Pixelizing Data Cubes: A Block-Based Approach

Yeow Wei Choong[1,2], Anne Laurent[3], and Dominique Laurent[2]

[1] HELP University College, BZ-2 Pusat Bandar Damansara, 50490 Kuala Lumpur, Malaysia
choongyw@help.edu.my
[2] ETIS-CNRS, Université de Cergy Pontoise, 2 av. Chauvin, 95302 Cergy-Pontoise, France
dominique.laurent@dept-info.u-cergy.fr
[3] LIRMM-CNRS, Université Montpellier 2, 161 rue Ada, 34392 Montpellier, France
anne.laurent@lirmm.fr

Abstract. Multidimensional databases are commonly used for decision making in the context of data warehouses. Considering the multidimensional model, data are presented as hypercubes organized according to several dimensions. However, in general, hypercubes have more than three dimensions and contain a huge amount of data, and so cannot be easily visualized. In this paper, we show that data cubes can be visualized as images by building blocks that contain *mostly* the same value. Blocks are built up using an APriori-like algorithm and each block is considered as a set of pixels which colors depend on the corresponding value. The key point of our approach is to set how to display a given block according to its corresponding value while taking into account that blocks may overlap. In this paper, we address this issue based on the *Pixelization paradigm.*

1 Introduction

On-Line Analytical Processing (OLAP) is currently a major research area in the database community aiming at managing huge amounts of data for decision making. The modelling of a multidimensional database has been extensively studied [28] and data visualization is one of the major issues in this context [20,17,18].

In this paper, we consider a complete process of multidimensional databases visualization. This process relies on the discovery of blocks of homogeneous data which are then displayed to the user. The complete process can thus be considered as a two-layer process, as in [20]:

- a logical layer that deals with the discovery of blocks of similar measure values,
- a presentational layer that deals with the presentation of the blocks to the user.

In this paper, we give a brief overview of the techniques used in discovering blocks and we focus on the presentational layer.

The rest of the paper is organized as follows. We survey the concepts of data warehouse and OLAP in Section 2. Section 3 provides the background for block discovery and visual data analysis is introduced in Section 4. Finally, Section 5 concludes our work and presents topics for future work.

P.P. Lévy et al. (Eds.): VIEW 2006, LNCS 4370, pp. 63–76, 2007.

2 Data Warehouses

2.1 Main Features

A primary objective of a data warehouse is to provide intuitive access to information stored in a database that is used in decision making. According to [14], "A data warehouse is a subject-oriented, integrated, non-volatile and time-variant collection of data in support of management's decision making process".

Data warehousing refers to the process of constructing and exploitating of the data warehouse. Data warehousing thus includes the integration of the data from multiple sources into a unified schema at a single location to facilitate data analysis for decision making. Thus, the construction of a data warehouse includes data integration, data cleansing, data consolidation and On-Line Analytical Processing (OLAP) [5,9,12].

OLAP systems are constructed in a data warehouse environment that serves as a repository of the data to be processed. From a data warehouse perspective, data mining can be viewed as an advanced stage of on-line analytical processing [12].

2.2 Multidimensional Data Model

The most popular data model for a data warehouse is a multidimensional data model (MDD). MDD models typically have two types of data: (a) *measures* which are numerical in nature and (b) *dimensions* which are mostly textual data characterizing the measures [24].

A typical MDD example is in the case of a retail business where *Product*, *Location* and *Time* are dimensions and *Quantity*, *Price* and *Sales* are measures. Intuitively, it is possible to think of a business as a data cube with these dimensions and measures, as shown in Figure 1. Any point, known as *cell* in OLAP literature, is characterized by a value on each dimension and by the corresponding measure values. Thus, one can think of a cell in the cube as the measurements of the business (e.g. *Quantity*, *Price* and *Sales*) for a particular combination of *Product*, *Location* and *Time*.

Given the above structure, OLAP tools provide fast answers to the ad-hoc queries that aggregate the data from the data warehouse. Such process is known as multidimensional data analysis or OLAP analysis.

Generally, a multidimensional database is a set of *hypercubes* (hereafter *cubes*). We note that *hierarchies* may be defined over dimensions, for instance to describe sales in function of states and not of cities. As hierarchies are not considered in the present approach, we shall not consider this feature in the following definition.

