the same formal framework as for a precedence query. To simplify the presentation, we distinguish again two cases for the scoring function: scoring over a single attribute, and scoring over two or more attributes.

Definition 9 (Scoring Over a Single Attribute). *Let* $\mathsf{S}_\mathsf{q} = (\mathsf{Q}, \mathsf{S})$ *be a precedence query, where* S *is a scoring function over a single attribute* A*. The answer to* S_q*, denoted* $\mathsf{ans}(\mathsf{S}_\mathsf{q})$ *is defined as follows:* $\mathsf{ans}(\mathsf{S}_\mathsf{q}) = \mathsf{pr}[(\mathsf{S} \circ \mathsf{A})/\mathsf{ans}(\mathsf{Q})]$

In the case of multiple attributes the only difference with precedence queries lies in the definition of a scoring function over tuples from the scoring functions over single attributes. More precisely, let S/A denote a scoring function over attribute A, let $\mathsf{def}(\mathsf{S}/\mathsf{A})$ denote the domain of definition of S/A, and for a set of scoring functions $\mathsf{S} = \{\mathsf{S}/\mathsf{A}_1, \cdots, \mathsf{S}/\mathsf{A}_\mathsf{k}\}$ define: $\mathsf{tuples}(\mathsf{S}) = \mathsf{def}(\mathsf{S}/\mathsf{A}_1) \times \cdots \times \mathsf{def}(\mathsf{S}/\mathsf{A}_\mathsf{k})$ Given a tuple t in $\mathsf{tuples}(\mathsf{S})$, one can combine the scores of the individual attribute values of t in order to assign a score to t. For example, two ways of doing this is to define the score of t to be either the maximum or the minimum of all scores in t. Whatever the method used, let us denote by $\mathsf{S}^\times$ the resulting scoring function on $\mathsf{tuples}(\mathsf{S})$.

Definition 10 (Scoring Over Multiple Attributes). *Let* $\mathsf{S}_\mathsf{q} = (\mathsf{Q}, \mathsf{S})$ *be a scoring query, where* $\mathsf{S} = \{\mathsf{S}/\mathsf{A}_1, \cdots, \mathsf{S}/\mathsf{A}_\mathsf{k}\}$*. The answer to* S_q*, denoted by* $\mathsf{ans}(\mathsf{S}_\mathsf{q})$*, is defined in three steps as follows:*

1. *Use the scoring functions* $\mathsf{S/A_1}, \cdots, \mathsf{S/A_k}$ *to define a scoring function on the set* $\mathsf{tuples(S)}$*; call the resulting scoring function* $\mathsf{S}^{\times}$*, where* $\mathsf{S}^{\times}$ *can be derived by taking max, min, or some other operation over the scores of each tuple (a choice to be made by the user).*
2. *Define the function* $\mathsf{A_1} \times \cdots \times \mathsf{A_k} : \mathsf{dom(Oid)} \rightarrow \mathsf{dom(A_1)} \times \cdots \times \mathsf{dom(A_k)}$ *such that* $(\mathsf{A_1} \times \cdots \times \mathsf{A_k})(\mathsf{i}) = (\mathsf{A_1(i)}, \cdots, \mathsf{A_k(i)})$
3. *Define the answer to* $\mathsf{S_q}$ *as follows:* $\mathsf{ans(S_q)} = \mathsf{pr}[(\mathsf{S}^{\times} \circ (\mathsf{A_1} \times \cdots \times \mathsf{A_k}))/\mathsf{ans(Q)}]$

Note that when S consists of a scoring function over a single attribute, then we find again the answer defined in Definition 9.
The remarks made for precedence queries carry over, in a straightforward manner, to the evaluation of a scoring query $\mathsf{S_q} = (\mathsf{Q}, \mathsf{S})$, in which S is either a scoring function over a single attribute (Definition 9) or a set of scoring functions over multiple attributes (Definition 10); we only have to replace $\mathsf{Rank_P}$ with S, in the case of a single attribute, and $\mathsf{Rank_{P^{\times}}}$ with $\mathsf{S}^{\times}$, in the case of multiple attributes.

