International Journal of Approximate Reasoning 52 (2011) 38–48

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Strict and tolerant antidivision queries with ordinal layered preferences

Patrick Bosc, Olivier Pivert [*], Olivier Soufflet

Irisa – Enssat, University of Rennes 1, Technopole Anticipa, 22305 Lannion Cedex, France

A R T I C L E I N F O

Article history:
Received 18 November 2009
Received in revised form 8 April 2010
Accepted 13 April 2010
Available online 18 April 2010

Keywords:
Database querying
Antidivision
Ordinal preferences
Quotient
Query relaxation
Fuzzy quantifiers

A B S T R A C T

In this paper, we are interested in taking preferences into account for a family of queries inspired by the antidivision. An antidivision query aims at retrieving the elements associated with none of the elements of a specified set of values. We suggest the introduction of preferences inside such queries with the following specificities: (i) the user gives his/her preferences in an ordinal way and (ii) the preferences apply to the divisor which is defined as a hierarchy of sets. Different uses of the hierarchy are investigated, which leads to queries conveying different semantics and the property of the result delivered is characterized. Furthermore, the case where a conjunctive stratified antidivision query returns an empty set of answers is dealt with, and an approach aimed at relaxing such queries is proposed.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Queries including preferences have received a growing interest during the last decade. Motivations for introducing preferences inside database queries are manifold [1]. 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 items in the database.

Approaches to database preference queries may be classified into two categories according to their qualitative or quantitative nature [1]. In the latter, preferences are expressed quantitatively by a monotone scoring function (the overall score is positively correlated with partial scores), often taken as a weighted linear combination of attribute values (which have therefore to be numerical). Since the scoring function associates each tuple with a numerical score, tuple t_1 is preferred to tuple t_2 if the score of t_1 is higher than the score of t_2. Representatives of this family of approaches are top-k queries [2], fuzzy-set-based approaches (e.g., [3]), and the model proposed in [4]. However, it is well known that scoring functions cannot represent all preferences that are strict partial orders [5], not even those that occur in database applications in a natural way [6]. Another issue is that devising the scoring function may not be simple. In the qualitative approach, preferences are defined through binary preference relations. Since binary preference relations can be defined in terms of scoring functions, the qualitative approach is more general than the quantitative one. Among the representatives of this second family of approaches, let us mention the system called *Preferences* [7], an approach based on CP-nets [8], and those relying on a dominance relation, e.g. Pareto order, in particular *PreferenceSQL* [9], Skyline queries [10] and the approach presented in [6].

In this paper, a qualitative view of preference queries is adopted, considering that it is less demanding for a user in terms of elicitation. Indeed, it is sufficient to use a completely ordered scale involving symbolic weights for assessing the

[*] Corresponding author.
E-mail addresses: bosc@enssat.fr (P. Bosc), pivert@enssat.fr (O. Pivert), soufflet@enssat.fr (O. Soufflet).

0888-613X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2010.04.004

importance of the different preference criteria. Another aspect worthy of comment concerns the type of result returned to the user. Pareto-order and CP-net-based approaches yield partially ordered answers, which can be somewhat frustrating for a user. Here, a type of preference queries for which a totally ordered result can be computed is considered. More precisely, preference queries dealt with involve a hierarchy of Boolean conditions (which induces an implicit ordinal scale).

Up to now, most of the research works on database preference queries have focused on fairly simple queries where preferences apply only to selections. The objective of this paper is to enlarge the scope of preference queries by considering more complex ones, founded on the association of an element with a given set of values, in the spirit of the division operation. Taking preferences into account will allow for keeping only the best k answers, in the spirit of top-k queries [2].

In the following, antidivision queries are considered. Let r be a relation of schema $R(X,A)$ and s a relation of schema $S(B)$, with A and B compatible (sets of) attributes, i.e., attributes defined on the same domain. The antidivision query $r[A \circledast B]s$ retrieves the X-values present in relation r which are associated in r with none of the B-values present in s. By analogy with a division, relation r may be called the dividend and relation s the divisor. Knowing that an antidivision delivers a non-discriminated set of elements, the idea is here to introduce preferences in this operator. Several lines for assigning preferences may be thought of, depending on whether preferences concern the divisor, the dividend or both, tuples individually (see e.g., [11–13]), or (sub) sets of tuples. In this paper, we investigate the case where: (i) preferences are purely ordinal and ii) they apply to the divisor only, which is structured as a hierarchy (a set of layers). An element x of the dividend will be all the more acceptable as it is not connected with a certain number of the subsets (S_i's) defined over the divisor. Three different roles allotted to the divisor (described as a hierarchical set) are envisaged in the remainder of this paper. They differ in the way the layers of the divisor are taken into account for discrimination.

The rest of the paper is organized as follows. Section 2 is dedicated to some reminders on the antidivision operator. Three types of stratified antidivision queries are studied and modeled in Section 3. In Section 4, it is shown that the result returned by these queries can be characterized as an "antiquotient", i.e., a largest relation according to a given inclusion constraint. In Section 5, the case where a conjunctive stratified antidivision query returns an empty set of answers is dealt with, and an approach aimed at relaxing such queries is proposed. The idea consists in replacing the crisp quantifier *not exists* underlying strict antidivision queries by a more tolerant fuzzy quantifier such as *almost none*. Section 6 deals with implementation aspects and presents some experimental results as to the performances of different algorithms implementing conjunctive stratified antidivision queries. The conclusion summarizes the contribution of the paper and draws some lines for future research in particular as to implementation issues.