Definition 1 – Cube. *A k-dimensional cube, or simply a cube, is a tuple* $\langle C, dom_1, \ldots, dom_k, dom_m, m_C \rangle$ *where*

- C *is the name of the cube,*
- $dom_1, \ldots, dom_k$ *are k finite sets of symbols for the members associated with dimensions* $1, \ldots, k$ *respectively,*
- $dom_m = dom_{mes} \cup \{\perp\}$, dom_{mes} *is a finite totally ordered set of all possible measure and* $\perp$ *is a constant not in* dom_{mes} *that represents null values.*
- m_C *is a mapping from* $dom_1 \times \ldots \times dom_k$ *to* dom_m.

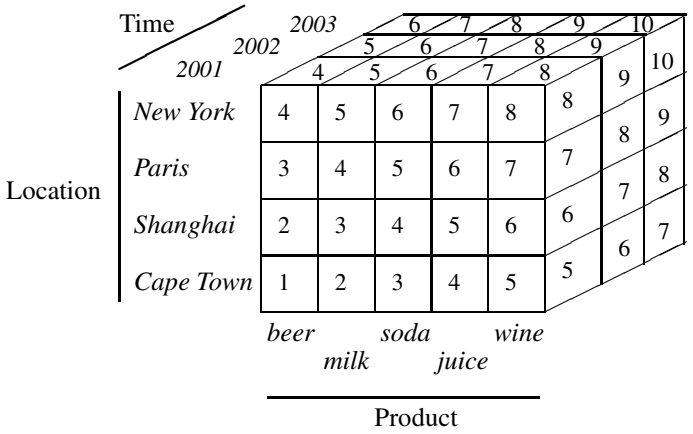

Fig. 1. A three-dimensional data cube

A cell c of a k-dimensional cube C is a $(k+1)$-tuple $\langle v_1,\ldots,v_k,m\rangle$ such that, for every $i=1,\ldots,k$, v_i is in dom_i and $m=m_C(v_1,\ldots,v_k)$. Moreover, m is called the *content* of c and c is called an *m-cell*.

As pointed out in [7], a cube can be associated with different *representations,* depending on the way member values are ordered in the sets dom_i $(i=1,\ldots,k)$.

Definition 2 – Representation. *A representation of a cube C is a set* $R=\{rep_1,\ldots,rep_k\}$ *where for every* $i=1,\ldots,k$, rep_i *is a one-to-one mapping from* dom_i *to* $\{1,\ldots,|dom_i|\}$.

Figure 2 displays two different representations of the same cube. In this paper, we consider a *fixed* k-dimensional cube C and a fixed representation of C, $R=\{rep_1,\ldots,rep_k\}$.

PRODUCT

P1	6	6	8	5	5	2	
P2	6	8	5	5	6	75	
P3	8	5	5	2	2	8	
P4	8	8	8	2	2	2	
	C1	C2	C3	C4	C5	C6	**CITY**

(a)

PRODUCT

P4	8	8	8	2	2	2	
P2	5	6	8	5	6	75	
P1	8	6	6	5	5	2	
P3	5	8	5	2	2	8	
	C3	C1	C2	C4	C5	C6	**CITY**

(b)

Fig. 2. Two representations of a data cube

2.3 OLAP Operations

As mentioned previously, in the multidimensional data model, data are organized according to multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. This way of representing the data allows the user

to view the data from different levels of detail. To achieve this, a number of OLAP operations are available for navigating through the data set. Navigation is a query-driven process, and a number of proposals have investigated formal models and languages to this end (see [12,22,29] for surveys). Some typical OLAP operations on multidimensional data are the following ones:

- *Roll-up.* This operation performs aggregation on a data cube. This is done by either *climbing up* a concept hierarchy for a dimension or by *dimension reduction*. Essentially, this operation increases the level of abstraction. An example of a roll-up operation is aggregating the *sales* of all cities to the level of country.
- *Drill-down.* This operation is essentially the reverse of roll-up. It puts more detail to a given dimension, thus decreasing the level of abstraction. For example, this operation *steps down* a concept hierarchy from the level of *month* to the level *day*.
- *Slice and dice.* The *slice* operation performs a selection on one dimension of a given cube, resulting in a subcube. For example, a selection of the months April, May and June from the *month* dimension is a *slice* operation. The *dice* operation defines a subcube by performing a selection on two or more dimensions.
- *Pivot* or *rotate*. This operation is a visualization operation that rotates the axes of the data in order to provide a different presentation of the data.
- *Switch.* This operation performs the interchanging of the positions of two members of a dimension of a cube.

Each of the above operations is illustrated in Figure 3, where the original cube is taken from Figure 1. There are many other OLAP operations such as *push*, *pull*, *join* and *merge* that are not covered here. A detailed list of OLAP operators and their descriptions can be found in [22]. More details on data warehouse and OLAP can be found in [3,10,11,14,15,19].

