Computing Appropriate Representations for Multidimensional Data

Yeow Wei Choong
LI - HELP Institute
Malaysia
choong_yw@help.edu.my

Dominique Laurent
LI - Université F. Rabelais
Tours - France
laurent@univ-tours.fr

Patrick Marcel
LI - Université F. Rabelais
Tours - France
marcel@univ-tours.fr

May 7, 2002

Abstract

On-Line Analytical Processing (OLAP) provides an interactive query-driven analysis of multidimensional data based on a set of navigational operators like roll-up or slice and dice. In most cases, the analyst is expected to use these operations intuitively to find interesting patterns in a huge amount of data of high dimensionality.

In this paper, we propose an approach to enhance this analysis by preparing the data set so that the analyst can explore it in a more systematic and effective manner. More precisely we define a measurement of the quality of the representation of multidimensional data and we present a framework for investigating the computation of appropriate representations. We identify the problems of computing such representations and study them w.r.t. an OLAP restructuring operator.

Keywords: Multidimensional Database, On-Line Analytical Processing, Representation.

1 Introduction

On-Line Analytical Processing (OLAP) [1, 3] technology provides a platform for analysing data according to multiple dimensions (e.g., product, location, time) and multiple granularities (e.g., city, district, country). Data is presented under the form of a cube. A cube can be seen as a set of cells, and a cell represents the association of a *measure* with one *member* in each dimension. For example, if dimensions are products, stores and days, the measures of a particular cell can be the sales of one product in a particular store on a given day.

The user is provided with a set of operators for navigating through the data set to identify interesting and relevant patterns. This navigation is a query-driven process, and a number of proposals have investigated formal models and languages to this end (see [6, 10] for surveys). Obviously, as the size and the dimensionality of the data set increase, the whole process becomes very tedious and complex. To deal with this complexity, it has been recently pointed out [9, 8] that the manual effort spent in analysis could be reduced by anticipating the user strategy.

In typical OLAP analysis, the strategy is mostly based on observing the measures, whereas most of the OLAP restructuring operators are parameterized by members.

For example, consider the cube of Figure 1 (a). This cube displays sales of beer, milk, soda, water and wine in different continents during year 2000. Assume that the analyst wants to visualize the sales having the highest values on the one hand, and the lowest values on the other hand. The way the cube is represented does *not* provide such a visualization easily, because the cells are displayed according to the lexical ordering of the members in each dimension, and *not* according to the measures. On the other hand, it can be seen that the cube of Figure 1 (b) contains the same information as that of Figure 1 (a), but displays the sales in an appropriate way for the analyst. Indeed, the lowest values of sales are located down-left in the cube, whereas the highest values are located top-right. It should be noticed from the example that a clear distinction between a cube and its representation is needed here. This is precisely what we propose in this paper.

In our approach, the representation of an n-dimensional cube consists of n functions, each of them being a numbering of the members of a dimension. Given a cube C and one of its representations R, we assume that C is displayed according to the ordering defined by R. For example, the numbering defining the dimension product for the representation (a) of Figure 1 associates beer with 1, milk with 2, soda with 3, water with 4 and wine with 5. The numbering defining this dimension for the representation (b) associates beer with 1, water with 2, milk with 3, wine with 4 and soda with 5.

Representation (b) of Figure 1 can be interactively constructed by the user from representation (a) via some restructuring operators proposed in the OLAP context. These operators allow users to change the representation of the cube but *not* its logical structure: the association between one member in each dimension and the measure is preserved. For example, the *switch* operator [5, 6] allows users to exchange the position of 2 members on the axis corresponding to a given dimension while preserving the cells. The order over the columns in representation (b) of Figure 1 can be obtained from representation (a) by 1/ switching *milk* and *soda*, 2/ switching *soda* and *wine* and 3/ switching *wine* and *water*.

As a contribution to automating OLAP analysis, we propose to study how to arrange the representation of the cube according to its measures. We believe

year 2000 sales					
Africa	3	5	6	3	5
America	4	6	7	5	7
Asia	2	4	6	2	5
Europe	4	5	7	4	6
	beer	milk	soda	water	wine

(a)

year 2000 sales					
America	4	5	6	7	7
Europe	4	4	5	6	7
Africa	3	3	5	5	6
Asia	2	2	4	5	6
	beer	water	milk	wine	soda

(b)

Figure 1: A 2-dimensional cube before and after restructuring

that computing appropriate representations can help to identify patterns which would otherwise remain unknown to the user. This contributes also to obtain the result of typical OLAP ranking queries like top-n.

We notice that even dimensions that are inherently ordered like e.g., time, can be rearranged so as to make some patterns apparent. For example, consider the cube of Figure 2 that displays monthly sales of chocolate in various regions. In representation (a) the months are depicted in the standard ordering, whereas in representation (b) the ordering is imposed by the measures. Representation (b) can be exploited by the analyst to discover that e.g., chocolate sales are the highest around new year and easter.

This paper presents a framework for investigating the quality of cube representations. Obviously there may be several ways of considering what an appropriate representation is and how to reach it.

Concerning appropriate representations, we define a cell as *misplaced* if there exists at least one other cell with lower measure and with greater or equal numberings in all dimensions. For example, the cell containing the sales of soda in Europe is misplaced in representation (a) of Figure 1. Indeed the cell containing the sales of water in Europe 1/ contains a lower measure and 2/ has greater numbering in dimension product, and the same numbering in dimension continent. We call *appropriate* the representations having the least number of misplaced cells, and we study the problem of finding these representations. To this end, we show that the *switch* operation proposed in the context of OLAP [5, 6] is the basic operator that allows us to compute these representations.

The main results of the paper are:

- First, we define a measurement for the quality of the representation of a

chocolate sales												
east	8	5	4	6	6	3	1	0	2	4	5	7
north	9	5	5	7	7	4	1	1	3	4	6	8
south	7	3	2	5	4	1	0	0	1	2	3	5
west	6	3	3	6	5	2	0	0	1	2	4	7
	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec

(a)

chocolate sales												
north	1	1	3	4	4	5	5	6	7	7	8	9
east	0	1	2	3	4	4	5	5	6	6	7	8
west	0	0	1	2	2	3	3	4	5	6	7	7
south	0	0	1	1	2	2	3	3	4	5	5	6
	aug	jul	sep	jun	oct	mar	feb	nov	may	apr	dec	jan

(b)

Figure 2: Restructuring a 2-dimensional cube with an inherently ordered dimension

cube by computing the number of its misplaced cells, and

- Second, we identify several problems related to the representation of cubes w.r.t. this measurement:
  - Test for the existence of a representation with no misplaced cells (called a *perfect* representation. Representation (b) of Figure 1 is an example of a perfect representation). In this case, we give a sequence of restructuring operations for reaching such a representation, if it exists, and we compute the total number of perfect representations. We show that this problem is polynomial with respect to the size of the cube.
  - If no representation having no misplaced cells exists, we outline the problems of finding representations having the least number of misplaced cells.

Related work A variant of the switch operator has been defined in [5] in the context of 2-dimensional tabular databases. This operator allows users to exchange two rows of a matrix regardless of the status of the rows (members and measures are treated uniformly). However, in [5], the authors did not consider the problem of using this operation to restructure matrices in a more appropriate way for the user.

In [7], Mäkinen and Siirtola study the problem of reordering tabular representations by interchanging rows and columns. They show that in general this problem is NP-complete. The problem of computing a perfect representation of a n-dimensional cube we consider in this paper can be seen as a particular case of the problem studied by Mäkinen and Siirtola. In our approach, the definition of perfect representation allows to propose polynomial time algorithms for computing such representations.

In [8, 9], Sarawagi & al. propose a new set of operators for reducing the number of roll-ups and drill-downs (changing the granularity of the representation) needed to discover abnormalities or to explain drops or increases in the values of the measures. Their work concentrates on the "vertical" aspect of OLAP data where the link between aggregated data is exploited.

While our motivations are essentially the same as the authors of [8, 9], our work is orthogonal to their approach in the sense that we concentrate on the "horizontal" aspect of OLAP data. Our goal is to reduce the number of restructuring operations used during the analysis. We are interested in the representation of the data at a given level and we do not take granularity into account.

