On Some Uses of a Stratified Divisor in an Ordinal Framework

Patrick Bosc and Olivier Pivert

Abstract. In this paper, we are interested in taking preferences into account for division-like queries. The interest for introducing preferences is first to cope with user needs, then to get discriminated results instead of a flat set of elements. Here, the idea is to use ordinal preferences which are not too demanding for a casual user. Moreover, the type of query considered is inspired by the division operator and some of its variations where preferences apply only to the divisor. The division aims at retrieving the elements associated with a specified set of values and in a similar spirit, the anti-division looks for elements which are associated with none of the values of a given set. One of the focuses of this paper is to investigate queries mixing those two aspects. In order to remain coherent with the denomination of (anti-)division, the property of the result delivered is characterized. Last, a special attention is paid to the implementation of such queries using a regular database management system and some experimental results illustrate the feasibility of the approach.

Keywords: Relational databases, division, anti-division, quotient, ordinal preferences.

1 Introduction

Queries including preferences have received a growing interest during the last decade [1, 2, 5, 6, 8, 11, 12, 13]. 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 queries concerned with preferences by considering more complex queries,

Patrick Bosc and Olivier Pivert
IRISA/ENSSAT - University of Rennes 1, 6 Rue de Kerampont
Technopole Anticipa BP 80518, 22305 Lannion Cedex, France

J. Kacprzyk et al. (Eds.): Uncer. Appro. for Spatial Data Model. and Process., SCI 271, pp. 133–154.
springerlink.com

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. Last, taking preferences into account will allow for keeping only the best k answers, in the spirit of top-k queries [5].

Knowing that a regular division delivers a non discriminated set of elements, the idea is to call on preferences related to the divisor. Two major lines for assigning preferences may be thought of, depending on whether they concern tuples individually (see e.g., [2]), or (sub)sets of tuples, which is the choice made here and we will use the term "stratified divisor".

Moreover, we will not only consider the division, but a neighbor operator called the anti-division. As the division retrieves elements associated with a given set of values, the anti-division looks for elements that are associated with none of the elements of a specified set.

In both cases, the first layer of the divisor may be seen as an initial divisor and the following layers serve to break ties between elements associated with all (respectively none of) the values of the first layer. In other words, to be satisfactory, an element x of the dividend must be associated with all (respectively none of) the values of the first layer. The way the next layers are taken into account is discussed in more details in the body of the paper.

Such extended division (respectively anti-division) queries can be expressed in natural language as:

> "find the elements x connected **in priority** with all (respectively none) of $\{set\}_1$ **then if possible** with all (respectively none) of $\{set\}_2$ …
> **then if possible** with all (respectively none) of $\{set\}_n$".

This type of statement has some relationship with bipolarity [9, 10]. Indeed, this falls in the third category of bipolarity reported in [10] where the two types of criteria are of a different nature. Here, the connection with all (respectively none of) the values of $\{set\}_1$ is a constraint and those with all (respectively none of) the values of $\{set\}_2$ to $\{set\}_n$ represent wishes which are not mandatory (in the sense of acceptance/rejection).

Let us illustrate the idea of an extended division with 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. Similarly, an anti-division is of interest if one is interested in food products which do not contain some additives, where some are totally forbidden and other more or less undesired.

The rest of the paper is organized as follows. Section 2 is devoted to some reminders about the division and anti-division operators in the usual relational setting. In section 3 a stratified version of these operators is presented along with their syntax and modeling. It is also shown that the result they deliver has the same property as in the usual case. In Section 4, the issue of considering queries involving both a stratified division and a stratified anti-division is tackled. Implementation issues for all these queries involving stratified operations are discussed in section 5. The conclusion summarizes the contributions of the paper and evokes some lines for future work.

2 The Regular Division and Anti-division Operators

In the rest of the paper, the dividend relation r has the schema (A, X), while without loss of generality that of the divisor relation s is (B) where A and B are compatible sets of attributes, i.e., defined on the same domains of values.

2.1 The Division

The relational division, i.e., the division of relation r by relation s is defined as:

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

$$= \{x \mid (x \in R[X]) \wedge (\forall a, a \in s \Rightarrow (a, x) \in r)\} \quad (2)$$

where r[X] denotes the projection of r over X and $\Omega_r(x) = \{a \mid \langle a, x\rangle \in r\}$. In other words, an element x belongs to the result of the division of r by s if and only if 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 res obtained with expression (1) has the double characteristic of a quotient:

$$s \times res \subseteq r \quad (3a)$$

$$\forall res' \supset res, s \times res' \not\subseteq r \quad (3b)$$

$\times$ denoting the Cartesian product of relations. Expressions (3a) and (3b) express the fact that the relation res resulting from the division (according to formula (1) or (2)) is a quotient, i.e., **the largest relation** whose Cartesian product with the divisor returns a result smaller than or equal to the dividend (according to regular set inclusion).

In an SQL-like language, the division of r by s can be expressed thanks to a partitioning mechanism:

select X **from** r [**where** condition] **group by** X
having set(A) **contains** (**select** B **from** s **where** …).

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 the product whose number is 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 thanks to a division as:

$$\text{div}(\text{o-g35}, \text{p-u127}, \{np\}, \{np\})$$

where relation o-g35 corresponds to pairs (n, s) such that product n has been ordered from store s in a quantity over 35 and relation p-u127 gathers products

whose price is under \$127. From the following extensions of relations o-g35 and p-u127:

o-g35 = {<15, 32>, <12, 32>, <34, 32>, <26, 32>, <12, 7>, <26, 7>,
<15, 19>, <12, 19>, <26, 19>},
p-u127 = {<15>, <12>, <26>},

the previous division using formula (1) leads to a result made of two elements {<32>, <19>}. It can easily be checked that this result satisfies expressions (3a) and (3b). ♦

2.2 The Anti-division

Similarly, we call anti-division the operator ⅋ defined the following way:

$$r\ [A \parr B]\ s = \{x \mid (x \in r[X]) \wedge (s \subseteq cp(\Omega_r(x)))\} \quad (4)$$

$$= \{x \mid (x \in r[X]) \wedge (\forall a, a \in s[B] \Rightarrow (a, x) \notin r)\} \quad (5)$$

where cp(rel) denotes the complement of rel. The result ad-res of the anti-division may be called an "anti-quotient", 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:

$$s \times \text{ad-res} \subseteq cp(r) \quad (6a)$$

$$\forall \text{ad-res'} \supset \text{ad-res},\ s \times \text{ad-res'} \not\subseteq cp(r). \quad (6b)$$

