Advanced Visualization for OLAP

Andreas S. Maniatis[1]
Tel. +30-210-7721436
andreas@dblab.ntua.gr

Panos Vassiliadis[2]
Tel. +30-26510-98814
pvassil@cs.uoi.gr

Spiros Skiadopoulos[1]
Tel. +30-210-7721402
spiros@dblab.ntua.gr

Yannis Vassiliou[1]
Tel. +30-210-7722526
yv@cs.ntua.gr

[1]National Technical University of Athens
Dept. of Elec. and Computer Eng.
9, Iroon Polytechneiou St.
15780 Athens, Hellas

[2]University of Ioannina
Dept. of Computer Science
45110 Ioannina, Hellas

ABSTRACT

Data visualization is one of the big issues of database research. OLAP as a decision support technology is highly related to the developments of data visualization area. In this paper we demonstrate how the Cube Presentation Model (CPM), a novel presentational model for OLAP screens, can be naturally mapped on the Table Lens, which is an advanced visualization technique from the Human-Computer Interaction area, particularly tailored for cross-tab reports. We consider how the user interacts with an OLAP screen and based on the particularities of Table Lens, we propose an automated proactive users support. Finally, we discuss the necessity and the applicability of advanced visualization techniques in the presence of recent technological developments.

Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical Design – *data models.*
H.2.3 [**Database Management**]: Languages – *report writers*
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces (GUI), user-centered design.*

General Terms: Design, Human Factors.

Keywords: On-Line Analytical Processing, Visualization

1. INTRODUCTION

In the last years, On-Line Analytical Processing (OLAP) and data warehousing have become major research areas in the database community [3,7]. Although the *modeling of data* [16,17] has been extensively dealt with, an equally important issue of the OLAP domain, the *visualization of data*, has not been adequately investigated. In the context of OLAP, data visualization deals with the techniques and tools used for presenting OLAP specific information to end-users and decision makers. The database community expects visualization to be of significant importance in the area, during the next years [7], and although research has provided results dealing with the presentation of vast amounts of data [5,4,1,15], OLAP has not been part of advanced visualization techniques so far.

In this paper, we start by adopting a newly introduced presentation model for OLAP called *Cube Presentation Model* - CPM [11] and demonstrate how it can be combined with non-traditional visualization techniques. The CPM model distinguishes representation from data retrieval. It is separated in two layers: a logical that deals with data retrieval and representation and a presentational that provides a generic model for data representation. In this paper, we present a quick informal overview of the main characteristics of CPM and accompany them with its respective UML modeling for ease of understanding. Then, we proceed with the contributions of this paper, which can be listed as follows:

- Initially, we present a mapping of the generic presentational scheme of CPM to the particularities of an advanced visualization technique coming from the field of Human Computer Interaction. The *Table Lens* technique [14,12] is particularly tailored for cross-tab reports, which are most commonly used for OLAP purposes and it is accompanied by a set of handy features for the exploration of data sets which are presented in this way.
- Next, we provide algorithms for the automated proactive support of the user during his interaction with an OLAP screen, based on the particularities of Table Lens. Specifically, Table Lens employs a particular distortion of the presentation to highlight areas of increasing interest to the user. We provide a generic algorithm to support this task proactively.
- Finally, we discuss the necessity and the applicability of such visualization techniques in the presence of current technological developments.

The remainder of this paper is structured as follows. In Section 2, we summarize the logical and the presentation layers of CPM. Section 3 shows how CPM can be naturally combined with Table Lens. Moreover, Section 3 demonstrates the automate proactive support to the user. In Section 4, we discuss the necessity and applicability of the proposed ideas. Finally, Section 5 concludes our results and presents topics for future work. A longer version of this paper, with more details can be found in [8].

2. THE CUBE PRESENTATION MODEL

Although OLAP has been an active research area for the past few years, the efforts devoted to the visualization of OLAP screens are very scarce. To our knowledge, only two such efforts exist [10,1]. The first is from the industrial field, where Microsoft has issued a

DOLAP '03, November 7, 2003, New Orleans, Louisiana, USA.

commercial standard for multidimensional databases and where the presentational issues form a major part [10]. In this approach, a powerful query language is used to provide the user with complex reports, created from several cubes (or actually subsets of existing cubes). The second is an academic approach, the Tape Model [1], based on the notion of “Tapes”, called so due to their look and feel. Tapes are infinite and can overlap (if they contain shared data dimensions), or intersect with each other. A two dimensional intersection is called a matrix and represents a kind of cross-tab between the corresponding dimensions. Each tape comprises of a variable number of *tracks*. The most important operations on tapes include: (a) insertion and deletion of tracks, (b) changing the sequence of tracks (i.e., sorting) and (c) scrolling on tracks. The Tape Model offers the possibility of defining *hierarchical structures* within a tape.