The paper is organized as follows. The next section introduces basic definitions on the multidimensional data model, on the notion of representation, and on the quality measurement. In Section 3, we define the problems of finding appropriate representations, and in Section 4 we study and solve the particular problem of computing a perfect representation. We conclude and discuss future work in Section 5.

Due to lack of space, proofs are omitted and can be found in [2].

2 Preliminaries

In this section, we give the formal definitions of the concepts used in this paper. The terminology concerning OLAP (members, measures, ...) is that of [6].

2.1 The multidimensional model

In our model, we distinguish a cube from its representation. Intuitively, a cube is a logical multidimensional structure, and a representation can be seen as a way of displaying the cube to the analyst.

Definition 2.1 An n-dimensional cube, or simply a cube, is a tuple $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ where

- C is the name of the cube,
- $dom_1, \ldots, dom_n$ are n finite sets of symbols for the members associated with dimension $1, \ldots, n$, respectively,
- let dom_{mes} be a finite totally ordered set of measures. Let $\perp$ be a constant not in dom_{mes} used to represent null values. Then $dom_m = dom_{mes} \cup \{\perp\}$, and $\perp$ cannot be compared to the elements of dom_{mes},

- m_C is a mapping from $dom_1 \times \ldots \times dom_n$ to dom_m.

In what follows, we denote by $|dom_i|$ the cardinality of dom_i for every dimension i.

Definition 2.2 A *representation* $R_C = \{rep_1, \ldots, rep_n\}$ of a cube $\langle C, dom_1, \ldots, dom_n, dom_m, m_C\rangle$ is a set of n bijective mappings $rep_1, \ldots, rep_n$ such that for every $i \in [1, n]$, rep_i is a mapping from dom_i to the initial segment of $\mathbb{N}$ $\{1, \ldots, |dom_i|\}$. The set of all different representations of a cube $\langle C, dom_1, \ldots, dom_n, dom_m, m_C\rangle$ is denoted by S_{R_C}.

Given a representation R of a cube C, for every i in $[1, n]$ and for every $m \in dom_i$, $rep_i(m)$ is called the *position* of m on dimension i in R.

Note that the notion of representation we propose does not associate a dimension with a particular axis (e.g., for 2-dimensions the vertical axis or the horizontal axis) for displaying the members. Only the relative position of the members in one dimension is relevant. On each dimension i, the values of dom_i are ordered according to their representation rep_i. In other words, placing value m of dom_i at the j^{th} position means that $rep_i(m) = j$.

The cardinality of S_{R_C} (i.e., the number of different representations of C) is the product of the number of different *rep* mappings for each dimension. Therefore, we have $|S_{R_C}| = \Pi_{i\in[1,n]}(|dom_i|!)$.

Example 2.1 Consider the 2-dimensional cube $\langle C, \{a, b\}, \{x, y\}, \{1, 2, 3, 4\}, m_C\rangle$, where $m_C(a, x) = 1, m_C(a, y) = 2, m_C(b, x) = 3, m_C(b, y) = 4$. The number of different representations of this cube is $2! \times 2! = 4$. These representations, called R_1, R_2, R_3 and R_4 respectively, are displayed below. The representation R_1 is the set $\{rep_1, rep_2\}$ where rep_1 and rep_2 are defined by $rep_1(a) = 2, rep_1(b) = 1, rep_2(x) = 1, rep_2(y) = 2$.
As a convention throughout the paper, in this 2-dimensional example and the other examples, the horizontal axis is oriented from left to right and the vertical axis is oriented from bottom to top.

R_1

a	1	2
b	3	4
	x	y

R_2

b	3	4
a	1	2
	x	y

R_3

b	4	3
a	2	1
	y	x

R_4

a	2	1
b	4	3
	y	x

We note that all of these representations are different representations of the same cube C. Indeed, in C, we have for instance $m_C(a, y) = 2$, which holds in R_1, R_2, R_3 and R_4. The representations differ only in the ordering according to which the rows and the columns are displayed. On the other hand, the table below is *not* a representation of C since for instance, the measure associated with $\langle a, y\rangle$ is not 2.

a	1	3
b	2	4
	y	x

$\square$

A cell is the association of a member in each dimension with a measure.

Definition 2.3 A *cell* c of a cube $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$, is a tuple $\langle m_1, \ldots, m_n, m \rangle$ where $\forall i \in [1, n], m_i \in dom_i, m \in dom_m$ and $m_C(m_1, \ldots, m_n) = m$.

A cell c of a cube C is an element of the graph of the function m_C. Therefore we feel allowed to consider a cube C as the set of its cells, and we write $c \in C$ to mean that c is a cell of C. A cell containing $\perp$ is called an *empty cell.*

Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube, $R_C = \{rep_1, \ldots, rep_n\}$ a representation of C and $c = \langle m_1, \ldots, m_n, m \rangle$ a cell of C. The position of c in C according to R_C is the tuple $\langle x_1, \ldots, x_n \rangle$ where $rep_i(m_i) = x_i$, for every $i \in [1, n]$.

Note that the position of a cell in a representation is only based on the functions rep_i. This means that the position is invariant w.r.t. a rotation of the cube.

Example 2.2 Consider representation R_1 of Example 2.1. For this representation, the position of the cell $c_1 = \langle a, x, 1 \rangle$ is the tuple $\langle 2, 1 \rangle$, and the position of the cell $c_4 = \langle b, y, 4 \rangle$ is the tuple $\langle 1, 2 \rangle$. $\square$

2.2 Cell arrangement

We can now define the ordering over cell positions.

Definition 2.4 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube and $R_C = \{rep_1, \ldots, rep_n\}$ a representation of C. Let $c = \langle m_1, \ldots, m_n, m \rangle$ and $c' = \langle m'_1, \ldots, m'_n, m' \rangle$ be two cells of C. We define the relation $\prec_{R_C}$ as a partial ordering over cells by $c \prec_{R_C} c' \iff \forall i \in [1, n], rep_i(m_i) \leq rep_i(m'_i)$.

Example 2.3 Consider the cube of Example 2.1. This cube has cells $c_1 = \langle a, x, 1 \rangle$, $c_2 = \langle a, y, 2 \rangle, c_3 = \langle b, x, 3 \rangle$, and $c_4 = \langle b, y, 4 \rangle$. Considering the representation R_1, we have $c_3 \prec_{R_1} c_1$, $c_3 \prec_{R_1} c_2$, $c_3 \prec_{R_1} c_4$, $c_1 \prec_{R_1} c_2$, $c_4 \prec_{R_1} c_2$. Note that c_1 cannot be compared with c_4 w.r.t. $\prec_{R_1}$. $\square$

Now, we define what we call a *misplaced* cells.

Definition 2.5 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube and R_C a representation of C. A cell $c = \langle m_1, \ldots, m_n, m \rangle$ of C is *misplaced w.r.t.* R_C if $m \neq \perp$, and

- $\exists c_1 = \langle m'_1, \ldots, m'_n, m' \rangle \in C$ such that $c \prec_{R_C} c_1$ and $m > m'$, or
- $\exists c_2 = \langle m''_1, \ldots, m''_n, m'' \rangle \in C$ such that $c_2 \prec_{R_C} c$ and $m'' > m$.

For a cube C, a representation R_C of C and a cell $c \in C$, we define the function $f_{R_C}(c) = 1$ if c is misplaced w.r.t. R_C, 0 otherwise.

Then, the measurement we propose is simply the total number of misplaced cells in a cube.

Definition 2.6 Given a cube C and a representation R_C of C, we define $M_{R_C}(C)$ by $M_{R_C}(C) = \sum_{c_i \in C} f_{R_C}(c_i)$. $M_{R_C}(C)$ is the total number of misplaced cells in C w.r.t. the representation R_C.

With this measurement, we can characterize the representations of a cube.

Definition 2.7 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube and let S_{R_C} be the set of all representations of C.

- A representation R_C of C is a *Perfect Representation (PR)* if $M_{R_C}(C) = 0$.
- A representation R_C of C is an *Optimal Representation (OR)* if $\nexists R'_C \in S_{R_C}, M_{R'_C}(C) < M_{R_C}(C)$.