In an SQL-like language, the anti-division of r by s can be expressed in a way similar to a division:

select X **from** r [**where** condition] **group by** X
having set(A) **contains-none** (**select** B **from** s **where** …)

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

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

Example 2. Let us consider the following relations Prod(product, component, proportion), which describes the composition of some chemical products and Nox(component) which gathers the identifications of noxious components:

Prod = {<p_1, c_1, 3>, <p_1, c_2, 4>, <p_1, c_3, 54>, <p_2, c_1, 9>, <p_2, c_4, 30>,
<p_3, c_2, 8>, p_3, c_6, 22>},

Nox = {<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 anti-division of the relation Prod1 derived from Prod (on the basis of a proportion over 5%) made of:

{<p_1, c_3>, <p_2, c_1>, <p_2, c_4>, <p_3, c_2>, <p_3, c_6>}

by Nox, whose result according to (4) or (5) is $\{p_1\}$ and it is easy to check that formulas (6a-6b) both hold. ♦

3 Stratified Division and Anti-division Queries

In this section, we first give some characteristics of the stratification. Then, the expression of stratified division and anti-division queries in an SQL-like fashion is proposed as well as the modelling of such queries. Finally, the property of the result delivered is discussed.

3.1 About the Stratification Mechanism

As mentioned before, the key idea is to use a divisor made of several layers. So, there is a preference relation over the subsets of the divisor, namely:

$$(S_1 = \{v_{1,1}, \ldots, v_{1,j_1}\}) \succ \ldots \succ (S_n = \{v_{n,1}, \ldots, v_{n,j_n}\})$$

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:

$$l_1 > \ldots > l_n > l_{n+1}$$

which will be also used to assign levels of satisfaction to elements pertaining to the result of any stratified division or anti-division. In this scale, l_1 is the maximal element for the highest satisfaction and the last label l_{n+1} expresses rejection. These two specific levels play the role of 1 and 0 in the unit interval.

Example 3. Coming back to the example of the query related to wine shops evoked in the introduction, there are three layers:

S_1 = {Saint Emilion Grand Cru, Pomerol, Margaux},

S_2 = {Gewurztraminer Vendanges Tardives, Chablis Premier Cru},

S_3 = {Pommard, Chambertin},

along with the scale $L = l_1 > l_2 > l_3 > l_4$. ♦

According to the view adopted in this paper, the first stratum S_1 is considered mandatory, whereas the next ones (S_2 to S_n) define only wishes. In other words, S_1 is a regular divisor and S_2, …, S_n are introduced as complementary components in order to discriminate among the elements of the dividend associated with all (respectively none) of the values of S_1. In addition, the layers are considered in a hierarchical fashion, which means that a given layer intervenes only if the association (or non-association) with all the previous ones holds. This behavior is similar to what is done in the systems Preferences [13] and PreferenceSQL [12] or with the operator winnow [6] when cascades of preferences are used. Finally, an element x of the dividend is all the more acceptable as it is (respectively it is not) connected with a "long" succession of layers of the divisor starting with S_1. In other words, x is preferred to y if x is associated with all (respectively none) of the

values of the sets S_1 to S_p and y is associated with all (respectively none) of the elements of a shorter list of sets.

Example 4. Let us consider the stratified divisor :

$$s = \{\{a, b, c\}, \{d\}, \{e, f\}\}$$

and the dividend:

$$\begin{aligned} r = \{&<x1, a>, <x1, b>, <x1, c>, <x1, d>, <x1,e>,\\ &<x2, a>, <x2, b>, <x2, c>,\\ &<x3, a>, <x3, b>,<x3, c>, <x3, e>, <x3, f>,\\ &<x4, a>, <x4, e>, <x4, f>,\\ &<x5, b>, <x5, c>,\\ &<x6, d>, <x6, e>\}. \end{aligned}$$

The stratified division of r by s discards x4, x5 and x6 which are not exhaustively associated with $S_1 = \{a, b, c\}$ and it delivers the result: $x1 \succ \{x2, x3\}$. ♦

It must be noticed that the view adopted here is somehow conjunctive. An alternative would be to model a behavior that takes into account all the layers in a hierarchical way and build, for a given x, a vector E(x) of Boolean values (E(x)[i] = 1 if x is associated with all (respectively none) of the values from layer S_i, 0 otherwise). The different x's could then be ranked according to the lexicographic order over the vectors.

3.2 Syntax of Stratified Operations

Division queries are expressed in an SQL-like style where the dividend may be any intermediate relation (not only a base relation) and the divisor is either explicitly given by the user, or stated thanks to subqueries, along with his/her preferences. This is done in way quite similar to the usual division (i.e., thanks to a partitioning mechanism), namely:

select top k X **from** r [**where** condition] **group by** X
having set(A) **contains** $\{v_{1,1}, \ldots, v_{1,j_1}\}$ **and if possible** …
and if possible $\{v_{n,1}, \ldots, v_{n,j_n}\}$.

Coming back to the example of wines evoked before, 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}.

In the context of medical diagnosis, the following example illustrates the use of subqueries to build the stratified divisor. Let us consider: i) a relation disease(name, symptom, frequency) which describes the symptoms associated with some diseases as well as the frequency with which a given symptom appears for a given disease, ii) a relation patient(#person, symptom) which describes the

symptoms shown by some patients. The following stratified division query looks for the patients which have all of the 100% frequent symptoms of influenza, and if possible all of the symptoms whose frequency is above 80%, and if possible all of the symptoms whose frequency is above 50%:

select top 10 #person **from** patient
group by #person
having set(symptom) **contains**
(**select** symptom **from** disease **where** name = 'flu' **and** frequency = 100)
and if possible
(**select** symptom **from** disease
where name = 'flu' **and** frequency **between** 80 **and** 99)
and if possible
(**select** symptom **from** disease
where name = 'flu' **and** frequency **between** 50 **and** 79)

The anti-division is similarly formulated as:

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

It is worth noticing that an expression based on one (or several) difference(s) would be complicated to formulate and thus would not be natural at all (especially for a user), while the one chosen above is. Moreover, the query specifies dislikes which are given in a hierarchical manner. So, S_1 contains the values the most highly (indeed totally excluded) and S_n those which are the most weakly unwanted. Here also, associated with the preference relation sustaining the hierarchy, is an ordinal scale L with labels l_i's (such that $l_1 > \ldots > l_n > l_{n+1}$) which will be used to assign levels of satisfaction to the elements of the result of any stratified anti-division.

