A Presentation Model and Non-Traditional Visualization for OLAP

Andreas Maniatis (contact author)[1], Panos Vassiliadis[2], Spiros Skiadopoulos[1], Yannis Vassiliou[1], George Mavrogonatos[1], Ilias Michalarias[1]

[1] National Technical University of Athens
Dept. of Elec. and Computer Eng.,
Knowledge & Database Systems Laboratory
15780 Athens, Hellas
{andreas,spiros,yv}@dblab.ece.ntua.gr
{el99201,el99031}@central.ntua.gr

[2] University of Ioannina,
Dept. of Computer Science
45110 Ioannina, Hellas
pvassil@cs.uoi.gr

Abstract. Data visualization is one of the major issues of database research and OLAP, being a decision support technology, is clearly in the center of this effort. Still, so far, visualization has not been incorporated in the abstraction levels of DBMS architecture (conceptual, logical, physical), neither has it been formally treated in this context. In this paper we start by reconsidering the separation of the aforementioned abstraction levels to take visualization into consideration. Then, we present the Cube Presentation Model (CPM), a novel presentational model for OLAP screens. The proposal lies on the fundamental idea of separating the logical part of a data cube computation, from the presentational part of the client tool. Then, CPM 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. Based on the particularities of Table Lens, we propose automated proactive support to the user for the interaction with an OLAP screen. Finally, we discuss implementation and usage issues in the context of an academic prototype system (CubeView) that we have implemented.

Keywords: OLAP, Visualization, Presentation Model

INTRODUCTION

In the last years, On-Line Analytical Processing (OLAP) and data warehousing (DW) have become major research areas in the database community (Inmon, 1996, Abiteboul et al., 2003). Although the *modeling* of data (Tsois et al., 2001, Vassiliadis & Sellis, 1999) has been extensively dealt with, an equally important issue in the OLAP domain, the *presentation* of data has not been adequately investigated.

As the Lowell report (Abiteboul et al., 2003) 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*". It is easy to understand that of all fields of database research, decision support and OLAP are the ones to be affected most out of this phenomenon.

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 (Abiteboul et al., 2003), and although research has provided results dealing with the presentation of vast amounts of data (Keim, 1997, Inselberg, 2001, Gebhardt et al., 1997), to our knowledge, OLAP has not been part of advanced visualization techniques so far.

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). Still, this is not the case with modern user interfaces for datasets, especially in the context of OLAP. A single front-end screen typically involves the combination of more than one back-end queries. Still, to the best of our knowledge, there are no modeling techniques and languages (from the relational model, to SQL and the OLAP modeling efforts proposed in the academia) that build upon 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.

In this paper, we try to approach the problem from a clean sheet of paper. Although we do not claim to provide a generic answer for all kinds of database visualization problems, we focus on the specifics of the OLAP field. Having observed that presentational models are not really part of the classical conceptual-logical-physical hierarchy of database models (depicted in Fig. 1 below), we propose a new separation of layers. In the sequel, we will refer to the different layers of abstraction, or models, that help us design, manage and operate an OLAP environment through the term *layers*.

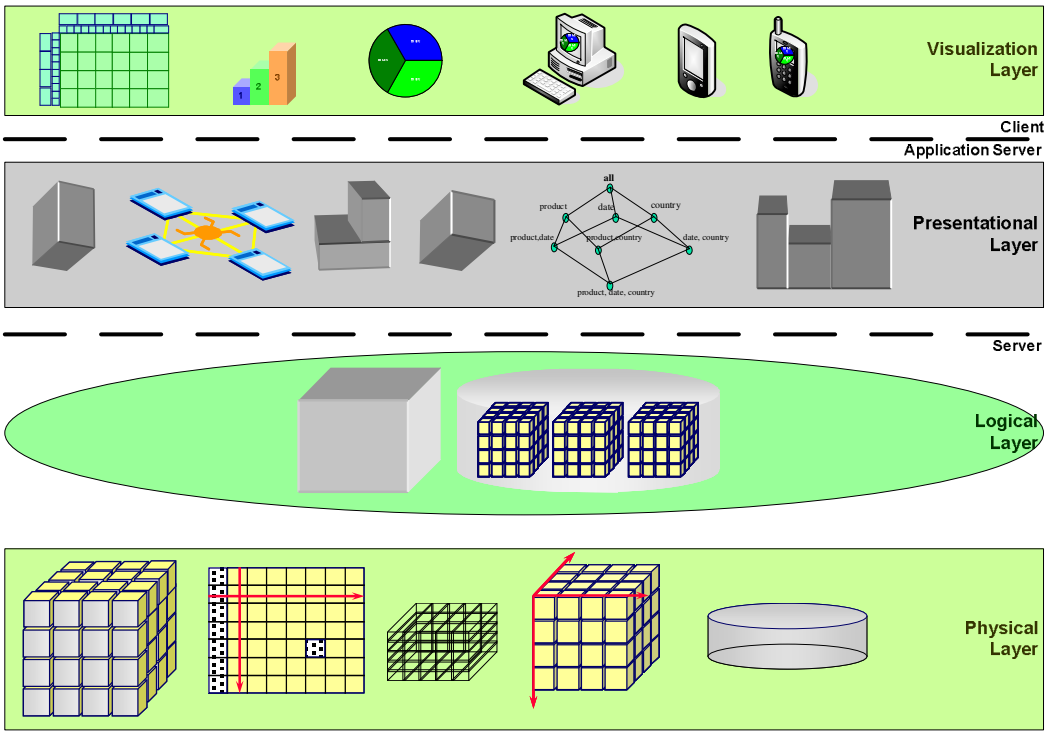

Fig. 1: The general framework for CPM

In the middle, there is a *logical layer* that abstracts from the particularities of data storage and describes cubes and dimensions. This layer is naturally mapped to physical storage entities, like relational tables (ROLAP), or proprietary structures like multidimensional matrices (MOLAP). This kind of physical entities form the *physical layer*. Having these structures covers well enough

the part pertaining to the query formulation part. Still, whereas the logical layer deals with the representation of data in an abstract form, as well as the formulation of queries and operations over them, we need a way to model how the answer to a query is represented in the client part. The role of the logical layer for the server is played by the *presentational layer* for the client, which involves a simple and generic model to abstract from the particularities of data retrieval. The ultimate representation is performed by the specific visualization techniques (pies, bar charts, etc) handled by the *visualization layer*. The presentational layer is generic enough to discuss the broad strategy of how data are to be visualized (e.g., by 2D vs. 3D means, focused for tabular vs. multimedia data, and so on), whereas the visualization layer deals with the particularities of the final visualization means (e.g., palmtop, printer, virtual reality environment, and so on).

As one can see in Fig. 1, there is a part of the functionality that pertains to the server and part of it that pertains to the application server and the client. Naturally, the distinguishing lines between this 3-tier architecture can be rearranged for a 2-tier or a 4-tier architecture easily. Note also that conceptual modeling is orthogonal to this classification for two reasons: (a) conceptual models are not really part of the OLAP engines or environments and (b) in the OLAP field, the logical and conceptual level are quite often indistinguishable (Kimbal et al., 1998).

Someone could possibly question the need for new models. Is it really necessary to depart from the well known classification of models? Our answer is positive and we base our proposal on the following reasons:

- First, we need to allow the formal definition of the presentation of the result of a database query – in our case of a cube. This is currently done in an ad hoc manner from the client tools; no formal foundations are given for this kind of representation.
- Second, we need to decouple the definition of the logical underpinnings of user operations from the way the result is presented. Even if we create a model for the presentation of the results, we need to keep it loosely coupled with the model of the underlying data. One could claim that a generic, complete model could cover all cases; still, to our understanding, such a model is too hard to achieve. Therefore, the natural antidote to the lack of completeness should be applied and this is genericity: by decoupling logical and presentational models, we can decide how we match a particular logical model to a presentational one, among many choices.
- Third, we need to be able to depart from the traditional thinking of treating visualization as appropriate only for computer screens. On the contrary, we live in an age where a computer screen is just one of the possible choices for the presentation of data. Bundling the choices for different presentation devices with the logical models and languages would probably create hard-to-use constructs.

In the context of all the aforementioned issues, in this paper, we move on to make the following contributions.