Obviously for a given cube, a PR may not exist, and there exists at least one OR. Moreover, if a PR exists it may not be unique.

Example 2.4 Consider the representations R_1 and R_2 of the cube C in Example 2.1. The number of misplaced cells in R_1 is $M_{R_1}(C) = 4$, whereas R_2 is a PR of C (i.e., $M_{R_2}(C) = 0$). Now if we consider the table below as a representation of a cube, there exists no PR of this cube. This is so because the lowest and highest measures are on the same row. Since this must hold in every representation of the cube although this cannot hold in any PR, this cube has no PR.

2	3
1	4

□

3 The problems

In this section we study the problems of using the measurement of Definition 2.6 to find appropriate representations of cubes. We first define the operation used to change the representation of a cube.

3.1 Arranging the cube

The *switch* operation [5, 6] is an OLAP operation that consists in interchanging the positions of two members of a dimension of a cube. In our framework, the switch operation is the basic operation to go from one representation of a cube to another.

Definition 3.1 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube and S_{R_C} the set of all representations of C. A switch on dimension j of members p and q, denoted by $switch(j, p, q)$, is a function from S_{R_C} to S_{R_C} such that, for every $R_C = \{rep_1, \ldots, rep_n\}$ in S_{R_C}, $switch(j, p, q)(R_C) = R'_C$ where $R'_C = \{rep'_1, \ldots, rep'_n\}$ is defined by:

- for every i in $[1, n]$, if $i \neq j$, then $rep_i = rep'_i$,
- $rep_j(p) = rep'_j(q)$ and $rep_j(p) = rep'_j(q)$
- for every m in dom_j different than p and q, $rep_j(m) = rep'_j(m)$.

Notice that according to the first point of Definition 3.1, applying a switch operation on two members in one dimension leaves unchanged the positions of the members in the other dimensions.

Example 3.1 Consider the cube of Example 2.1 and its representations R_1 and R_2. R_2 is the result of the operation $switch(1, a, b)$ applied to R_1. In other words, $R_2 = switch(1, a, b)(R_1)$. □

Definition 3.2 A finite composition of switches is called an *arrangement*.

Example 3.2 Consider the representations of Example 2.1. We have $switch(1, a, b)(R_1) = R_2$, $switch(2, x, y)(R_2) = R_3$. Thus $switch(2, x, y)(switch(1, a, b)(R_1)) = R_3$. Therefore, $R_3 = arr(R_1)$ where arr is the arrangement defined by $switch(2, x, y) \circ switch(1, a, b)$. □

As for the switch operation, it is obvious that applying an arrangement involving only one dimension leaves the position of the members of the other dimensions unchanged.

The following proposition shows that all representations of a cube can be obtained through arrangements.

Proposition 3.1 *Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube and let S_{R_C} be the set of all representations of C. Given any two representations R_1 and R_2 of S_{R_C}, there exists an arrangement arr such that $arr(R_1) = R_2$.*

3.2 The Perfect Representation problem

We are interested in the following problem that we call the Perfect Representation (PR) problem: For a given cube and a given representation of this cube, test whether there exists at least one PR, and if so, compute one PR. If more than one PR exists, then compute the total number of PRs.

The approach we use to present the algorithms for solving the PR problem is the following: We first consider the simple case of a cube for which at least one row in each dimension contains no duplicates and no null values. This gives rise to a basic algorithm for solving the PR problem. Then we consider cubes for which no such row exists, and we concentrate on dealing with duplicates in the absence of null values. Then we concentrate on dealing with null values in the absence of duplicates. Finally, we give the algorithm that solves the PR problem in the general case of cubes where duplicates and null values can appear in any rows.

We now introduce formally the notion of row for the sake of readability. Intuitively, a row is a set of cells where all coordinates but one are fixed.

Definition 3.3 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube. A row r in dimension k is the set of cells of C $\{\langle m_1, \ldots, m_{k-1}, j, m_{k+1}, \ldots, m_n, m\rangle \mid j \in dom_k\}$. This row is identified by the tuple $\langle m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n\rangle$, where $m_i \in dom_i$ for every i in $[1, k-1] \cup [k+1, n]$.

As for cells and cubes, we feel allowed to denote by $r \in C$ the fact that every cell belonging to r also belongs to C.

Given a representation $R = \{rep_1, \ldots, rep_k, \ldots, rep_n\}$ of a cube, a row r in dimension k, and a cell $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle$ of r, the position of c in r is simply $rep_k(m_k)$.

Definition 3.4 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C \rangle$ be a cube, let R_C be a representation of C. A row r is sorted in R_C if $\forall c = \langle m_1, \ldots, m_n, m\rangle, c' = \langle m'_1, \ldots, m'_n, m'\rangle \in r$ with $m \neq \perp$ and $m' \neq \perp$, $c \prec_{R_C} c' \Longrightarrow m \leq m'$. Otherwise the row r is unsorted.

Given a representation R and a row r in dimension k, sorting r is simply changing rep_k. Note that in a sorted row, empty cells can appear anywhere. Based on usual algorithms for sorting one-dimensional arrays, we have the following lemma.

Lemma 3.2 *For a given cube C, a given representation R_C of C and a given row r there is an arrangement that sorts the row r.*

If r is a row and R is a representation, sorting a row means applying an arrangement to R so that r is sorted in the resulting representation. Obviously, sorting a row in dimension k implies assigning a position to the members of dom_k.

Example 3.3 Consider Example 2.1. The row $\langle y \rangle$ is the set $\{\langle a, y, 2\rangle, \langle b, y, 4\rangle\}$. Moreover, this row is sorted in R_2. □

The following theorem, of which the proof is an immediate consequence of Definition 2.5, is the basic result on which rely all proofs of the subsequent propositions and corollaries.

Theorem 3.3 *A representation of a cube is a PR if and only if every row in every dimension is sorted.*

This theorem implies that each dimension of a cube can be processed independently when computing a PR. In the following section, we propose algorithms for solving the PR problem in the following cases:

- Case 1: A row with no duplicates and no null values exists in each dimension
- Case 2: Each row of the cube can contain duplicates but no null values,
- Case 3: Each row of the cube can contain null values but no duplicates,
- Case 4: Each row of the cube can contain both duplicates and null values.

4 Solving the PR problem

4.1 Case 1: A row with no duplicates and no null values exists in each dimension

We first consider the case where at least one row in every dimension contains no duplicates and no null values. In this case, we show that the existence of a PR can be efficiently tested by sorting only one row in each dimension. Moreover, when a PR actually exists, it is unique and our method computes it. Our method is based on the following propositions and corollary.

Proposition 4.1 *Let C be a cube such that at least one row in every dimension contains no duplicates and no null values. There exists at most one PR of C.*

Proposition 4.2 *Let C be a cube such that at least one row in every dimension contains no duplicates and no null values. If there exists a representation such that for one dimension, a row r containing no duplicates and no null values is sorted and another row r' is unsorted, then there exists no PR.*

Corollary 4.3 *Let C be a cube such that at least one row in every dimension contains no duplicates and no null values, and for which a PR exists. Let R be a representation of C. If in R one row containing no duplicates and no null values is sorted in each dimension, then R is a PR.*

At this point, a simple algorithm can be given to solve the PR problem for a cube where one row in every dimension contains no duplicates and no null values.

Algorithm 4.1
Input: A representation of a cube C
Output: The PR of C or the indication "no PR"

for each dimension k of C do

    choose a row r in dimension k containing no duplicates and no null values

    sort r

    for every other row r' in dimension k do

        check if r' is sorted

        if r' is unsorted then

            exit with output "no PR"

Based on the previous propositions and corollary, we can present the following Theorem.

Theorem 4.4 *Let C be a cube such that at least one row in every dimension contains no duplicates and no null values. Let R be a representation of C. If the call to Algorithm 4.1 outputs "no PR" then there exists no PR of C. Otherwise, the output is the only PR of C.*

Algorithm 4.1 is polynomial in the number of cells of the cube, since it only sorts one-dimensional arrays (one row in each dimension) or tests if one-dimensional arrays are sorted.