2.4 Data Warehouse Visualization

For effective data analysis, it is important to include the user in the data exploration process and to combine the flexibility, creativity and the common sense knowledge of human with the computational power of the computers [4]. The fundamental idea of visual data analysis is to present the data in some visual forms, allowing the user to gain insight into the data [25]. Data visualization involves examining data represented by dynamic images (colors) rather than pure numbers. Normally these images are rendered in colors rather than shades of gray, that can be used as data value indicators [21].

Visual data exploration usually follows a three-step process (*overview first*, *zoom and filter*, and *details-on-demand*) known as the Information Seeking Mantra [23].

The process of visualization can be considered as a kind of numerical data transformation. In the context of OLAP, this transformation process is performed on measure values. In the last decade, a large number of novel information visualization techniques have been developed, allowing visualizations of multidimensional data sets [13], and supporting the hierarchical structure of the dimensions of a data cube [26].

In [16], a presentational model for OLAP data is introduced. This model is based on the geometrical representation of a cube and the human perception in the space, and it is shown how the logical and the presentation models can be integrated.

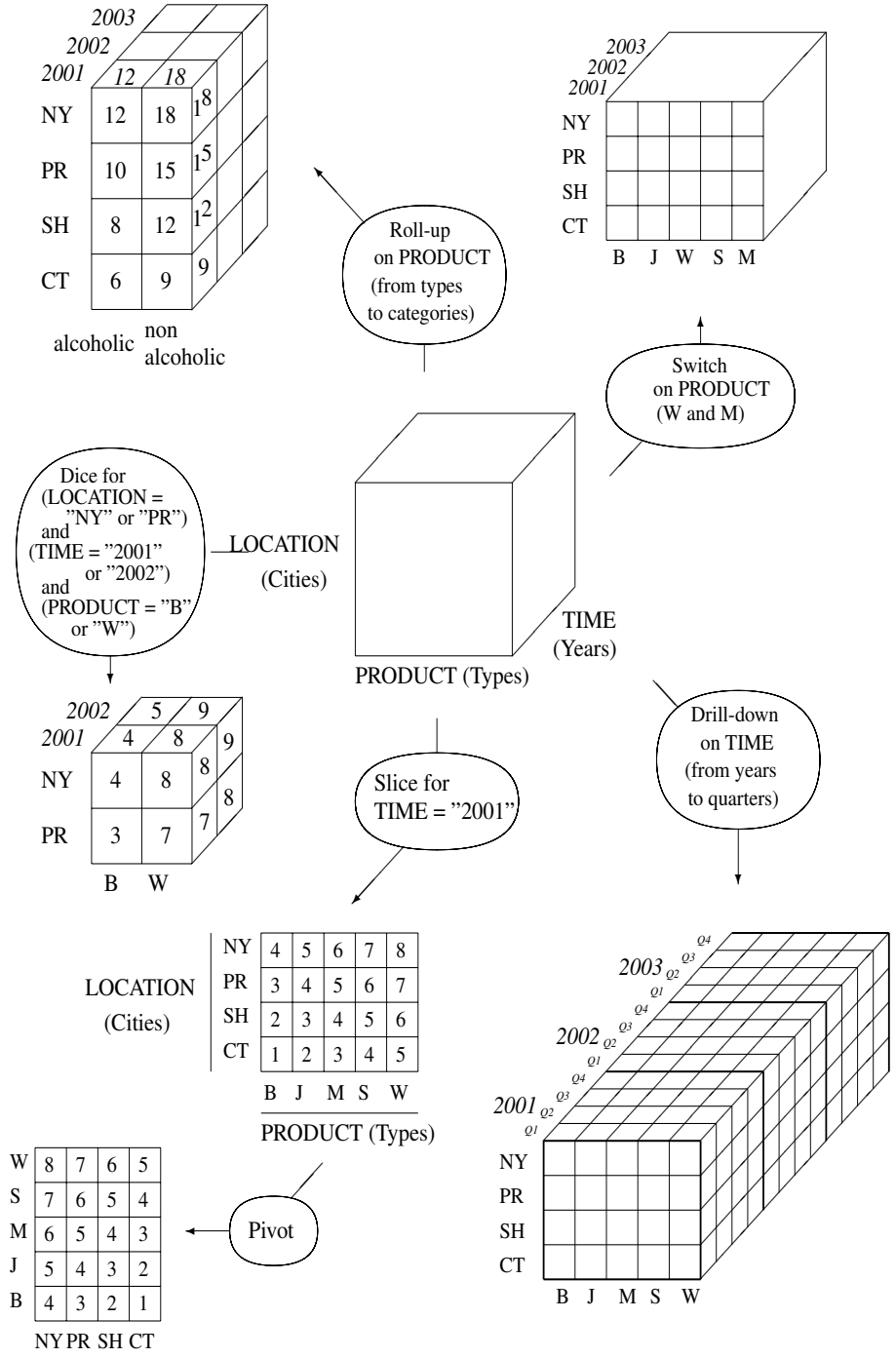

Fig. 3. Examples of typical OLAP operations on a cube

In [2], the authors present an approach to visualizing multidimensional data in the presence of constraints and user preferences. In this approach, constraints refer to the fact that the device on which data are displayed imposes restrictions (e.g. the size of the screen), and user preferences are seen as partial orderings over the member values. In this approach, dimensions are nested so as to fit the constraints, whereas in this paper, we focuss on the visualization of blocks that summarize large data cubes.

3 Summarizing Data Cubes

In this section, we review from [8] the APriori-like method used to discover blocks of homogeneous data from a multidimensional database.

Given a k-dimensional cube C and one of its representations $\{rep_1, \ldots, rep_k\}$, let d_i be a dimension of C. Two elements v_1 and v_2 in dom_i are said to be *contiguous* if $|rep_i(v_1) - rep_i(v_2)| = 1$. Moreover, for all v and v' in dom_i, the *interval* $[v, v']$ is the set of all contiguous values between v and v'. In our approach, a block of C is a subcube of C defined as follows.

Definition 3 – Block. *Given a k-dimensional cube C, a block b is a set of cells in C defined by $b = \delta_1 \times \ldots \times \delta_k$ where δ_i is an interval of contiguous values from dom_i, for $i = 1, \ldots, k$.*

