J Intell Inf Syst (2011) 37:315–331
DOI 10.1007/s10844-010-0147-y

On three classes of division queries involving ordinal preferences

Patrick Bosc · Olivier Pivert · Olivier Soufflet

Received: 1 December 2009 / Revised: 3 September 2010 / Accepted: 2 December 2010 /
Published online: 17 December 2010

Abstract In this paper, we are interested in taking preferences into account for a family of queries inspired by the relational division. A division query aims at retrieving the elements associated with a specified set of values and usually the results remain not discriminated. So, 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 in terms of a quotient is studied. Special attention is paid to the implementation of such extended division queries using a regular database management system along which some experiments to support the feasibility of the approach. Moreover, the issue of empty or overabundant answers is dealt with.

Keywords Relational database · Division · Preference · Ordinal scale · Empty answers · Overabundant answers

1 Introduction

Queries including preferences have received growing interest during the last decade (Bőrzsőnyi et al. 2001; Bosc et al. 2007; Bruno et al. 2002; Chomicki 2003; Dubois and Prade 1996; Hadjali et al. 2008; Kießling and Köstler 2002; Lacroix and Lavency

P. Bosc (✉) · O. Pivert · O. Soufflet
Irisa—Enssat, University of Rennes 1, Technopole Anticipa,
22305, Lannion Cedex, France
e-mail: bosc@enssat.fr, Patrick.Bosc@enssat.fr

O. Pivert
e-mail: pivert@enssat.fr

O. Soufflet
e-mail: soufflet@enssat.fr

1987). One of their main advantages is to allow for some discrimination among the elements of their result (which is no longer a flat set) thanks to the compliance with the specified preferences. However, up to now, most of the research works 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. Moreover, a purely ordinal framework is chosen and the user has only to deal with an ordinal scale, which we think to be not too demanding. Lastly, taking preferences into account will allow for keeping only the best k answers, in the spirit of top-k queries (Bruno et al. 2002). Knowing that a regular division delivers a non-discriminated set of elements, the idea is here to call on preferences related to the divisor. In the majority of previous works, preferences are assigned to tuples individually, in particular in Bosc et al. (2007) where the divisor relation is made of weighted tuples stemming from some fuzzy selection condition (e.g., medium-priced wines or fairly ancient monuments). On the contrary, this paper investigates another line based on preferences in terms of a hierarchy of subsets of tuples and we will use the words "stratified divisor/division". Consequently, an element x of the dividend will be all the more acceptable as it is connected with a large number of the subsets (S_i's) defined over the divisor. In fact, different roles can be allotted to the divisor when it is described as a hierarchical set. Three of them, which seem to be useful and natural are envisaged:

1. a direct extension of the division in a conjunctive way, where the first layer of the divisor is mandatory and the following ones are considered only desirable: find the elements x connected with S_1 *and if possible ... and if possible* S_n (which has some relationship with bipolarity (Dubois and Prade 2008)),
2. a disjunctive view where x is all the more satisfactory as it is connected with all the values of a highly preferred (sub)set of the divisor: find the elements x connected with S_1 *or else ... or else* S_n,
3. an intermediate approach where x is all the more highly ranked as it is connected with numerous and preferred (sub)sets of the divisor: find the elements x connected with S_1 *and-or ... and-or* S_n.

As an example, let us consider the case of a user looking for wine shops offering Saint Emilion Grand Cru, Pomerol and Margaux and if possible Gewurztraminer Vendanges Tardives and Chablis Premier Cru and if possible Pommard and Chambertin. Of course, "or else" or "and-or" could be used as well.

The rest of the paper is organized as follows. Section 2 is devoted to some reminders about the division and the expression of the extended division queries dealt with here. A detailed presentation (in terms of syntax and semantics) of the three types of queries considered is made in Section 3. In Section 4, it is shown that the result returned in each case can be characterized as a quotient, i.e., a largest relation according to a given inclusion constraint. Implementation issues are discussed in Section 5 and some experiments are reported in order to assess the performances of such queries using a commercially available DBMS. Section 6 is devoted to the case where a conjunctive query yields an empty answer or a disjunctive query returns too many answers. The conclusion summarizes the contributions of the paper and opens some lines for future research.

2 Reminders and syntax

2.1 Some reminders on the division

The division is a well-established high-level operator of the relational algebra. It applies to a couple of relations and it imposes some constraint in terms of associations of values between them, which turns out to be a useful concise way for expressing this type of need. 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 X, A and B are sets of attributes taking their values in corresponding domains of values (wine names, colors, etc.). In addition, A and B are assumed to be compatible which means that they share the same domain(s) of values (they serve as a semantic link between the dividend and the divisor relations). The division of relation r by relation s is defined as:

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

or equivalently as:

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

where $r[X]$ denotes the projection of r over X (in other words, the set of values taken by X in r) and $\Omega_r(x) = \{a \mid (a,\ x) \in r\}$. 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, while the result q of the division of the two integers m and n, is such that: i) $q * n \leq m$, ii) $\forall q' > q, q' * n > m$, the resulting relation *res* obtained with expression (1) has the double characteristic:

$$\forall t \in res,\ s \times \{t\} \subseteq r \tag{3}$$

$$\forall t \notin res,\ s \times \{t\} \nsubseteq r \tag{4}$$

$\times$ denoting the Cartesian product of relations.

Expressions (3) and (4) express the fact that relation *res* is a quotient, i.e., the largest relation whose Cartesian product with the divisor returns a result which is included in the dividend.