4.2 Case 2: Dealing with duplicates

We consider in this section cubes for which duplicates but no null values can appear in a row. In this case, sorting a row in each dimension is necessary but is no more sufficient for computing a PR. For instance, consider the cube of which representations R_1 and R_2 are depicted below. Sorting row $\langle a \rangle$ may lead to representation R_1 which is not perfect, since row $\langle b \rangle$ is unsorted. On the other hand, sorting row $\langle b \rangle$ leaves row $\langle a \rangle$ sorted and gives a PR.

R_1

b	4	3
a	1	1
	x	y

R_2

b	3	4
a	1	1
	y	x

Definition 4.1 Let $C = \langle dom_1, \ldots, dom_n, dom_m, m_C\rangle$ be a cube, $R = \{rep_1, \ldots, rep_n\}$ be a representation of C, and $r = \langle m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n\rangle$ be a row of dimension k. A sequence of duplicates in r is an interval $I = [i_1, i_2]$ of $\mathbb{N}$ such that for all $i, j \in I$, $m_C(m_1, \ldots, m_{k-1}, rep_k^{-1}(i), m_{k+1}, \ldots, m_n) = m_C(m_1, \ldots, m_{k-1}, rep_k^{-1}(j), m_{k+1}, \ldots, m_n)$. Given a row r, a sequence of duplicates I in r is maximal if there is no sequence of duplicates J in r such that $I \subset J$.

Given a representation of a cube, a row r in dimension k, and an interval I of $\mathbb{N}$, the contiguous part of r w.r.t. I is defined by:

$$r_I = \{c \in r \mid c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle \text{ and } rep_k(m_k) \in I\}.$$

Proposition 4.5 *Let C be a cube and R a representation of C. Let r be a sorted row in R containing p maximal sequences of duplicates $I_1, \ldots, I_p$. If there exists a row r' in the same dimension that is still unsorted after having sorted every contiguous part of r' w.r.t. $I_1, \ldots, I_p$, then there exists no PR.*

Example 4.1 Consider a cube of which representations R_1 and R_2 are depicted below. Suppose we sort row $\langle b\rangle$ first, so as to obtain representation R_1. The next step is to sort row $\langle a\rangle$ *without affecting row* $\langle b\rangle$. The only possibility is to switch members x and y. Once done, we obtain representation R_2 where row $\langle a\rangle$ is still unsorted. Therefore, according to Proposition 4.5 above, there is no PR of this cube.

R_1

a	4	3	1
b	1	1	2
	x	y	z

R_2

a	3	4	1
b	1	1	2
	y	x	z

□

At this point we can give an algorithm that outputs a PR of a cube where the rows contain duplicates but no null values, if any. Otherwise, the algorithm indicates that no PR exists.

Algorithm 4.2
Input: A representation of an n-dimensional cube C
Output: A PR of C or the indication "no PR"
Variable: Two sets D and D' of sequences of duplicates

for each dimension k of C do

let $D = \{I\}$ with $I = [1, |dom_k|]$

choose a row r in dimension k

repeat until every row is marked

sort r_I for every $I \in D$

check if r is sorted

if r is unsorted then

    exit with output "no PR"

else

    for each I in D do

        $D' = \emptyset$

        compute $I_1, \ldots, I_p$ the sequences of duplicates in r_I

        $D' = D' \cup \{I_1, \ldots, I_p\}$

    $D = D'$

    mark r

    choose an unmarked row r

The following Theorem is a consequence of Proposition 4.5.

Theorem 4.6 *Let C be a cube for which each row can contain duplicates but no null values. Let R be a representation of C. If the call to Algorithm 4.2 outputs "no PR" then there exists no PR of C. Otherwise, the output is a PR of C.*

It is easy to see that this algorithm is polynomial in the number of cells of the cube.

Computing the total number of PRs in this case. Given a cube C for which each row can contain duplicates but no null values, more than one PR of C might exist. To compute the total number of PRs in this case we need to define what are identical slices in this context. We begin with the definition of a slice.

Definition 4.2 Let $\langle C, dom_1, \ldots, dom_n, dom_m, m_C\rangle$ be a cube. A slice s in dimension k is the set of cells of C $\{\langle j_1, \ldots, j_{k-1}, m_k, j_{k+1}, \ldots, j_n, m\rangle \mid j_i \in dom_i, i \in [1, k-1] \cup [k+1, n]\}$. This slice is identified by the member $m_k \in dom_k$.

As for cells, cubes and rows, we feel allowed to denote by $s \in C$ the fact that every cell belonging to slice s also belongs to C.

Definition 4.3 Let C be a n-dimensional cube. Let s and s' be two slices in dimension k. s and s' are identical for a given representation R of C if for each pair of cells $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle \in s$ and $c' = \langle m'_1, \ldots, m'_k, \ldots, m'_n, m'\rangle \in s'$, $rep_i(m_i) = rep_i(m'_i) \implies m = m'$, for all $i \in [1, k-1] \cup [k+1, n]$.

Based on Definition 4.3 above, we have the following proposition:

Proposition 4.7 *Let C be a n-dimensional cube and S_{R_C} be the set of every representation of C. Let R be a particular representation of C. Let s and s' be two slices in dimension k. If s and s' are identical for R then s and s' are identical for every representation $R' \in S_{R_C}$.*

It appears that the number of different PRs depends on the presence of identical slices within a dimension. Indeed

- switching two identical slices of a PR gives another PR, and
- computing a PR from a PR means applying an arrangement that preserves the order of every row, which can be done only if the arrangement involves only identical slices.

Proposition 4.8 *Let C be a cube of which a PR exists. Then there exists more than one PR of C if and only if C contains at least two identical slices in one of its dimensions.*

It is to be noticed that, if one looks only at the measures, every PR of a cube looks the same. The following corollary allows to compute the total number of PRs in this case.

Corollary 4.9 *Let C be an n-dimensional cube and R be a PR of C. Let p_i be the number of different sets of identical slices in dimension $i \in [1, n]$, and let $m^i_j, j \in [1, p_i]$, be the cardinality of each such set in dimension i. Then the total number of PRs is $\Pi_{i\in[1,n]}(\Pi_{j\in[1,p_i]}(m^i_j!))$.*

Therefore, outputting every PR is clearly not polynomial. However computing the total number of PRs can be done by counting the number of identical slices in each dimension, which is polynomial.

4.3 Case 3: Dealing with null values

In what follows, we assume that the rows of a cube can contain null values but no duplicates. We recall from Definition 2.5 that changing the position of a null value in a row does not affect the fact that the row is sorted or not. Thus, a row containing null values can be sorted in different ways, which results in more flexibility when looking for PRs. For instance, consider the cube of which representations R_1 and R_2 are depicted below. Sorting row $\langle a \rangle$ may lead to representation R_1 which is not perfect, since row $\langle b \rangle$ is unsorted. On the other hand, sorting row $\langle b \rangle$ does not affect the fact that row $\langle a \rangle$ is still sorted, and gives a PR.

R_1

	x	y
b	4	3
a	1	$\perp$

R_2

	x	y
b	3	4
a	$\perp$	1

This flexibility for sorting rows imposes that many combinations have to be explored when looking for PRs. For example, suppose we must arrange the following representation.

⋮	⋮	⋮	⋮	⋮	⋮
b	1	⊥	4	2	⊥
a	1	2	3	⊥	⊥
	v	w	x	y	z

Suppose we have sorted row $\langle a \rangle$ and we must sort row $\langle b \rangle$. As $\perp$ can be placed anywhere, the following two possibilities are valid.

b	1	2	⊥	4	⊥
a	1	⊥	2	3	⊥
	v	y	w	x	z

b	⊥	1	⊥	2	4
a	⊥	1	2	⊥	3
	z	v	w	y	x

We begin with an example to illustrate the intuition of the algorithm used to compute a PR in this case, if it exists.