Note that in the case where the interval on dimension d_i is the whole set dom_i, δ_i is denoted by *ALL*.

The *relative size* of a block b in C, denoted by $size(b)$, is the number of cells in b, *i.e.,*

$$r\text{-}size(b) = \frac{\#\ cells\ in\ b}{\#\ cells\ in\ C}$$

Moreover, the support and confidence of a block are defined as follows.

Definition 4 – Support and Confidence. *Let C be a k-dimensional, b a block in C and m a measure value.*

1. *The support of b for m, denoted by $supp(b,m)$ is the ratio*

$$supp(b,m) = \frac{\#\ cells\ in\ b\ containing\ m}{\#\ cells\ in\ C}$$

   Considering a user-given minimum support threshold σ and a measure value m, a block b such that $supp(b,m) > \sigma$ is called σ-frequent for m.
2. *The confidence of b for m, denoted by $conf(b,m)$, is defined as:*

$$conf(b,m) = \frac{\#\ cells\ in\ b\ containing\ m}{\#\ cells\ in\ b}$$

Given a k-dimensional cube C, a support threshold σ and a confidence threshold γ, we have presented in [8] an algorithm for computing a set $\mathcal{B}$ of blocks of C such that for every b in $\mathcal{B}$, there exists a measure value m in C for which $supp(b,m) > \sigma$ and

$conf(b,m) > \gamma$. As argued in [8], the blocks in $\mathcal{B}$ are meant to summarize the content of the cube C, either by associating them with a rule (that can be fuzzy due to overlappings) or by visualizing them appropriately, which is the subject of the present paper. Roughly speaking, our algorithm works as follows:

1. For every $i = 1,\ldots,k$, compute all maximal intervals I of values in dom_i such that, for every v in I, the block $b_v = \delta_1 \times \ldots \times \delta_k$ where $\delta_i = [v,v]$ and $\delta_j = ALL$ for $j \neq i$, is σ-frequent.
2. Combine the intervals in a level wise manner as follows: at level l, compute all σ-frequent blocks $b = \delta_1 \times \ldots \times \delta_k$ such that exactly l intervals defining b are different than ALL. Assuming that all σ-frequent blocks have been computed at the previous levels, this step can be achieved in much the same way as frequent itemsets are computed in the algorithm Apriori (see [8] for more details).
3. Considering the set of all blocks computed in the previous step, sort out those that are not minimal with respect to the inclusion ordering and then those having a confidence for m less than or equal to γ.

We illustrate the computation of the blocks based on the cube C and its representation as displayed in Figure 2(a). This cube has two dimensions, namely CITY and PRODUCT and contains 24 cells. Moreover, in this example, we consider the following thresholds: $\sigma = 1/15$ and $\gamma = 3/5$.