Example 1 Let us take a database involving the two relations order (o) and product (p) with respective schemas $O(np, store, qty)$ and $P(np, price)$. Tuples (n, s, q) of o and (n, pr) of p state that product n has been ordered from store s in quantity q and that its price is pr. Retrieving the stores which have been ordered all the products priced under \$127 in a quantity greater than 35, can be expressed using a division as: $div(o\text{-}g35, p\text{-}u127, np, np)$ where relation $o\text{-}g35$ corresponds to pairs (n, s) such that product n has been ordered from store s in a quantity over 35 and relation $p\text{-}u127$ gathers products whose price is under \$127. Here, the formal dividend (respectively

divisor) relation $r(A, X)$ (respectively $s(B)$) is $o\text{-}g35(np, store)$ (respectively $p\text{-}u127(np)$). From the extensions of relations $o\text{-}g35$ and $p\text{-}u127$ given hereafter:

$$o\text{-}g35 = \{(p_{15}, s_{32}), (p_{12}, s_{32}), (p_{34}, s_{32}), (p_{26}, s_{32}), (p_{12}, s_7), (p_{26}, s_7), (p_{15}, s_{19}), (p_{12}, s_{19}), (p_{26}, s_{19})\}$$

$$p\text{-}u127 = \{p_{15}, p_{12}, p_{26}\}$$

the division returns $\{s_{32}, s_{19}\}$, which satisfies (3) and (4).

2.2 General syntactical framework

The three types of queries studied later are expressed in an SQL-like style where the dividend may be any intermediate relation and the stratified divisor is either explicitly given by the user (case which will be considered further) or results from a series of subqueries. Usually, the division of relation r of schema $R(A, X)$ by relation s of schema $S(B)$ is expressed using a partitioning mechanism and we suggest a similar expression here:

select top k X from r [**where** condition] **group by** X

having set(A) **contains** $\{v_{1,1}, ..., v_{1,j_1}\}$ *connector ... connector* $\{v_{n,1}, ..., v_{n,j_n}\}$

where *connector* stands for either "and if possible", or "or else", or "and-or". Such a statement induces an ordering over the divisor, namely $(S_1 = \{v_{1,1}, ..., v_{1,j_1}\}) \succ ... \succ (S_n = \{v_{n,1}, ..., v_{n,j_n}\})$ where $a \succ b$ denotes the preference of a over b. Associated with this preference relation is an ordinal scale L with labels l_i's (such that $l_1 > ... > l_n > l_{n+1}$) which will be used to assign levels of satisfaction to elements pertaining to the result of a stratified division (l_1 corresponds to the highest satisfaction and l_{n+1} expresses rejection and they are the counterparts of 1 and 0 in the unit interval when a numeric framework is used, e.g., fuzzy sets). Coming back to the context evoked in the introduction, an example of such a query could be:

select top 6 shop-name **from** wineshops **group by** shop-name

having set(wine) **contains** {Saint Emilion Grand Cru, Pomerol, Margaux}

and if possible {Gewurztraminer Vendanges Tardives, Chablis Premier Cru}

and if possible {Pommard, Chambertin}

along with the scale $L = l_1 > l_2 > l_3 > l_4$. In such a context, shops selling all the wines receive the grade l_1, those which sell all the wines except either Pommard, or Chambertin are assigned the grade l_2, those which sell Saint Emilion Grand Cru, Pomerol, Margaux and not at least one of the wines of the second layer (Gewurztraminer Vendanges Tardives or Chablis Premier Cru) get the grade l_3 and those which do not sell either Saint Emilion Grand Cru, or Pomerol, or Margaux are assigned the last grade of the scale, which means that are are rejected.

3 Stratified division-like queries

The three types of queries called Q1, Q2 and Q3 are the following:

- find the best k elements associated with S_1 and if possible ... and if possible S_n (Q1),
- find the best k elements associated with S_1 or else ... or else S_n (Q2),
- find the best k elements associated with S_1 and-or ... and-or S_n (Q3).

3.1 Conjunctive queries (Q1)

As to Q1 queries, to be qualified, an element x must be connected with all the elements having the maximal importance (S_1). In addition, as soon as it is not connected with all 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 associated with all the values of the succession of sets S_1 to S_i where i is large (if possible n for "perfection"). In other words, x is preferred to y if x is associated with all the values of the sets S_1 to S_p and y is associated with a shorter list of sets. More formally, let us denote:

$$I(x) = \{i \mid S_i \nsubseteq \Omega_r(x)\} \text{ and if } I(x) \neq \emptyset,\ imin(x) = \min(I(x)).$$

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

$$sat(x) = l_1 \text{ if } I(x) = \emptyset,\ l_{n+2-imin(x)} \text{ otherwise.} \tag{5}$$

So doing, the satisfaction is seen as a composition of the results of the division of the dividend with each of the layers of the divisor.

An alternative way of modeling conjunctive queries stems from the extension of formula (2) by: (i) dealing with the preferences applying to the divisor and (ii) using a symbolic implication. Indeed, we will use an augmented relational framework where each tuple of a relation *rel* is assigned a (symbolic) level of preference taken from the scale L, denoted by $pref_{\text{rel}}(t)$ and any tuple can be written $pref_{\text{rel}}(t)/t$. Since the dividend relation is not concerned with explicit preferences, its tuples are assigned the maximal level l_1 while the tuples which are absent are (virtually) assigned the lowest level l_{n+1}. The level of preference attached to a tuple of the divisor is directly induced by the place of the corresponding element in the hierarchy provided by the user. As to the implication, it can be chosen among fuzzy implications (Fodor and Yager 1999) provided that: i) it is compatible with an ordinal context, and ii) it conveys the semantics of importance associated with the layered divisor. It turns out that the implication (derived from the regular material implication) defined as:

$$s1 \Rightarrow_o s2 = \max(rev(s1), s2)$$

where $s1, s2$ are two symbols of an ordinal scale and *rev* denotes order reversal, meets the previous requirements. Let us remark that with the scale $L = l_1 > \ldots > l_{n+1}$, one

has: $rev(l_i) = l_{n+2-i}$. So equipped, if V denotes the values of the divisor, the stratified division is defined as follows:

$$sat(x) = \min_{v \in V} pref_s(v) \Rightarrow_o pref_r(v, x). \tag{6}$$

As a consequence, if x is associated with all of the values of the entire divisor, the maximal level of preference l_1 is obtained and as soon as an association $\langle v, x \rangle$ is missing, the level of preference of x decreases all the more as v is highly preferred.