In [11], we have presented the *Cube Presentation Model (CPM)*, a novel proposal towards a presentation model for OLAP screens. CPM is composed of two parts: (a) a *logical* layer which involves the formulation of cubes and (b) a *presentational layer* that involves the presentation of these cubes (normally, on a 2D screen). The main idea behind CPM lies in the separation of *logical data retrieval* (which we encapsulate in the logical layer of CPM) and *data presentation* (captured from the presentational layer of CPM). This duality provides the flexibility of possibly replacing one of the two layers with an alternative proposal smoothly. The logical layer that we propose is based on an extension of a previous proposal [18] with additional functionality that allows us to incorporate more complex cubes. In a nutshell, the logical model involves (a) *dimensions* defined as lattices of dimension *levels*, (b) *ancestor functions* (in the form of $\mathtt{anc}^{L_1;L_2}$) mapping values between related levels of a dimension, (c) *detailed data sets*, practically modeling fact tables at the lowest granule of information for all their dimensions and (d) *cubes*, defined as aggregations over detailed data sets. In this paper, we will not deal with the formal presentation of the underlying logical layer of CPM (the reader is referenced to [11] for a detailed and in depth presentation) but focus on the mapping of our presentation layer to alternative visualization techniques from the area of Human Computer Interaction.

Following, we give an intuitive and informal description of the *presentation layer* of CPM that provides a formal model for OLAP screens.

The most important entities – as far as display aspects are concerned – of the presentation layer of CPM include:

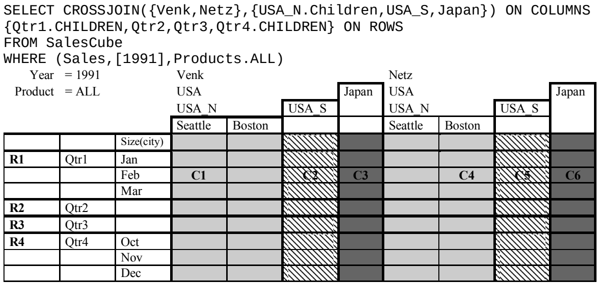
- **Points:** A *point over an axis* resembles the classical notion of points over axes in mathematics. In the simple case, a point is characterized by an equality selection condition over a level (e.g., `City=Seattle`). Nevertheless, as we shall see, we can multiplex several logical dimensions to one presentational axis; therefore, a point will be formally defined to handle this kind of situations, too.
- **Axis:** An axis can be viewed as a set of points. We introduce two special purpose kinds of axes, `Invisible` and `Content`. An `Invisible` axis is a placeholder for the levels of the data set which are to be presented to the user. The `Content` axis has a more elaborate role: it is a place holder for the content of the multicube, as computed over the detailed data.
- **Multicubes.** A multicube is defined over (a) a multidimensional space, comprising a set of axes, (b) an underlying data set providing all the data which will be filtered and aggregated before presented to the user and (c) a mapping among the multidimensional space and the underlying data set that shows the computation of the multicube contents.
- **2D-slice:** A 2D slice is a 2D layer of data that can be presented on the screen. Consider a multicube `MC`, composed of `K` axes. A *2D-slice over* `MC` can be sufficiently defined by a set of (`K-2`) points, each from a separate axis. Intuitively, a 2D-slice pins the axes of the multicube to specific points, except for 2 axes, which will be presented on the screen (or a printout). In Fig. 2, we depict such a 2D slice over a multicube. *//check whether it is the MS picture…*
- **Tape:** Intuitively, a tape is column or a row over a 2D-slice, i.e., a construct parallel to an axis. Again, if we consider a 2D-slice `SL` over a multicube `MC`, composed of `K` axes, a tape is sufficiently defined by a set of (`K-1`) points, where the (`K-2`) points are the points of `SL`. A tape is always parallel to a specific axis: out of the two "free" axis of the 2D-slice, we pin one of them to a specific point which distinguishes the tape from the 2D-slice.
- **Cross-join:** Intuitively, if we take one tape parallel to the horizontal axis and another parallel to the vertical axis, their intersection is a cell. In the most general case, as we shall see, it can be a set of cells. In both cases, the intersection of two non-parallel tapes is called a cross-join. Consider a 2D-slice `SL` over a multicube `MC`, composed of `K` axes and two tapes t_1 and t_2 which are not parallel to the same axis. A *cross-join over* t_1 *and* t_2 is defined by a set of `K` points, where the (`K-2`) points are the points of `SL` and each of the two remaining points is a point on a different axis of the remaining axes of the slice.
- **Content Function**: At the schema level, we assume a function assigning the computation of measures to the `Content` axis of the multicube, along with ordering and other restrictions. We also assume a function, mapping combinations of multicube coordinates, one from each of the coordinate axis of the multicube to the measure axis. Each such assignment is practically a row in the result set of one of the queries/expressions/… computing the multicube, which we call cell[1]. For brevity, in the sequel, we simply tag the `Content` axis with this information.

To make the discussion easier, we will use an example taken from [10], throughout the paper (Figure 1). In this example, we assume a cube `SalesCube` is defined over the dimensions `Products`, `Salesman`, `Time`, and `Geography`, each involving several levels of aggregation. In this query, we restrict the `Time` dimension to the sales of `Year 1991`. We ignore the `Products` dimension (`Products=ALL`) in the subsequent aggregation of detailed data. Whenever we need to present a 2D screen and more than two dimensions are involved, we need to merge (`CROSSJOIN` in [10] terminology) as many dimensions as necessary in a single axis. In this case, we combine the dimensions `Salesman` (restricted on two

[1] The name *cell* stems from the regular terminology of OLAP, referring to points in the multidimensional space. Although in the classical tabular representation of data, *cell* is actually a successful name, for other representation techniques this does not apply (e.g., in the proposal of [4], a cell should be represented by a line).

salesmen) and `Geography` on the `COLUMNS` axis and leave the dimension `Time` on the `ROWS` axis. Note that the `Geography` dimension involves more than one levels of aggregation (both `City` and `Region`). The same applies for the `Time` dimension, where both `Quarters` and `Months` are employed.