According to the definitions of support and confidence given previously, this means that, given a measure value m in C, in order to be output, a block b must contain at least 2 m-cells and that, at least 3 of its cells out of 5 are m-cells. Considering the measure value 8, the three steps above are run as follows:

1. For dimension CITY, we compute successively the supports of the blocks $[c,c] \times ALL$, for every c in $\{C1,\ldots,C6\}$. This computation gives the only interval $[C1,C3]$ since
   (*i*) $C1$, $C2$ and $C3$ are contiguous values such that $supp(C1,8) = supp(C2,8) = supp(C3,8) = 1/12$ and $1/12 > 1/15$,
   (*ii*) for $c \in \{C4,C5,C6\}$, $supp(c,8) \leq 1/15$.
   Similarly, for dimension PRODUCT, we obtain the interval $[P3,P4]$.
2. Since we have only two dimensions, this step only combines the two intervals above, which gives the block $b = [C1,C3] \times [P3,P4]$. Since $supp(b,8) = 1/6$, this block is $1/15$-frequent and thus is considered at the next step.
3. The confidence of b is computed as the ratio $4/6$ because b contains 6 cells among which 4 are 8-cells. Since this ratio is greater than $3/5$, b is output as the only 8-block.

Similar computations are run for every measure value other than 8 occuring in C, leading to the following blocks shown in Figure 4: $[C1,C1] \times [P1,P2]$ for measure value 6, $[C3,C4] \times [P1,P3]$ for measure value 5, and $[C4,C6] \times [P3,P4]$ for measure value 2.

In [8], one rule per block is automatically built up, and in order to convey block overlappings, these rules are fuzzy. In this paper, we do not consider rules, but rather, we define a policy for the visualization of the blocks, according to criteria defined by the user.

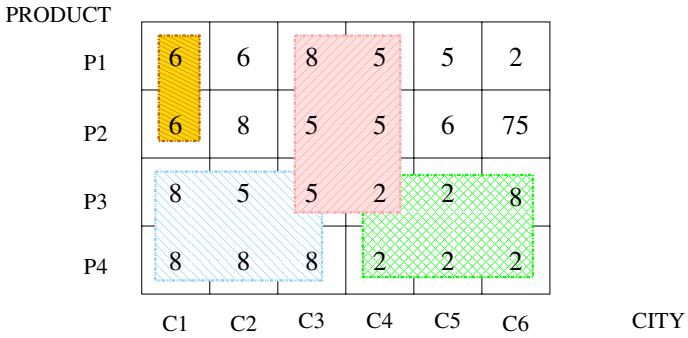

Fig. 4. Data cube and associated blocks

4 Visualization of the Blocks

4.1 Dealing with Multidimensionality

The multidimensionality of the data prevents users from having a global view of the data in a glance, since one cannot visualize more than two or three dimensions at a time. Since only a 2- or 3-dimensional subcube can be visualized, our approach aims to provide a way for displaying the intersections of the blocks with this subcube.

To this end, given a k-dimensional cube C and the set B of associated blocks, we assume that the user can choose

- the 2 (or 3) dimensions that are to be displayed, and
- a measure value for all other dimensions.

In this way, the corresponding 2- or 3-dimensional cube to be displayed is a block b_0 and for each b in B overlapping b_0, the intersection $b \cap b_0$ is also a 2- or 3-dimensional block.

Example 1. *Let us consider a cube containing sale counts according to the following four dimensions: Product, City, Month and Customer-Category. If a user chooses the two dimensions Product and City as being the* displayed dimensions*, and if this user chooses particular values for the dimensions Month and Customer-Category, say* m_0 *and* c_0*, respectively, then the block* b_0 *is defined by* $b_0 = dom_1 \times dom_2 \times [m_0, m_0] \times [c_0, c_0]$*. Thus* $dom_1 \times dom_2$ *can be displayed on the device together with a reference to the values* m_0 *and* c_0*.*

Moreover, for every block $b = \delta_1 \times \delta_2 \times \delta_3 \times \delta_4$ *in* B *such that* $m_0 \in \delta_3$ *and* $c_0 \in \delta_4$*, the block* $\delta_1 \times \delta_2$ *can be highlighted as a sub block of* $dom_1 \times dom_2$*.*

In what follows, given a k-dimensional cube, we assume that the 2- or 3-dimensional block b_0 that is to be visualized is known.

4.2 From Blocks to Pixels

Encoding a quantitative variable such as measure value requires only one psychophysical variable such as hue. In the process of encoding a measure domain, a predefined palette is considered to select the color for each domain entry. The colors in the palette are presented in a color spectrum, predominantly on hue [27]. Colors are ordered according to the measure values to display a gradual transformation of colors according to the gradual transformation of the measure values.