Example 2 Let us take the divisor: $\{a, b\} \succ c \succ \{d, e\}$ and the dividend relation:

$$r = \{(a, x_1), (b, x_1), (c, x_1), (d, x_1), (a, x_2), (b, x_2), \\ (a, x_3), (b, x_3), (d, x_3), (e, x_3), (e, x_4), (b, x_5), (d, x_5)\}.$$

Due to the presence of three layers in the divisor, the scale used to assess the elements of the result is $L = l_1 > l_2 > l_3 > l_4$. In addition, one has: $n = 3$, $I(x_1) = \{3\}$, $imin(x_1) = 3$ and according to (5) $sat(x_1) = l_2$; similarly, $sat(x_2) = sat(x_3) = l_3$, $sat(x_4) = sat(x_5) = l_4$ and the final result is: $x_1 \succ \{x_2, x_3\}$.

Now, referring to (6), one has:

$$\begin{aligned} sat(x_1) = \min(&pref_s(a) \Rightarrow_o pref_r(a, x_1), pref_s(b) \Rightarrow_o pref_r(b, x_1), \\ &pref_s(c) \Rightarrow_o pref_r(c, x_1), pref_s(d) \Rightarrow_o pref_r(d, x_1), \\ &pref_s(e) \Rightarrow_o pref_r(e, x_1)) \\ = \min(&l_1 \Rightarrow_o l_1, l_1 \Rightarrow_o l_1, l_2 \Rightarrow_o l_1, l_3 \Rightarrow_o l_1, l_3 \Rightarrow_o l_4) \\ = \min(&l_1, l_1, l_1, l_1, l_2) = l_2. \end{aligned}$$

Similarly, one gets: $sat(x_2) = \min(l_1, l_1, l_3, l_2, l_2) = l_3$, $sat(x_3) = \min(l_1, l_1, l_3, l_1, l_1) = l_3$, $sat(x_4) = \min(l_4, l_4, l_3, l_2, l_1) = l_4$, $sat(x_5) = \min(l_4, l_1, l_3, l_1, l_2) = l_4$.

As expected, it can be observed that these results coincide with those obtained previously with formula (5), i.e.: $x_1 \succ \{x_2, x_3\}$. Let us notice that x_4 and x_5 do not appear in the result since they got the grade l_4, expressing rejection (they are totally unsatisfactory since they are not associated with (at least) a and b in the dividend).

3.2 Disjunctive queries (Q2)

While Q1 queries have a conjunctive behavior, Q2 queries are meant disjunctive instead, and S_1 is no longer a mandatory 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 all 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 all the values of the set S_k (and no S_j with $j < k$) and y is associated with S_p (and no S_m with $m < p$) and $p > k$. Let us denote:

$$I'(x) = \{i \mid S_i \subseteq \Omega_r(x)\} \text{ and if } I'(x) \neq \emptyset, ipmin(x) = \min(I'(x)).$$

The grade of satisfaction attached to an element x is expressed as:

$$sat(x) = l_{n+1} \text{ if } I'(x) = \emptyset,\ l_{ipmin(x)} \text{ otherwise.} \tag{7}$$

The satisfaction is still a combination of the results of the division of the dividend with each of the layers of the divisor.

Example 3 Let us take the divisor: $\{a, b\} \succ c \succ \{d, e\}$ and the dividend:

$$r = \{(a, x_1), (d, x_1), (e, x_1), (c, x_2), (d, x_2), (e, x_2),$$
$$(a, x_3), (b, x_3), (c, x_3), (d, x_3), (e, x_3), (c, x_4), (b, x_5)\}.$$

One has: $n = 3$, $I'(x_2) = \{2, 3\}$, $ipmin(x_2) = 2$ and $sat(x_2) = l_2$; similarly, $sat(x_1) = l_3$, $sat(x_3) = l_1$, $sat(x_4) = l_2$, $sat(x_5) = l_4$ and then the final result is: $x_3 \succ \{x_2, x_4\} \succ x_1$. For the reason evoked in Example 2, x_5 is absent from the result.

3.3 Full discrimination-based queries (Q3)

Queries of type Q3 are designed so as to counter the common disability of Q1 and Q2 queries in distinguishing between elements which are equally ranked because additional associations are not taken into account (e.g., x_2 and x_4 in the above example). So, the principle for interpreting Q3 queries is to consider all the layers for which a complete association occurs. An element is all the more preferred as it is associated with a set S_i highly preferred and this same point of view applies to break ties. In this case, the grade of satisfaction for x may be expressed using a vector $V(x)$ of dimension n where $V(x)[i] = 1$ if x is associated with all the values of S_i, 0 otherwise. Ordering elements boils down to comparing such vectors according to the lexicographic order ($\succ_{\text{lex}}$):

$$x \succ y \Leftrightarrow V(x) \succ_{\text{lex}} V(y) \Leftrightarrow \exists k \in [1,\ n] \text{ such that}$$
$$\forall j < k,\ V_j(x) = V_j(y) \text{ and } V_k(x) > V_k(y).$$

In this view, the scale L is not used directly even if the order of the elements of the vectors reflects it in the sense that, if $i < j$, $V_i(x)$ is more important than $V_j(x)$ as $l_i > l_j$. It is however possible to use a scale to perform the comparison of elements in the context of Q3 queries, even if this scale is not the initial one and is much larger (2^n levels instead of $(n + 1)$ in the original one). Let us consider the function which maps a vector V into an integer score s as follows:

$$sat(x) = \sum_{i=1..n} V_i(x) * 2^{n-i}. \tag{8}$$

It is straightforward to prove that the preference of x over y as defined before is equivalent to $sat(x) > sat(y)$. In addition, it turns out that dealing with such scores is easier than comparing vectors from a calculus point of view.