2. Some reminders about the antidivision

In the rest of the paper, the dividend relation r has the schema (A, X), while that of the divisor relation s is (B) where A and B are compatible sets of attributes. The division of relation r by relation s is defined as:

$$r[A \div B]s = \{x | x \in r[X] \wedge \ s \subseteq \Omega_r(x)\} \tag{1}$$

or equivalently as:

$$r[A \div B]s = \{x | x \in r[X] \ \wedge \ \forall a, \quad a \in s \Rightarrow (a,x) \in r\} \tag{2}$$

where $r[X]$ denotes the projection of r over X and $\Omega_r(x) = \{a | (a,x) \in r\}$. In other words, an element x belongs to the result of the division of r by s iff it is associated in r with at least all the values a appearing in s. The justification of the term "division" assigned to this operation relies on the fact that a property similar to that of the quotient of integers holds. Indeed, the resulting relation *d-res* obtained with expression (1) or (2) has the double characteristic:

$$\forall t \in d\text{-}res, \quad s \times \{t\} \subseteq r \tag{3}$$

$$\forall t \in (r[X] - d\text{-}res), \quad s \times \{t\} \not\subseteq r \tag{4}$$

$\times$ denoting the Cartesian product of relations. Expressions (3) and (4) express the fact that relation *d-res* is a quotient, i.e., the largest relation whose Cartesian product with the divisor returns a result included in the dividend. In a similar way, we call antidivision the operator $\circledast$ defined the following way:

$$r[A \circledast B]s = \{x | x \in r[X] \ \wedge \ s \subseteq cp(\Omega_r(x))\} \tag{5}$$

where $cp(r)$ is the complement of r, or equivalently:

$$r[A \circledast B]s = \{x | x \in r[X] \ \wedge \ \forall a, \quad a \in s \Rightarrow (a,x) \notin r\} \tag{6}$$

The result *ad-res* of the antidivision may be called an "antiquotient", i.e., the largest relation whose Cartesian product with the divisor is included in the complement of the dividend. Thus, the following two properties hold:

$$\forall t \in ad\text{-}res, \quad s \times \{t\} \subseteq cp(r) \tag{7}$$

$$\forall t \in (r[X] - ad\text{-}res), \quad s \times \{t\} \not\subseteq cp(r) \tag{8}$$

where $E - F$ denotes the difference between E and F. In an SQL-like language, the division of r by s may be expressed:

select X **from** r [**where** *condition*] **group by** X **having set** (A) **contains** $\{v_1, \ldots, v_n\}$

and the antidivision similarly as:

select X **from** r [**where** *condition*] **group by** X **having set**(A) **contains-none** $\{v_1, \ldots, v_n\}$ (9)

where the operator "contains-none" states that the two operand sets do not overlap. An alternative expression of the latter can be based on a difference:

(**select** X **from** r) **minus** (**select** X **from** r **where** A **in** (**select** B **from** s)). (10)

Example 1. Let us consider the following relations P (product, component, proportion), which describes the composition of some chemical products and N (component) which gathers the identifications of noxious components:

$$P = \{(p_1, c_1, 3), (p_1, c_2, 4), (p_1, c_3, 54), (p_2, c_2, 30), (p_3, c_2, 8), (p_3, c_6, 22)\},$$
$$N = \{c_1, c_2, c_5\}.$$

The query "retrieve any product which does not contain any noxious component in a proportion higher than 5%" can be expressed as the antidivision of the relation Prod′ derived from Prod made of $\{(p_1, c_3), (p_2, c_2), (p_3, c_2), (p_3, c_6)\}$ by Nox, whose result according to (5) or (6) is $\{p_1\}$ and it is easy to check that formulas (7) and (8) both hold.

3. Three types of stratified antidivision queries

3.1. Antidivision and preferences

What has been said until now concerns what we could call traditional antidivision queries inasmuch as no preferences come into play. We now move to more advanced queries mixing antidivision and the expression/handling of preferences. The three types of queries investigated here are the following:

- *CJ queries*: a direct extension of the antidivision in a conjunctive way, where the connection with the first layer of the divisor is forbidden and the non-association with the following ones is considered only desirable: find the elements x not connected with S_1 and if possible... and if possible S_n.
- *DJ queries*: a disjunctive view where x is all the more satisfactory as it is connected with none of the values of a highly preferred sub (set) of the divisor: find the elements x not connected with S_1 or else... or else S_n,
- *FD queries*: an intermediate approach where x is all the more highly ranked as it is not connected with numerous and preferred (sub) sets of the divisor: find the elements x not connected with S_1 and-or... and-or S_n.