Example 5. 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 5 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 tells that the additives AS27, BT12 and C3 are completely forbidden, that the absence of both AS5 and D2 is appreciated and that it is still better if D8 is not in the product. ♦

3.3 Modeling Stratified Operations

We consider a stratified division or anti-division of a relation r whose schema is (A, X) by a relation s defined over attribute B with A and B compatible attributes

(in fact, A and B could be compatible sets of attributes as well). The principle for defining these operations is to extend expressions (2) and (5). This point of departure entails: i) dealing with the preferences applying to the divisor and ii) using an ordinal (symbolic) implication. This is why we 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 $pref_{rel}(t)$ and any tuple can be written $pref_{rel}(t)/t$. 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 worst level l_{n+1}. For the divisor, the level of preference attached to a tuple is directly stemming from the place of the corresponding element in the hierarchy provided by the user. As to the implication, it can be chosen among fuzzy implications with two requirements: i) to work in a purely ordinal context, and ii) to convey the semantics of importance associated with the layered divisor. It turns out that Kleene-Dienes implication usually defined as:

$$p \Rightarrow_{KD} q = \max(1 - p, q)$$

meets the goal provided that the complement to 1 is changed into order reversal over L. In other words, we will use a symbolic version of the previous implication, denoted by $\Rightarrow_{sKD}$:

$$l_i \Rightarrow_{sKD} l_j = \max(rev(l_i), l_j)$$

where $\forall l_i \in L = l_1 > \ldots l_{n+1}$, $rev(l_i) = l_{n+2-i}$. In other words, if a symbol s has the position k on the scale, rev(s), its negation, has the position k when the scale is read from the end.

Example 6. Let L be the scale:

completely important > highly important > fairly important >
not very important > not at all important.

The inverse scale is :

[rev(completely important) = not at all important] <
[rev(highly important) = not very important] <
[rev(fairly important) = fairly important] <
[rev(not very important) = highly important] <
[rev(not at all important) = completely important]. ♦

So equipped, if V denotes the values of the divisor, the stratified division and anti-division are defined as follows:

$$\begin{aligned} pref_{strat\text{-}div(r, s, A, B)}(x) &= \min_{v \in V} pref_s(v) \Rightarrow_{sKD} pref_r(v, x)) \\ &= \min_{v \in V} \max(rev(pref_s(v)), pref_r(v, x)) \end{aligned} \quad (7)$$

$$\begin{aligned} pref_{strat\text{-}antidiv(r, s, A, B)}(x) &= \min_{v \in V} pref_s(v) \Rightarrow_{sKD} rev(pref_r(v, x))) \\ &= \min_{v \in V} \max(rev(pref_s(v)), rev(pref_r(v, x))). \end{aligned} \quad (8)$$

Due to the fact that $pref_r(v, x)$ takes only the two values l_1 and l_{n+1} depending on the presence or absence of <v, x> in relation r:

i) in expression (7), each term $\max(rev(pref_V(v)), pref_r(v, x))$ equals l_1 if x is associated with v in r ($<v, x> \in r$), $rev(pref_V(v))$ otherwise,
ii) in expression (8), each term $\max(rev(pref_V(v)), rev(pref_r(v, x)))$ equals l_1 if x is not associated with v in r ($<v, x> \notin r$), $rev(pref_V(v))$ otherwise.

In other words, if x is associated with all (respectively none) of the values of the entire divisor, the maximal level of preference l_1 is obtained and as soon as an association <v, x> is missing (respectively found), the level of preference of x decreases all the more as v is highly preferred (respectively undesired).

Example 7. Let us consider the following dividend relation r:

$r = \{<a1, x>, <a2, x>, <a4, x>, <a1, y>, <a3, y>, <a5, z>, <a2, t>\}$

and the stratified divisor:

$s = \{a1\} \succ \{a2\} \succ \{a3, a4\}$

which induces the four-level scale $L = l_1 > l_2 > l_3 > l_4$. These relations rewrite:

r

A	X	pref
a1	x	l_1
a2	x	l_1
a4	x	l_1
a1	y	l_1
a3	y	l_1
a2	z	l_1
a3	z	l_1
a4	z	l_1
a2	t	l_1

s

B	pref
a1	l_1
a2	l_2
a3	l_3
a4	l_3

According to formula (7), the result d-res of the division of r by s is:

$pref_{d\text{-}res}(x) = \min(l_1, l_1, rev(l_3), l_1) = l_2$,

$pref_{d\text{-}res}(y) = \min(l_1, rev(l_2), l_1, rev(l_3)) = l_3$,

$pref_{d\text{-}res}(z) = \min(rev(l_1), rev(l_2), rev(l_3), rev(l_3)) = l_4$,

$pref_{d\text{-}res}(t) = \min(rev(l_1), l_1, rev(l_3), rev(l_3)) = l_4$,

which means that x is preferred to y on the one hand and that z and t are rejected on the other hand. Similarly, using formula (8), the following result ad-res of the anti-division of r by s is obtained:

$pref_{ad\text{-}res}(x) = \min(rev(l_1), rev(l_2), l_1, rev(l_3)) = l_4$,

$pref_{ad\text{-}res}(y) = \min(rev(l_1), l_1, rev(l_3), l_1) = l_4$,

$\text{pref}_{\text{ad-resr}}(z) = \min(l_1, l_1, l_1, l_1) = l_1,$

$\text{pref}_{\text{ad-res}}(t) = \min(l_1, \text{rev}(l_2), l_1, l_1) = l_3,$

which states that z is fully satisfactory and t significantly less, while x and y are quite unsatisfactory. ♦

3.4 Property of the Result of Stratified Divisions and Anti-divisions

In order to be qualified a division (respectively anti-division), the extended operator defined above must deliver a result having the characteristic property of a quotient. This means that one must have valid properties similar to 3a-b (respectively 6a-b). In [2], it is shown that the division of fuzzy relations (i.e., where each tuple is assigned a membership degree taken in the unit interval) leads to a result which is a quotient as far as the implication used is either an R-implication, or an S-implication. The key point of the proof lies in the fact that these implications ($\Rightarrow_f$) may be written in a common format, namely :

$$p \Rightarrow_f q = \sup \{y \in [0, 1] \mid \text{cnj}(p, y) \leq q\}$$

where cnj is an appropriate conjunction operator (see [2, 7] for more details). In the specific case considered here, the ordinal version of Kleene-Dienes implication (which belongs to the family of S-implications) writes:

$$l_i \Rightarrow_{sKD} l_j = \max(\text{rev}(l_i), l_j) = \sup \{y \in [l_1, l_{n+1}] \mid \text{cnj}(l_i, y) \leq l_j\} \quad (9)$$

$$\text{with cnj}(a, b) = \begin{cases} l_{n+1} \text{ if } a \leq \text{rev}(b), \\ b \text{ otherwise.} \end{cases} \quad (10)$$