In terms of *CPM* terminology, the query of Figure 1 is a 2D-Slice, say `SL` (see also Figure 2). In `SL` one can identify 4 horizontal tapes denoted as `R1`, `R2`, `R3` and `R4` in Figure 1) and 6 vertical tapes (numbered from `C1` to `C6`). The meaning of the horizontal tapes is straightforward: they represent the `Quarter` dimension, expressed either as quarters or as months. The meaning of the vertical tapes is somewhat more complex: they represent the combination of the dimensions `Salesman` and `Geography`, with the latter expressed in `City`, `Region` and `Country` level. Moreover, two constraints are superimposed over these tapes: the `Year` dimension is pinned to a specific value (i.e., `Year=1991`) and the `Product` dimension is ignored. One can also consider the cross-join `t1` defined by the common cells of the tapes `R1` and `C1`.



```
SELECT CROSSJOIN({Venk,Netz},{USA_N.Children,USA_S,Japan}) ON COLUMNS
{Qtr1.CHILDREN,Qtr2,Qtr3,Qtr4.CHILDREN} ON ROWS
FROM SalesCube
WHERE (Sales,[1991],Products.ALL)
```

Year	= 1991		Venk				Netz			
Product	= ALL		USA			Japan	USA			Japan
			USA_N		USA_S		USA_N		USA_S	
			Seattle	Boston			Seattle	Boston		
		Size(city)								
R1	Qtr1	Jan								
		Feb	**C1**		**C2**	**C3**		**C4**	**C5**	**C6**
		Mar								
R2	Qtr2									
R3	Qtr3									
R4	Qtr4	Oct								
		Nov								
		Dec								

Figure 1. Motivating example for the cube model [10,11].

Interpreting our motivating example in terms of CPM, we assume a detailed data set, named `SalesCube`, under the schema:

```
S = [Quarter.Day, Salesman.Salesman,
        Geography.City, Time.Day,
        Product.Item, Sales, PercentChange,
        BudgetedSales]
```

The following *axis schemata* can also be discerned in Figure 2:

```
Row_S  =  {[Quarter],  [Month,Quarter,  Quarter,
        Month]}
```

$$\texttt{Column\_S = \{[Salesman}\times\texttt{Geography],}\\ \texttt{[Salesman]}\times\texttt{[[City,Size(City)],}\\ \texttt{Region,Country]\}}$$

```
Section_S = {[Time],[Year]}
Invisible_S = {[Product],[Product.ALL]}
```

$$\texttt{Content\_S = \{[Sales],[sum(Sales}^{0}\texttt{)]\}}$$

along with their respective *axes*:

$$\texttt{Rows = \{Row\_S,[anc}^{\texttt{month;day}}\texttt{(Month)=Qtr1,}\\ \texttt{Quarter=Qtr2, Quarter=Qtr3,}\\ \texttt{anc}^{\texttt{month;day}}\texttt{(Month)=Qtr4]\}}$$

$$\texttt{Columns = \{Column\_S, \{[Salesman='Venk',}\\ \texttt{Salesman='Netz'],}\\ \texttt{[anc}^{\texttt{region;city}}\texttt{(City)='USA\_N',}\\ \texttt{Region='USA\_S', Country='Japan']\}}$$

```
Sections = {Section_S,[Year=1991,Year=1992]}
Invisible = {Invisible_S,[ALL='all']}
Content = {Content_S}
```

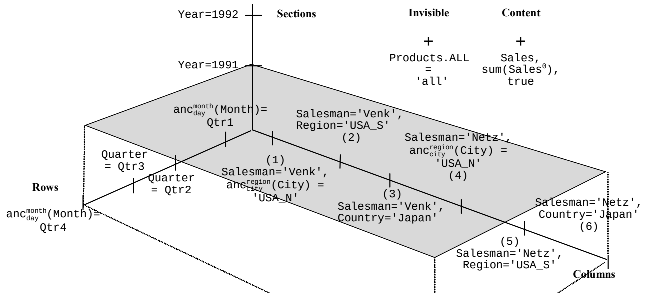

Year=1992
Sections
Invisible
Content
Year=1991
Products.ALL = 'all'
Sales, sum(Sales⁰), true
anc month day (Month)= Qtr1
Salesman='Venk', Region='USA_S' (2)
Salesman='Netz', anc region city (City) = 'USA_N' (4)
Quarter = Qtr3
(1) Salesman='Venk', anc region city (City) = 'USA_N'
Quarter = Qtr2
Rows
anc month day (Month)= Qtr4
(3) Salesman='Venk', Country='Japan'
Salesman='Netz', Country='Japan' (6)
(5) Salesman='Netz', Region='USA_S'
Columns

Figure 2. The 2D-Slice `SL` for the example of Figure 2 [11].

In Figure 2, we can also observe an exemplary *point over an axis*, incorporating equality conditions for each of the involved dimensions of the axis:

$$\texttt{p}_1\texttt{ = ([Salesman,[}\underline{\texttt{City}}\texttt{,Size(City)]],}\\ \texttt{[Salesman='Venk',anc}^{\texttt{region;city}}\texttt{(City)='USA\_N'])}$$

Thus, a multicube `MC` can be defined as:

```
MC = {Rows, Columns, Sections, Invisible,
                    Content}
```

Finally, in Figure 3, we present some more comprehensive visualization representations of multicubes, axes, points, 2D slices and cross-joins on a 3D and 2D layout.