Knowing that the dividend may be any intermediate relation and the divisor is explicitly given by the user along with his/her preferences, the expression of these three types of antidivision queries is inspired from (9):

select top k X **from** r [**where** condition] **group by** X
having set(A) **contains-none**$\{v_{1,1},\ldots, v_{1,j_1}\}$ connector...connector$\{v_{n,1},\ldots, v_{n,j_n}\}$

where "connector" is either "and if possible", or "or else", or "and-or", and not from (10) inside which the integration of the layers of the divisor would not be easy. Such a statement induces an order over the divisor, namely $(S_1 = \{v_{1,1}, \ldots, v_{1,j_1}\}) \succ \cdots \succ (S_n = \{v_{n,1}, \ldots, v_{n,j_n}\})$ where $a \succ b$ denotes the preference of a over b. Actually, this order is about dislikes, i.e., S_1 contains the values the most highly undesired (even excluded for CJ queries) and S_n those which are the most weakly unwanted. Associated with this preference relation is an ordinal scale L with labels l_i's (such that $l_1 > \cdots > l_n > l_{n+1}$) which will be used to assign levels of satisfaction to elements pertaining to the result of stratified antidivisions (l_1 and l_{n+1} are extreme elements similar to 1 and 0 in the unit interval).

Example 2. Let us consider the case of a consumer who wants food products (e.g., noodles or vegetal oil) without certain additive substances. In the presence of the relation Products (p-name, add-s) describing which additives (add-s) are involved in products, a possible query is:

select top 6 p-name **from** Products **group by** p-name
having set(add-s) **contains-none** {AS27, BT12, C3}
and if possible {AS5, D2} **and if possible** {D8}

which induces the scale $L = l_1 > l_2 > l_3 > l_4$.

3.2. Conjunctive queries (CJ)

As mentioned before, CJ queries are basically seen as an extension of the regular antidivision. To be more or less satisfactory, an element x must be connected with none of the elements having the maximal importance (S_1). In addition, as soon as it is connected with at least one of the elements of a set S_k, its association with values of any set S_{k+p} does not intervene for its final ranking. An element x is all the more preferred as it is not associated with any of the values of the succession of sets S_1–S_i where i is large (if possible n for "perfection"). In other words, x is preferred to y if x is associated with none of the values of the sets S_1 to S_p and y is not associated with a shorter list of sets. This behavior is formalized using two approaches.

First, we consider the formal framework of relations assorted with preferences where every tuple t of a relation r is assigned a symbolic level of preference denoted by $pref_r(t)$. Tuples of the divisor are graded according to the ordering given by the user and since no preference applies to the dividend, its tuples have the maximal grade l_1 (conversely, any tuple absent from it is considered as having the grade l_{n+1}).

The usual implication ($p \Rightarrow q$=(not p) or q) involved in Formula (6) is extended to this context, which requires an adequate definition for both the negation and the disjunction. This latter is expressed thanks to the maximum (denoted by max), which satisfies most of the usual properties of the regular disjunction (associative, commutative, increasingly monotone with respect to each argument, admittance of l_{n+1} as the neutral element). Note that the maximum is meant with respect to the order of the labels. As to the negation, it corresponds to order reversal (denoted by $rev(-)$) defined as:

$$\forall i \in [1,\ n+1], \quad rev(l_i) = l_{n+2-i}$$

which is involutive, i.e., $rev(rev(l_i)) = l_i$.

Example 3. Let us consider the following scale related to the importance of a phenomenon: *complete* > *high* > *medium* > *low* > *no*. The inverse scale is:

$$\begin{array}{lllll} rev(complete) < & rev(high) < & rev(medium) < & rev(low) < & rev(no) \\ = & = & = & = & = \\ no & low & medium & high & complete \end{array}$$

This leads to the symbolic (or ordinal) version of Kleene-Dienes' implication defined as: $p \to q = max(rev(p), q)$. This implication coincides with the regular one when p and q take only the values l_1 and l_{n+1} (corresponding to true and false) and it obeys most of its properties, in particular contraposition and monotony with respect to the arguments.

Adapting the antidivision (formula (6)) to ordinal relations leads to assign each x of the dividend r the level of satisfaction $sat(x)$ defined as:

$$sat(x) = min_{v \in s}\ pref_s(v) \to rev(pref_r(v,\ x))) = min_{v \in s}\ max(rev(pref_s(v)),\ rev(pref_r(v,\ x))) \tag{11}$$

Knowing that $pref_r(v,x)$ takes only the values l_1 and l_{n+1} depending on the presence or absence of the tuple (v,x) in relation r, each term

$$max(rev(pref_s(v)), \quad rev(pref_r(v,\ x)))$$

equals l_1 if x is not connected with v in r ($(v, x) \notin r$) and $rev(pref_s(v))$ otherwise. In particular, if x is connected with none of the values of the divisor, the maximal level l_1 is obtained and as soon as an association is encountered, the level of satisfaction decreases all the more as the undesired element is highly rejected.

Another description of CJ queries may also be provided. Its interest lies in its closeness to those given later for DJ and FD queries, which cannot be modeled by means of (logical) expressions in the spirit of (11). The grade of satisfaction for x may be expressed thanks to a vector $W(x)$ of dimension n where $W(x)[i]$ = 1 if x is associated with none of the values of S_i, 0 otherwise. Let us denote:

$$I(x) = \{i | W[i] = 0\} \text{ and } imin(x) = \min(I(x))\ (n+1) \text{ if } I(x) = \emptyset).$$

The grade of satisfaction obtained by an element x (t sat(x)) is expressed thanks to the scale L (implicitly) provided by the user as follows:

$$sat(x) = l_{n+2-imin(x)}. \tag{12}$$

So doing, the satisfaction is seen as a composition of the results of the antidivision of the dividend with each of the layers of the divisor. It is easy to prove that formulas (11) and (12) deliver the same result.