First, we introduce *(CPM – Cube Presentation Model)*, a presentation model for OLAP and we combine it with non-traditional visualization techniques. 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). The logical layer that we propose is based on an extension of a previous proposal (Vassiliadis & Skiadopoulos, 2000) to incorporate more complex cubes. The presentational layer, at the same time, provides a formal model for OLAP screens. To our knowledge, there is no such result in the related literature.

Once CPM has been introduced, we move on to give 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 (Rao & Card, 1994, Pirollo & Rao, 1996) 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. In the sequel, 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 and customize it to a particular instance to show how it could actually work. By exploiting Table Lens along with suitable coloring schemes, we provide a new presentation technique which we call *OLAP Lens*.

Moreover, an academic software platform specifically designed and implemented to support both CPM and OLAP Lens is introduced. The architecture of the platform, which is called *CubeView*, is particularly tailored to support for Mobile OLAP, a term used to denote the porting of OLAP Visualization and Analysis applications onto portable, mobile and wireless devices.

To motivate the discussion, throughout the paper we will use a running example, where we customize the example presented in (Microsoft, 1998) to an international publishing company, with traveling salesmen, selling books and CD's to other bookstores all over the world (Fig. 2). In this example we assume that 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 (Microsoft, 1998) terminology) as many dimensions as necessary in a single axis. In this case, we combine the dimensions `Salesman` (restricted by the query author to two particular salesmen, i.e., `Salesman` in `{'Venk','Netz'}`) 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.

```
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
Product = ALL

			C1		C2	C3	C4		C5	C6
			Venk				Netz			
			USA			Japan	USA			Japan
			USA_N		USA_S		USA_N		USA_S	
		Size (City)	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	Qtr2		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

Fig. 2: Motivating example for the cube model (taken from (Microsoft, 1998)).

The remainder of this paper is structured as follows: In Section 2, we summarize the logical layer of CPM. In Section 3, we present in detail the presentation layer of CPM. In Section 4, we show how CPM can be naturally combined with Table Lens and how we can automate the task of proactively supporting the user with highlighted areas of interest. In Section 5, we describe a prototype platform, CubeView, where we have implemented the proposed visualization schemes. In Section 6 we present work closely related to our research and finally, in Section 7 we conclude our results and present topics for future work.

THE LOGICAL FOUNDATIONS

In this section, we present the logical layer of CPM; to this end, we extend a logical model (Vassiliadis & Skiadopoulos, 2000) in order to compute more complex cubes. In a nutshell, the logical layer involves (a) *dimensions* defined as lattices of dimension *levels*, (b) *ancestor functions* (in the form of anc_{L2}^{L1}) 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.

Formally, the constructs of the (Vassiliadis & Skiadopoulos, 2000) model are:

- Four countable pairwise disjoint infinite sets exist: a set of *level names* (or simply *levels*) U_L, a set of *measure* names (or simply *measures*) U_M, a set of *dimension* names (or simply *dimensions*) U_D and a set of *cube* names (or simply *cubes*) U_C. The set of *attributes* U is defined as $U=U_L \cup U_M$. For each $A \in U_L$, we define a countable totally ordered set $dom(A)$, the domain of A, which is isomorphic to the integers. Similarly, for each $A \in U_M$, we define an infinite set $dom(A)$, the domain of A, which is isomorphic to the real numbers. We can impose the usual comparison operators to all the values participating to totally ordered domains $\{<, >, \leq, \geq\}$. We also assume the existence of two attributes ALL, RANK. The role of the special attribute ALL will be analyzed in the sequel. Level ALL has a single value in its domain, namely 'all'. RANK is a special purpose measure, which will be used for the ordering of a cube. The domain of RANK is the set of integers.
- A *dimension* D is a lattice $(\mathbf{L}, \prec)$ such that:
  - $\mathbf{L}=(L_1, \ldots, L_n)$, is a finite subset of U_L.
  - $dom(L_i) \cap dom(L_j) = \varnothing$ for every $i \neq j$.
  - $\prec$ is a partial order defined among the levels of $\mathbf{L}$.

  Each path in the dimension lattice, beginning from its upper bound and ending in its lower bound is called a *dimension path*.
- A family of functions $anc_{L_1}^{L_2}$ is defined, satisfying the following conditions:
  - For each pair of levels L_1 and L_2 such that $L_1 \prec L_2$ the function $anc_{L_1}^{L_2}$ maps each element of $dom(L_1)$ to an element of $dom(L_2)$.