As blocks have several properties (e.g. support, confidence, relative size), we consider the HSB (Hue, Saturation, Brightness) encoding in order to convey several chracteristics of the block within a single pixel. HSB encoding is represented as a cone, as shown in Figure 5.

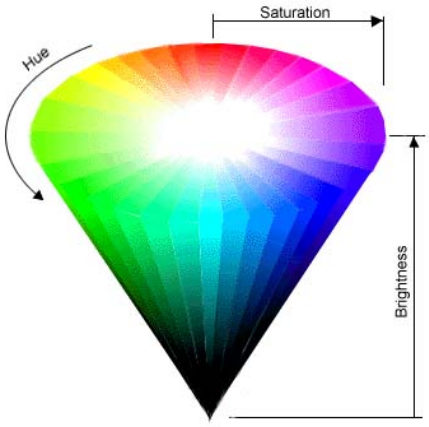

Fig. 5. HSB color space

Let C be the cube to be visualized, M (respectively m) the maximum (respectively minimum) measure value appearing in one block of C, and H_M be the maximum hue value being considered. For each measure value μ, a hue is assigned according to the following formula:

$$Hue(\mu) = \frac{H_M(\mu - m)}{M - m}$$

Note that the hue is expressed as an angle ranging from 0 to 360. In our approach, we map the measure values to an angle ranging from 0 to H_M. We argue that H_M must not be equal to 360 so that the highest measure values do not resemble to lowest ones. For instance, we can consider $H_M = 300$.

For each block associated with a measure value μ, the cells are normally displayed in the hue corresponding to μ. Moreover, as in [1], in order to point out exceptions or outliers, we also consider that cells having a measure value *very different* from μ can be displayed in the hue corresponding to their own value, instead of the hue corresponding

to μ. In order to decide whether a cell measure value is *very different* from the block measure value, a threshold expressed as a percentage has to be considered. Denoting by π_H this threshold, μ' is said to be *very different from* μ if

$$\frac{|\mu - \mu'|}{|\mu|} \cdot 100 > \pi_H$$

Saturation and Brightness are also computed regarding the blocks, except for outliers. These values are set to 100% for the outliers, and range from 0 to 100% for any other cell in a block, based on the following criteria:

- the support,
- the relative size,
- the confidence,
- the average measure value,
- the number of outliers.

We note that some of these criteria (*i.e.,* support, relative size, confidence) are very efficiently computed since they are involved in the process of block discovery, while others require additional computations (*i.e.,* average measure value, number of outliers).

Let γ_S be the criterion used to compute the Saturation value and γ_B the criterion used to compute the Brightness value. The set of all blocks can be ordered according to their values for γ_S and for γ_B.

Given a block b, we denote by $rank_S(b)$ (respectively $rank_B(b)$) the rank of b with respect to γ_S (respectively γ_B), and we have:

$$Saturation(b) = \frac{rank_S(b)}{|B|} \cdot 100 \qquad \text{and} \qquad Brightness(b) = \frac{rank_B(b)}{|B|} \cdot 100$$

where $|B|$ is the number of blocks computed from the cube C. Notice that, if no criterion is defined by the user, then a default value will be used for Saturation and Brightness.

4.3 Managing Overlappings

As seen previously, our approach for computing the blocks implies that some of them may overlap. In that case, one way to display the cells shared by several blocks could be to "mix" the hues associated to these blocks. However, we claim that this could be confusing for the user, because mixing hues may result in another hue corresponding to a different measure value. Instead, we propose to display overlapping blocks in a fore- and background manner, as explained below.

If a given cell is shared by several blocks, we have to decide which block is the most relevant for this cell, so as to determine which block to display on the foreground. To this end, we consider an additional criterion, chosen among those listed above.

Then, this criterion, denoted by γ_F, defines an ordering according to which the blocks are displayed in a fore- or background manner. More precisely, if b and b' are two overlapping blocks such that the value of γ_F for b is greater than that for b', then the overlapping area of b is displayed on the foreground.

Algorithm 1 shows how the blocks are displayed, assuming that the three criteria γ_F, γ_S and γ_B have been chosen by the user. Note that cells belonging to several blocks are treated more than once, which guarantees that they appear with the HSB code corresponding to the block displayed in the foreground based on γ_F.