3.3. Disjunctive queries (DJ)

While CJ queries have a conjunctive behavior, DJ queries are meant disjunctive instead, and S_1 is no longer a completely forbidden subset. Here, the order of the subsets according to user's preferences is used so that an element x is all the more preferred as it is connected with none of the values of S_k and k is small (ideally 1 for "perfection"). In this case again, the associations with the subsets of higher index (>k), and then lower importance, do not play any role in the discrimination strategy. In other words, x is preferred to y if x is associated with at least one of the values of each set S_1 to S_{k-1} and with none of the values of S_k and y is associated with at least one of the values of each set S_1 to S_{p-1} and with none of the values of S_p and $k < p$. Let us denote:

$$I'(x) = \{i | W[i] = 1\} \text{ and } imin'(x) = min(I'(x))(n+1 \text{ if } I'(x) = \emptyset).$$

Here again, the grade of satisfaction obtained by an element x is expressed using the ordinal scale L and:

$$sat(x) = l_{imin'(x)}. \tag{13}$$

The satisfaction is still a combination of the results of the antidivision of the dividend with each of the layers of the divisor. The grade l_1 is obtained if x is associated with none of the values of S_1, while l_{n+1} expresses rejection when the connection with at least one element of each of the S_i's holds.

3.4. Full discrimination queries (FD)

Queries of type FD are designed so as to counter the common disability of CJ and DJ queries in distinguishing between elements which are equally ranked because additional associations are not taken into account. So, the principle for interpreting FD queries is to consider all the layers for which no association occurs. An element is all the more preferred as it is connected with none of the elements of a set S_i highly excluded and this same point of view applies to break ties. Here, ordering the elements is a matter of comparison between the vectors W according to the lexicographic order ($>_{lex}$):

$$x \succ y \iff W(x) >_{lex} W(y) \iff \exists k \in [1,\ n] \text{ s.t. } \forall j < k,\ W(x)[j] = W(y)[j] \text{ and } W(x)[k] > W(y)[k]. \tag{14}$$

Here, scale L is not used directly even if the order of the elements of the vectors reflects it in the sense that, if $i < j$, $W(x)[i]$ is more important than $W(x)[j]$ as $l_i > l_j$.

Example 4. Let us take the divisor $\{\{a,b\} \succ c \succ \{d,e\}\}$ and the dividend: $r = \{(c,x_1), (g,x_1), (f,x_2), (c,x_3), (e,x_3), (a,x_4), (k,x_4)\}$. Here $n = 3$ and according to formula (11) or (12):

$$sat(x_1) = l_3, sat(x_2) = l_1, sat(x_3) = l_3,\ sat(x_4) = l_4 \text{ and } x_2 \succ \{x_1,\ x_3\} \succ x_4.$$

With formula (13), one gets:

$$sat(x_1) = sat(x_2) = sat(x_3) = l_1,\ sat(x_4) = l_2 \text{ and } \{x_1,\ x_2,\ x_3\} \succ x_4.$$

Last, using formula (14), we get a refinement of the previous two orderings, namely: $x_2 \succ x_1 \succ x_3 \succ x_4$ where the tie between x_1, x_2 and x_3 (resp. x_1 and x_3) in the result obtained with (13) (resp. (12)) is broken.

4. Characterizing the result of antidivision queries

In this section, we provide a characterization (in terms of an "antiquotient") of the result delivered by the three previous types of queries. In other words, the result returned by each of these queries is a maximal relation and it obeys formulas similar to (7) and (8). The validity of the characterization formulas given hereafter straightforwardly follows from the definitions of the three types of stratified division operators. Since relations are graded, (symbolic or ordinal) levels of satisfaction (l_i's) in both the result and the dividend have to be considered for the characterization.

For CJ queries, if tuple x of the result is assigned the grade l_i ($i \in [1, n+1]$), the following properties hold:

$$\text{if } i \in [1,\ n],\ \forall k \in [1,\ n-i+1],\ S_k \times \{x\} \subseteq cp(r) \tag{15}$$

$$\text{if } i \in [2,\ n],\ S_{n-i+2} \times \{x\} \not\subseteq cp(r). \tag{16}$$

In a similar way, for DJ queries, if x has received the grade of satisfaction l_i (letting S_{n+1} be empty) one has the double property:

$$S_i \times \{x\} \subseteq cp(r) \tag{17}$$

$$\forall k \in [1,\ i-1],\quad S_k \times \{x\} \not\subseteq cp(r) \tag{18}$$

As to FD queries, let us recall that the values of the vector $W(x)$ state whether x is connected ($W(x)[i] = 0$) or not ($W(x)[i] = 1$) with at least one of the values of layer i of the divisor. So, the following properties hold:

$$\forall i \in [1,\ n] \text{ such that } W(x)[i] = 1,\quad S_i \times \{x\} \subseteq cp(r), \tag{19}$$

$$\forall i \in [1,\ n] \text{ such that } W(x)[i] = 0,\quad S_i \times \{x\} \not\subseteq cp(r) \tag{20}$$

which means that if $sat(x)$ were higher, constraint (19) would be violated. The validity of (15)–(20) can easily be checked over Example 4.