Example 4 Let us take the divisor: $\{a, b\} \succ c \succ \{d, e\}$ and the dividend:

$$r = \{(a, x_1), (d, x_1), (e, x_1), (c, x_2), (d, x_2), (e, x_2),$$
$$(a, x_3), (b, x_3), (d, x_3), (c, x_4), (b, x_5)\}.$$

One has: $V(x_1)=(0,0,1)$, $V(x_2)=(0,1,1)$, $V(x_3)=(1,0,0)$, $V(x_4)=(0,1,0)$, $V(x_5)=(0,0,0)$, $sat(x_1)=1$, $sat(x_2)=3$, $sat(x_3)=4$, $sat(x_4)=2$, $sat(x_5)=0$ and then the final result is: $x_3 \succ x_2 \succ x_4 \succ x_1$. Here again, x_5 is discarded since it is not at all satisfactory.

3.4 Relationship with the lexicographic order

As it has been indicated, Q3 queries potentially use all the layers of the divisor in order to discriminate between the elements returned by a query and is founded on the lexicographic order. It turns out that the other two types of queries can also be situated in this setting. Interpreting any Q1 or Q2 query can be done through a vector accounting for the association with the values of each complete stratum of the divisor, as it is done for Q3 queries. For Q1 queries:

$$sat(x) = l_{n-k+2} \text{ where } k \text{ is the smallest index s.t. } V_k(x) = 0 \ (n+1 \text{ if none}).$$

In other words, the comparison of x and y can be based on a modified vector V' obtained from V by propagating the first 0 to the right, which yields the equivalence $(sat(x) > sat(y)) \Leftrightarrow (V'(x) \succ_{\text{lex}} V'(y))$. Similarly, for Q2 queries:

$$sat(x) = l_k \text{ where } k \text{ is the smallest indice s.t. } V_k(x) = 1 \ (k = n+1 \text{ if none}).$$

Here also, the comparison of x and y can be based on a modified vector V'' obtained from V by propagating the first 1 to the right, which yields the equivalence $(sat(x) > sat(y)) \Leftrightarrow (V''(x) \succ_{\text{lex}} V''(y))$.

Example 5 Let us come back to the data of Example 4, in particular the vectors V obtained. From them, we obtain the modified vectors: $V'(x_1)=(0,0,0)$, $V'(x_2)=(0,0,0)$, $V'(x_3)=(1,0,0)$, $V'(x_4)=(0,0,0)$, $V'(x_5)=(0,0,0)$ which leads to keeping only x_3 as the result of query Q1. Similarly, the modified vectors V'' are: $V''(x_1)=(0,0,1)$, $V''(x_2)=(0,1,1)$, $V''(x_3)=(1,1,1)$, $V''(x_4)=(0,1,1)$, $V''(x_5)=(0,0,0)$ and the result of Q2 is: $x_3 \succ \{x_2, x_4\} \succ x_1$. These two results are exactly those obtained with formulas (5), (6) and (7).

4 Property of quotient of the result delivered

We now give a characterization as a quotient (in the spirit of formulas (3) and (4)) of the result delivered by the three types of division queries. Here, we have of course to consider the satisfaction level (l_i) assigned to an element x of the result. For Q1 queries, if it is assumed that x is assigned the grade l_i ($i \in [1,\ n]$) the following property holds:

$$\forall k \in [1,\ n-i+1],\ S_k \times \{x\} \subseteq r \tag{9}$$

$$S_{n-i+2} \times \{x\} \not\subseteq r. \tag{10}$$

Formulas (9) and (10) convey both a maximality requirement and a containment constraint between the product of the result with the divisor on the one hand and the

dividend on the other hand. In addition, any value x which is not (at all) in the result (grade of satisfaction l_{n+1}) is such that: $S_1 \times \{x\} \not\subseteq r$, which expresses that its grade cannot even be increased from l_{n+1} to l_n.

Example 6 Let us take the data of Example 2 where $n = 3$ and $sat(x_1) = l_2$. One has: $S_1 \times \{x_1\} = \{(a, x_1), (b, x_1)\} \subseteq r$, and $S_2 \times \{x_1\} = \{(c, x_1)\} \subseteq r$, whereas $S_3 \times \{x_1\} = \{(d, x_1), (e, x_1)\} \not\subseteq r$. The same situation occurs for x_2.

Similarly, for x_3, one has: $S_1 \times \{x_3\} = \{(a, x_3), (b, x_3)\} \subseteq r$, whereas $S_2 \times \{x_3\} = \{(c, x_3)\} \not\subseteq r$.

Last, for $i = 4$ and 5: $S_1 \times \{x_i\} = \{(a, x_i), (b, x_i)\} \not\subseteq r$.

This illustrates the validity of formulas (9) and (10).

Similarly, for Q2 queries, if x got the grade of satisfaction l_i, letting $S_{n+1} = \emptyset$, one has:

$$S_i \times \{x\} \subseteq r \tag{11}$$

$$\forall k \in [1, i-1],\ S_k \times \{x\} \not\subseteq r. \tag{12}$$

In other words, i corresponds to the index of the first layer S_i of the divisor such that x is associated with every value of S_i in the dividend.

Example 7 Let us take the data of Example 3 where $n = 3$. One has: $sat(x_1) = l_3$ and $S_3 \times \{x_1\} = \{(d, x_1), (e, x_1)\} \subseteq r$, whereas both $S_2 \times \{x_1\} = \{(c, x_1)\} \not\subseteq r$ and $S_1 \times \{x_1\} = \{(a, x_1), (b, x_1)\} \not\subseteq r$.