So, if we denote by d-res (repectively ad-res) the result of a stratified division (respectively anti-division), due to the very nature of expression (9), the following expressions hold:

$$s \times \text{d-res} \subseteq r \quad (11a) \qquad \forall \text{d-res'} \supset \text{d-res}, s \times \text{d-res'} \not\subseteq r \quad (11b)$$

$$s \times \text{ad-res} \subseteq \text{cp}(r) \quad (12a) \qquad \forall \text{ad-res'} \supset \text{ad-res}, s \times \text{ad-res'} \not\subseteq \text{cp}(r) \quad (12b)$$

where the Cartesian product ($\times$), inclusion and complement are respectively defined as:

$$r \times s = \{p3/uv \mid p1/u \in r \wedge p2/v \in s > \wedge\ p3 = \text{cnj}(p1, p2)\},$$

$$r \subseteq s \Leftrightarrow \forall\ p1/u \in r, \exists\ p2/u \in s \text{ such that } p1 \leq p2,$$

$$\text{cp}(r) = \{\text{rev}(p)/u \mid p/u \in r\}$$

which means that d-res is a quotient and that ad-res is an anti-quotient.

Example 8. Let us come back to the relations of example 7. According to (11a), we must have:

s

B	pref
a1	l_1
a2	l_2
a3	l_3
a4	l_3

$\times$ **d-res**

X	pref
x	l_2
y	l_3

$\subseteq$ **r**

A	X	pref
a1	x	l_1
a2	x	l_1
a4	x	l_1
a1	y	l_1
a3	y	l_1
a2	z	l_1
a3	z	l_1
a4	z	l_1
a2	t	l_1

on the one hand, and with respect to (11b), if any grade is increased in d-res (yielding d-res'), the Cartesian product of s and d-res' is not included in r. We will illustrate what happens for x and t (which may be considered to be in d-res with the level of preference l_4) and it would be easy to observe that the same conclusions can be drawn for y and z.

In the Cartesian product, we have the tuples:

$cnj(l_1, l_2)/<a1, x> = l_2/<a1, x>$,
$cnj(l_2, l_2)/<a2, x> = l_2/<a2, x>$,
$cnj(l_3, l_2)/<a3, x> = l_4/<a3, x>$,
$cnj(l_3, l_2)/<a4, x> = l_4/<a4, x>$,
$cnj(l_1, l_4)/<a1, t> = l_4/<a1, t>$,
$cnj(l_2, l_4)/<a2, t> = l_4/<a2, t>$,
$cnj(l_3, l_4)/<a3, t> = l_4/<a3, t>$,
$cnj(l_3, l_4)/<a4, t> = l_4/<a4, t>$,

and the inclusion in r holds. If we suppose that the level of preference of x in d-res is increased (from l_2 to l_1), the partial Cartesian product of s and l_1/x yields:

$cnj(l_1, l_1)/<a1, x> = l_1/<a1, x>$,
$cnj(l_2, l_1)/<a2, x> = l_1/<a2, x>$,
$cnj(l_3, l_1)/<a3, x> = l_1/<a3, x>$,
$cnj(l_3, l_1)/<a4, x> = l_1/<a4, x>$,

for which the inclusion in r does not hold (presence of the tuple $l_1/<a3, x>$ which does not belong to r). Similarly, let us increase the level of preference of t in d-res (from l_4 to l_3), the partial Cartesian product of s and l_3/t is:

$cnj(l_1, l_3)/<a1, t> = l_3/ <a1, t>$,
$cnj(l_2, l_3)/<a2, t> = l_4/<a2, t>$,
$cnj(l_3, l_3)/<a3, t> = l_4/<a3, t>$,
$cnj(l_3, l_3)/<a4, t> = l_4/<a4, t>$,

and the tuple l_3, <a1, t> violates the inclusion in r. Due to the increasing monotonicity of cnj with respect to its second argument, any other increase of the level of preference of t in d-res would also lead to the non inclusion of the Cartesian product in r.

We now consider the anti-division of r by s and, for illustration purpose, only the elements l_4/y and l_1/z of its result ad-res. In order to check formula (12a), the Cartesian product of s and these two tuples has to be performed, which results in:

$cnj(l_1, l_4)/<a1, y> = l_4/<a1, y>$,
$cnj(l_2, l_4)/<a2, y> = l_4/<a2, y>$,
$cnj(l_3, l_4)/<a3, y> = l_4/<a3, y>$,
$cnj(l_3, l_4)/<a4, y> = l_4/<a4, y>$,
$cnj(l_1, l_1)/<a1, z> = l_1/<a1, z>$,
$cnj(l_2, l_1)/<a2, z> = l_1/<a2, z>$,
$cnj(l_3, l_1)/<a3, z> = l_1/<a3, z>$,
$cnj(l_3, l_1)/<a4, z> = l_1/<a4, z>$,

and the inclusion in the complement of r holds. As to the satisfaction of (12b), clearly the level of preference of z (l_1) is maximal and if that of y is increased from l_4 to l_3, we have the Cartesian product:

$cnj(l_1, l_3)/<a1, y> = l_3/<a1, y>$,
$cnj(l_2, l_3)/<a2, y> = l_4/<a2, y>$,
$cnj(l_3, l_3)/<a3, y> = l_4/<a3, y>$,
$cnj(l_3, l_3)/<a4, y> = l_4/<a4, y>$,

and the tuple l_3/<a1, y> violates the desired inclusion ($l_3 > rev(l_1) = l_4$). ♦

4 Stratified Queries Mixing Division and Anti-division Features

4.1 A Basis for Safe Mixed Queries

The starting point of this section is the analogy between division queries and the search for documents indexed by a certain set of keywords, since these two activities are concerned with the association of an element (respectively a document) with a set of values (respectively keywords). On this line, it seems convenient to extend/enhance the basis of document retrieval with a set of undesired keywords, which has a direct counterpart in terms of anti-division. Last, if we introduce the notion of levels of importance of the keywords in both the positive and negative parts, we end up with a query involving a stratified division (corresponding to the desired keywords/positive part) and a stratified anti-division (corresponding to the unwanted keywords/negative part). Consequently, in the following, we consider queries where the association and non association conditions relate to a same attribute, even it would make sense to envisage more general situations.