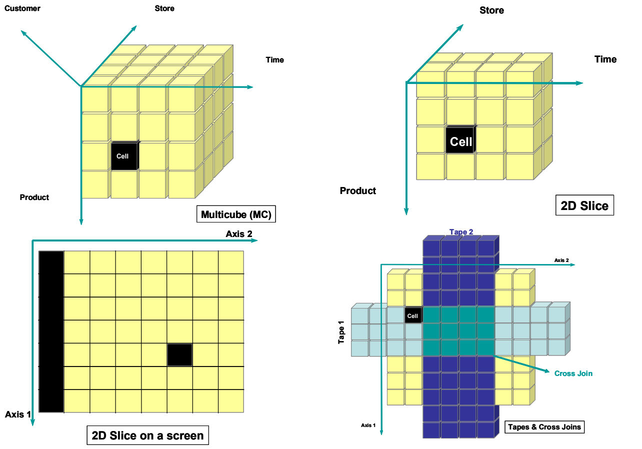


Figure 3. Mapping CPM objects to 3D and 2D Cross Tabular layouts.

3. MAPPING CPM TO VISUALIZATION TECHNIQUES

In this section, we will demonstrate how CPM can be combined with Table Lens (TL) [14,12], a traditional cross-tabular presentation model from the Human Computer Interaction area. This model is widely used in applications and platforms for the visualization of tabular, multivariate and multidimensional data and appears to be quite appropriate for OLAP purposes. Table Lens is based on the "focus plus context" technique that allows visualizing and manipulating large 2-D tables [14]. Using Table Lens, we can easily examine patterns and correlations in large tables and effectively zoom in without losing the global picture of our data. We have chosen Table Lens as an advanced visualization technique due to the fact that it is based on a cross-tabular paradigm for the presentation of information; a paradigm quite popular in OLAP screens, too.

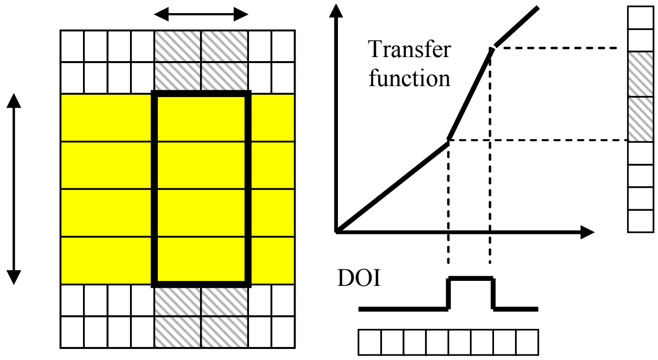


Figure 4. A Table Lens example: (a) a 2x4 focus window is defined over a space of 8x8 points; (b) Table Lens distortion of the Columns axis

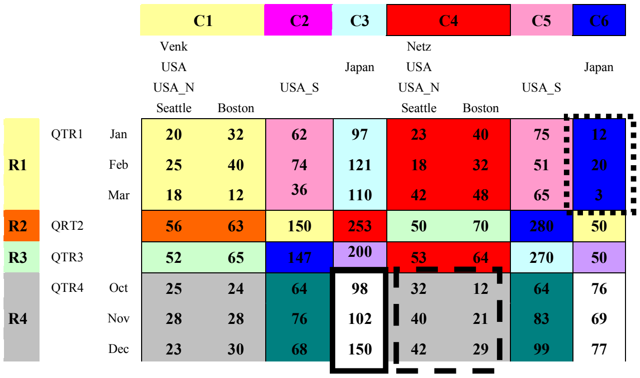
3.1 Mapping CPM to Table Lens

In this subsection, we will present the main features of Table Lens, and then we will link it to the CPM model. The main constructs of the Table Lens technique involve:

- **Axes**: The Table Lens model assumes two axes. For clarity, we will use Rows and Columns to denote these two axes.
- **2D space**: The 2D space is constructed from the Cartesian product of the two Table Lens axes. It is a (finite) matrix of cells.
- **Degree of Interest Function (DOI)**: DOI is a function that maps each axis point to a value that indicates the level of interest for that point. For each axis a different DOI function is prescribed.
- **Transfer Function**: A transfer function maps each cell to its physical locations, indicating the level of zoom for each cell. Practically, the transfer function is the translation of the respective DOI function (operating at the "interest" space) to the "pixel" space.

One of the basic ideas behind the Table Lens technique is that not all cells are considered equal. In fact, certain cells comprising a concrete region of the 2D space are assigned to occupy more surface of the screen than the rest of the cells. This is the essence of zooming into the particular region of the 2D space. To implement this, in the simplest setting of Table Lens, each DOI function is a simple "pulse" function, meaning that it has a standard value for all points, except for the points of a certain interval that are mapped to a higher value. Remember that each axis has its own DOI function, thus a 2D space is characterized by 2D windows of focus. In Figure 4a, we depict an 8x8 space with a 2x4 focus window. In Figure 4b we show (i) how the originally equally important cells of the Columns axis are assigned importance values by the DOI function: notice the pulse on two particular cells that assigns them greater importance than the rest of the cells and (ii) how the Transfer function, defined as a weighted integral of the DOI function maps the points to pixel areas. For reasons of efficient representation [14], in Figure 4b, the produced axis is rotated by 90°.Finally, another interesting feature of Table Lens is the ability to define more than one windows of focus. This is quite helpful in situations where two areas can be contrasted and compared. As we shall see in the next section, this feature is particularly useful in the case of OLAP.