7 Concluding Remarks

We have presented a unified formal framework for the definition and evaluation of both, qualitative and quantitative preference queries. For qualitative queries, in particular, we have introduced a language for expressing preferences, namely the language of precedence relations, which is strictly more expressive than the languages that have been proposed in the literature. Concerning the declaration of preferences over multiple attributes, we have adopted the approach of declaring preferences over individual attributes, then combining the declarations to derive preferences over tuples. An alternative approach is to declare preferences directly on tuples. Clearly, in such a scenario, the definitions of answer given earlier still hold. However, we think that it is more difficult for a user to express preferences by comparing or scoring whole tuples rather than individual attribute values.
Ongoing research aims at two objectives: (a) designing efficient algorithms for the evaluation of preference queries and (b) designing a user friendly interface for the declaration of preference queries. Both these tasks are under way.

References

1. Agrawal, R., and Wimmers, E. L. A Framework for Expressing and Combining Preferences. In Proceedings of the ACM SIGMOD International Conference on Management of Data (Dallas, USA), 2000, 297-306.
2. Amartya K. Sen, 1970 [1984], Collective Choice and Social Welfare, ISBN 0-444-85127-5
3. Arrow, K.J., Social Choice and Individual Values, New York: John Wiley and Sons, 1951; second edition, 1963.
4. Balke, W.-T., Güntzer U., and Kießling W. On Real-time Top k Querying for Mobile Services. International Conference on Cooperative Information Systems, Irvine, USA, 2002.

5. Beardon, A.F., Candeal, J.C., Herden, G., Indurain, E., and G.B. Mehta: "The non-existence of a utility function and the structure of non-representable preference relations," Journal of Mathematical Economics 37 (2002), 17-38.
6. Börzsönyi, S., Kossman, D., and Stocker, K. The Skyline Operator. In Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, 2001.
7. Boutilier, C., Brafman, R., Hoos, H., and Poole, D. Reasoning with conditional ceteris paribus preference statements. In UAI-99, pages 71- 80, 1999.
8. Bouyssou D.; Vincke P,Introduction to topics on preference modelling, Annals of Operations Research, 80, 1998, i-xiv
9. . Bridges, D.S., A numerical representation of preferences with intransitive indifference, Journal of Mathematical Economics, 1983, vol. 11, issue 1, pages 25-42
10. Bridges, D.S., Numerical representation of intransitive preferences on a countable set, Journal of Economic Theory, Elsevier, vol. 30(1), 1983, pages 213-217.
11. Bruno, N., Gravano, L., and Marian, A. Evaluating Top-k Queries over Web-Accessible Databases. ICDE, 2002, 369-279.
12. Chang, C. L. Deduce: A deductive query language for relational databases. In Pattern Rec. and Art. Int., C. H. Chen, Ed. Academic Press, New York, 1976, 108-134.
13. Chaudhuri, S., and Gravano, L. Evaluating Top-k Selection Queries. In Proceedings of the 25th International Conference on Very Large Data Bases, 1999.
14. Chebotarev, Pavel Yu.; Shamis, Elena; "Characterizations of Scoring Methods for Preference Aggregation"; Annals of Operations Research; Vol. 80; 1998; 299-332; #3685
15. Chomicki J, Iterative Modification and Incremental Evaluation of Preference Queries. FoIKS 2006: 63-82
16. Chomicki, J. Preference formulas in relational queries. ACM Transactions on Database Systems (TODS), 28(4), 2003, 427-466.
17. Chomicki, J. Querying with Intrinsic Preferences. In Proceedings of the 8th International Conference on EDBT, Prague, Czech Rep., 2002, 34-51.
18. Chomicki, J. Semantic optimization of preference queries. In 1st Int. Sym. on Appl. of Constraint Databases, Springer (LNCS 3074), 2004.
19. Domshlak C. and Brafman R.. Cp-nets - reasoning and consistency testing. In KR-02, pages 121-132, 2002.
20. Fagin R et al., Comparing Partial Rankings, SIAM J. Discrete Mathematics
21. Fagin R. et al. Comparing and aggregating rankings with ties. In PODS, 2004.
22. Fagin R., Kumar R., and Sivakumar D. Comparing top k lists. In SODA, 2003.
23. Fagin R., Kumar, R. and Sivakumar D. Efficient similarity search and classification via rank aggregation. In SIGMOD, 2003.
24. Fishburn P.C.. Preference structures and their numerical representations. Theoretical Computer Science, 217(2):359-383, April 1999
25. Fishburn, P. C., 1988 Nonlinear Preference and Utility Theory, John Hopkins University Press, Baltimore.
26. Fishburn, P.C., Intransitive indifference with unequal indifference intervals, Journal of Mathematical Psychology 7, 1970, 144-149
27. Fishburn, P.C., Inverted orders for monotone scoring rules, Discrete Applied Mathematics, 3 (1981), 27-36
28. Fishburn, P.C., Nontransitive Preferences in Decision Theory, Journal of Risk and Uncertainty, Springer, vol. 4(2), 1991, pages 113-34
29. Fishburn, P.C., The Theory of Social Choice, Princeton: Princeton University Press, 1973