Similarly, for $i = 2$ and 4, $sat(x_i) = l_2$ and $S_2 \times \{x_i\} = \{(c, x_i)\} \subseteq r$ and $S_1 \times \{x_i\} = \{(a, x_i), (b, x_i)\} \not\subseteq r$.

For x_3, $sat(x_3) = l_1$ and $S_1 \times \{x_3\} = \{(a, x_3), (b, x_3)\} \subseteq r$.

Last, $sat(x_5) = l_4$ and $S_4 \times \{x_5\} = \emptyset \subseteq r$ on the one hand, and $S_3 \times \{x_5\} = \{(d, x_5), (e, x_5)\} \not\subseteq r$, $S_2 \times \{x_5\} = \{(c, x_5)\} \not\subseteq r$, $S_1 \times \{x_5\} = \{(a, x_5), (b, x_53)\} \not\subseteq r$ on the other hand.

It turns out that formulas (11) and (12) hold.

As to Q3 queries, recall that the grade of satisfaction of x is basically expressed as a function of the values of the vector V stating whether x is connected or not with all the values of the different layers of the divisor (cf. formula (8)). The following property holds:

$$\forall i \in [1,\ n] \text{ such that } V_i(x) = 1,\ S_i \times \{x\} \subseteq r \tag{13}$$

$$\forall i \in [1,\ n] \text{ such that } V_i(x) = 0,\ S_i \times \{x\} \not\subseteq r \tag{14}$$

which means that if the grade $sat(x)$ is increased, some inclusion constraint(s) of type (13) will be violated.

Example 8 Let us come back to Example 4. $V(x_1) = (0, 0, 1)$ and $S_1 \times \{x_1\} = \{(a, x_1), (b, x_1)\} \not\subseteq r$, $S_2 \times \{x_1\} = \{(c, x_1)\} \not\subseteq r$ and $S_3 \times \{x_1\} = \{(d, x_1), (e, x_1)\} \subseteq r$. $V(x_2) = (0, 1, 1)$ and $S_1 \times \{x_2\} = \{(a, x_2), (b, x_2)\} \not\subseteq r$, $S_2 \times \{x_2\} = \{(c, x_2)\} \subseteq r$ and $S_3 \times \{x_2\} = \{(d, x_2), (e, x_2)\} \subseteq r$.

$V(x_3) = (1, 0, 0)$ and $S_1 \times \{x_3\} = \{(a, x_3), (b, x_3)\} \subseteq r$, $S_2 \times \{x_3\} = \{(c, x_3)\} \not\subseteq r$ and $S_3 \times \{x_3\} = \{(d, x_3), (e, x_3)\} \not\subseteq r$.

$V(x_4) = (0, 1, 0)$ and $S_1 \times \{x_4\} = \{(a, x_4), (b, x_4)\} \not\subseteq r$, $S_2 \times \{x_4\} = \{(c, x_4)\} \subseteq r$ and $S_3 \times \{x_4\} = \{(d, x_4), (e, x_4)\} \not\subseteq r$.

$V(x_5) = (0, 0, 0)$ and $S_1 \times \{x_5\} = \{(a, x_5), (b, x_5)\} \not\subseteq r$, $S_2 \times \{x_5\} = \{(c, x_5)\} \not\subseteq r$ and $S_3 \times \{x_5\} = \{(d, x_5), (e, x_5)\} \not\subseteq r$.

Here, we have the illustration of the fact that formulas (13) and (14) hold.

5 Implementation and experimental results

In this section, we describe how queries Q1–Q3 can be implemented using a commercial DBMS. Moreover, we report some experiments made in order to assess the extra cost induced by the handling of preferences in division queries.

5.1 Principle of the algorithms

The general principle retained for implementing the previous queries is to use regular SQL queries for accessing the data, embedded in programs in charge notably of computing the grade of satisfaction (denoted by *sat* in the algorithm hereafter) assigned to each element of the result. The algorithms proposed have the following two characteristics: i) they are inspired by the usual way of expressing a division query by means of a counting, and ii) one takes advantage of the stratification of the divisor so as to first access certain tuples of the dividend. In the algorithm hereafter, *specific-condition* and *specific-conclusion* depend on the type of query under consideration (Q1, Q2 or Q3). The layers are scanned in decreasing order of importance (S_1 to S_n), which really matters only for Q1 and Q2 queries. Moreover, *specific-condition* and *specific-conclusion* are so that for a Q1 (resp. Q2) query, the loop stops as soon as an exhaustive association does not hold (resp. holds) and the grade of satisfaction is computed accordingly. For a Q3 query, all the layers must be examined.

```
declare c1 cursor for select distinct X from r;
open c1; fetch c1 into :x;
while not end-of-c1 do
   i := 1;
   while specific-condition do
     select count(*) into :nb from r where X = :x and A in Si;
     if :nb = card(Si) then specific-conclusion endif; i := i + 1;
   endwhile;
   if appropriate then res := res + {sat/x} endif;
```

```
   fetch c1 into :x;
endwhile;
close c1;
```

5.2 Experiments

The final objective of the experiments is to assess the extra cost to pay when dealing with preference division queries. Queries Q1–Q3 are evaluated with dividend relations of different sizes (300, 3,000 and 30,000 tuples), with and without index, with a same divisor made of five layers involving 3, 2, 1, 2 and 2 values. A reference query is taken, namely a division without preferences where the divisor is made of the ten tuples of the five layers of the previous stratified divisor. The DBMS used is Oracle™8.0 with a 2-processor Alpha™ server 4000 and a 1.5 Gb main memory. The results obtained are gathered in Table 1, knowing that:

- we used synthetic data and the selectivity of each value a from the divisor relatively to any x from the dividend was set to 75%,
- each algorithm is run eight times in order to avoid any bias,
- the results reported in the case of an index concern an index on X in r,
- the size of the result is 2 (resp. 15, 234) for a dividend of 300 (resp. 3,000, 30,000) tuples,
- the time unit corresponds to 1/60 s.