5. Tolerant conjunctive antidivision queries

When every element x from the dividend is associated with at least one value from the first layer of the divisor, a CJ antidivision query returns an empty set of answers. In order to provide for a more flexible interpretation of the antidivision, it makes sense to envisage a less demanding view of conjunctive queries, in other terms, to consider a relaxed form of CJ queries. In this section, such a relaxation approach is presented, which rests on a weakening of the universal quantifier. Since the only layer of the divisor which has a "discarding power" is the first one—the most prioritary one—, obviating empty answers only implies to relax that layer—with no guarantee that the relaxed query will produce a non-empty result, of course, since the situation can occur where every x from the dividend is associated with *too many* values from the first layer. However, in the approach we describe hereafter, all of the layers are relaxed in order to increase the discrimination capability of the query as well as to handle the different layers of the divisor in an homogeneous way.

5.1. Softened universal quantifier

The idea is to replace the universal quantifier in each layer of the query by a fuzzy relative quantifier [14,15] which allows for some tolerance as to the number of non-associations required. In addition, the softening mechanism is gradual, i.e., the level of satisfaction corresponding to a given proportion p (of associations) lies in the unit interval. Among others, representatives of such a quantifier are *almost none* (*an*), and *as little as possible* (*alap*) modeled as follows:

- $an(p) = 1$ if $p \leqslant lb$, 0 if $p \geqslant ub$, linear in-between (where lb and ub are two constants belonging to $[0,1]$ such that $lb < ub$),
- $alap(p) = 1 - p$.

For instance, with a referential of ten elements and using $lb = 5\%$ and $ub = 25\%$, we get:

$$an\left(\frac{10}{10}\right) = \cdots = an\left(\frac{3}{10}\right) = 0, \quad an\left(\frac{2}{10}\right) = 0.25, \; an\left(\frac{1}{10}\right) = 0.75, \quad an(0) = 1$$
$$alap\left(\frac{10}{10}\right) = 0, \quad alap\left(\frac{9}{10}\right) = 0.1, \; \ldots, \; alap\left(\frac{2}{10}\right) = 0.8, \quad alap\left(\frac{1}{10}\right) = 0.9, \quad alap(0) = 1$$

5.2. Relaxation approach

Let us denote by Q the fuzzy relative quantifier used to relax the query. It is assumed that this quantifier is specified by the user who either chooses it from a list of default ones, or defines it through an appropriate interface. In any case, such an interaction between the system and the user takes place only when the initial user query fails.

The evaluation of the relaxed query is as follows. For each x, one builds a vector $V(x)$ of scores of length n (the number of layers in the divisor) in the following way:

$$\forall i \in [1,\; n], \quad V(x)[i] = Q\left(\frac{k(x,\; i)}{n_i}\right) \tag{21}$$

where $k(x,i) = |\{a \in S_i \text{ such that } (a,x) \in r\}|$ and n_i denotes the cardinality of layer S_i.

The result of the tolerant antidivision query could then be ranked using the lexicographic order on the vectors:

$$x \succ x' \iff V(x) >_{lex} V(x')$$

However, by doing so, one takes into account a layer S_i in the ordering of x even if there exists a layer S_j with $j < i$ such that $Q(k(x,j)/n_j) = 0$, which is somewhat contradictory with the hierarchical behavior expressed by "and if possible"—even though different possible interpretations of this operators may be envisaged. Indeed, we think that two items x and x' such that $V(x) = [0.75,0,1,0.33]$ and $V(x') = [0.75,0,0,0]$ should be considered equivalently satisfactory since both of them perform the same on layer S_1 and totally fail on layer S_2. This can be modeled by modifying vector $V(x)$ into $V'(x)$ in the following way:

$$\forall i \in [1,\; n], \; V'(x)[i] = min_{j=1\ldots i} \; V(x)[j] \tag{22}$$

The ranking formula then becomes:

$$x \succ x' \iff V'(x) >_{lex} V'(x')$$

Example 5. Let us consider relation r of schema (patient, symptom) and the strict antidivision query:

select patient **from** r **group by** patient
having set(symptom) **contains-none** {'headache', 'fever', 'chills'}
and if possible {'back pain', 'chest pain'}
and if possible {'nausea', 'dizziness'}.

Table 1
Extension of relation *r*.

Patient	Symptom
Smith	Headache
Smith	Diarrhea
Smith	Cough
Smith	Nausea
Jones	Fever
Jones	Chest pain
Jones	Sneezing

Evaluating it on the content represented in Table 1, one gets an empty result since both Smith and Jones have a symptom from the first layer. Now, if a tolerant form of the previous query is used, where *contains-none* is replaced by *contains-as-little-as-possible*, we get the vectors:

$$V'(\text{Smith}) = [0.67,\ 1,\ 0.5] \text{ and } V'(\text{Jones}) = [0.67,\ 0.5,\ 1]$$

and the final ordered result is: Smith $\succ$ Jones.