There is an easy way to map the underlying constructs of the CPM to the ones of the Table Lens. The axis points of CPM are mapped to axis points of Table Lens and a 2D slice in CPM is implemented as a 2D space in Table Lens. The contents function provides the values of the cells of the 2D space. Naturally, CPM is generic enough to lack the particularities of the axis distortion due to the DOI function. The naïve way to overcome the limitation is simply to ask the user to define a certain window of focus over the presented 2D slice, specifying both its size and position. Still, we can automate the process on the basis of the structure and the contents of a 2D slice.



			C1		C2	C3	C4		C5	C6
			Venk				Netz			
			USA			Japan	USA			Japan
			USA_N		USA_S		USA_N		USA_S	
			Seattle	Boston			Seattle	Boston		
R1	QTR1	Jan	20	32	62	97	23	40	75	12
		Feb	25	40	74	121	18	32	51	20
		Mar	18	12	36	110	42	48	65	3
R2	QRT2		56	63	150	253	50	70	280	50
R3	QTR3		52	65	147	200	53	64	270	50
R4	QTR4	Oct	25	24	64	98	32	12	64	76
		Nov	28	28	76	102	40	21	83	69
		Dec	23	30	68	150	42	29	99	77

Figure 5. Instantiation of the motivating example with values; different shading determine different cross-joins and thick borders highlight the cross-joins with the highest, lowest and closest to average values.

3.2 Which Window of Interest to Choose?

In this subsection, we will deal with the problem of providing the user with proactive automated support for the exploration of an OLAP report. Our main tool towards this end is the window of interest as determined by the DOI functions and the basic idea is to provide an algorithm *to proactively determine the window of interest over a 2D slice*. We want to define an algorithm that automatically determines this window whenever a user invokes an OLAP report. It appears that we can come up with a generic algorithm, where the stopping conditions, error range and other parameters can be tuned by the user. Actually, we can even treat as a parameter a choice on whether the user is simply interested of having a window of a certain surface or he is actually interested to see a focus on a range of cells satisfying certain statistical properties (e.g., minimum/maximum/closest to average set of values). Having determined algorithmically the window of interest, the two involved DOI functions, which are independent from each other, are directly derived.

3.3 Motivation and Assumptions

Before providing the generic algorithm, let us clarify our contribution through a specific example. We instantiate the example of Figure 2 with the values of Figure 5. Let us assume that when the user activates this OLAP screen, he would like to be informed on three particular cross-joins: one involving the maximum sales, another involving the lowest and a third involving the cross-join with behavior closest to the average of the whole screen. Practically, this involves three windows of focus, which we depict through a thick border around the involved cross-joins. In this particular case, the cross-join R1/C6 is the one with the lowest summary of values, the cross-join R4/C4 the one with the highest sum and the cross-join R2/C3 is the one closest to the average sales per cross-join (which amounts to 240.5 sales per cross-join).

A vanilla algorithm to compute the aforementioned quantities proceeds as follows: (a) summarizes all cells per cross-join; (b) sorts cross-joins and computes the average cross-join value and (c) pinpoints the three regions of interest. This algorithm has linear (precisely, one-pass) complexity on the number of cells and *nlogn* (due to sorting) complexity on the number of cross-joins. Actually, if we are simply to keep the `max`, `min` or `closest-to-avg` cross-join, a linear single pass from all the cells is sufficient, without any sorting. In the case of `avg`, each time that we summarize the cells from a cross-join we can compute the average of the individual cross-join summaries and compute the closest cross-join to the current value of this average.

Assumptions: Underlying this proactive notification to the user, we have made the following assumptions:

- *Cross-joins constitute homogeneous pieces of information*. This means that we can assume a certain level of semantic cohesion among the cells of a certain cross-join. Moreover, we can assume that each cross-join can be considered as a distinct semantic unit and that cross-joins are comparable to each other. For example, we assume that it makes sense to compare sales from Japan to the sales of Southern USA. Naturally, the user choices for the axes points (and the produced cross-joins) may severely affect this assumption.
- Statistically speaking, *we are allowed to perform certain aggregate operations over our data*. Specifically, we assume that the underlying detailed data set has been summarized by a distributive aggregate function.

In [6] aggregation functions are categorized as (a) *distributive* functions, like `max`, `min`, `sum` or `count`, meaning that there is a way to compute the result of the application of the aggregation function to the overall data set by composing the individual results of its application to subsets of the dataset; (b) *algebraic functions* that are expressed as finite algebraic expressions over distributive functions, like `avg`; and (c) *holistic* functions for all other functions.

To forestall any possible criticism, we want to point out that the *exact* result of aggregation operations over a 2D slice is handled by the logical layer. In the case of the [18] model, all operations are formally defined as operations over the detailed data set; optimization results for the obvious cases are also provided. Nevertheless, in the case of this paper, we want a *quick approximation* of the statistical measures under consideration, to be used for the determination of the focus window and not of the values of the report. Thus, problems like the *Simpson's paradox* or the *non invariance* property [6] are considered as out of the scope of this paper. Finally, as a general comment, since it is quite cumbersome to ask the user each time to characterize the statistical nature of his underlying data, we employ the idea that one can have an *indication* of the statistical nature of the information of screen by observing the aggregate function that has been applied to compute them. Thus, since in our case we are starting with a `sum` aggregate function, we conclude that we can apply further distributive operations to the measure `Sales` in order to obtain our indicative approximations.