4.3.1 Intuition of the method

We illustrate the algorithm on the following representation R_1.

	c_1	c_2	c_3	c_4
r_1	7	⊥	5	⊥
r_2	4	8	⊥	3
r_3	⊥	6	2	⊥

Note that as for the previous cases, dimensions can be treated independently. Hence we consider only the horizontal dimension in the example. Our method consists mainly in three steps that are explained below.

Step 1: Sort one row. We begin by sorting a row in the considered dimension, by treating $\perp$ as being greater than every other measure. Suppose we sort row $\langle r_1 \rangle$. We obtain the following representation R_2.

	c_3	c_1	c_2	c_4
r_1	5	7	⊥	⊥
r_2	⊥	4	8	3
r_3	2	⊥	6	⊥

Then we check if the first two cells of each row are sorted. Since it is the case in our example, this means that a PR might exist. Thus we proceed to the next step.

Step 2: Compute intervals. We now consider row $\langle r_2 \rangle$. We sort the last two cells of this row, and we obtain the following representation R_3.

	c_3	c_1	c_4	c_2
r_1	5	7	$\perp$	$\perp$
r_2	$\perp$	4	3	8
r_3	2	$\perp$	$\perp$	6

Since the last two cells of each row are sorted, a PR might exist. Thus we continue the current step by trying to find a valid position among the first three positions of $\langle r_2 \rangle$ for the cell at position $\langle r_2, c_4 \rangle$ in $\langle r_2 \rangle$. Since this cell contains 3, it should be on the left hand side of the cell containing 4 at position $\langle r_2, c_1 \rangle$, i.e., in column 1 or 2. Then we associate this cell with the interval $[1, 2]$. The cell containing $\perp$ at position $\langle r_3, c_4 \rangle$ can be placed anywhere among the first three positions in $\langle r_3 \rangle$, thus we associate it with the interval $[1, 3]$. Hence row $\langle c_4 \rangle$ can be placed either at the first or at the second position, i.e., the interval of possible valid positions for $\langle c_4 \rangle$ is $[1, 2] \cap [1, 3] = [1, 2]$.

We apply the same reasoning to find an interval of valid positions for the cell containing 8 at position $\langle r_2, c_2 \rangle$. It should be placed on the right hand side of the cell containing 4. Then $\langle c_2 \rangle$ should be placed after c_1 and we associate the interval $[3, 3]$ with this cell. The cell containing 6 at position $\langle r_3, c_2 \rangle$ should be placed on the right hand side of the cell containing 2. Then $\langle c_2 \rangle$ should be placed after $\langle c_3 \rangle$ and we associate the interval $[2, 3]$ with this cell. Therefore the possible valid positions for $\langle c_2 \rangle$ are given by $[3, 3] \cap [2, 3] = [3, 3]$, meaning that $\langle c_2 \rangle$ must be the row right after $\langle c_1 \rangle$.

Step 3: Arrange in intervals. We first choose a position for $\langle c_4 \rangle$ since the interval computed at step 2 above is $[1, 2]$. Assume that we choose 1. This entails that rows $\langle c_3 \rangle$ and $\langle c_1 \rangle$ have to be shifted to the right for $\langle c_4 \rangle$ to be the first row in this dimension. This implies that the interval of positions for $\langle c_2 \rangle$ must now be $[4, 4]$ instead of $[3, 3]$. So we obtain the following representation R_3.

	c_4	c_3	c_1	c_2
r_1	$\perp$	5	7	$\perp$
r_2	3	$\perp$	4	8
r_3	$\perp$	2	$\perp$	6

Since $\langle c_2 \rangle$ has not to be moved, the algorithm stops for this dimension, and rows $\langle r_1 \rangle$, $\langle r_2 \rangle$ and $\langle r_3 \rangle$ are sorted. Applying the same method to the rows $\langle c_1 \rangle$, $\langle c_2 \rangle$, $\langle c_3 \rangle$ and $\langle c_4 \rangle$ does not change the representation if we first consider $\langle c_2 \rangle$ as we did for $\langle r_1 \rangle$. As a consequence, R_3 is a PR.

4.3.2 The algorithm

In order to present the algorithm implementing our method, we need the following definitions.

Definition 4.4 Let C be a n-dimensional cube, k a dimension of C and R a representation of C. The set row_k is the set of rows in dimension k, that is
$row_k = \{r = \langle m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n\rangle \in C$
$\mid m_i \in dom_i$ for every i in $[1, k-1] \cup [k+1, n]\}$.
Given an interval $I = [a, b]$ with $a < b \leq |dom_k|$, row_k^I is the set of rows in dimension k w.r.t. I. That is $row_k^I = \{r_I \mid r \in row_k\}$.

Intuitively, row_k^I is the set of all slices s in dimension k such that $rep_k(s) \in I$.

As in the example above, we consider only one particular dimension k. Given a particular representation R of a cube C, we first sort a chosen row r in dimension k, such that all cells of r containing $\perp$ are located at the right hand side of r. To this end, we use a function called $extSort(R, r, I)$ where R is a representation, r is a row and I is an interval, that sorts r_I by considering that $\perp$ is greather than any other measure of dom_{mes}.

The representation $R_1 = extSort(R, r, [1, |dom_k|])$ we obtain partitions the rows in row_k into two parts (Figure 3 is an example in a 2-dimensional case):

1. The first part $p_1 = [1, i]$ corresponds to the members of dimension k for which the cells of row r contain no null values.

2. The second part $p_2 = [i + 1, |dom_k|]$ corresponds to the members of dimension k for which the cells of row r contain only null values.

Concerning point 1 above, we use a function called $testOrder(R, k, p_1)$ that tests if every row in $row_k^{p_1}$ is sorted. Based on Proposition 4.2, we have the following lemma, stating that if at least one row in $row_k^{p_1}$ is unsorted, then there exists no PR.

Lemma 4.10 *Let C be a cube, k be a dimension of C, r be a row of C in dimension k, and I be an interval. Let R' be the output of $extSort(R, r, I)$. If the computation of $testOrder(R', k, I)$ outputs $false$ then there exists no PR of C.*

Suppose we are in the case where every row in $row_k^{p_1}$ is sorted, then a PR might exist. In this case, we consider another row r' of the same dimension k. We sort the part r'_{p_2} of r' by calling $extSort(R_1, r', p_2)$. Let R_2 be the representation obtained. With representation R_2 we obtain an interval $p_3 \subseteq p_2$ such that r'_{p_3} is the contiguous part of r'_{p_2} containing no null values (see Figure 4). Note that if $p_3 = p_2$ then this row has not to be considered. For notational convenience, we assume that $p_1 = [1, i]$ and that $p_3 = [i + 1, j]$.

For the representation R_2, we check if every row of $row_k^{p_3}$ is sorted by calling the function $testOrder(R, k, p_3)$. By Lemma 4.10, if the function returns false then there exists no PR.

Assuming that every row in $row_k^{p_3}$ is sorted, we now want to find a representation where $row_k^{[1,j]}$ is sorted. To this end, we define the notion of valid positions for a cell in a contiguous subpart of a row as follows:

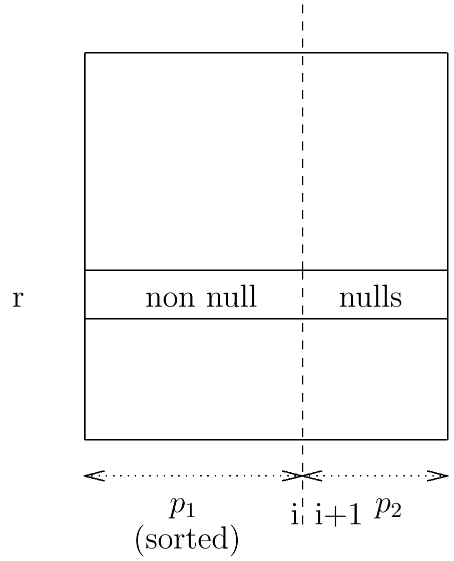

Figure 3: First call to *extSort*

Definition 4.5 Let r be a row in dimension k of a cube C and R be a representation of C such that r_I is sorted for a given interval $I = [\alpha, \beta]$. Given a cell $c = \langle m_1, \ldots, m_n, m \rangle$ of r that does not belong to r_I, let Sup and Inf be defined as follows:

- $Sup = \{\lambda = rep_k(m_k^\lambda) \mid \exists c' = \langle m'_1, \ldots, m_k^\lambda, \ldots, m'_n, m' \rangle \; m > m', \alpha \leq \lambda \leq \beta\}$
- $Inf = \{\mu = rep_k(m_k^\mu) \mid \exists c' = \langle m'_1, \ldots, m_k^\mu, \ldots, m'_n, m' \rangle \; m < m', \alpha \leq \mu \leq \beta\}$

The interval of valid positions of c in r_I is the interval $J = [a, b]$ defined as follows:

1. If $Sup = \emptyset$ then $a = \alpha$, otherwise $a = max(Sup) + 1$,
2. If $Inf = \emptyset$ then $b = \beta + 1$, otherwise $b = min(Inf)$.