A query is made of two parts: i) the positive part which gathers the values which are desired (at different levels of importance) and ii) the negative part which collects the unwanted values, still with different importances. In fact, such queries call on two types of bipolarity: i) one tied to the fact that some conditions

(the association with all (respectively none) of the values of the first set) are mandatory whereas others (the association (respectively non association) with the values of the next sets) are only desirable, and ii) another related to the fact that the association with some values is expected (those of the positive part), while one would like the non association with other values (those of the negative part). Clearly, these two types of bipolarity impact the semantics of a query in two quite different ways. The first one entails handling the associations (respectively non associations) with the values of the first stratum as constraints (whose satisfaction or not causes acceptance or rejection) and the associations (respectively non associations) with the values of the other layers as wishes (whose satisfaction or not influences the discrimination between selected elements). The second aspect leads to distinguish between values which are desired and values which are unwanted, then to look for the association with the former ones and for the non association with the latter ones.

A first approach to mixed stratified queries is to consider them as made of two components according to the following pattern:

select top k X **from** r [**where** condition] **group by** X
having set(A) **contains** $\{v_{1,1}, \ldots, v_{1,j_1}\}$ **and if possible** …
and if possible $\{v_{n,1}, \ldots, v_{n,j_n}\}$ **and**
contains-none $\{w_{1,1}, \ldots, w_{1,k_1}\}$ **and if possible** …
and if possible $\{w_{p,1}, \ldots, w_{p,k_p}\}$.

This means that the query refers to two scales:

$L1 = l_1 > \ldots > l_n > l_{n+1}$ for the positive part

and:

$L2 = l'_1 > \ldots > l'_p > l'_{p+1}$ for the negative part

and the overall satisfaction of a given x would require to combine two symbols (one from each scale), which raises a serious problem.

To avoid this difficulty, we suggest to build mixed queries in such a way that a single scale comes into play. Each level of the scale used in a query will be assigned a set of desired values (contributing the positive part) and a set of unwanted values (subset of the negative part), one of them being possibly empty. A mixed stratified query will be expressed according to the following model:

select top k X **from** r [**where** condition] **group by** X
having set(A) **contains** [**pos**: $\{v_{1,1}, \ldots, v_{1,j_1}\}$, **neg**: $\{w_{1,1}, \ldots, w_{1,k_1}\}$]
and if possible …
and if possible [**pos**: $\{v_{n,1}, \ldots, v_{n,j_n}\}$, **neg**: $\{w_{n,1}, \ldots, w_{n,k_n}\}$]

where "pos (respectively neg): S", at a given level of importance, stands for a set of desired (respectively unwanted) values, which x must be (repectively not be) associated with. In addition, note that it is possible to have "pos : {}", "neg : {}" (but not both) at each layer.

4.2 Syntax, Semantics and Modeling of Mixed Queries

The above type of query is interpreted in a straightforward manner as follows. To be somewhat satisfactory, a element x: i) must be associated with all the values $\{v_{1,1}, \ldots, v_{1,j_1}\}$ and none of the values of $\{w_{1,1}, \ldots, w_{1,k_1}\}$, and ii) it receives a level of satisfaction (pref) all the larger as it satisfies the association with all the values $\{v_{2,1}, \ldots, v_{2,k_2}\}$, …, $\{v_{j,1}, \ldots, v_{j,k_j}\}$ and none of the values of $\{w_{2,1}, \ldots, w_{2,p_2}\}$, …, $\{w_{j,1}, \ldots, w_{j,p_j}\}$ with j taking a high value (n for the maximal level pref = l_1). In other words, an element x is preferred to another y if x is connected with $\{v_{1,1}, \ldots, v_{1,k_1}\}$, …, $\{v_{i,1}, \ldots, v_{i,k_i}\}$ and none of $\{w_{1,1}, \ldots, w_{1,p_1}\}$, …, $\{w_{i,1}, \ldots, w_{i,p_i}\}$, while y is associated with $\{v_{1,1}, \ldots, v_{1,k_1}\}$, …, $\{v_{j,1}, \ldots, v_{j,k_j}\}$ and none of $\{w_{1,1}, \ldots, w_{1,p_2}\}$, …, $\{w_{j,1}, \ldots, w_{j,p_j}\}$ and $i > j$.

Let us denote by $s = \{V_1, \ldots, V_n\}$ the different layers of the divisor where each V_i is made of a positive part P_i and a negative part N_i. Alternatively, s writes as s = (P, N), its positive and negative parts. The mixed stratified division is defined as:

$$\begin{aligned}
&\text{pref}_{\text{mix-strat-div(r, s, A, B)}}(x) = \\
&\qquad \min_{i \in [1, n]} \min(\min_{v \in P_i} \text{pref}_s(v) \Rightarrow_{sKD} \text{pref}_r(v, x)), \\
&\qquad\qquad \min_{w \in N_i} \text{pref}_s(w) \Rightarrow_{sKD} \text{rev}(\text{pref}_r(w, x))) \\
&= \min_{i \in [1, n]} \min(\min_{v \in P_i} \max(\text{rev}(l_i), \text{pref}_r(v, x)), \\
&\qquad\qquad \min_{w \in N_i} \max(\text{rev}(l_i), \text{rev}(\text{pref}_r(w, x)))) \\
&= \min(\min_{v \in P} \max(\text{rev}(\text{pref}_s(v)), \text{pref}_r(v, x)), \\
&\qquad \min_{w \in N} \max(\text{rev}(\text{pref}_s(w)), \text{rev}(\text{pref}_r(w, x)))). \qquad (13)
\end{aligned}$$

By construction, the result delivered by the above operation is a quotient in the sense that it is a maximal relation. More precisely, it is the largest (ordinal) relation whose Cartesian product (using the conjunction given in expression (10)) with the positive and negative parts of the divisor is included in the dividend. So, if m-res denotes the result delivers by expression (13), the following characterization formulas hold:

$$\left.\begin{array}{l} P \times \text{m-res} \subseteq r \\ \quad \text{and} \\ N \times \text{m-res} \subseteq \text{cp}(r) \end{array}\right\} \qquad (14a)$$

$$\forall \text{m-res'} \supset \text{m-res}, \quad \left.\begin{array}{c} P \times \text{m-res'} \not\subset r \\ \text{or} \\ N \times \text{m-res'} \not\subset \text{cp}(r) \end{array}\right\} \qquad (14b)$$

4.3 A Complete Example

Let us consider a relation Prod(product, component) where a tuple <p, c> expresses that c is one of the components of product p and the mixed division query:

select top 5 product **from** Prod **group by** product
having set(product) **contains** [**pos**: {c_1}, **neg**: {c_5, c_6}]
and if possible [**pos**: {c_2}, **neg**: {}]
and if possible [**pos**: {c_3, c_4}, **neg**: {c_7}]

expressing that the double stratification:

P : {c_1} (l_1) > {c_2} (l_2) > {c_3, c_4} (l_3)
N : {c_5, c_6} (l_1) > ∅ > {c_7} (l_3).