The analysis of these results leads to the following three main comments:

- in the absence of an index, the cost of all the queries is non linear (in the size of the dividend relation r), while on the contrary, it is linear when r is indexed on X. This is clearly due to the construction of the cursor (c1) which requires a sort of relation r when no index on X is available;
- the cost of queries Q1 and Q2 are roughly the same and in the range of one half of that of query Q3. Unsurprisingly, this means that for Q1 and Q2, in the average, for a given value x of the dividend the SQL query "select count(*) ..." is performed for one half of the layers, while this query is run for all of the layers for Q3;
- in the presence (resp. the absence) of an index, the extra cost induced by preferences is in the range of 40–50% (resp. 50–100%) for queries Q1 and Q2, and roughly 200% (resp. 300%) for queries Q3. This is not negligible, but this is clearly more acceptable with an index on X.

These first results are encouraging, even if they must be completed to reach definitive conclusions as to the way of developing tools on top of existing DBMS's for processing such queries.

Table 1 Experimental results

Size (dividend)	300		3,000		30,000	
	no idx	idx	no idx	idx	no idx	idx
Reference query	17	16	267	144	16,930	1,556
Query Q1	24	22	529	221	30,948	2,231
Query Q2	28	25	447	220	30,012	2,266
Query Q3	55	49	1,074	480	74,471	4,754

6 Empty or overabundant answers to stratified division queries

It can be observed that conjunctive queries (Q1) are somewhat demanding since an element x which is not associated with all the values of the first stratum is discarded (whatever its associations with the next strata). On the other hand, disjunctive queries (Q2) are quite permissive since one looks for the most highly desired layer for which the association holds, here also whatever the validity of the associations with the values of the other layers. As a consequence, there is a risk on the one hand for conjunctive queries to return an empty answer and on the other hand for disjunctive queries to deliver a too large (thus difficult to manage) result. In both cases the calibration specified by the user (*top k* clause) is not met. Of course, several approaches may be envisaged to overcome these situations. One would be to replace the unsatisfactory query (in terms of the size of its result) by the corresponding Q3 query in order to take advantage of a finer discrimination scale. However, one might argue that we have shifted too far from the semantics of the initial query (either conjunctive, or disjunctive). This is why we suggest to investigate an alternative direction where the universal quantifier used in the initial query is weakened and this issue is discussed in Section 6.1. Then, in Section 6.2, the softening of a conjunctive division query (Q1) is addressed while Section 6.3 tackles the strengthening of a disjunctive division query (Q2).

6.1 Relaxation of the universal quantifier

The idea is to replace the universal quantifier in a given layer of an initial Q1 or Q2 query by a fuzzy relative quantifier (Kerre and Liu 1998; Zadeh 1983) which allows for some tolerance as to the number of missing associations tolerated. Moreover, the level of satisfaction corresponding to a given proportion p (of associations) lies in the unit interval which will make the relaxation a gradual one. Among others, representatives of such a quantifier are *almost all* (*aa*), and *as many as possible* (*amap*) modeled as follows:

- $aa(p) = 1$ if $p \geq ub$, 0 if $p \leq lb$, linear in-between (where lb and ub are two constants belonging to $[0, 1]$ such that $lb < ub$),
- $amap(p) = p$.

For instance, with a referential of 10 elements and using $lb = 75\%$ and $ub = 95\%$, we get:

$$aa(0) = ... = aa(\frac{7}{10}) = 0, aa(\frac{8}{10}) = 0.25, aa(\frac{9}{10}) = 0.75, aa(\frac{10}{10}) = 1.$$

$amap(0) = 0, amap(\frac{1}{10}) = 0.1,\ amap(\frac{2}{10}) = 0.2,\ amap(\frac{3}{10}) = 0.3,\ amap(\frac{4}{10}) = 0.4,$ $amap(\frac{5}{10}) = 0.5, amap(\frac{6}{10}) = 0.6, amap(\frac{7}{10}) = 0.7, amap(\frac{8}{10}) = 0.8, amap(\frac{9}{10}) = 0.9,$ $amap(\frac{10}{10}) = 1.$

It appears that the scale used is numeric and no longer ordinal, contrary to the initial one (l_1, ..., l_n for both Q1 and Q2 stratified queries), but we will see that that does not really matter since the scale used to rank-order the answers remains hidden from the user.

6.2 Tolerant conjunctive division queries

As mentioned earlier, when every element x of the dividend is not associated with at least one value of the first layer (S_1), the result of a Q1 division query is empty. In such a situation, it seems convenient to define a relaxed form of the query in order to try to get some answers. The softening approach presented hereafter relies on the weakening of the universal quantifier which will apply only to the *first* stratum. It may be remarked that there is no guarantee that the relaxed query returns a non-empty answer.

Let us denote by q the fuzzy relative quantifier used to relax the query. This quantifier is assumed to be specified by the user who either chooses it from a list of default ones, or defines it through an appropriate interface. This interaction between the system and the user takes place when the initial user query returns an empty answer. Formally, the relaxed query can be stated as follows:

select top k X from r [**where** condition] **group by** X

having set(A) **contains** *rel-quant* **of** $\{v_{1,1}, ..., v_{1,j_1}\}$ **and if possible all of** ... **and if possible all of** $\{v_{n,1}, ..., v_{n,j_n}\}$

where *rel-quant* stands for the softened quantifier q.