  - Given levels L_1, L_2 and L_3 such that $L_1 \prec L_2 \prec L_3$, the function $anc_{L_1}^{L_3}$ equals to the composition $anc_{L_1}^{L_2} \circ anc_{L_2}^{L_3}$. This implies that:
    1. $anc_{L_1}^{L_1}(x)=x$.
    2. if $y=anc_{L_1}^{L_2}(x)$ and $z=anc_{L_2}^{L_3}(y)$, then $z=anc_{L_1}^{L_3}(x)$.
  - for each pair of levels L_1 and L_2 such that $L_1 \prec L_2$ the function $anc_{L_1}^{L_2}$ is monotone (preserves the ordering of values). In other words:

    $$\forall x, y \in dom(L_1): x<y \Rightarrow anc_{L_1}^{L_2}(x) \leq anc_{L_1}^{L_2}(y), L_1 \prec L_2$$
- A *schema* S is a finite subset of U. Normally, we will represent a schema as divided in two parts: $S=[D_1.L_1, \ldots, D_n.L_n, A_1, \ldots, A_m]$, where:
  - $(L_1, \ldots, L_n)$ are levels from a dimension set $\mathbf{D}=(D_1, \ldots, D_n)$ and level L_i comes from dimension D_i, for $1 \leq i \leq n$.
  - $(A_1, \ldots, A_m)$ are attributes, i.e. measures and levels.
- A *detailed schema* S^0 is a schema whose levels are the lowest in the respective dimensions. When we refer to a level L as the *lowest* in the dimension, it means that there does not exist any other level L', such that $L' \prec L$.
- A *tuple* t over a schema $S=[L_1, \ldots, L_n, A_1, \ldots, A_m]$ is a total and injective mapping from S to $dom(L_1) \times \ldots \times dom(L_n) \times dom(A_1) \times \ldots \times dom(A_m)$, such that $t[X] \in dom(X)$ for each $X \in S$.

- A *data set* DS over a schema $S=[L_1,\ldots,L_n,A_1,\ldots,A_m]$ is a finite set of tuples over S such that:
  - $\forall\ t_1,t_2 \in DS,\ t_1[L_1,\ldots,L_n]=t_2[L_1,\ldots,L_n] \Rightarrow t_1=t_2$.
  - for no strict subset $X \subset \{L_1,\ldots,L_n\}$, the previous also holds.

  In other words, $A_1,\ldots,A_m$ are functionally dependent (in the relational sense) on levels $(L_1,\ldots,L_n)$ of schema S. A *detailed data set* DS^0 is a data set over a detailed schema S.
- An atom is `true`, `false` (with obvious semantics) or an expression of the form $x\ \partial\ y$, where x and y can be one of the following: (a) a level L_1 (i.e. not a measure); (b) a value l; (c) an expression of the form $anc_{L_1}^{L_2}(L_1)$ where $L_1 \prec L_2$; (d) an expression of the form $anc_{L_1}^{L_2}(l)$ where $L_1 \prec L_2$ and $l \in dom(L_1)$. If x and y are levels then they should belong to isomorphic dimensions. ∂ is an operator from the set ($>$, $<$, $=$, $\geq$, $\leq$, $\neq$).
- A *selection condition* φ is a formula involving atoms and the logical connectives $\wedge$, $\vee$ and $\neg$. A selection condition is always applied to a data set such that all the level names occurring in the selection condition – either \ in the form (1) or (3) – belong to the schema of the data set. Let DS be a data set over schema S. The expression $\varphi(DS)$ is a set of tuples $\mathbf{x}$ belonging to DS such that when, for all the occurrences of level names in φ, we substitute the respective level values of every $x \in \mathbf{x}$, the formula φ becomes true. A *detailed selection condition* φ^0 is a selection condition where all participating levels are the detailed levels of their dimensions.
- A *primary cube* c (over the schema $[L_1,\ldots,L_n,M_1,\ldots,M_m]$), is an expression of the form:

  $c=(DS^0,\varphi,[L_1,\ldots,L_n,M_1,\ldots,M_m],[agg_1(M_1^0),\ldots,agg_m(M_m^0)])$, where:
  - DS^0 is a detailed data set over the schema $S=[L_1^0,\ldots,L_n^0,M_1^0,\ldots,M_k^0], m \leq k$.
  - φ is a detailed selection condition.
  - $M_1, \ldots, M_m$ are measures.
  - L_i^0 and L_i are levels such that $L_i^0 \prec L_i$, $1 \leq i \leq n$.
  - $agg_i \in \{sum,min,max,count\}$, $1 \leq i \leq m$.

  The expression characterising a cube has the following semantics:

$$c=\{x \in Tup(L_1,\ldots,L_n,M_1,\ldots,M_m) \mid \exists y \in \varphi(DS^0),$$

$$x[L_i]=anc_{L_i^0}^{L_i}(y[L_i^0]), 1 \leq i \leq n,$$

$$x[M_j]=agg_j(\{q \mid \exists z \in \varphi(DS^0), x[L_i]=anc_{L_i^0}^{L_i}(z[L_i^0]), 1 \leq i \leq n,\ q=z[M_j^0]\}), 1 \leq j \leq m\}$$

In other words, a cube c is a set of tuples. To compute it, first we apply the selection condition to the detailed data set. Then, we replace the values of the levels for the tuples of the result, with their respective ancestor values (at the levels of the schema of c) and group them into a single value for each measure, through the application of the appropriate aggregate function.

Coming back to our motivating example, we can detect the following dimensions:

- `arrival date` and `departure date` (when the salesman arrives/leaves the store).
- `salesman` (instantiated in Figure 4).
- `product` (instantiated in Figure 5).
- `location` (instantiated in Figure 6).

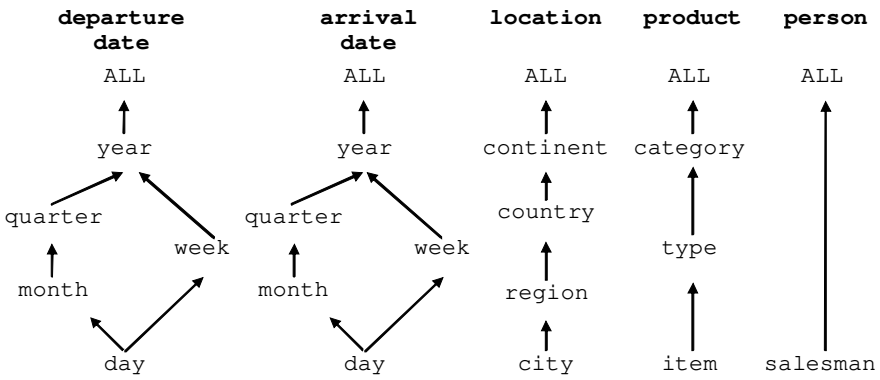

Fig. 3: The dimensions `arrival date`, `departure date`, `location`, `product` and `salesman`.

The functionally dependent measures are `Sales`, `PercentChange`. Our data set DS_0 is depicted in Fig. 7. Based on this data set we can define a basic primary cube as:

```
c⁰=(DS⁰,true,[arrival day.day, departure day.day, product.item,
      person.salesman, location.city,sales],sum(sales))
```

For brevity we can write c^0 as:

```
c⁰=(DS⁰,true,[arrival day,departure
day,item,salesman,city,sales],sum(sales)).
```

Salesman
Venk
Netz

Fig. 4: Dimension `person`.

Category	Type	Item
Books	Literature	"Report to El Greco"
		"Karamazof brothers"
	Philosophy	"Zarathustra"
		"Symposium"
Music	Heavy Metal	"Piece of Mind"
		"Ace of Spades"

Fig. 5: Dimension `product`.

Continent	Country	Region	City
Europe	Greece	Greece-North	Salonica
		Greece-South	Athens
			Rhodes
North America	USA	USA-East	New York
			Boston
		USA-West	Los Angeles
			San Francisco
		USA-North	Seattle
Asia	Japan	Kiusiu	Nagasaki
		Hondo	Tokyo
			Yokohama
			Kioto

Fig. 6: Dimension `location`.

Arrival Day	Departure day	Item	Salesman	City	% Change	Sales
1-Jan-97	3-Jan-97	"Report to El Greco"	Netz	Rhodes	10	10
1-Jan-97	3-Jan-97	"Symposium"	Netz	Rhodes	20	5
6-Feb-97	17-Feb-97	"Symposium"	Netz	Athens	-30	7
6-Feb-97	17-Feb-97	"Karamazof brothers"	Netz	Athens	-50	10
6-Feb-97	17-Feb-97	"Piece of Mind"	Netz	Athens	+35	13
18-Feb-97	10-May-97	"Karamazof brothers"	Netz	Seattle	-50	5
11-May-97	7-Jun-97	"Report to El Greco"	Netz	Los Angeles	100	2
11-May-97	7-Jun-97	"Ace of Spades"	Netz	Los Angeles	100	20
3-Sep-97	5-Sep-97	"Zarathustra"	Netz	Nagasaki	0	50
3-Sep-97	5-Sep-97	"Report to El Greco"	Netz	Nagasaki	0	30
6-Sep-97	16-Dec-97	"Piece of Mind"	Netz	Tokyo	10	10
1-Jul-97	4-Aug-97	"Ace of Spades"	Venk	Salonica	30	13
1-Jul-97	4-Aug-97	"Piece of Mind"	Venk	Salonica	50	34
6-Sep-97	12-Oct-97	"Symposium"	Venk	Boston	-30	7
6-Sep-97	12-Oct-97	"Zarathustra"	Venk	Boston	0	10
1-Feb-98	10-Apr-98	"Ace of Spades"	Venk	Seattle	50	15
1-Feb-98	10-Apr-98	"Piece of Mind"	Venk	Seattle	6	53
4-May-98	7-Jun-98	"Report to El Greco"	Venk	Kyoto	-30	14
13-Jun-98	15-Jul-98	"Zarathustra"	Venk	Nagasaki	0	50
13-Jun-98	15-Jul-98	"Report to El Greco"	Venk	Nagasaki	0	30

Fig. 7: Data Set.

A primary cube can be defined as follows:

$$c=(DS^0, anc^{departure.year}_{departure.day}='1997',\ [arrival.month, departure.month, category, salesman.ALL, continent, sum_sales], sum(sales))$$

with the following data values:

Arrival month	Departure month	Category	Salesman.ALL	Continent	Sales
Jan-97	Jan-97	Books	all	Europe	15
Feb-97	Feb-97	Books	all	Europe	17
Feb-97	Feb-97	Music	all	Europe	13
Feb-97	May-97	Books	all	North-America	5
May-97	Jun-97	Books	all	North-America	2
May-97	Jun-97	Music	all	North-America	20
Jul-97	Aug-97	Music	all	Europe	47
Sep-97	Sep-97	Books	all	Asia	80
Sep-97	Oct-97	Books	all	North-America	17
Sep-97	Dec-97	Music	all	Asia	10

Fig. 8: A primary cube

The limitations of primary cubes is that, although they model accurately `SELECT-FROM-WHERE-GROUPBY` queries, they fail to model (a) ordering, (b) computation of values through functions and (c) selection over computed or aggregate values (i.e., the `HAVING` clause of a SQL query). To compensate this shortcoming, we extend the aforementioned model with the following entities:

- Let $\mathbf{F}$ be a set of *functions* mapping sets of attributes to attributes. We distinguish the following major categories of functions: *property functions*, *arithmetic functions* and *control functions*. For example, for the level `Day`, we can have the property function `holiday(Day)` indicating whether a day is a holiday or not. An arithmetic function is for example `Profit=(Price-Cost)*Sold_Items`. Control functions simulate the control statements of the programming languages.
- A *secondary selection condition* ψ is a formula in disjunctive normal form. An atom of the secondary selection condition is `true`, `false` or an expression of the form $x \theta y$, where x and y can be one of the following: (a) an attribute A_i (including `RANK`), (b) a value l, an expression of the form $f_i(\mathbf{A}_i)$, where $\mathbf{A}_i$ is a set of attributes (levels and measures) and (c) θ is an operator from the set ($>$, $<$, $=$, $\geq$, $\leq$, $\neq$). With this kind of formulae, we can compute relationships between measures (`Cost>Price`), ranking and range selections (`ORDER BY...;STOP after 200, RANK[20:30]`), measure selections (`sales>3000`), property based selection (`Color(Product)='Green'`).
- Suppose a data set `DS` over the schema $[A_1, A_2, \ldots, A_z]$. Without loss of generality, suppose a non-empty subset of the schema $S=A_1, \ldots, A_k, k \leq z$. Then, there is a set of *ordering operations* O_S^{θ}, used to sort the values of the data set, with respect to the set of attributes participating to S. θ belongs to the set $\{<,>,\varnothing\}$ in order to denote ascending, descending and no order, respectively. An ordering operation is applied over a data set and returns another data set which obligatorily encompasses the measure `RANK`.
- A *secondary cube* over the schema $S=[L_1, \ldots, L_n, M_1, \ldots, M_m, A_{m+1}, \ldots, A_{m+p}, \text{RANK}]$ is an expression of the form: $s=[c, [A_{m+1}:f_{m+1}(\mathbf{A}_{m+1}), \ldots, A_{m+p}:f_{m+p}(\mathbf{A}_{m+p})], O_{\mathbf{A}}^{\theta}, \psi]$ where $c=(DS^0, \varphi, [L_1, \ldots, L_n, M_1, \ldots, M_m], [agg_1(M_1^0), \ldots, agg_m(M_m^0)])$ is a primary cube, $[\mathbf{A}_{m+1}, \ldots, \mathbf{A}_{m+p}] \subseteq [L_1, \ldots, L_n, M_1, \ldots, M_m]$, $\mathbf{A} \subseteq S-\{\text{RANK}\}$, $f_{m+1}, \ldots, f_{m+p}$ are functions belonging to $\mathbf{F}$ and ψ is a secondary selection condition.

A secondary cube has the following formal semantics:

$s=\{x \in Tup(L_1, \ldots, L_n, M_1, \ldots, M_m, A_{m+1}, \ldots, A_{m+p}, \text{RANK}) \mid \exists$ data set DS^s defined over the schema $[L_1, \ldots, L_n, M_1, \ldots, M_m, A_{m+1}, \ldots, A_{m+p}], y \in DS^s, y_1 \in c^1 : y[L_i]=y_1[L_i], \quad 1 \leq i \leq n, \quad y[M_i]=y_1[M_i], 1 \leq i \leq m,$ $y[A_{m+i}]=f_{m+i}[\mathbf{A}_{m+i}], 1 \leq i \leq p, x \in \psi \quad (O_{\mathbf{A}}^{\theta}(DS^s))\}$

- A *star schema* $(\mathbf{D}, S^0)$ is a couple comprising a finite set of dimensions $\mathbf{D}$ and a detailed schema S^0 defined over (a subset of) these dimensions
- A *star databases instance* over a star schema $(\mathbf{D}, S^0)$ is a triplet $[DS^0, \mathbf{C}, \mathbf{C}^S]$, where
    - DS^0 is a detailed data set defined over S^0;
    - $\mathbf{C}$ is a finite set of cubes, defined over DS^0;
    - $\mathbf{C}^S$ is a finite set of secondary cubes, defined over $\mathbf{C}$.

With these additions, primary cubes are extended to secondary cubes that incorporate: (a) computation of new attributes ($\mathbf{A}_{m+i}$) through the respective functions ($\mathbf{f}_{m+i}$), (b) ordering ($O^{\Theta}_{\mathbf{A}}$) and (c) the HAVING clause, through the secondary selection condition ψ.

THE PRESENTATIONAL LAYER

In this section, we present the *presentation layer* of CPM. As already mentioned, the reason for introducing a presentation layer is two-fold: (a) decouple the definition of the logical underpinnings of user operations or queries from the way the result is presented and (b) allow the formal definition of the presentation of the result. Our approach is based on the introduction of points, axes and layers with a particular focus on two-dimensional spaces that can be represented easily in the regular devices used for data representation (e.g., screens, paper, etc.). Areas of this space can then be mapped to underlying database constructs, such as cubes. In the rest of this section, we will first give an intuitive, informal description of the presentation layer. Then, we will present its formal definition. Throughout this section, we will use the example of Fig. 2, as our reference example.

The most important entities of the presentational layer of CPM – depicted in Fig. 9 – include:

– **Axis:** OLAP is based on the *multidimensional* representation of information. Each dimension of an OLAP environment conceptually represents a different categorization of measures, in the mind of the knowledge worker. In principle, we would prefer each dimension to be graphically represented as an axis (e.g., an axis for `Salesman`, another axis for `Product` and so on). In this case, an axis can be viewed as a set of points and combinations of axes define a multidimensional space, just like in geometry. Nevertheless, in CPM we focus on two dimensional devices for the representation of information. Therefore, for presentation reasons, we will see how we can handle the reduction of spaces with high dimensionality to two-dimensional representations, in order to depict data on 2D devices.

– **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`) and represents a dimension value in any of the levels of the dimension. Still, since we need to multiplex several logical dimensions to one presentational axis, a point will be formally defined to handle this kind of situations, too.

– **Multicubes.** A combination of axis forms a multidimensional space. The data values of the points of the multidimensional spaces are the (aggregate) measure values that have to be computed over the detailed data of the logical model. In this section, we show how constructs in the logical model will be mapped to constructs of the multidimensional space.

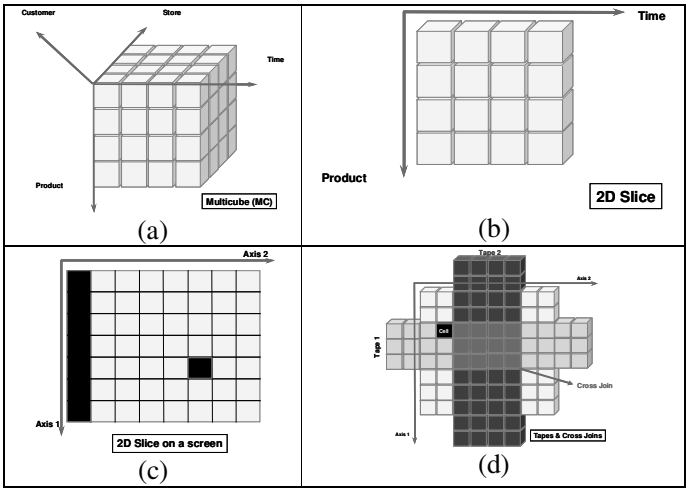

Fig. 9: Mapping CPM objects to 3D and 2D Cross Tabular layouts.

- **2D-slice:** As already mentioned, the focus of CPM is set on two dimensional devices. A 2D-slice is a two dimensional layer of data over the multidimensional space, 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 both Fig. 2 and Fig. 10, we depict such a 2D-slice over a multicube.
- **Tape:** Intuitively, a tape is column or a row over a 2D-slice, i.e., a construct parallel to one of the axis of the 2D-slice. 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. Cross-joins are directly mapped to secondary cubes of the logical model and form the basis of mapping an OLAP screen to a *set* of queries over the underlying database.

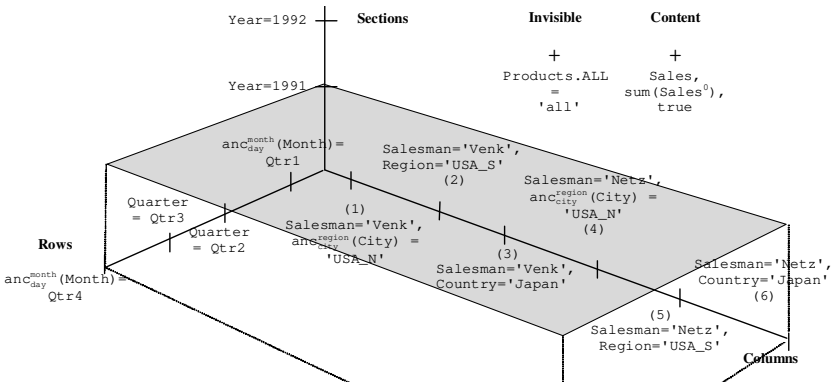

Fig. 10: The 2D-Slice SL for the example of **Fig. 2**.

In terms of *CPM* terminology, the query of Fig. 2 is a 2D-Slice, say `SL` (see also Fig. 3). In `SL` one can identify 4 horizontal tapes denoted as `R1`, `R2`, `R3` and `R4` in Fig. 2) 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 and the Product dimension is ignored. In this multidimensional world of 5 axes, the tapes C1 and R1 are defined as:

C1 = [(Salesman='Venk'∧anc^{region}_{city}(city)='USA_N'),(Year='1991'),
anc^{ALL}_{item}(Products)='all'),(Sales,sum(Sales))]

R1 = [(anc^{month}_{day}(Month)='Qtr1'∧Year='1991'),(Year='1991'),
(anc^{ALL}_{item}(Products)='all'),(Sales,sum(Sales))]

One can also consider the cross-join t1 defined by the common cells of the tapes R1 and C1. Remember that City defines an attribute group along with [Size(City)].

t1=([SalesCube,(Salesman='Venk'∧anc^{region}_{city}(city)='USA_N ∧
anc^{month}_{day}(Month)='Qtr1'∧Year='1991'∧anc^{ALL}_{item}(Products)='all'),
[Salesman,City,Month,Year,Products.ALL,Sales],sum],[Size(City)],true)

In the rest of this section, we will describe the presentation layer of CPM in its formality. First, we will introduce the multidimensional space and then we will show how the contents of the space are computed.

The Multidimensional Space

In this subsection, we will introduce the multidimensional space of the presentational layer. The main entities of the multidimensional space are *axes* and *points*. Before their formal definition though, we will introduce auxiliary entities, necessary to cover the multiplexing of more than one logical dimension to a single axis.

Preliminaries

We assume the existence of the following pairwise disjoint infinitely countable sets: a set of *point names* (or simply *points*) U_P, a set of *axes* names (or simply *axes*) U_X, and a set of *multicube* names (or simply *multicubes*) U_{MC}.

We will also assume their finite subsets, **P** for points, **A** for axes and **MC** for multicubes, each time that we deal with a particular instance.

Before proceeding, we need to extend the definition of dimensions and to deal with multiplexing of dimensions. First, we extend the notion of dimension to incorporate any kind of *attributes* (i.e., results of functions, measures, etc.). Consequently, we consider every attribute not already belonging to some dimension, to belong to a single-level dimension (with the same name as the attribute), with no ancestor functions or properties defined over it. We will distinguish between the dimensions comprising levels and functionally dependent attributes through the terms *level dimensions* and *attribute dimensions*, wherever necessary. The dimensions involving arithmetic measures will be called *measure dimensions*.

Now, we are ready to deal with multiplexing dimensions in a single axis. This is necessary due to the fact that typically data are presented by 2D means, e.g., a screen, meaning that the multidimensional space has to be folded in a 2D projection. For example, observe Fig. 11: the logical dimensions Salesman and Geography have been multiplexed in order to be presented on the same axis. This practically means that for every value of Salesman, all the values of

Geography are repeated. Therefore, in order to be able to represent this kind of structures we need to define groups of attributes, to be multiplexed in the same axis.

An *attribute group* AG is a pair [**A**,**DA**], where **A** is a list of attributes (called the *key* of the group) and **DA** is a list of attributes *dependent* on the attributes of **A**. With the term *dependent* we mean (a) measures dependent over the respective levels of the data set and (b) function results depending on the arguments of the function. One can consider examples of the attribute groups such as ag_1=([City],[Size(City)]),ag_2=([Sales,Expenses],[Profit]).

A *dimension group* DG is a pair [**D**,**DD**], where **D** is a list of dimensions (called the *key* of the dimension group) and **DD** is a list of dimensions *dependent* on the dimensions of **D**. With the term *dependent* we simply extend the respective definition of attribute groups, to cover also the respective dimensions. For reasons of brevity, wherever possible, we will denote an attribute/dimension group comprising only of its key simply by the respective attribute/dimension.

Axes and Points

An *axis schema* is a pair [**DG**,**AG**], where **DG** is a list of K dimension groups and **AG** is an ordered list of K finite ordered lists of attribute groups, where the keys of each (inner) list belong to the same dimension, found in the same position in **DG**, where K>0. The members of each ordered list are not necessarily different. We denote an axis schema as a pair
$AS_K=([DG_1 \times DG_2 \times \ldots \times DG_K],[[ag_1^1,ag_1^2,\ldots,ag_1^{k_1}]\times[ag_2^1,ag_2^2,\ldots,ag_2^{k_2}]\times\ldots\times[ag_k^1,ag_k^2,\ldots,ag_k^{k_k}])\}$.

In other words, one can consider an axis schema as the Cartesian product of the respective dimension groups, instantiated at a finite number of attribute groups. For instance, in the example of Fig. 1, we can observe two axes schemata, having the following definitions:
Row_S = {[Quarter],[Month,Quarter,Quarter,Month]}
Column_S = {[Salesman×Geography], [Salesman]×[[City,Size(City)], Region, Country]}

A *point* is a member of the set U_P.

A *point over an axis schema* AS, is a point tagged with a set of equality selection conditions, one for the key of each attribute group of the axis schema.

For example, given the axis schema [Salesman,[City,Size(City)]], a point can be defined as
p_1=([Salesman='Venk',anc_{city}^{region}(City)='USA_N'])

or, if we wish to incorporate the axis schema in the definition,
p_1=([Salesman,[City,Size(City)]], [Salesman='Venk',anc_{city}^{region}(City)='USA_N'])

An *axis over an axis schema* AS, is a finite list of points, all defined over the axis schema AS.