The user wants product c_1, if possible c_2 and if possible c_3 and c_4, and he/she dislikes c_5 and c_6 (respectively c_7) as much as he/she desires c_1 (respectively c_3 and c_4). Notice that there is no counterpart for c_2 (in other words c_3 and c_4 are forbidden and c_7 is only weakly undesired). If the dividend relation is:

$r = \{<c_1, x>, <c_2, x>, <c_4, x>, <c_1, y>, <c_2, y>, <c_3, y>, <c_4, y>, <c_7, y>, <c_2, z>, <c_3, z>, <c_4, z>, <c_1, t>, <c_2, t>, <c_5, t>, <c_1, u>\}$.

According to formula (13), the levels of satisfaction assigned to x, y, z and t are:

$$\begin{aligned}\text{pref}(x) &= \min(\min(l_1 \Rightarrow_{sKD} \text{pref}_r(c_1, x), l_2 \Rightarrow_{sKD} \text{pref}_r(c_2, x), \\ &\qquad l_3 \Rightarrow_{sKD} \text{pref}_r(c_3, x), l_3 \Rightarrow_{sKD} \text{pref}_r(c_4, x)), \\ &\quad \min(l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_5, x)), l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_6, x)), \\ &\qquad l_3 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_7, x))) \\ &= \min(l_1, l_1, l_2, l_1, l_1, l_1, l_1) = l_2\end{aligned}$$

$$\begin{aligned}\text{pref}(y) &= \min(\min(l_1 \Rightarrow_{sKD} \text{pref}_r(c_1, y), l_2 \Rightarrow_{sKD} \text{pref}_r(c_2, y), \\ &\qquad l_3 \Rightarrow_{sKD} \text{pref}_r(c_3, y), l_3 \Rightarrow_{sKD} \text{pref}_r(c_4, y)), \\ &\quad \min(l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_5, y)), l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_6, y)), \\ &\qquad l_3 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_7, y))) \\ &= \min(l_1, l_1, l_1, l_1, l_1, l_1, l_2) = l_2\end{aligned}$$

$$\begin{aligned}\text{pref}(z) &= \min(\min(l_1 \Rightarrow_{sKD} \text{pref}_r(c_1, z), l_2 \Rightarrow_{sKD} \text{pref}_r(c_2, z), \\ &\qquad l_3 \Rightarrow_{sKD} \text{pref}_r(c_3, z), l_3 \Rightarrow_{sKD} \text{pref}_r(c_4, z)), \\ &\quad \min(l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_5, z)), l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_6, z)), \\ &\qquad l_3 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_7, z))) \\ &= \min(l_4, l_1, l_1, l_1, l_1, l_1, l_1) = l_4\end{aligned}$$

$$\begin{aligned}\text{pref}(t) &= \min(\min(l_1 \Rightarrow_{sKD} \text{pref}_r(c_1, t), l_2 \Rightarrow_{sKD} \text{pref}_r(c_2, t), \\ &\qquad l_3 \Rightarrow_{sKD} \text{pref}_r(c_3, t), l_3 \Rightarrow_{sKD} \text{pref}_r(c_4, t)), \\ &\quad \min(l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_5, t)), l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_6, t)), \\ &\qquad l_3 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_7, t))) \\ &= \min(l_1, l_1, l_2, l_2, l_4, l_1, l_1) = l_4\end{aligned}$$

$$\begin{aligned}\text{pref}(u) &= \min(\min(l_1 \Rightarrow_{sKD} \text{pref}_r(c_1, u), l_2 \Rightarrow_{sKD} \text{pref}_r(c_2, u), \\ &\qquad l_3 \Rightarrow_{sKD} \text{pref}_r(c_3, u), l_3 \Rightarrow_{sKD} \text{pref}_r(c_4, u)), \\ &\quad \min(l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_5, u)), l_1 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_6, u)), \\ &\qquad l_3 \Rightarrow_{sKD} \text{rev}(\text{pref}_r(c_7, u))) \\ &= \min(l_1, l_3, l_2, l_2, l_1, l_1, l_1) = l_3.\end{aligned}$$

Finally, one has the resulting relation: $\{l_2/x, l_2/y, l_3/u\}$.

It turns out that x and y are equally ranked since the absence of $<c_3, x>$ has the same impact for x as the presence of $<c_7, y>$ for y (the level of desire for c_3 equals the level of dislike for $c_7 - l_3$). Similarly, the absence of c_1 (mandatory) for z has the same effect (rejection) as the presence of c_5 (forbidden) for t.

Using the non commutative conjunction defined in formula (10) (and discarding the tuples whose level is l_4), the Cartesian product of the positive part of the divisor and $\{<l_2, x>, <l_2, y>, <l_3, u>\}$ is:

$$\{l_1/<c_1, x>, l_1/<c_2, x>, l_2/<c_1, y>, l_2/<c_2, y>, l_3/<c_1, u>\}$$

which is included in the dividend r. Similarly, the Cartesian product of the negative part of the divisor and the previous result yields:

$$\{l_2/<c_5, x>, l_2/<c_6, x>, l_2/<c_5, y>, l_2/<c_6, y>, l_3/<c_5, u>, l_3/<c_6, u>\}$$

which is included in the complement of the dividend (i.e., none of these tuples appears in the dividend). We observe that formula (14a) holds.