Data : C cube (containing measure values ranging from m to M and where m_C is the function associating each cell to its measure value), B set of blocks in C, γ_F foreground criterion (mandatory), γ_S Saturation criterion, γ_B Brightness criterion, π_H outlier threshold, H_M Hue maximal angle
Result : Pixelization(C)
Order B according to γ_F ;
Order B according to γ_S ;
Order B according to γ_B ;
foreach $b \in B$, *in ascending order according to* γ_F **do**
  Let $m(b)$ be the measure value associated to b ;
  Hue(b) $\leftarrow \frac{H_M(m(b)-m)}{M-m}$;
  Saturation(b) $\leftarrow \frac{rank_S(b)}{|B|} \cdot 100$;
  Brightness(b) $\leftarrow \frac{rank_B(b)}{|B|} \cdot 100$;
  foreach *cell* $c \in b$ **do**
    if $\frac{|m_C(c)-m(b)|}{|m(b)|} \cdot 100 > \pi_H$ **then**
      Hue(c) = $\frac{H_M(m_C(c)-m)}{M-m}$
    else
      Hue(c) = Hue(b)
    Saturation(c) $\leftarrow$ Saturation(b)
    Brightness(c) $\leftarrow$ Brightness(b)

Algorithm 1. Algorithm for displaying the blocks

The following example illustrates Algorithm 1 above.

Example 2. *Referring back to Example 1, we assume that the blocks to be displayed are those shown in Figure 4. Moreover, we assume that the threshold* π_H *has been set to* 50% *and that the criteria associated with* γ_F, γ_S *and* γ_B *are respectively the confidence, the average measure value and the number of outliers.*

The table below summarizes the computations of the associated values for each block and Figure 6 displays what users can visualize in this case as output by Algorithm 1. It can be seen in this figure that the cells $(P3, C6, 8)$ *and* $(P1, C3, 8)$ *are computed as outliers, both associated with the same measure value* 8*. Therefore, their hue, saturation and brightness are respectively* 300, 100 *and* 100*. Note that the cell* $(P3, C4, 2)$ *is computed as an outlier as well for the* 5*-block. However, this cell also belongs to the* 2*-block in which it is not an outlier. Since the* 5*-block is put at the back of the* 2*-block (because of their respective confidences), the cell* $(P3, C4, 2)$ *is not displayed as an outlier. On the other hand, the cells* $(P3, C2, 5)$ *and* $(P3, C3, 5)$ *are displayed as outliers.*

Block	*Average*	*rank$_S$(b)*	*number of outliers*	*rank$_B$(b)*	*Confidence*	*rank$_F$(b)*	*H*	*S*	*B*
2-*Block*	*3*	*1*	*1*	*1*	*83.33%*	*2*	*0*	*25*	*50*
5-*block*	*5*	*2*	*2*	*3*	*66.67%*	*3*	*150*	*50*	*75*
6-*block*	*6*	*3*	*0*	*4*	*100%*	*1*	*200*	*75*	*25*
8-*block*	*7*	*4*	*2*	*2*	*66.67%*	*3*	*300*	*100*	*75*

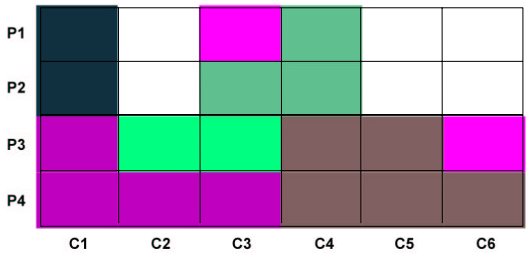

Fig. 6. Visualization example

5 Conclusion

In this paper, we have extended our work on block discovery in OLAP environment to the effective presentation of these blocks using pixelization techniques. This approach is not only particularly relevant to OLAP but it serves as a contribution to the enhancement of multidimensional data analysis.

However, we recall that, in the present approach, we assume that the blocks are available for visualization. We note in this respect that, due to the fact that measure values in a cube are numerical, it is likely that few blocks of similar values can be discovered. This is clearly a limitation for the visualization, that can be overcome by discretizing measure values before computing the blocks. We are currently considering this issue and preliminary results can be found in [6]. Another limitation of our approach is that of being able to display a two-dimensional cube on a given device. Indeed, the number of member values on each displayed dimension is generally very high and so, might not fit on the device. Therefore, constraints have to be taken into account in our visualization technique, and we think that the work reported in [2] provides an appropriate framework to deal with this problem.

A preliminary implementation of the computation of the blocks and their visualization is now available and shows the effectiveness of our approach. We note that the visualization approach proposed in this paper does not require significant additional computation. We are currently working on some extensions, including using other existing methods of HD data visualization, designing non linear scales for the computation of the hue according to the distribution of the measure values, visualizing overlappings more accurately (*e.g.,* using transparency), and pointing out outliers based on additional paradigms (*e.g.,* using different textures or flashing colors).