Notice that the use of a fuzzy quantifier leads to dropping the initial ordinal scale for expressing the "quality level" of an element in the result. The lexicographic order which is used for ranking the answers to a relaxed query is more refined than the initial scale in the sense that it involves more levels, but it provides only a relative view of the satisfaction of the elements, whereas the scale used for evaluating an initial query provides an absolute assessment of the "quality" of these elements.

6. Implementation aspects and experimental results

In this section, we first outline some evaluation strategies and algorithms suited to the different types of antidivision queries. Then, we present an experimentation which concerns strict conjunctive antidivision queries, as well as the results obtained.

6.1. Sequential scan of the dividend (SSD)

In this first algorithm, the idea is to access the tuples from the dividend relation (r) "in gusts", i.e., by series of tuples which share the same X-attribute value (in the spirit of what is performed by a *group by* clause). Moreover, the tuples (x, a) inside a group are ranked in increasing order of their A-attribute value. All this is performed by the query:

$$\textbf{select} \ * \ \textbf{from} \ r \ \textbf{order by} \ X,\ A.$$

Thanks to a table which gives, for each value (*val-A*) of the divisor, the layer to which it belongs (*str-A*), one can update the number of values from each layer which are associated with the current element x, while scanning the result of the query above. At the end of a group of tuples from the dividend, one checks the layers in decreasing order of their importance. In the case of CJ queries, this step stops as soon as the current element x is associated with at least one of the values from a layer S_i. Three cases may appear:

1. Element x is associated with none of the values from any layer of the divisor and it gets the preference level l_1.
2. The stop occurs while checking layer S_i whose importance is not maximal ($i > 1$) and x gets the preference level $rv(l_i) = l_{n+2-i}$.
3. The stop occurs while checking layer S_1; element x gets the level l_{n+1} and is thus rejected.

The worst-case data complexity of this algorithm is:

$$cmax(SSD) = n_r \cdot log_2(n_r) + n_r$$

where n_r denotes the cardinality of relation r. The factor $n_r \cdot log_2(n_r)$ comes from the *order-by* clause which sorts relation r. The overall complexity is thus in $\theta(n_r \cdot log_2(n_r))$.

In the case of DJ queries, the scan of the layers stops as soon as a layer S_i is met such that no element of S_i is associated with x.

On the other hand, for Full Discrimination queries, it is necessary to scan all of the layers in order to build the vector $W(x)$ of scores according to formula (14).

In case of an empty set of answers returned by a CJ query, the user may choose to relax his/her query according to the principle described in Section 5, which implies running a modified version of the algorithm suited to CJ queries. The objective is then to build a vector $V'(x)$ according to formula (22). Notice that the scan of the layers can be stopped as soon as a layer S_i such that $Q\left(\frac{k(x,\ i)}{n_i}\right) = 0$ (cf. formula (21)) is met.

6.2. Access guided by the divisor (AGD)

In this second algorithm, instead of scanning the dividend exhaustively and then checking the layers satisfied by a given x by means of the aforementioned table, one first retrieves the X-values from the dividend, and for each such x, one checks the associations with the different layers by means of an SQL query involving the aggregate *count*. Again, in the case of CJ queries, a layer is checked only if the layers of higher importance had none of their values associated with x.

The first step is to retrieve the distinct values of attribute X present in r by means of the query: **select distinct** X **from** r. Then, for each value x returned, one counts the A-values from S_1 which are associated with x in r:

```
select count (*) from r where X =: x and A in
                (select A from s where pref = $l_1$);
```

If the value returned equals zero (for CJ queries), one checks layer S_2 by means of a similar query and so on; otherwise the loop stops. The preference level assigned to x is computed according to the same principle as described in the previous subsection.

The data complexity of this algorithm is upper-bounded by:

$$cmax(AGD) = n_r + |r[X]| * n_r * n_s$$

where n_r (resp. n_s denotes the cardinality of relation r (resp. s).

In the case of DJ queries, the scan of the layers stops as soon as a layer S_i is met such that the value of the aggregate *count* returned by the inner block query equals 0.

On the other hand, for FD queries, it is still necessary to check all of the layers (which means processing n counting queries for a given x) in order to build the vector $W(x)$ of scores according to formula (14).

6.3. Series of regular antidivision queries (SRA)

This third strategy consists of two steps:

1. Process as many regular antidivision queries as there are layers in the divisor.
2. Merge the different results and compute the final preference degrees (for CJ and DJ queries only).

The algorithm has the following general shape:

Step 1: For each layer S_i of the divisor, one processes an antidivision query which retrieves the x's which are associated in r with none of the values from S_i. The layers are examined in decreasing order of their importance.