Let $p'_1 = [1, i+1]$. If we find an interval I of valid positions in r'_{p_1} for a cell c in r'_{p_3}, then we can find an arrangement leading to a representation where the position of c is in I and where $r'_{p'_1}$ is sorted. Note that if such a representation exists, then $r_{p'_1}$ is also sorted since all cells of r_{p_3} contain $\perp$.

Note also that such an interval always exists, and that it might not be restricted to one position (see Figure 5).

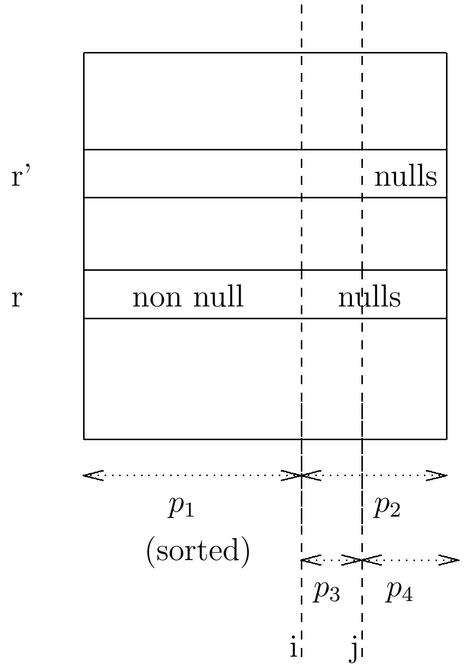

Figure 4: Sorting part r_{p_3}

The problem now becomes: for a representation R such that $row_k^{p_1''}$ is sorted with $p_1'' = [1, x-1]$, find an interval I of valid position in $r'_{p_1''}$ for the cell c at position x in r' such that I is also an interval of valid positions in $r'_{p_1''}$ for every cell belonging to slice m_k.

Definition 4.6 Let C be a cube and R a representation of C. Let r be a row in dimension k, I be an interval and $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle$ be a cell of r that does not belong to r_I. The interval of valid positions in row_k^I for c is the interval $J = \bigcap I_{c'}$, where $I_{c'}$ is the interval of valid positions in r_I for each c' belonging to slice m_k.

At this point, we use a function called *computeInterval* that computes an interval J of valid positions in $row_k^{p_1}$ for a cell $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle$. This function performs the following two steps (Figure 6):

1. for each c' belonging to slice m_k compute $I_{c'}$
2. compute $J = \bigcap I_{c'}$ and return J.

The function *computeInterval* is as follows:

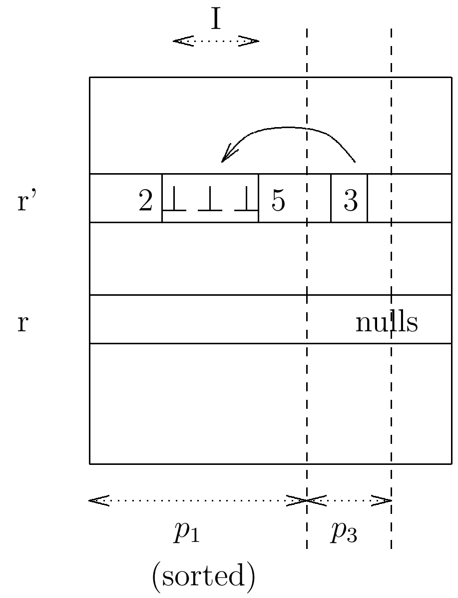

Figure 5: Finding an interval of valid positions in p_1 for a cell in p_3

Algorithm 4.3
Function: computeInterval
Input: a representation R, a dimension k, a contiguous subpart r_I of a row r, a cell $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle$ of r
Output: an interval of valid positions for c in row_k or $\emptyset$
Variable: a list L of intervals

$L = \emptyset$
for each c' belonging to slice m_k do

compute $I_{c'}$ the interval of valid positions for c' in r_I

add I'_c to L

compute $J = \bigcap_{I_{c'} \in L} I_{c'}$
return J

If the interval J computed by this function is empty, then there exists no PR, as stated by the following lemma.

Lemma 4.11 *Let C be a cube, R be a representation of C, r be a row of C in dimension k, $I = [a, b]$ be an interval and c be a cell of r. If the call to $computeInterval(R, k, r_I, c)$ outputs $\emptyset$ then there exists no PR of C. Otherwise, if the function outputs the interval $J \neq \emptyset$, then there exists a representation of*

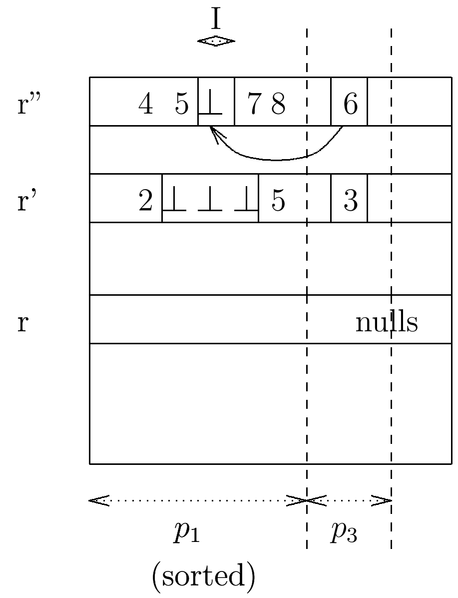

Figure 6: Computing the interval of valid positions in $row_k^{p_1}$

C such that the position of c in r belongs to J and every row in $row_k^{[a,b+1]}$ *is sorted.*

We call this function for every cell in r'_{p_3}. Suppose a non empty interval I_c exists for every such cell c. We call L the list of all such intervals. Note that for each pair of cells $c = \langle m_1, \ldots, m_n, m\rangle$, $c' = \langle m'_1, \ldots, m'_n, m'\rangle$ of r'_{p_3} such that $m < m'$ and associated with two intervals of L, respectively $I_c = [a, b]$ and $I_{c'} = [a', b']$, then $a \leq a'$ and $b \leq b'$.

Now we have to choose for all cells c in r'_{p_3} a position in I_c such that the arrangement leading to a representation where $row_k^{p_1^1}$ is sorted, with $p_1^1 = [1, j]$.

This is done using a function called $arrangeInIntervals(R, k, r'_{p_3}, L)$ that proceeds as follows: For each cell $c = \langle m_1, \ldots, m_n, m\rangle$, of r'_{p_3} associated with interval $I_c = [a, b]$, the function:

- chooses a position x for the cell c in I_c, e.g., the smallest position in I_c
- computes a new representation R' from representation R as follows:
  - place c at position x in r and,
  - place each cell c' of r which position is x' in R, $x < x' < |dom_k|$ at position $x' + 1$ in r
- update each interval $I_{c'} = [a', b']$ of L associated with cell $c' = \langle m'_1, \ldots, m'_n, m'\rangle$ in r'_{p_3} such that $m < m'$, as follows:

– $I_{c'} = [a' + 1, b' + 1]$ if $x < a'$, or
– $I_{c'} = [x + 1, b' + 1]$ if $x \geq a'$.