Based on this work, we plan to study how to extend our visualization technique to OLAP operations, *i.e.,* based on the fact that a cube *C* is visualized, how to efficiently

visualize a cube obtained form C through a combination of one or several operations applied to C. We think that this problem is particularly relevant for data exploration, especially when considering the roll-up or drill-down operators, in order to display the most relevant level of granularity of the cube.

References

1. D. Barbara and M. Sullivan. Quasi-cubes: Exploiting approximation in multidimensional data sets. *SIGMOD Record*, 26(3), 1997.
2. L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for olap queries. In *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 9–18, New York, NY, USA, 2005. ACM Press.
3. L. Cabibbo and R. Torlone. A Logical Approach to Multidimensional Databases. In *Sixth International Conference on Extending Database Technology (EDBT'98)*, pages 183–197, Valencia, Spain, 1998. Lecture Notes in Computer Science 1377,Springer-Verlag.
4. S.K. Card, J.D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization : Using Vision to Think*. Morgan Kaufmann Publishers, 1999.
5. S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *ACM-SIGMOD Records*, 26(1):65–74, 1997.
6. Y.W. Choong, A. Laurent, and D. Laurent. Building fuzzy blocks from data cubes. In *11th IPMU International Conference*, 2006, to appear.
7. Y.W. Choong, D. Laurent, and P. Marcel. Computing appropriate representation for multidimenstional data. *DKE Int. Journal*, 45:181–203, 2001.
8. Y.W. Choong, P. Maussion, A. Laurent, and D. Laurent. Summarizing multidimensional databases using fuzzy rules. In *Proc. 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'04)*, pages 99–106, Perugia. Italy, 2004.
9. E.F. Codd, S.B. Codd, and C.T. Salley. Providing olap (on-line analytical processing) to user-analysts: An it mandate. In *White Paper*, 1993.
10. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tabs, and Sub-Totals. *Journal of Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.
11. M. Gyssens and L.V.S. Lakshmanan. A Foundation for Multidimensional Databases. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, pages 106–115, Athens, Greece, August 1997.
12. J. Han and M. Kamber. *Data Mining - Concepts and Techniques*. Morgan Kaufmann, 2001.
13. C.G. Healey. *Effective Visualization of Large Multidimensional Datasets*. PhD thesis, University of British Columbia, 1996.
14. W.H. Inmon. *Building the Datawarehouse*. John Wiley & Sons, 1996.
15. M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis. *Fundamentals of Data Warehouses*. Springer-Verlag, 1998.
16. Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors. *Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2003, Prague, Czech Republic, September 3-5,2003, Proceedings*, volume 2737 of *Lecture Notes in Computer Science*. Springer, 2003.
17. D.A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1), 2000.

18. D.A. Keim and H.-P. Kriegel. Issues in visualizing large databases. In *Proc. Int. Conf. on Visual Database Systems*. Chapman & Hall Ltd., 1995.
19. R. Kimball. *The Datawarehouse Toolkit*. John Wiley & Sons, 1996.
20. A.S. Maniatis, P. Vassiliadis, S. Skiadopoulos, and Y. Vassiliou. Advanced visualization for olap. In *Proceedings of ACM 6th International Workshop on Data Warehousing and OLAP (DOLAP)*, New Orleans, 2003.
21. G.M. Marakas. *Modern Data Warehousing, Mining and Visualization: Core Concepts*. Prentice Hall, 2003.
22. P. Marcel. Modeling and querying multidimensional databases: An overview. *Networking and Information Systems Journal*, 2(5-6):515–548, 1999.
23. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society Press, 1996.
24. A. Shoshani. Statistical Databases: Characteristics, Problems, and some Solutions. In *Eigth International Conference on Very Large Data Bases, September 8-10, 1982, Mexico City, Mexico, Proceedings*, pages 208–222. Morgan Kaufmann, 1982.
25. R. Spence. *Information Visualization*. Addison-Wesley (ACM Press), 2000.
26. C. Stolte, D. Tang, and P. Hanrahan. Query analysis and visualization of hierarchically structured data using polaris. In *Proc. of the 8th ACM SIGKDD Intl. Conference of Knowledge Dicovery and Data Mining*, 2002.
27. D. Travis. *Effective Color Displays: Theory and Practice*. Academic Press, London, 1991.
28. P. Vassiliadis. Modeling Databases, Cubes and Cube Operations. In M. Rafanelli and M. Jarke, editors, *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM)*, Capri, Italy, July 1998. IEEE Computer Society.
29. P. Vassiliadis and T. Sellis. A survey of logical Models for OLAP Databases. *SIGMOD Record*, 28(4), 1999.