30. Fishburn, Peter C. Utility Theory for Decision Making. Huntington, NY. Robert E. Krieger Publishing Co. 1970
31. Hafenrichter B, Kießling W: Optimization of Relational Preference Queries. ADC 2005: 175-184
32. Hoppe, Hans-Hermann, 1993, "The Economics and Ethics of Private Property," Kluwer Academic Publishers
33. Kendall M. and Gibbons J. D.. Rank Correlation Methods. Edward Arnold, London, 1990.
34. Kendall M. G. The treatment of ties in ranking problems. Biometrika, 33(3):239251, 1945.
35. Kießling W., Preference Queries with SV-Semantics., in COMAD (J. Haritsa and T. Vijayaraman,. eds.), pp. 1526, Computer Society of India, 2005
36. Kießling, W. Foundations of Preferences in Database Systems. In Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China, 2002, 311-322.
37. Kießling, W., and Köstler, G. Preference SQL Design, Implementation, Experiences. In Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China, 2002, 990-1001.
38. Koutrika G, Ioannidis Y, Personalized Queries under a Generalized Preference Model. ICDE 2005: 841-852
39. Koutrika, G., and Ioannidis, Y. Personalization of Database Queries Using Stored Preferences. In Proceedings of Seminar on Preferences: Specification, Inference, Applications, Schloss, Dagstuhl, DE, 2004.
40. Kreps, David M. Notes on the Theory of Choice. Boulder, CO. Westview Press. 1988
41. Manzini Paola & Mariotti Marco, 2003. "How vague can one be? Rational preferences without completeness or transitivity," Game Theory and Information 0312006, EconWPA, revised 16 Jul 2004.
42. Mas-Colell, Andreu; Whinston, Michael; & Green, Jerry (1995). Microeconomic Theory. Oxford: Oxford University Press. ISBN 0-19-507340-1
43. May, K. O.: 1954, Intransitivity, utility, and the aggregation of preference patterns, Econometrica, 22, 1-13.
44. May, R. M.: 1971, Some mathematical remarks on the paradox of voting, Behavioral Science, 16, 143-151.
45. Truchon, Michel, 2004, Aggregation of Rankings in Figure Skating, Cahiers de recherche 0414, CIRPEE.
46. van Acker, P.: 1990, Transitivity revisited, Annals of Operations Research, 23, 1-35.
47. Wang Xuzhu, An investigation into relations between some transitivity-related concepts, Fuzzy Sets and Systems, Volume 89, Issue 2, 16 July 1997, Pages 257-262.
48. Delos Network of Excellence in Digital Libraries (http://www.delos.info/)
49. N. Spyratos: A Functional Model for Data Analysis, Intl. Conference Flexible Query Answering Systems (FQAS 2006: 51-64), Milano, Italy, June 8-10, 2006
50. P.Fejer, D. Simovici, Mathematical Foundations of Computer Science, Vol.I: Sets, Relations and Induction, Springer 1991