The function *arrangeInIntervals* is as follows:

Algorithm 4.4
Function: arrangeInIntervals
Input: a representation R, a dimension k, a contiguous subpart r_I of a row r with $I = [a, b]$, a list L of intervals
Output: a representation R'
Variables: a representation R', an integer i

```
for each c = ⟨m_1, ..., m_k, ..., m_n, m⟩ in r_I do
    choose a position x in I_c
    for i = m_k down to x + 1 do
        R' = switch(k, i, i − 1)
    I = [a + 1, b]
    for each cell c' = ⟨m'_1, ..., m'_n, m'⟩ in r_I do
        find in L the interval I_c' = [a', b'] of valid positions for c'
        if x < a' then
            I_c' = [a' + 1, b' + 1]
        else
            I_c' = [x + 1, b + 1]
return R'
```

Lemma 4.12 *Let C be a cube for which each row can contain null values but no duplicates. Let R be a representation of C, k be a dimension, r be a row of C, $J = [1, i], I = [i+1, j]$ be two intervals and L be a list of intervals of valid positions in row_k^J for every cell in r_I. The call to $arrangeInIntervals(R, k, r_I, L)$ outputs a representation R' where every row in $row_k^{[1,j]}$ is sorted.*

Once every cell of r'_{p_3} has been processed, the algorithm iterates on the other rows of dimension k with $p_1 = [1, j]$ (see Figure 7). Once dimension k has been processed, if p_2 is not empty, then it corresponds to the members of dimension k for which every combination with the members of the other dimensions contains a null value.

If every dimension has been successfully processed, the representation obtained is a PR.

We are now ready to present the main function that solves the PR problem in the case of null values (but no duplicates).

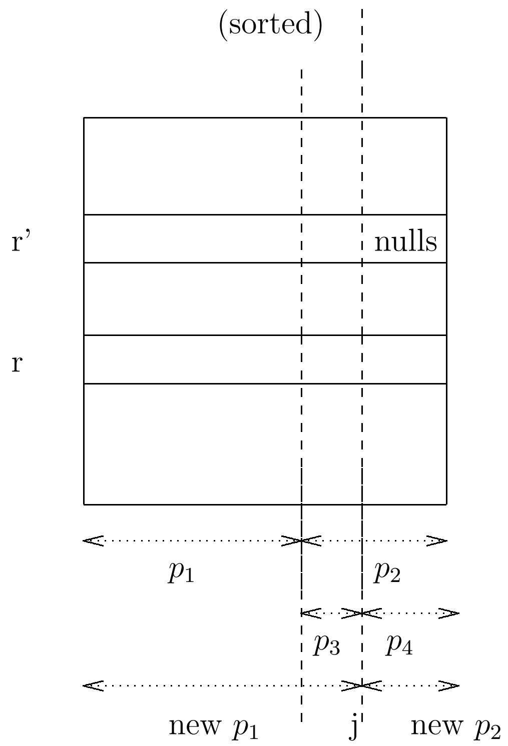

Figure 7: Row r' has been successfully processed

Algorithm 4.5
Function: main
Input: A representation R of a cube C
Output: A PR of C or the indication "no PR"
Variables: a representation R', a boolean $existsPR$, a list L of interval

for each dimension k of C

let r be a row in dimension k having $|dom_k| - i$ null values

$R' = extSort(R, r, [1, |dom_k|])$

let $p_1 = [1, i]$

let $p_2 = [i + 1, |dom_k|]$

mark r

$existsPR = testOrder(R', k, p_1)$

if not $existsPR$ then exit with "no PR" else while there exists an unmarked row r' in dimension k do

$L = \emptyset$

$R' = extSort(R', r', p_2)$

let $p_4 = [j+1, |dom_k|]$ be the only sequence of null values in r'_{p_2}

let $p_3 = [i+1, j]$

$existsPR = testOrder(R', k, p_3)$

if not $existsPR$ then exit with "no PR"

else for every cell c of r'_{p_3} do

$I = computeInterval(R', k, r'_{p_1}, c)$

if $I = \emptyset$ then exit with "no PR"

else add interval I to list L

$R' = arrangeInIntervals(R', k, r'_{p_3}, L)$

mark r'

$p_1 = [1, j]$

$p_2 = [j+1, |dom_k|]$

return(R')

We can now present the following Theorem, which is a consequence of the previous Lemmas.

Theorem 4.13 *Let C be a cube and R be a representation of C. If the call to Algorithm 4.5 outputs "no PR" then there exists no PR of C. Otherwise, the output is a PR of C.*

This algorithm is polynomial in the number of cells of the cube. Indeed:

- Step 1 consists in sorting one row, which is polynomial.
- For a given dimension k, step 2 consists in comparing every cell of a slice in dimension k to the other cells of the row in dimension k it belongs to. For each dimension k, the number of comparisons is at most $|dom_{k'}|^n$, where k' is the dimension having the greatest number of members, and n is the number of dimensions. Indeed for an n-dimensional cube, a slice contains at most $|dom_{k'}|^{n-1}$ cells, each of which being compared to at most $|dom_{k'}|$ other cells.
- Step 3 consists mostly in switch operations. For a given row, the number of switches is at most $|dom_{k'}|^2$, where k' is the dimension having the greatest number of members. Indeed no more than $|dom_{k'}|$ cells can be switched and for each no more than $|dom_{k'}|$ switches are necessary to move a cell to a valid position.

Note that the row r to be sorted first can be chosen so as to optimize the algorithm. Indeed we have every interest to take a row having the least number of nulls. For instance, if one row contains no null, then choosing this row reduces this case to case 1 (i.e., no null values in the cube).

Computing the total number of PRs in this case. Given a cube C for which each row can contain null values but no duplicates, more than one PR of C might exist. First, we have to adapt the notion of identical slice to this case.

Definition 4.7 Let C be a n-dimensional cube. Let s and s' be two slices in dimension k. s and s' are identical for a given representation R of C if for each pair of cells $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle \in s$ and $c' = \langle m'_1, \ldots, m'_k, \ldots, m'_n, m'\rangle \in s'$, $rep_i(m_i) = rep_i(m'_i) \implies m = \perp$ or $m' = \perp$, for all $i \in [1, k-1] \cup [k+1, n]$.

Obviously as in case 2, we have the following proposition:

Proposition 4.14 *Let C be a n-dimensional cube and S_{R_C} be the set of every representation of C. Let R be a particular representation of C. Let s and s' be two slices in dimension k. If s and s' are identical for R then s and s' are identical for every representation $R' \in S_{R_C}$.*

The number of different PRs in this case depends on the presence of identical slices. We have the following proposition.

Proposition 4.15 *Let C be a cube of which a PR exists. Then there exists more than one PR of C if and only if C contains at least two identical slices in one of its dimensions.*

Note that a particular case of identical slices is the case of a slice containing only cells that contain a null value. We call such a slice a *null-slice* in the following. This kind of slices can be placed anywhere in a representation without affecting the number of misplaced cells. The following corollary allows to compute the total number of PRs in this case.

Corollary 4.16 *Let C be an n-dimensional cube and R be a PR of C. Let p^i_{null} be the number of null-slices in dimension $i \in [1, n]$, let p_i be the number of different sets of identical non null-slices in dimension i, and let $m^i_j, j \in [1, p_i]$ be the cardinality of each such set in dimension i. Then the total number of PRs is $\Pi_{i\in[1,n]} \left[(\Pi_{j\in[1,p_i]}(m^i_j!)) \times \binom{n}{p^i_{null}} \right]$.*

Therefore, outputting every PR is clearly not polynomial. However computing the total number of PRs can be done by counting the number of identical slices in each dimension, which is polynomial.