For CJ queries, an element x is checked only if it belongs to the result associated with the previous layer:

```
create view $T_1$ as select distinct $X$ from $r$ where $X$ not in
          (select $X$ from $r$, $s$ where $r.A = s.B$ and $s$.pref = $l_1$);
for i := 2 to n do
  create view $T_i$ as select $X$ from $T_{i-1}$ where $X$ not in
          (select $X$ from $r$, $s$ where $r.A = s.B$ and $s$.pref = $l_i$);
```

For DJ queries, an element x is checked only if it does not belong to the result associated with the previous layer. Therefore, the query inside the loop must be replaced by:

```
create view $T_i$ as select $X$ from $r$ where $X$ not in
  (select $X$ from $T_{i-1}$) and $X$ not in
  (select $X$ from $r$, $s$ where $r.A = s.B$ and $s$.pref = $l_i$)
```

For FD queries, every layer is checked for every x and Step 1 becomes:

```
for i := 1 to n do
  create view $T_i$ as select distinct $X$ from $r$ where $X$ not in
    (select $X$ from $r$, $s$ where $r.A = s.B$ and $s$.pref = $l_i$);
```

Step 2: For CJ queries, the results of the previous antidivision queries are merged by taking them in decreasing order of the priority of the corresponding layers. An element x which belongs to the result of layer S_i but not to that of layer S_{i+1} gets the preference level l_{n-i+1} in accordance with formula (12). It is assumed hereafter that there exists a table T_{n+1} which is empty. The algorithm corresponding to that merging step is given hereafter.

```
for i := 1 to n do
begin
declare cursor $c_i$ as
  select $X$ from $T_i$ where $X$ not in (select $X$ from $T_{i+1}$);
open $c_i$; fetch $c_i$ into: x;
while not end (active set) do
begin
  result := result + $\{l_{n-i+1}/x\}$;
  fetch $c_i$ into: x;
 end;
end;
```

The data complexity of this algorithm is upper-bounded by:

$$cmax(SRA) = \sum_{i=1}^{n} |T_{i-1}| * n_r * n_s + \sum_{i=1}^{n-1} |T_i| * |T_{i+1}|$$

where $|T_i|$ is the cardinality of $(r \divideontimes S_i) \cap T_{i-1}$ $\forall i = 2,\ldots,n$ and $|T_1|$ is the cardinality of $r \divideontimes S_1$. $|T_0| = 1$ is assumed in the formula.

In the case of DJ queries, the results of the different antivididivision queries are scanned in decreasing order of the priority of the corresponding layers. An element x which belongs to the result of layer S_i gets the degree l_i in accordance with formula (13).

As for FD queries, Step 2 consists in (i) building the vector $W(x)$ for every x present in the union of the results, (ii) ranking the x's by applying the lexicographic order on the vectors.

6.4. Experimental measures

The objectives of the experimentation are:

1. To assess the additional processing cost related to the handling of preferences in strict antidivision queries of type CJ, and
2. to compare the performances of the three algorithms presented above.

The experimentation was performed with the DBMS Oracle™ Enterprise Edition Release 8.0.4.0.0 running on an Alpha server 4000 bi-processor with 1.5 GB memory. Even though the scope of the experiment presented here is still limited and should be extended in the future, it shows an interesting trend as to the cost of such queries.

A generic stratified antidivision query has been run on dividend relations of 300, 3000 and 30,000 tuples, and a divisor including five layers made of respectively, 3, 2, 1, 2 and 2 values.

The query taken as a reference is the analogous antidivision query without preferences, where the divisor is made of the sole first layer (which corresponds to a "hard constraint" as mentioned before). The reference query has been evaluated using two methods:

- Algorithm AGD without preferences, denoted by REF2.
- First step of algorithm SRA with one layer only, denoted by REF3.

Notice that algorithm SSD without preferences would have the same complexity as SSD itself since it would also involve an exhaustive scan of the dividend (this is why there is no reference method "REF1"). Moreover:

- We used synthetic data generated in such a way that the selectivity of each value v_i from the divisor relatively to any x from the dividend is equal to 25% (for a given value v_i from the divisor and a given x from the dividend, tuple (x, v_i) has one chance out of four to be present in the dividend).
- Each algorithm was run eight times, so as to avoid any bias induced by the load of the machine.
- The time unit equals 1/60 s.

We tested different variants of algorithm SRA where set differences in both steps are performed either by means of the operator *minus* (instead of *not in*) or are expressed by an outer join. It appears that the most efficient expression is that where the set differences are based on: (i) the operator *minus* in Step 1, (ii) an outer join in Step 2. The results discussed further consider this expression for SRA. The results reported in Table 2 show that:

- Among the reference methods for non-stratified antidivisions, REF3 is much more efficient than REF2.
- The performances of REF2, AGD, and SSD vary linearly w.r.t. the size of the dividend. As to REF3 and SRA, their complexity is less than linear.
- The best algorithm for stratified antidivisions is SRA, which is significantly better than AGD, itself much more efficient than SSD.

Table 2
Experimental results.

Size of the dividend	300	3000	30,000
REF2	41.4	400.7	4055
REF3	13.2	81.4	760.2
SSD	108.6	960.5	10,418
AGD	54.2	645.2	6315
SRA	106	375.1	4353
Number of answers (top layer)	37	427	4365

- The extra cost of SRA w.r.t. the most efficient reference algorithm, namely REF3, is still rather important (multiplicative factor between 4.6 and 8).

What all these measures show was somewhat predictable: the best way to process an antidivision query (stratified or not) is to express it by means of a single query that can be efficiently handled by the optimizor of the system, and not by external programs which induce a more or less important overhead. The extra cost attached to SRA w.r.t. REF3, is also explainable by the fact that SRA processes five regular antidivision queries—one for each layer—instead of one for REF3, and then has to merge the results of these queries. If the stratified antidivision functionality were to be integrated into a commercial DBMS, it is quite clear that it would have to be handled by the optimizor at an internal level, and processed as *one* query involving a new type of "having" clause, as in expression (9).