This query is interpreted as follows. For each x, a vector V(x) of scores of length n (the number of strata in the divisor) is built the following way:

$$V(x)[1] = q(\frac{card(\{(x,a)\ s.t.\ a \in S_1\})}{card(S_1)})$$

$$V(x)[2] = 1\ if\ \forall\, a \in S_2,\ (x,a) \in r,\ 0\ otherwise$$

$$\forall i \in [3,\ n],\ V(x)[i] = 1\ if\ \forall\, a \in S_i,\ (x,a) \in r\ and\ V(x)[i-1] = 1, 0\ otherwise.$$

The result of the tolerant division query could then be ranked using the lexicographic order (using the vectors) as defined in Subsection 3.3, i.e.:

$$x \succ y \Leftrightarrow\ V(x) \succ_{\text{lex}} V(y).$$

By doing so, an element x is preferred to y if it is strictly better with respect to the associations with the first stratum (according to q) or it is equal with respect to this stratum, but it is (fully) associated with a longer list of the next layers (2 to n). In this respect, it is worth noticing that we stay on the same line as that of the non-relaxed conjunctive queries. For instance:

- if $V(x) = [0.8, 1, 1, 0]$ and $V[y] = [0.4, 1, 1, 1]$, one has: $x \succ y$ (since x is better than y on the first layer),
- if $V(x) = [0.8, 1, 1, 0]$ and $V[y] = [0.8, 1, 0, 1]$, one has: $x \succ y$ (since x is equal to y on the first two layers and strictly better on layer 3,
- if $V(x) = [0.8, 0, 1, 1]$ and $V[y] = [0.8, 0, 0, 0]$, x and y are undistinguishable (although x is better than y on layers 3 and 4, because both fail on layer 2).

If the quantifier q returns m positive values (for the proportions strictly under 100%) and the tolerant query Q involves n strata, one has: $m * n$ levels in the scale used to discriminate among the answers to Q.

Example 9 Let us take the dividend relation $r(A, X)$:

$$r = \{(a, x_1), (b, x_1), (d, x_1), (e, x_1), (f, x_1), (a, x_2), (c, x_2), (f, x_2), (b, x_3), (c, x_3), (d, x_3), (e, x_3), (d, x_4), (b, x_5), (d, x_5)\}$$

and the query:

select top 3 X from r **group by** X

having set(A) **contains** $\{a, b, c\}$ **and if possible** $\{d\}$ **and if possible** $\{e\}$

and if possible $\{f\}$**.**

According to formula (5) or (6), the result of this non-relaxed stratified division is empty. So, we move to the tolerant version of the stratified division with the relaxed quantifier $q =$ *as many as possible*, i.e., the query:

select top 3 X from r **group by** X

having set(A) **contains as many as possible of** $\{a, b, c\}$ **and if possible** $\{d\}$

and if possible $\{e\}$ **and if possible** $\{f\}$.

We have $n = 4$ (layers) and $m = 2$ (corresponding to the proportions $\frac{2}{3}$ and $\frac{1}{3}$). Thus, we are provided with eight potential levels for discriminating among the results, a subset of which is used with the data of this example. We get:

$$V(x_1) = \left[\frac{2}{3}, 1, 1, 1\right];\ V(x_2) = \left[\frac{2}{3}, 0, 0, 1\right];\ V(x_3) = \left[\frac{2}{3}, 1, 1, 0\right];$$

$$V(x_4) = [0, 1, 0, 0];\ V(x_5) = \left[\frac{1}{3}, 1, 0, 0\right].$$

Then, the final ordering is: $x_1 \succ x_3 \succ x_2 \succ x_5 \succ x_4$ and the result returned to the user is: $x_1 \succ x_3 \succ x_2$.

6.3 Reinforcing disjunctive division queries

As mentioned previously, it may happen that a disjunctive stratified division query Q delivers an answer whose size is far over the desire of the user (k). It is then desirable to try to provide the user with an answer of a smaller size, which induces *some kind of strengthening* of Q.

A disjunctive stratified query delivers an overabundant answer if there is no x in the data associated with all the values of each of the layers 1 to p and there are M elements $x_1, \ldots, x_M$ (M far above k) satisfying the associations with all the values of layer $(p + 1)$ ($S_{p+1} = \{v_{p+1,1}, \ldots, v_{p+1,j_{p+1}}\}$). Then, every element of the answer is assigned the level of satisfaction l_{p+1}. Stratum $(p + 1)$ may be taken for responsible of the problem (similarly to stratum 1 in the case of empty answers dealt with in Section 6.2), but there is no obvious strengthening of this layer of the query without calling on the user (for instance to add new values in order to define S'_{p+1} instead of S_{p+1}). The idea chosen here is to take advantage of the stratification of the divisor to overcome the situation in trying to find a reasonable number of elements ($x'_1, \ldots, x'_{M'}$ with M' close to k) likely to obtain the *preceding level of satisfaction*, i.e., l_p. To this aim, the principle adopted is to weaken the requirement about the associations with

layer p (the values of $S_p = \{v_{p,1}, \ldots, v_{p,j_p}\}$) in the spirit of what was proposed in Section 6.2. Here again, let us mention that it is not sure at all that the process will be successful. In addition, there is no property of containment of the answers returned by the new query in those initially obtained. Notice also that when $(p+1) = 1$, this strategy cannot be applied. This leads to processing the query:

select top k X from r [**where** condition] **group by** X

having set(A) **contains** *rel-quant* **of** $\{v_{p,1}, ..., v_{p,j_p}\}$

where *rel-quant* represents q, a weakened form of the universal quantifier and is chosen by the user in an appropriate way (as in Section 6.2). Let us notice that the discrimination power is now exactly that of this quantifier over the considered layer (p), which can be restated in a purely ordinal way $(l_1, \ldots, l_m$ if the softened quantifier takes m distinct positive values) if needed.