3.4 A Generic Algorithm for Determining the Window of Focus

Naturally, we can do better than the aforementioned vanilla algorithm by adding extra criteria to the proactive selection of the starting window of focus. We propose a guided greedy generic algorithm, *GenericFocusWindow* (Figure 6), to deal with the issue. The simple idea underlying the algorithm is that there are certain conditions to be met for the focus window. For example, one could require that the focus window occupies at most/least a certain percentage of the screen size, or of a certain size of cells. Moreover, the selected window optimizes an objective function. The property *Determining Quality* of the algorithms captures exactly this requirement in the form of a certain function. Since our algorithm is greedy, we need an *Original Pick* routine to start the processing; in general this is closely related to the *Determining Quality* function and we require that it starts with a smallest value. Moreover, a *Guard Condition* checks for the satisfaction of the desired property (meaning that we can possibly allow a certain approximation error ε to out obtained solution). Finally, a function *Pick* provides the necessary details for working from the original small-in-value solution towards the final result, practically picking the next cross-join to enlarge the current window of focus.

One implicit assumption that our algorithm makes is that the *Original Pick* fits inside the allowed window. This constraint can easily be relaxed by an extension of the algorithm picking subparts of a cross-join in a similar fashion with the proposed algorithm, if we consider that we pick subparts of a 2Dslice. For lack of space we do not incorporate this extension too.

We will give two examples for the instantiation of the aforementioned generic algorithm. In the first case (Algorithm *FocusWindow_Min_3x3* in Figure 7), we are interested in a focus window which (a) includes the window with the minimum summary of values and (b) is not bigger than `3x3` (with a tolerance of the surface ε`=1`). We can observe that the guided greedy algorithm picks the window of minimum value as its starting point. The first constraint is met by the original pick and the second by the stop condition of the algorithm. During the expansion phase, each time we choose a cross-join such that (a) it is neighbouring with the current solution; (b) if merged with the current solution, it comprises a rectangle (easily determined by comparing the lengths of the opposite sides of the new solution and (c) has the smallest surface.

If we execute the algorithm on the data of Figure 5, the result will be **`Q`**`={R1/C6,R2/C6,R3/C6,R4/C6}` which is practically the tape `C6`. If instead of the minimum value, in function `Pick` we had chosen the maximum, then the result would be **`Q`**`={R1/C6,R1/C5}`. Another obvious extension would be to employ a 2-greedy algorithm: in this case the small cross-joins `R2/C5` and `R2/C6`, each comprising a single cell, could have been incorporated in the solution too.

```
Algorithm GenericFocusWindow
Input:
    A set of cross-joins GJ and a display grid of cells Grid related
    to GJ.
    Each cell belonging to Grid is characterized by coordinates
    (x,y) and each CJ belonging to GJ is characterized by the
    coordinates of its upper left and lower right cell. Each cross-join
    has a surface, determined by its coordinates.
Parameters:
    OriginalPick(GJ): a routine to determine the starting
    cross-join of the algorithm
    GuardCondition: a routine to determine whether the
    algorithm should stop
    $\varepsilon$: a tolerance, or error range for the acceptance of a solution or
    not
    Qualifies: a Boolean function that determines whether a
    solution satisfies a set of constraints
    DeterminingQuality: a property of a cross-join like
    surface, sum of values, …
    Pick(GJ,Q): a routine picking a cross-join to enlarge the
    produced solution
Output:
    A set of cross-joins, Q that satisfies the conditions set by the
    user.
Begin
    Q = {}
    C = OriginalPick(GJ)
    Add C to Q.
    While (GuardCondition) {
        CJ = Pick(GJ,Q);
        If CJ≠NULL Then add CJ to Q  Else exit the loop
    }
    Return Q
End.
```

Figure 6. Algorithm *GenericFocusWindow*

```
Algorithm FocusWindow_Min_3x3
Input:
    A set of cross-joins GJ and a display grid of cells Grid related
    to GJ.
    Each cell belonging to Grid is characterized by coordinates
    (x,y) and each CJ belonging to GJ is characterized by the
    coordinates of its upper left and lower right cell. Each cross-join
    has a surface, determined by its coordinates.
Parameters:
    OriginalPick(GJ): start with cross-join having the
    minimum summary
    GuardCondition (Q, $\varepsilon$): the surface is closest to 3x3
    $\varepsilon$: 1 cell$^2$
    Qualifies: a Boolean function that determines whether a
    solution satisfies a set of constraints
    DeterminingQuality(Q): distance from the ideal 3x3
    surface
    Pick(GJ,Q): a routine picking the cross-join with minimum
    distance from the ideal 3x3 surface
Output:
    A set of cross-joins, Q that satisfies the conditions set by the
    user.
Begin
    Q = {}
    C = OriginalPick(GJ)
    Add C to Q.
    While (GuardCondition){
        CJ=Pick(GJ,Q);
        If CJ≠NULL Then add CJ to Q  Else exit the loop
    }
    Return Q
End.
OriginalPick(GJ){
    Let the cross-join $C_r$ s.t., |sum($C_r$) | is the minimum;
    Among equals pick the upper and left-wise;
    Return ($C_r$);
}
DeterminingQuality(Q) {
    Return surface(Q)-surface(3x3);
}
GuardCondition (Q,1){
    If surface(Q)-surface(3x3) <1  Then Return true;
    Else Return false
}
Pick(CJ,Q){
    Let V be the subset of the cross-joins of CJ, s.t., for each
            v∈V: Qualifies(v,CJ,Q)
    Let $v_P$∈V be a cross-join s.t., |DeterminingQuality(Q)|
            is minimum, if $v_P$ is added to Q.
    Return $v_P$;
}
Qualifies(v,CJ,Q){
    If  (v is adjacent to a cross-join CJ∈CJ) &&
        (v ∪ Q forms a rectangle)
    Then Return true;
    Else Return false
}
```