Practically, an axis is a restriction of an axis schema to specific values, through the introduction of specific constraints for each occurrence of a level.
a = (AS_K,[$\varphi_1,\varphi_2,\ldots,\varphi_K$]),K≤N or
$a=\{[DG_1\times DG_2\times\ldots\times DG_K],[[ag_1^1,ag_1^2,\ldots,ag_1^{k_1}]\times[ag_2^1,ag_2^2,\ldots,ag_2^{k_2}]\times\ldots\times[ag_k^1,ag_k^2,\ldots,ag_k^{k_k}]],\ [[\varphi_1^1,\varphi_1^2,\ldots,\varphi_1^{k_1}]\times[\varphi_2^1,\varphi_2^2,\ldots,\varphi_2^{k_2}]\times\ldots\times[\varphi_k^1,\varphi_k^2,\ldots,\varphi_k^{k_k}]]\}$

We will denote the set of dimension groups of each axis a by dim(a).

In our motivating example, we can observe the following two axes:

Rows = {Row_S,[anc^{month}_{day}(Month)=Qtr1,Quarter=Qtr2,
Quarter=Qtr3,anc^{month}_{day}(Month)=Qtr4]}
Columns = {Column_S,{[Salesman='Venk',Salesman='Netz'],
[anc^{region}_{city}(City)='USA_N', Region='USA_S', Country='Japan']}

Lemma. An axis can be reduced to a finite set of points, if one calculates the Cartesian products of the attribute groups and their respective selection conditions. In other words,

$a=([DG_1 \times DG_2 \times ... \times DG_K],[[p_1,p_2,\ldots,p_l])$, $l=k_1 \times k_2 \times \ldots \times k_{kk}$.

Proof. Obvious. ■

In the sequel, we will mostly treat an axis as a finite set of pairwise disjoint points; therefore we impose the constraint that the selection conditions characterizing each point are pairwise disjoint, too.

We will differentiate between two types of points: atomic and hierarchically decomposable points. The former constitute points defined over single level or measure values, whereas the latter are defined over sets of values.

Atomic points are characterized by the fact that all the equality selection conditions for their attribute groups involve an attribute (level or measure) and a constant. In other words, atomic points are of the form `Level=constant` or `Measure=constant`.

Hierarchically decomposable points are characterized by the fact that the selection condition of one (or possibly more) of their attribute groups involves the usage of an ancestor function.

For example, p_1 is a hierarchically decomposable point

p_1=([Salesman,[City,Size(City)]], [Salesman='Venk',anc^{region}_{city}(City)='USA_N'])

whereas p_2 is an atomic point

p_2=([Time,[Quarter], [Quarter=Qtr2])

Naturally, a hierarchically decomposable point corresponds to a finite set of atomic points (directly stemming from the finiteness of the domain of ancestor functions). Therefore, the aforementioned point p_1 corresponds to the points $p_{1.1}$ and $p_{1.2}$, defined at the `City` level.

$p_{1.1}$=([Salesman,[City,Size(City)]], [Salesman='Venk',City='Seattle'])
$p_{1.2}$=([Salesman,[City,Size(City)]], [Salesman='Venk',City='Boston'])

An axis that comprises only atomic points is an *atomic-level* axis. An atomic-level axis X_a which comprises the atomic points produced from the hierarchical decomposition of the points of an axis X, is the *atomic-level equivalent* of X.

In the sequel, we will refer to points indiscriminately of their type; in the case where we will need to make a distinction, this will be shown clearly.

An *axis tag* is a characterization of an axis with respect to (a) its natural properties and (b) to the fact that it can be visualized in a 2D screen or not. Therefore an axis is characterized as

- `coordinate` vs. `measure`, depending on whether it represent values that determine the coordinates or the internal points of the multidimensional space (see section 3.2)
- `visible` vs. `invisible`, depending on whether we allow its representation on a 2D screen or we use the axis simply for pinpointing values of its points, without involving them in the visualization of the result. For example, in Fig. 2, there is an invisible axis, pinpointing `Year` to `1991` and `Products` to `ALL`.

Finally, we say that two axes schemata are *joinable over a data set* if their key dimensions (a) belong to the set of dimensions of the data set and (b) their points are disjoint. For instance, `Rows_S` and `Columns_S` are joinable.

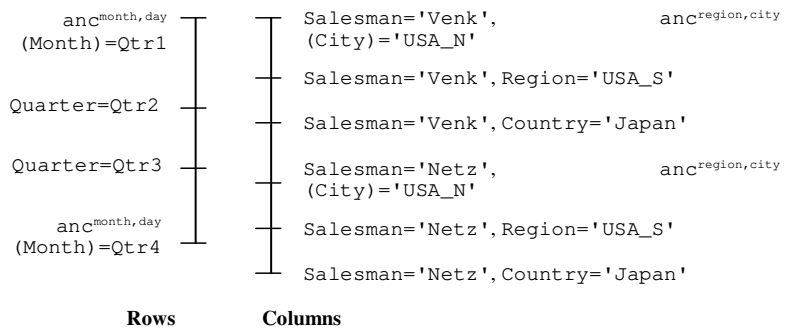

Fig. 11: The points for the axes `Rows` and `Columns`.

The Contents of Multidimensional Space

In this subsection, we will introduce the contents of the multidimensional space of the presentational layer. The main entities of the multidimensional space are *multicubes* which more or less correspond to n-dimensional structures. Prevalent entities in their context are *2D-slices*, standing for 2-dimensional structures that can be presented on a 2D screen, *tapes*, which are 1-dimensional entities and *cross-joins* which are areas of a 2D-slice where two tapes meet.

Multicubes

A *multicube schema* $MC^S=[\mathbf{A}^S]$ is a finite set of axis schemata $\mathbf{A}^S$.

A *multicube* $MC=[\mathbf{A},\mathbf{f}]$, where $\mathbf{A}$ is a finite set of axes and $\mathbf{f}$ is a contents function, mapping coordinates to measures. We require that $\mathbf{A} = \mathbf{C} \cup \{\mathbf{M}\}$, where $\mathbf{C}$ are the *coordinate* axes and $\mathbf{M}$ is the *measure* axis of the multicube.

Let also $\mathbf{M}$ be a measure axis. The points of $\mathbf{M}$ will be computed through queries to the underlying database. Still, it is a regular axis, with the only difference that the same point (e.g., `Sales = 40`) can be repeated more than once (since measures can have identical values). Remember that axes are finite *lists* of points; for measure axes we assume bag semantics underlying this list.

In the simple case, a point is characterized by a single equation of the form [measure=constant]. Still, we can multiplex more than one logical measure in a measure axis and each point of the measure axis is characterized by a set of equations of the form $[measure_1=constant_1,\ldots,measure_k=constant_k]$, depending on the attribute/dimension group that regulates the axis.

In the case of a measure axis, we can tag the schema of the axis with an aggregate function for each of the dimensions participating in the schema. Also, a secondary selection condition can be attached to the schema, acting as a filter for retrieved data.

In our motivating example of Fig. 2, we have a measure axis, named `Content`, comprising 64 points. Observe that the measure axis is defined in terms of the atomic-level equivalents of the involved coordinate axes. The `Time` axis is hierarchically decomposed in 8 values and the `Geography`×`Salesman` axis, also in another 8 points. The measure axis schema is also tagged with the aggregation $Sales=sum(Sales^0)$ and the secondary selection condition `true` (i.e., no selection is performed).

We assume the existence of a *contents* function for **M**. The contents function practically instantiates the points of the measure axis, by computing them as queries over the underlying data set. Each point in the measure axes is then dependent on the points of the atomic-level equivalents of the coordinate axes responsible for its identification. Formally, let $contents_M(\mathbf{C})$: $\mathbf{C} \rightarrow U_p$.

In other words, supposing that there are $K-1$ axes in **C**, $contents_M(\mathbf{C})$ is defined as $[\mathbf{A}_1, \ldots, \mathbf{A}_{K-1}]$, therefore, for every combination of points $[p_1, \ldots, p_{K-1}]$ (each point p_i coming from axis $\mathbf{A}_i$) there exists a point μ in U_p., as the result of the $contents_M(\mathbf{C})$ function. Based on the fact that **C** comprises a finite number of points, then $contents_M(\mathbf{C})$ returns also a finite number of points; nevertheless, as already mentioned, more than one coordinates can map to the same measure value. This fact disqualifies the existence of an inverse function; to compensate for this shortcoming, we assume the mapping $coordinates(\mu)$, such that $coordinates(\mu)=[p_1, \ldots, p_{K-1}]$.

In our motivating example, we can observe the following axes schemata and axes:

```
Row_S = {[Quarter],[Month,Quarter,Quarter,Month]}
Column_S = {[Salesman×Geography], [Salesman]×[[City,Size(City)],
      Region, Country]}
Invisible_S = {[Product×Time],[[Product.ALL]×[Year]]}
Content_S = {[Sales],[Sales=sum(Sales⁰), true]}
```

and their respective axes

```
Rows={[anc_day^month(Month)=Qtr1,Quarter=Qtr2,Quarter=Qtr3, anc_day^month(Month)=Qtr4]}
Columns = {[Salesman='Venk',Salesman='Netz'] ×
      [anc_city^region(City)='USA_N', Region='USA_S', Country='Japan']}
Invisible = {[Year=1991] × [ALL='all']}
Content = {64 points}
```

Then, a multicube MC can be defined as

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

Assume now that we want to present data in multiple spreadsheet sheets, and each sheet comprises a certain year, e.g., the first sheet involves 1991 and the second involves 1992. We can resolve this by adding an extra axis, `Sections` (see also, Fig. 12). The changes are as follows:

Axes schemata:

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

and axes:

```
Sections = {Section_S,[Year=1991,Year=1992]}
Invisible = {Invisible_S,[ALL='all']}
Content = {128 points}
```

Then, the multicube MC can be defined as

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

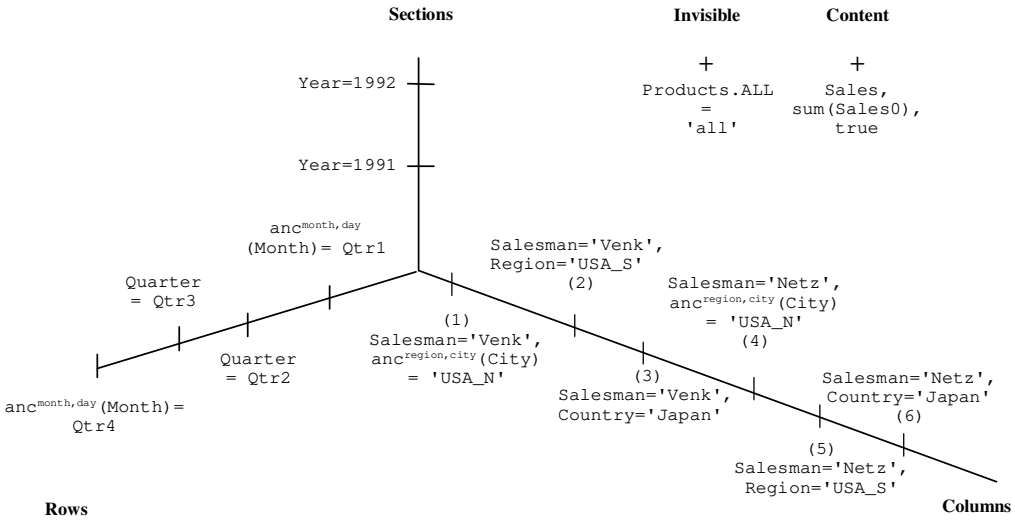

Fig. 12: The multidimensional space for the variant of the motivating example, extended with `Sections`

2D-Slices

In the beginning of this section, we have informally introduced 2D-Slices. Intuitively, a 2D-slice represents a bounded two dimensional plane. To achieve this it is only necessary to pin the axes of the multicube to specific points, except for 2 axes, which are left free. Then, these two axes define a two dimensional plane that can be presented on a screen (or a printout).

Formally, consider a multicube `MC`, composed of `K` axes. A *2D-slice over `MC`* is a set of (`K-2`) points, each from a separate axis, where the points of the `Invisible` and the `Content` axis are comprised within the points of the 2D-slice.

In our motivating example, Fig. 2 and Fig. 11 represent the same 2D slice.

Tapes

Tapes represent 'one-dimensional' parts of a 2D slice. In fact, out of the two free axis of the 2D slice, we have only one left free and the other pinpointed to a particular point, say `p`. In this case, a tape is parallel to this particular axis. Tapes are not considered 'lines', due to hierarchically decomposable points: if the pinpointed point `p` is hierarchically decomposable, the tape will be visualized as a set of parallel lines.

Formally, consider a 2D-slice `SL` over a multicube `MC`, composed of `K` axes. A *tape over `SL`* is 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. A tape is more restrictively defined with respect to the 2D-slice by a single point: we will call this point the *key of the tape with respect to its 2D-slice*. Moreover if a 2D-slice has two axes a_1, a_2 with `size(`a_1`)` and `size(`a_2`)` points each, then one can define `size(`a_1`)*size(`a_2`)` tapes over this 2D-slice.

Observe Fig. 2. All `C1`, `C2`, `C3`, `C4`, `C5`, `C6` and `R1`, `R2`, `R3`, `R4` are tapes. Observe `C1` or `R1`: due to the fact that they are pinpointed to hierarchically decomposable points, they involve more than one 'lines'. The different colors correspond to different vertical tapes (`C1`-`C6`).

Cross-joins

Intuitively, a cross join is a set of cells produced by the intersection of two tapes. If the two tapes are defined over atomic points, the cross-join involves a single cell (e.g., in the case of tapes R2 and C3); otherwise a set of cells is produced, as in the case of tapes R2 and C1. Note that a 'cell' corresponds to a point of the measure axis.

Formally, consider a 2D-slice SL over a multicube MC, composed of K axes. Consider also 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 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.

Two tapes are *joinable* if they can produce a cross-join.

The only difference between a tape and a cross-join is that the cross-join restricts all of its dimensions with equality constraints, whereas the tape constraints only a subset of them.

Bridging the Presentation and the Logical Layers of CPM

Cross-joins form the bridge between the logical and the presentational layers. In this section we provide a theorem proving that a cross-join is a secondary cube. Then, we show how common OLAP operations can be performed on the basis of our model.

Theorem 1. Assume a star schema database $[DS^0, \mathbf{C}, \mathbf{C}^S]$, over a star schema $[\mathbf{D}, S^0]$. Assume also a cross-join, say c, defined over a subset of the dimensions **D**. Then, c can be mapped to a secondary cube over the star schema database.

Proof. We will constructively obtain the definition of the secondary cube. Remember that the cross-join is practically defined by a set of K points over the axes of a multicube.

1. The detailed data set is naturally DS^0.
2. Each of the dimensions of the cross-join is a subset of **D**, and we can assume that the levels referring to each point are $[L_1, \ldots, L_n, M_1, \ldots, M_m]$ with the first being coordinate axes and the latter being measure axes.
3. A selection condition φ can be derived from the points of the coordinate axes.
4. If there are any functions applied, they are also defined over the attributes of the data set: suppose that we have $A_{m+1}: f_{m+1}(\mathbf{A}_{m+1}), \ldots, A_{m+p}: f_{m+p}(\mathbf{A}_{m+p})$ attributes in the definitions of the attribute groups of the multicube. The measure axis also has a set of aggregate expressions over measures, say $[agg_1(M_0^1), \ldots, agg_m(M_0^m)]$ and a secondary selection condition.
5. The function **f** of the cross-join's multicube is defined as a mapping of $[L_1, \ldots, L_n]$ to $[M_1, \ldots, M_m]$, possibly exploiting the use of the functions f_{m+i}.
6. Consequently, one can produce the following secondary cube out of the cross-join c:

$$c = [DS^0, \varphi, [L_1, \ldots, L_n, M_1, \ldots, M_m], [agg_1(M_0^1), \ldots, agg_m(M_0^m)], [A_{m+1}: f_{m+1}(\mathbf{A}_{m+1}), \ldots, A_{m+p}: f_{m+p}(\mathbf{A}_{m+p})], O_{\varnothing}^{<}, \psi]$$

where φ is the conjunction of the primary selection conditions of the levels and ψ is the conjunction of the secondary selection conditions of the rest of the attributes. ■

The only difference between a tape and a cross-join is that the cross-join restricts all of its dimensions with equality constraints, whereas the tape constraints only a subset of them. Moreover, from the definition of the *joinable* tapes it follows that a 2D-slice contains as many cross-joins as the number of pairs of joinable tapes belonging to this particular slice. This

observation also helps us to understand why a tape can also be viewed as a collection of cross-joins (or cubes). Thus we have the following lemma.

Lemma. A tape is a finite set of secondary cubes.
Proof. Each of this cross-joins is defined from the $k-1$ points of the tape and one point from all its joinable tapes. This point belongs to the points of the axis the tape is parallel to. Consequently, we are allowed to treat a tape as a set of cross-joins, or cubes: $t=[c_1,\ldots,c_k]$. ■

THE MAPPING TO TABLE LENS

In the previous section, we have showed how a generic presentational model, CPM, can represent multi-query screens that can be mapped to constructs of an underlying logical model. At the same time, there is more that we can do over the presentation model and this involves advanced visualization techniques. It would be straightforward to visualize the CPM constructs simply as tabular data. Nevertheless, we can do better than that, and apply advanced visualization techniques over the CPM constructs. In this section, we will demonstrate how CPM can be combined with Table Lens (TL) (Rao & Card, 1994, Pirollo & Rao, 1996), a non-traditional cross-tabular presentation model from the Human Computer Interaction area. This model, based on the "focus plus context" technique, is used in applications and platforms for the visualization of tabular data and appears to be quite appropriate for OLAP purposes. 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 a 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.

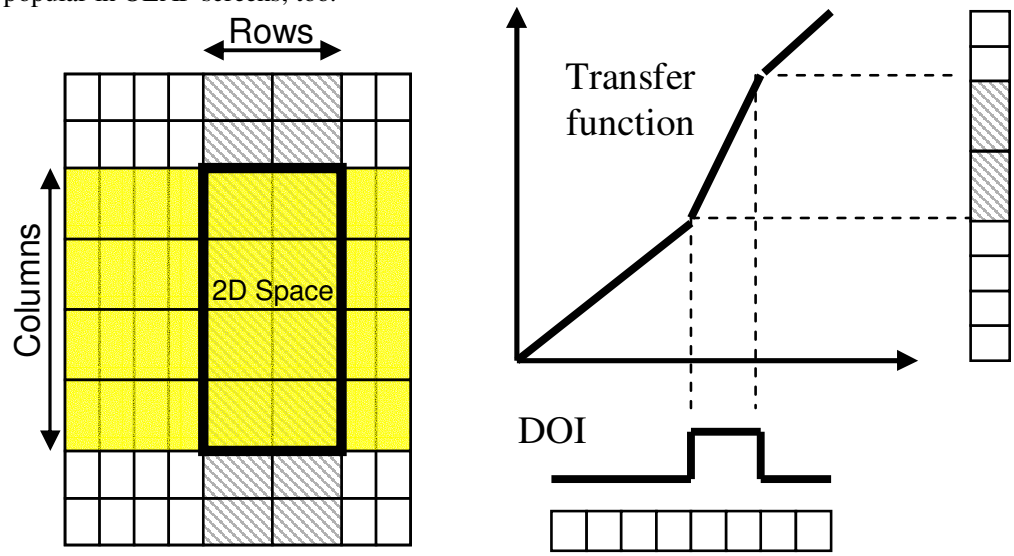

Fig. 13: A Table Lens example: (a) a 2x4 focus window is defined over a space of 8x8 points; (b) Table Lens distortion of the Columns axis

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, as shown in Fig. 13.