It is easy to check that if the level of preference of any element (x, y, z, t or u) is upgraded, the property conveyed by formula (14b) is valid. For instance, if we consider l_1/x instead of l_2/x, the Cartesian product (with P) becomes:

$$\{l_1/<c_1, x>, l_1/<c_2, x>, l_1/<c_3, x>, l_1/<c_4, x>, l_2/<c_1, y>, l_2/<c_2, y>, l_3/<c_1, u>\}$$

and the presence of the third tuple shows the non-inclusion in the dividend. Similarly, if the tuple l_3/z is introduced, the Cartesian product (with P) becomes:

$$\{l_1/<c_1, x>, l_1/<c_2, x>, l_2/<c_1, y>, l_2/<c_2, y>, l_3/<c_1, u>, l_3/<c_1, z>\}$$

and the last tuple proves that the inclusion in r does not hold (then that property (14b) is valid). Last, if l_2/y is replaced by l_1/y, the Cartesian product of N and the modified result is:

$$\{l_2/<c_5, x>, l_2/<c_6, x>, l_1/<c_5, y>, l_1/<c_6, y>, l_1/<c_7, y>, l_3/<c_5, u>, l_3/<c_6, u>\}$$

and the presence of the tuple $l_1/<c_7, y>$ makes the inclusion in the dividend fail, which, once again, shows the validity of property (14b). ♦

5 Implementation Issues

Now, we tackle processing strategies issues for division and anti-division queries. The objective is to suggest several algorithms which are suited to a reasonably efficient evaluation of such queries (subsections 5.1 and 5.2) and to assess the extra cost with respect to queries involving no preferences (subsection 5.3).

5.1 Processing of Division Queries

Three algorithms implementing formula 7 are successively described. The first algorithm is based on a sequential scan of the dividend (SSD). 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, inside a cluster the tuples (x, a) are ordered increasingly on A. This is performed by the query:

select * **from** r **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, one checks the layers in decreasing order of their importance. The process stops as soon as the current element x is not associated with all of the values from a layer V_i. Three cases can appear: i) x is associated with all of the values from all the layers of the divisor and it gets the preference level l_1, ii) the stop occurs while checking layer V_i whose importance is not maximal (i > 1) and x gets the preference level $rev(l_i) = l_{n+2-i}$, iii) the stop occurs while checking layer V_1 and x gets the level l_{n+1} meaning that it is rejected.

In the second algorithm, data accesses are guided by the divisor (AGD). Thus, 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, the associations with the different layers are checked by means of an SQL query involving the aggregate count. Again, a layer is checked only if the layers of higher importance had all 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 V_1 which are associated with x (whose current value is denoted by :x below) in r by means of the query:

select count(*) **from** r **where** X = :x **and** A **in** (**select** A **from** V_1).

If the value returned equals the cardinality of V_1, one checks layer V_2 by means of a similar query, and so on. The loop stops as soon as a missing association with the current layer is detected. The preference level assigned to x is computed according to the same principle as in the previous algorithm.

The last strategy relies on a series of regular division queries (SRD). It consists of two steps: i) to process as many regular division queries as there are layers in the divisor, and ii) to merge the different results and compute the final preference degrees. The algorithm has the following general shape:

step 1: for each layer V_i of the divisor, one processes a division query which retrieves the x's which are associated in r with all of the values from V_i. The layers are examined in decreasing order of their importance and an element x is checked only if it belongs to the result of the query related to the previous layer.

step 2: the results T_1, …, T_n of the previous division queries are merged by taking them in decreasing order of the corresponding layers. An element x which belongs to T_i (the result of layer V_i) but not to T_{i+1} gets the preference level l_{n-i+1} (assuming that there exists a table T_{n+1} which is empty). We have used an algorithm where the query (in step 2) rests on an outer join.

5.2 Processing of Anti-division Queries

Each of the previous methods can be adapted so as to apply to anti-division queries. In the SSD algorithm, after running the query:

select * **from** r **order by** X, A,

using the table connecting each value of the divisor with its layer, it is possible to identify the occurrence(s) of unwanted values. At the end of a cluster of tuples, the level of preference assigned to the current element x is determined by checking the layers in decreasing order of their importance. Here also, the process can stop as soon as the current element x is associated with one of the values from a layer V_i (x receives the level l_{n+1} if i = 1, l_{n+2-i} if i ∈ [2, n]) and if no undesired association is detected, x is assigned the level l_1.

Similarly, the algorithm AGD is transformed as follows. As originally, the distinct values of attribute X present in r are retrieved by means of the query:

select distinct X **from** r.

Then, for each value x, the number of A-values from V_1 (the totally excluded values specified in the divisor) which are associated with x in r, is computed by means of the query:

select count(*) **from** r **where** X = :x **and** A **in** (**select** A **from** V_1).

If the value returned is zero, one checks layer V_2 by means of a similar query, and so on. The loop stops as soon as an unwanted association with the current layer is detected. The preference level assigned to x is computed according to the same principle as in the previous algorithm.

The strategy SRD now means "a series of regular differences". The first step rests on queries of type:

(**select** X **from** r) **differ** (**select** X **from** r **where** A **in** (**select** B **from** V_i))

for each set V_i corresponding to a layer of the divisor. The second step takes all the successive pairs of results produced previously in order to assign the preference level l_{n-i+1} to an element x which belongs to the result of layer V_i but not to that of layer V_{i+1}.

5.3 Experiments

As mentioned previously, the objectives of the experimentation are mainly to assess the additional processing cost related to the handling of preferences and to

compare the performances of the 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.

A generic stratified division (respectively anti-division) query has been run on dividend relations of 300, 3000 and 30000 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 division (respectively anti-division) query without preferences, where the divisor is made of the sole first layer of the divisor (which corresponds to a "hard constraint" as mentioned before). So doing, we can assess the extra cost related only to the "preference part" of the query, i.e., to the presence of the non-mandatory layers. The reference division query has been evaluated using three methods: i) sequential scan of the dividend (i.e., algorithm SSD without preferences, denoted by REF1), ii) access guided by the divisor (i.e., algorithm AGD without preferences, denoted by REF2), iii) algorithm REF3 based on a query involving a "group by" clause and a counting, as in the first step of algorithm SRD. The reference anti-division query has been evaluated using these same methods. However, it is worth notocing that REF1 shows the same performances for both the division and anti-division since the only difference lies in the final comparison of the cardinality of the current subset (with that of the layer for the division and with 0 for the anti-division. Moreover:

- we used synthetic data generated in such a way that the selectivity of each value b from the divisor relatively to any x from the dividend is equal to 75% in the case of a division query (for a given value b from the divisor and a given x from the dividend, tuple (x, b) has three chances out of four to be present in the dividend), and it is equal to 25% in the case of an anti-division query,
- each algorithm was run 8 times so as to avoid any bias induced by the load of the machine,
- the time unit equals 1/60 second.