4.4 Case 4: Dealing with both null values and duplicates

Solving the PR problem in this case can be done based on the methods given in the previous sections. Before giving the corresponding algorithm, we illustrate this case by an example. Consider the following representation R_1 of a two-dimensional cube C.

	c_1	c_2	c_3	c_4	c_5
r_1	2	2	1	1	1
r_2	$\perp$	2	1	1	$\perp$
r_3	3	2	2	1	2

Step 1: Call to *extSort*. We first treat dimension 1, with $dom_1 = \{c_1, c_2, c_3, c_4, c_5\}$. Sorting row $\langle r_3 \rangle$ by using the function $extSort(R_1, \langle r_3 \rangle, [1, 5])$ gives the following representation R_2.

	c_4	c_2	c_3	c_5	c_1
r_1	1	2	1	1	2
r_2	1	2	1	$\perp$	$\perp$
r_3	1	2	2	2	3

Step 2: Recursive call on the sequences of duplicates. Next, we identify in R_2 all sequences of duplicates in $\langle r_3 \rangle$: The only sequence in this example corresponds to the interval $[2, 4]$. We consider another row of the same dimension and we apply the algorithm recursively. Suppose we consider row $\langle r_2 \rangle$. This means that we sort $\langle r_2 \rangle_{[2,4]}$ by calling $extSort(R_2, \langle r_2 \rangle, [2, 4])$. This gives the following representation R_3.

	c_4	c_3	c_2	c_5	c_1
r_1	1	1	2	1	2
r_2	1	1	2	$\perp$	$\perp$
r_3	1	2	2	2	3

As there is no sequence of duplicates in $\langle r_2 \rangle_{[2,4]}$ for R_3, no other recursive call is processed.

Step 3: Call to *computeInterval* and *arrangeInInterval*. At this step, we still need to process the cell $\langle r_2, c_5, \perp \rangle$. Thus we compute an interval of valid positions for this cell in $\langle r_2 \rangle_{[2,4]}$ by calling $computeInterval(R_3, 1, \langle r_2 \rangle_{[2,4]}, \langle r_2, c_5, \perp \rangle)$. In our example, the interval of valid positions for $\langle r_2, c_5, \perp \rangle$ is $[2, 3]$. Then a call to $arrangeInInterval(R_3, 1, \langle r_2 \rangle_{[4]}, \{[2, 3]\})$ is used to arrange the representation, and we obtain the following representation R_4.

	c_4	c_5	c_3	c_2	c_1
r_1	1	1	1	2	2
r_2	1	$\perp$	1	2	$\perp$
r_3	1	2	2	2	3

At this point, the sequence of duplicates has been successfully processed. Then the algorithm stops for dimension 1 since every row in this dimension is sorted. Applying the same principle to dimension 2 results in the following representation which is a PR.

	c_4	c_5	c_3	c_2	c_1
r_3	1	2	2	2	3
r_1	1	1	1	2	2
r_2	1	$\perp$	1	2	$\perp$

The algorithm. We now present the algorithm for solving the PR problem in the general case. This algorithm consists in a function called $solvePR$ that calls the functions given in the previous sections. For a representation R of a cube C, $solvePR(R, k, [1, |dom_k|])$ is called for every dimension k of C.

Algorithm 4.6
Function: solvePR
Input: a representation R of a cube C, a dimension k an interval I
Output: a PR of C or the indication "no PR"
Variable : a list L of intervals, two intervals J and K

choose a row r in dimension k

$R' = extSort(R, r, I])$

for every sequence of duplicates J in r do

    $R'' = solvePR(R', k, J)$

$L = \emptyset$

for every cell c in r_I containing a null value

    $K = computeInterval(R'', k, r_I, c)$

    add the interval K to L

$R''' = arrangeInInterval(R'', k, r_I, L)$

if r is unsorted then exit with "no PR"

As a consequence of the Propositions and Theorems given in cases 2 and 3, we can present the following Theorem.

Theorem 4.17 *Let C be a cube and R be a representation of C. If the call to Algorithm 4.6 outputs "no PR" then there exists no PR of C. Otherwise, the output is a PR of C.*

As every function called in this algorithm is polynomial, it is easy to see that this algorithm is polynomial in the number of cells of the cube.

Computing the total number of PRs in this case Obviously in this case more than one PR of a cube might exist. The notion of identical slices has to be generalized to this case.

Definition 4.8 Let C be a n-dimensional cube. Let s and s' be two slices in dimension k. s and s' are identical for a given representation R of C if for each pair of cells $c = \langle m_1, \ldots, m_k, \ldots, m_n, m\rangle \in s$ and $c' = \langle m'_1, \ldots, m'_k, \ldots, m'_n, m'\rangle \in s'$, $rep_i(m_i) = rep_i(m'_i) \implies m =\perp$ or $m' =\perp$ or $m = m'$, for all $i \in [1, k-1] \cup [k+1, n]$.

As in case 2 and 3, the following proposition holds:

Proposition 4.18 *Let C be a n-dimensional cube and S_{R_C} be the set of every representation of C. Let R be a particular representation of C. Let s and s' be two slices in dimension k. If s and s' are identical for R then s and s' are identical for every representation $R' \in S_{R_C}$.*

With this definition of identical slice, the same reasoning as in case 3 applies. Therefore we have the following proposition and corollary.

Proposition 4.19 *Let C be a cube of which a PR exists. Then there exists more than one PR of C if and only if C contains at least two identical slices in one of its dimensions.*

Corollary 4.20 *Let C be an n-dimensional cube and R be a PR of C. Let p^i_{null} be the number of null-slices in dimension i, let p_i be the number of different sets of identical non null-slices in dimension $i \in [1, n]$, and let $m^i_j, j \in [1, p_i]$ be the cardinality of each such set in dimension i. Then the total number of PRs is $\Pi_{i\in[1,n]} \left[(\Pi_{j\in[1,p_i]}(m^i_j!)) \times \binom{n}{p^i_{null}} \right]$.*

As in case 2 and 3, outputting every PR is not polynomial, but computing the total number of PRs is polynomial.

5 Conclusion

In this paper we have introduced an approach to enhance the query-driven analysis of multidimensional data, based on representations of cubes according to their measures. We have introduced a measurement to compute the quality of the representation, and we have proposed an algorithm to find the representation of a cube for which this measurement is optimal, if it exists.

Our current and future work encompasses the following open issues:

- Implementation of the approach discussed in the paper. The algorithms given in Section 4 are naive algorithms, that should be reworked in order to propose an efficient implementation.

- Study of other problems in this framework. As stated in Section 2, a PR may not exist. Thus we can define two other problems that we shall study in the future:
  - The OR problem (cf. Definition 2.7): for a given cube and a given representation of this cube, find all ORs, and list all arrangements leading to these ORs. Based on the result of Mäkinen and Siirtola [7], we conjecture that this problem is not polynomial.
  - The t-OR problem: given a cube C and a threshold t, find a representation R_C of C such that $M_{R_C}(C) \leq t$ if it exists. If there exists at least one such representation, list all arrangements leading to these representations.
- Use of other OLAP operations to solve the problems. In this paper we restrict ourselves to the switch operation to compute appropriate representations. It would be interesting to study how the other OLAP operations [4, 5, 6] behave w.r.t. the problems introduced above. For example in the presence of hierarchies, can we use the roll-up operator to reach a PR?

References

[1] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.

[2] Yeow Wei Choong, Dominique Laurent, and Patrick Marcel. Computing appropriate representations for multidimensional data. In *ACM DOLAP*, 2001.

[3] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to user-analysts: An IT mandate [on-line]. 31p. White Paper, 1993.

[4] Marc Gyssens and Laks V. S. Lakshmanan. A foundation for multi-dimensional databases. In *VLDB*, pages 106–115, 1997.

[5] Marc Gyssens, Laks V. S. Lakshmanan, and Iyer N. Subramanian. Tables as a paradigm for querying and restructuring. In *ACM PODS*, pages 93–103, 1996.

[6] P. Marcel. Modeling and querying multidimensional databases: An overview. *Networking and Information Systems Journal*, 2(5-6):515–548, 1999.

[7] Erkki Mäkinen and Harri Siirtola. Reordering the reorderable matrix as an algorithmic problem. In *Diagrams 2000*, volume 1889 of *LNAI*, pages 453–467, 2000.

[8] Sunita Sarawagi. Explaining differences in multidimensional aggregates. In *VLDB*, pages 42–53, 1999.

[9] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of OLAP data cubes. In *EDBT*, volume 1377 of *LNCS*, pages 168–182, 1998.

[10] Panos Vassiliadis and Timos K. Sellis. A survey of logical models for OLAP databases. *SIGMOD Record*, 28(4):64–69, 1999.