Figure 7. Algorithm *FocusWindow_Min_3x3*

In a different example (Figure 8), we also demonstrate an algorithm producing a focus window which (a) includes the window with the maximum summary of values and (b) is not bigger than 3x3 (with a tolerance of the surface •=0) and (c) optimizes a combined formula over surface and value. Each time we pick the cross-joins that brings us closest to the 3x3 desideratum, while having the highest summary value. In the case of our motivating example, the solution is **Q**={R4/C3,R4/C4} with an exact 3x3 surface.

4. DISCUSSION

At this point, we would like to take the time to discuss the larger framework of the contribution of this paper. First, as the Lowell report [7] mentions, visualization is one of the big issues of database research for the next years. To copy from the Lowell report, "The original Laguna-Beach report lamented that there was little research on user interfaces to DBMSs. … There have not been comparable advances in the last 15 years. There is a crying need for better ideas in this area". We claim that of all fields of database research, decision support and OLAP are the ones to be affected most out of this phenomenon.

```
Algorithm FocusWindow_Max_Weighted
Input:
    Same as in Algorithm FocusWindow_Min_3x3
Parameters:
    OriginalPick(GJ): start with cross-join having the
    maximum summary
    GuardCondition (Q, $\varepsilon$): the surface is closest to
    3x3
    $\varepsilon$: 0 cell$^2$
    Qualifies: a Boolean function that determines
    whether a solution satisfies a set of constraints
    DeterminingQuality(Q): a combined formula each
    time picking the cross-join that brings us closest to the
    3x3 desideratum, while having the highest summary
    value.
    Pick(GJ,Q): a routine picking the cross-join with
    maximum Determining Quality
Output:
    A set of cross-joins, Q that satisfies the conditions set by
the user.
Begin
    Q = {}
    C = OriginalPick(GJ)
    Add C to Q.
    While (GuardCondition){
        CJ=Pick(GJ,Q);
        If CJ$\neq$NULL Then add CJ to Q  Else exit the loop
    }
    Return Q
End.
OriginalPick(GJ){
    Let the cross-join $C_r$ s.t., |sum($C_r$) | is the maximum;
    Among equals pick the upper and left-wise;
    Return ($C_r$);
}
DeterminingQuality(Q) {
    Return (1 - [surface(3x3)-surface(Q)] / 9 )*
            [sum(Q)-sum(OriginalPick(CJ))];
}
GuardCondition (Q,0){
    If surface(Q)-surface(3x3) < 0  Then
            Return true;
    Else Return false;
}
Pick(CJ,Q){
    Let V be the subset of the cross-joins of CJ, s.t., for each
      v$\in$V: Qualifies(v,CJ,Q)
     Let $v_P \in$ V be a cross-join s.t.,
      |DeterminingQuality(v $\cup$ Q)|  is maximum
    Return $v_P$;
}
Qualifies(v,CJ,Q){
     If  (v is adjacent to a cross-join CJ$\in$CJ)  &&
          (v  $\cup$  Q  forms a rectangle)  &&
          (surface(v$\cup$Q)-surface(3x3) ) < 0
    Then Return true;
    Else Return false
}
```

Figure 8. Algorithm *FocusWindow_Max_Weighted*

Someone could possibly question the need for a new model. For us it is clear that one of the main reasons for the research community not dealing with visualization issues so far, is the heritage of the computing paradigm of the past three decades. This paradigm silently made the assumption that the user sitting in front of a console makes *one* query and retrieves *one* answer (as would have happened in a UNIX terminal thirty years ago). This is not the case with modern user interfaces for datasets, especially in the context of OLAP. The user makes simultaneously *many* queries, combined in one or more screens; nevertheless, all our modeling techniques and languages so far (from the relational model, to SQL and the OLAP modeling efforts proposed in the academia) simply ignore this fact. Our effort tries to formalize the simultaneous presence of more than one queries and this is done in two layers. In the presentational layer we provide a uniform and generic model for the user interface, which hides the complexity of answer retrieval, detached in the logical layer. As a second interesting difference, note that the users work in *sessions* of queries, as opposed to sequences of unrelated queries. OLAP is a typical, but not the only, case for this behavior.

As a first attempt towards the issue, we have carefully selected a visualization technique from the fields of Human-Computer Interaction and Information Visualization with the particularity of being crafted specifically for tabular data and we have customized it for OLAP. Naturally, we do not claim that this is the ultimate solution to the problem, but rather we wish to indicate that there is quite an interesting research field in this area and a supportive body of knowledge from other disciplines, such as Human-Computer Interaction and Information Visualization.