The results obtained for the division are reported in the table hereafter:

Size of the dividend	300	3000	30000
REF1	15.8	144.6	1451
REF2	49.7	570	15536
REF3	11.4	40.5	361.9
SSD	99	1011	10451
AGD	84	1035	29927
SRD	89	332	2923
Number of answers	15	172	1693

One can notice that:

- among the reference methods for non-stratified operations, the most efficient is by far REF3. This is due to the fact that is is based on a single query involving a "group by" clause, which is very efficiently optimized by the system,

- the processing time of the algorithms based on a sequential scan of the dividend (i.e., SSD and REF1) vary linearly w.r.t. the size of the dividend, contrary to those from the second family (REF2 and AGD); as to algorithm SRD (implemented with an outer join), its complexity shows some linearity as soon as the size of the dividend is above a certain threshold (which means that there is a fixed cost attached to it, which depends on the number of layers of the divisor),
- algorithm SSD becomes better than AGD as soon as the size of the dividend is over 1000 tuples. But the best algorithm is SRD (implemented with an outer join), which outperforms all the others as soon as the dividend contains more than 300 tuples. It is worth noticing that the ratio between SRD and REF3 is almost constant (around 8), which is due to the fact that SRD performs one query of type REF3 per layer (here 5), plus the combination of the intermediate results.

To sum up, it appears that algorithm SRD based on an outer join is the most efficient, except for very small sizes of the dividend where AGD is slightly better. However, the extra cost of SRD with respect to the most efficient reference algorithm, namely REF3, is still important (the multiplicative factor is around 8).

The results obtained for the anti-division are reported in the next table:

Size of the dividend	300	3000	30000
REF1	15.8	144.6	1451
REF2	41.4	400.7	4055
REF3	13.2	81.4	760.2
SSD	108.6	960.5	10418
AGD	54.2	645.2	6315
SRD	106	375.1	4353
Number of answers	37	427	4365

These results show that:

- among the reference methods for non-stratified anti-divisions, REF3 is much more efficient than REF2; the fact that it outperforms REF2 by such a large margin means that the DBMS is not efficient at optimizing nested queries,
- the performances of REF2, AGD and SSD vary linearly with respect to the size of the dividend. As to REF3 and SRD, their complexity is less than linear.

It turns out that the best algorithm for stratified anti-divisions is SRD, which is significantly better than AGD, itself much more efficient than SSD. However, the extra cost of SRD with respect to 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 a division or anti-division 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. For instance, in the case of the anti-division, the extra cost attached to SRD with respect to REF3 is explainable by the fact that SRD processes five regular anti-division queries (one for each layer) instead of one for REF3, and then has to merge the results of these queries. Consequently, if the stratified division or anti-division functionality were to be integrated into a commercial DBMS, it is clear that it would have to be handled by the optimizor at an internal level, and processed as one query, according to the format given in Subsection 3.2 and in such a way that the evaluation of a given x is done in one step.

6 Conclusion

In this article, we dealt with division and anti-division queries involving user preferences expressed in an ordinal way. The principle consists in using a divisor made of a hierarchy of layers. The first layer corresponds to a set of mandatory values, whereas the other layers are used to discriminate among the elements in the result. So doing, the result is no longer a flat set but a list of items provided with a level of satisfaction. It has been shown that the stratified division (respectively anti-division) delivers a result that can be characterized as a quotient (respectively an anti-quotient).

Besides, some experimental measures have been carried out in order to assess the feasibility of such extended division or anti-division queries. Even though these measures still need to be completed, they show that the additional cost induced by the stratified nature of the divisor is quite high (multiplicative factor from 5 to 8 with respect to the relative classical operation) 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 division queries externally, as we did here.

This work opens several perspectives, among which : i) the enrichment of division or anti-division queries whose semantics could be disjunctive with respect to the role of the layers or even based on the lexicographic order as mentioned in the end of subsection 3.1, ii) making complementary experiments in order to take into account larger sizes for both the dividend and the divisor (in particular in the case where the divisor is not specified extensionally by the user, but results from subqueries), iii) the investigation of strategies suited for processing mixed queries in the sense of section 4, along with the corresponding experiments, and iv) checking whether the results obtained with Oracle are confirmed when another DBMS (e.g. PostgresQL or MySQL) is used.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline operator. In: Proc. of the 17th International Conference on Data Engineering, pp. 421–430 (2001)
2. Bosc, P., Pivert, O., Rocacher, D.: About quotient and division of crisp and fuzzy relations. Journal of Intelligent Information Systems 29, 185–210 (2007)

3. Bosc, P., Pivert, O.: On a parameterized antidivision operator for database flexible querying. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 652–659. Springer, Heidelberg (2008)
4. Bouchon-Meunier, B., Dubois, D., Godo, L., Prade, H.: Fuzzy sets and possibility theory in approximate and plausible reasoning. In: Bezdek, J., Dubois, D., Prade, H. (eds.) Fuzzy Sets in Approximate Reasoning and Information Systems, pp. 15–190. Kluwer Academic Publishers, Dordrecht (1999)
5. Bruno, N., Chaudhuri, S., Gravano, L.: Top-k selection queries over relational databases: mapping strategies and performance evaluation. ACM Transactions on Database Systems 27, 153–187 (2002)
6. Chomicki, J.: Preference formulas in relational queries. ACM Transactions on Database Systems 28, 427–466 (2003)
7. Dubois, D., Prade, H.: A theorem on implication functions defined from triangular norms. Stochastica 8, 267–279 (1984); Also in: Dubois, D., Prade, H., Yager, R.R. (eds.) Readings in Fuzzy sets for Intelligent Systems, pp. 105–112. Morgan & Kaufmann, San Francisco (1993)
8. Dubois, D., Prade, H.: Using fuzzy sets in flexible querying: why and how. In: Proc. of the Workshop on Flexible Query-Answering Systems, pp. 89–103 (1996)
9. Dubois, D., Prade, H.: Handling Bipolar Queries in Fuzzy Information Processing. In: Galindo, J. (ed.) Handbook of Research on Fuzzy Information Processing in Databases. Information Science Reference, Hershey (2008)
10. Dubois, D., Prade, H.: An introduction to bipolar representations of information and preference. International Journal of Intelligent Systems 23, 866–877 (2008)
11. Hadjali, A., Kaci, S., Prade, H.: Database preferences queries – A possibilistic logic approach with symbolic priorities. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 291–310. Springer, Heidelberg (2008)
12. Kießling, W., Köstler, G.: Preference SQL – Design, implementation, experiences. In: Proc. 28th Conference on Very Large Data Bases, pp. 990–1001 (2002)
13. Lacroix, M., Lavency, P.: Preferences: putting more knowledge into queries. In: Proc. 13th Conference on Very Large Data Bases, pp. 217–225 (1987)