7. Conclusion

In this paper, preferences for a family of queries stemming from the relational antidivision have been considered. The key idea is to make use of a divisor made of a hierarchy of subsets of elements. So doing, the result is no longer a flat set but a list of items provided with a level of satisfaction.

Three uses of the hierarchy have been investigated, which led to three distinct semantics of the corresponding queries. Moreover, a characterization of the result produced in all cases has been given: it is an "antiquotient", i.e., the largest relation whose product with the divisor remains included in the complement of the dividend. The case where a conjunctive stratified antidivision query returns an empty set of answers has also been dealt with, leading to an approach aimed at relaxing such queries. The idea consists in replacing the crisp quantifier *none* underlying strict antidivision queries by a more tolerant fuzzy quantifier such as *almost none*. Then, the elements of the result are not associated with a symbolic satisfaction level anymore, but each element is attached a vector of scores, which allows to rank the result by applying the lexicographic order on these vectors.

Besides, some experimental measures have been carried out in order to assess the feasibility of stratified antidivision queries. Even though these measures still need to be completed—only one of the three semantics has been considered so far—, they show that the additional cost induced by the stratified nature of the divisor is quite high (factor 4–8 w.r.t. a classical antidivision) but that the overall processing time is still acceptable for medium-sized dividend relations. To reach better performances, it would be of course necessary to integrate the new operator into the processing engine of the system, so as to benefit from a real internal optimization, instead of processing stratified antidivision queries externally, as we did here.

We are now planning to: (i) design algorithms for implementing antidivision queries of types DJ and FD, (ii) make experiments in order to evaluate these algorithms, as we did for CJ queries, (iii) check whether the results obtained are confirmed when another DBMS (e.g. PostgresQL or MySQL) is used, (iv) investigate some possible uses of stratified antidivision queries in an information retrieval context, as we did for fuzzy division queries [16].

References

[1] A. Hadjali, S. Kaci, H. Prade, Database preference queries – a possibilistic logic approach with symbolic priorities, in: Proceedings of the 5th Symposium on the Foundations of Information and Knowledge Systems (FoIKS'08), 2008, pp. 291–310.
[2] N. Bruno, S. Chaudhuri, L. Gravano, Top-*k* selection queries over relational databases: mapping strategies and performance evaluation, ACM Transactions on Database Systems 27 (2) (2002) 153–187.
[3] P. Bosc, O. Pivert, SQLf: a relational database language for fuzzy querying, IEEE Transactions on Fuzzy Systems 3 (1995) 1–17.
[4] R. Agrawal, E. Wimmers, A framework for expressing and combining preferences, in: Proceedings of SIGMOD'00, 2000, pp. 297–306.
[5] P. Fishburn, Preferences structures and their numerical representation, Theoretical Computer Science 217 (1999) 359–383.
[6] J. Chomicki, Preference formulas in relational queries, ACM Transactions on Database Systems 28 (4) (2003) 427–466.
[7] M. Lacroix, P. Lavency, Preferences: putting more knowledge into queries, in: Proceedings of of the 13th Conference on Very Large Data Bases (VLDB'87), 1987, pp. 217–225.
[8] R.I. Brafman, C. Domshlak, Database Preference Queries Revisited, in: Technical Report TR2004-1934, Cornell University, Computing and Information Science, 2004.
[9] W. Kießling, G. Köstler, Preference SQL — design, implementation, experiences, in: Proceedings of the 28th Conference on Very Large Data Bases (VLDB'02), 2002, pp. 990–1001.
[10] S. Bőrzsőnyi, D. Kossmann, K. Stocker, The skyline operator, in: Proceedings of the 17th IEEE International Conference on Data Engineering (ICDE'01), 2001, pp. 421–430.

[11] P. Bosc, O. Pivert, On a parameterized antidivision operator for database flexible querying, in: Proceedings of the 19th Conference on Database and Expert Systems Applications (DEXA'08), 2008, pp. 652–659.
[12] P. Bosc, O. Pivert, D. Rocacher, About quotient and division of crisp and fuzzy relations, Journal of Intelligent Information Systems 29 (2) (2007) 185–210.
[13] P. Bosc, O. Pivert, D. Rocacher, Characterizing the result of the division of fuzzy relations, International Journal of Approximate Reasoning 45 (3) (2007) 511–530.
[14] E. Kerre, Y. Liu, An overview of fuzzy quantifiers – interpretations, Fuzzy Sets and Systems 95 (1) (1998) 1–21.
[15] L. Zadeh, A computational approach to fuzzy quantifiers in natural languages, Computer Mathematics with Applications 9 (1983) 149–184.
[16] P. Bosc, V. Claveau, O. Pivert, L. Ughetto, Graded-inclusion-based information retrieval systems, in: M. Boughanem, C. Berrut, J. Mothe, C. Soulé-Dupuy (Eds.), Proceedings of ECIR'09, Lecture Notes in Computer Science, vol. 5478, Springer, 2009, pp. 252–263.