At the same time, new hardware developments pose new requirements for our visualization techniques. One of our goals is to implement OLAP visualization techniques for particularly small devices such mobile phones and palmtops. Although the processing power of these gadgets is no more negligible (actually, the buzzword 'thin client' seems to disappear from the standard vocabulary of the area) their screen sizes shrink over time. To make OLAP screens presentable to such devices one can follow several paths, such as: (a) show only high level summaries which involve small 2D slices or (b) show simply pie- or bar-charts. We choose an alternative approach where (a) the contents of the screen do not have to be squeezed in size in order to fit in the screen, and most importantly (b) the report does not have to be rewritten neither do we have to check for the aggregation level of the presented data. On the contrary, a certain part of the report is presented depending on the particularities of the device. Here, we make the reasonable assumption that either the device has the computational power to determine the amount of cells that can be presented to the user or, if this is not an option, the device can at least piggy-back its characteristics to the OLAP server and let the server decide on the focus window.

Third, making the discussion a little broader, we bring up the Table Lens technique to highlight the possibility of making proactive user decision support in the presence of large datasets (in our case, the value axis is quite larger than the size that someone can handle efficiently). Clearly, as report screens are limited not only due to hardware constraints, but also due to the particularities of human nature (e.g., the classical discussion on the limited capacity of persons in processing information [9]), it comes quite natural that automated proactive support to the users is thus one of the new requirements that decision support tools have to provide. Thus, this end of our contribution is related to a broader line of research [2,13].

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have demonstrated how the Cube Presentation Model, a novel presentation model for OLAP data can be naturally mapped into an advanced visualization technique, the Table Lens. Initially, we have defined the mapping scheme from the Cube Presentation Model to Table Lens entities and objects. Then, we have introduced suitable algorithms for proactive automated support of the user towards the highlighting of interesting areas of a report. Finally, we have discussed on the usefulness and applicability of the proposed techniques to modern technological developments.

Next steps in our research include the introduction of suitable, CPM specific, visualization techniques that comply to current standards and recommendations as far as usability and user interface design is concerned and its extension to address the specific visualization requirements of mobile and wireless OLAP, as this notion can be supported and implemented on mobile devices and palmtops.

6. REFERENCES

[1] M. Gebhardt, M. Jarke, S. Jacobs: A Toolkit for Negotiation Support Interfaces to Multi-Dimensional Data. ACM SIGMOD 1997, pp. 348 – 356.

[2] J. Han. Towards On-Line Analytical Mining in Large Databases. SIGMOD Record, 27(1): 97-107, 1998.

[3] W.H. Inmon: *Building the Data Warehouse*. John Wiley & Sons, 1996.

[4] Alfred Inselberg: *Visualization and Knowledge Discovery for High Dimensional Data*. 2nd Workshop Proceedings UIDIS, IEEE, 2001.

[5] D.A. Keim. *Visual Data Mining*. Tutorials of the 23rd International Conference on Very Large Data Bases, Athens, Greece, 1997.

[6] Hans-J. Lenz, Bernhard Thalheim. *OLAP Databases and Aggregation Functions*. In Proc. of the 13th International Conference on Scientific and Statistical Database Management (SSDBM'01), 2001.

[7] Various Authors: *The Lowell Database Research Self Assessment*. Lowell, Massachusetts USA, May 4-6, 2003. Available at: http://research.microsoft.com/~Gray/lowell/.

[8] Andreas Maniatis, Panos Vassiliadis, Spiros Skiadopoulos, Yannis Vassiliou: *Advanced Visualization for OLAP* (long version).

[9] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63, 81-97 (1956).

[10] Microsoft Corp. *OLEDB for OLAP February 1998*. Available at: http://www.microsoft.com/data/oledb/olap/.

[11] Andreas Maniatis, Panos Vassiliadis, Spiros Skiadopoulos, Yannis Vassiliou: *CPM: A Cube Presentation Model for OLAP*. DaWaK 2003, Prague, Czech Republic, September 3 – 5 2003.

[12] Peter Pirollo, Ramana Rao: *Table Lens as a Tool for Making Sense of Data*. Proceedings of the AVI '96 Workshop, Gubbio, Italy, June 1996.

[13] S. Sarawagi, R. Agrawal, N. Megiddo: *Discovery-Driven Exploration of OLAP data Cubes*, Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, March 1998.

[14] Ramana Rao, Stuart K. Card: The Table Lens: Merging Graphical and Symbolic Representations in an effective Focus + Context Visualization for Tabular Information. Proceedings of the ACM SIGCHI (CHI '94), Boston, Massachusetts USA, April 24-28, 1994.

[15] Thomas Ruf, Juergen Georlich, Ingo Reinfells: *Dealing with Complex Reports in OLAP Applications*. DaWaK '99, Florence, Italy, August 30th – September 1st 1999.

[16] Aris Tsois, Nikos Karayannidis, Timos Sellis: *MAC: Conceptual Data Modeling for OLAP*. Proc. of the International Workshop on DMDW 2001.

[17] P. Vassiliadis, T. Sellis: *A Survey on Logical Models for OLAP Databases*. SIGMOD Record, vol. 28, no. 4, December 1999.

[18] Panos Vassiliadis, Spiros Skiadopoulos: *Modeling and Optimization Issues for Multidimensional Databases*. Proc. of CAiSE'00, Stockholm, Sweden, 2000.