If q denotes the relative quantifier used, each element x appearing in the dividend relation r is assigned a satisfaction level given by:

$$sat(x) = q\left(\frac{card(\{a \text{ s.t. } (a,\ x) \in r \text{ and } a \in S_p\})}{card(S_p)}\right). \tag{15}$$

Example 10 Let us consider the dividend relation:

$$r = \{(a, x_1), (c, x_1), (d, x_1), (g, x_1), (b, x_2), (e, x_2), (f, x_2), (g, x_2),$$
$$(c, x_3), (d, x_3), (g, x_3), (a, x_4), (g, x_4), (c, x_5), (g, x_5),$$
$$(d, x_6), (e, x_6), (f, x_6), (g, x_6), (g, x_7), \ldots, (g, x_{15})\}$$

and the query:

select top 4 X from r **group by** X

having set(A) **contains** $\{a, b\}$ **or else** $\{c, d, e, f\}$ **or else** $\{g\}$.

According to formula (7), the result of the (non-relaxed) stratified division is made of the 15 elements x_1 to x_{15}, which receive the grade of satisfaction l_3 since each of them is associated with g and none of them is associated with either a and b, or c, d, e and f. Since the user desires only four answers, it is necessary to move to the strengthened version of the disjunctive stratified division. With the relaxed quantifier $q =$ *as many as possible*, we will have 4 discrimination levels since the second stratum involves four values (c, d, e, f) and the query to be processed is:

select top 4 X from r **group by** X

having set(A) **contains as many as possible of** $\{c, d, e, f\}$.

Using formula (15), we get:

$$sat(x_1) = \frac{2}{4} = 0.5;\ sat(x_2) = \frac{2}{4} = 0.5;\ sat(x_3) = \frac{3}{4} = 0.75;\ sat(x_4) = \frac{0}{4} = 0;$$
$$sat(x_5) = \frac{1}{4} = 0.25;\ sat(x_6) = \frac{3}{4} = 0.75;\ sat(x_7) = \ldots = sat(x_{15}) = \frac{0}{4} = 0.$$

The elements $\{x_4, x_7, \ldots, x_{15}\}$ are now totally unsatisfactory and the others are such that:

$$\{x_3, x_6\} \succ \{x_1,\ x_2\} \succ x_5.$$

So, it is possible to provide the user with the answer: $\{x_3, x_6\} \succ \{x_1,\ x_2\}$.

7 Conclusion

In this paper, preferences for a family of queries stemming from the relational division have been considered. The key idea is to use a divisor made of a hierarchy of subsets of elements. By doing so, 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 fairly distinct semantics of the corresponding queries. Moreover, it has been shown that the result delivered in all cases is a quotient. Special attention has been paid to the implementation of such queries using a regular DBMS. Some experimental results illustrate the feasibility of the approach and tend to prove that the extra cost induced by the handling of preferences can be kept acceptable, if not marginal. The last aspect tackled in the paper is about the cases where empty or overabundant answers are abtained. In both situations, the key idea of the solution advocated relies on the weakening of the universal quantifier which is replaced by a relative quantifier such as "almost all" or "as many as possible".

This works opens several perspectives among which: i) introducing disjunctions in the divisor, ii) investigating anti-division queries (Bosc and Pivert 2008) (looking for elements connected with none of the values of the divisor in the regular case) in the presence of a stratified divisor and iii) undertaking complementary experiments, in particular:

- using dividend and divisor relations of larger sizes,
- founded on other algorithms implementing queries Q1, Q2 and Q3. In particular, one might study what happens with a solution where one regular division query per stratum is submitted to the DBMS,
- replacing Oracle by another system, e.g., MySQL, to observe the stability/variability of the tendency of the measures.

References

Bőrzsőnyi, S., Kossmann, D., & Stocker, K. (2001). The skyline operator. In *Proc. of the 17th IEEE inter. conf. on data engineering* (pp. 421–430).

Bosc, P., & Pivert, O. (2008). On a parameterized antidivision operator for database flexible querying. In *Proc. of the 19th conference on database and expert systems applications* (pp. 652–659).

Bosc, P., Pivert, O., & Rocacher, D. (2007). About quotient and division of crisp and fuzzy relations. *Journal of Intelligent Information Systems, 29*(2), 185–210.

Bruno, N., Chaudhuri, S., & Gravano, L. (2002). Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Transactions on Database Systems, 27*(2), 153–187.

Chomicki, J. (2003). Preference formulas in relational queries. *ACM Transactions on Database Systems, 28*(4), 427–466.

Dubois, D., & Prade, H. (1996). Using fuzzy sets in flexible querying: Why and how. In *Proc. of the workshop on flexible query-answering systems (FQAS'96)* (pp. 89–103).

Dubois, D., & Prade, H. (2008). Handling bipolar queries in fuzzy information processing. In *Handbook of research on fuzzy information processing in databases* (pp. 97–114). IGI Global Publication.

Fodor, J., & Yager, R. (1999). Fuzzy-set theoretic operators and quantifiers. In *The handbook of fuzzy sets series* (pp. 125–193). Norwell: Kluwer.

Hadjali, A., Kaci, S., & Prade, H. (2008). Database preference queries—A possibilistic logic approach with symbolic priorities. In *Proc. of the 5th symposium on the foundations of information and knowledge systems (FoIKS'08)* (pp. 291–310).

Kerre, E., & Liu, Y. (1998). An overview of fuzzy quantifiers—Interpretations. *Fuzzy Sets and Systems, 95*, 1–22.

Kießling, W., & Köstler, G. (2002). Preference SQL—Design, implementation, experiences. In *Proc. of the 28th conference on very large data bases (VLDB'02)* (pp. 990–1001).

Lacroix, M., & Lavency, P. (1987). Preferences: Putting more knowledge into queries. In *Proc. of the 13th conference on very large data bases (VLDB'87)* (pp. 217–225).

Zadeh, L. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computer Mathematics with Applications, 9*, 149–183.