- **2D Space**: The 2D Space is constructed from the Cartesian product of the two Table Lens axes. It is a (finite) matrix of cells. One of the basic ideas behind the Table Lens technique is that not all cells are considered equal, in terms of presentation. 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 resembles of zooming into the particular region of the 2D Space.
- **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, thus a 2D Space is characterized by 2D windows of focus. 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. In Figure 13a, we depict an 8x8 space with a 2x4 focus window. In Figure 13b, we show 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.
- **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. In Figure 13b, we show how the Transfer function, defined as a weighted integral of the DOI function, maps the points to pixel areas. For reasons of efficient representation (Rao & Card, 1994), in Figure 13b, the produced axis is rotated by 90^{o}. 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 Space, specifying both its size and position. Still, we can automate the process on the basis of the structure and the contents of a 2D Space.

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 Space*. 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 controlling parameters (like stopping conditions, error range etc) can be tuned by the user. Actually, we can even treat as a parameter the 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.

Motivation and Assumptions

Before providing the generic algorithm, let us clarify our contribution through a specific example. We instantiate the example of Fig. 2 with the values in Fig. 14. Let us assume that when the user activates this OLAP screen, he would like to be informed on three particular cross-joins: a) one involving the maximum sales (`max`), b) another involving the lowest (`min`) and c) a third involving the cross-join with behavior closest to the average (`closest-to-avg`) of the whole 2D Space. 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/C3` 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 simple 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.

			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	Qtr2		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

Fig. 14: Instantiation of the motivating example with values; different coloring determines different cross-joins and thick borders highlight the cross-joins with the highest, lowest and closest to average values.

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.
- *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 (Lenz & Thalheim, 2001) 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 logical OLAP model presented in (Vassiliadis & Skiadopoulos, 2000), 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 (Lenz & Thalheim, 2001) are not considered in 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.

A Generic Algorithm for Determining the Window of Focus

Naturally, we can do better than the aforementioned algorithm by adding extra criteria to the proactive selection of the starting window of focus. We propose a guided greedy generic algorithm, *GenericFocusWindow* (Fig. 15), 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 2D-slice.

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
ε: 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.

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}

Fig. 15. (a)Algorithm *GenericFocusWindow* (b) Instantiation of the algorithm

We present an example for the instantiation of the aforementioned generic algorithm in Fig. 15b, where 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). To accomplish this, we initialize accordingly the parameters of the algorithm (GuardCondition, ε) and define accordingly the functions of the algorithms (OriginalPick, DeterminingQuality and Pick). The greedy algorithm is guided to pick 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) 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 Fig. 14, 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.

In (Maniatis et al. 2003a) we present more examples for the instantiation of the algorithm.

IMPLEMENTATION

In this section we will present a framework to support for *Mobile OLAP*, a term used to express the porting of requirements and specifications for OLAP applications into the wireless and mobile computing world as introduced in (Maniatis, 2004). In this context, we present *CubeView*, a pilot academic platform enabling OLAP visualization both for contemporary desktops as well as for mobile devices. We present details about the adopted system and software architecture of the system, along with explanations on the usage of the system.

The Architecture of CubeView

In this section we present the architecture of *CubeView*, organized as (a) system architecture, (b) software architecture and (c) implementations specifics.

The system architecture for CubeView is depicted in Fig. 16. The general idea is that the user on the mobile device (PDA, mobile phone or even remote desktop PC) uses a specific user interface on his device to navigate on his screen between OLAP data and perform OLAP analysis in general, based on data stored locally on his device in highly aggregated and summarized format.

The system is composed of three discrete and autonomous modules: A traditional *OLAP Server Module*, used as a black box in the process since it can be any of the existing commercial, open source or academic ones, a *Middleware Application Server*, serving as the mediator between the data stored in the OLAP Server and the mobile device and the *mobile Front-End Applications*, incorporating the local storage and the user interface, navigation and presentation options.

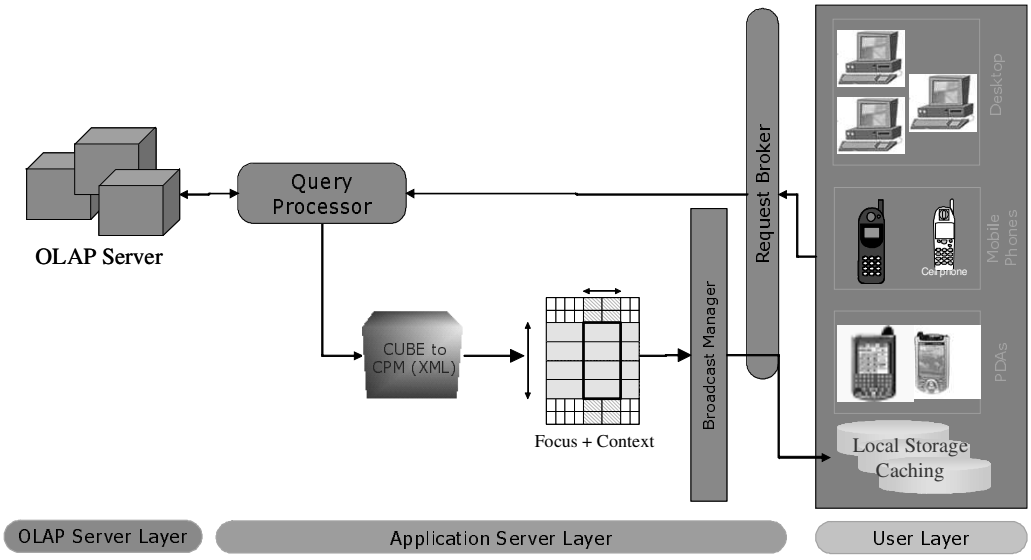

Fig. 16: The System Architecture of CubeView

The *Software Architecture of CubeView* is depicted in Fig. 17. Each distinct system layer holds a number of software modules, each – in turn – performing specific tasks in the context of the whole system. To be more specific, the software layers of CubeView are:

- The *OLAP Server Layer*, as mentioned before, is "transparent" to the user and is used as a "*black box*" from the system, meaning that only a specific API is used to query the server for OLAP data. Typically, any available OLAP server (MOLAP or ROLAP), commercial or academic platform could be used for storing and processing the actual OLAP queries.
- The *Application Server Layer* is the software layer holding the Java server-side application logic. It incorporates the following software modules: (*i*) The *Query Manager*, responsible for directly querying the OLAP Server according to the query posed by the mobile user; (*ii*) the *XML OLAP Data Manager*, a component responsible for formatting the requested data in XML format and which interacts with (*iii*) the *Cube to CMP (XML) Converter* and (*iv*) the *XML Cross-join Metadata*, that formats the OLAP data queried in a format suitable for presentation using CPM entities. A simple (*v*) *Caching Mechanism* is employed for performance purposes, which holds both the actual OLAP data and the necessary OLAP metadata, retrieved and managed through (*vi*) the *OLAP Metadata Manager*. Finally, (*vii*) the *Connection Manager* is the synchronization mediator between the Application Server and the User Layer, controlling the connection process when data from the OLAP Server are needed to refresh the OLAP data in CPM entities format, stored locally on the mobile device.
- The *User Layer* incorporates a number of software modules and client tools that supplies the mobile user with instruments for: (*i*) Storing OLAP data locally, on the *Local Storage RDBMS*, which holds the metadata of the system as well as highly aggregated OLAP data; (*ii*) Executing simple ad-hoc local queries – meaning on the OLAP data stored locally – though (*iii*) the *OLAP Query Manager*; (*iv*) the *OLAP Request Manager* interacts with the Application Server Layer – actually with the Connection Manager – to retrieve data from the OLAP Server if such a case is enforced due to the query posed by the user; and finally, (*v*) a suitable *User Interface* and *Local Metadata Browser* are the front end tools on the mobile device through which the user can pose queries, browse the local data and display the results of his queries on OLAP screens, employing the Table Lens visualization technique.

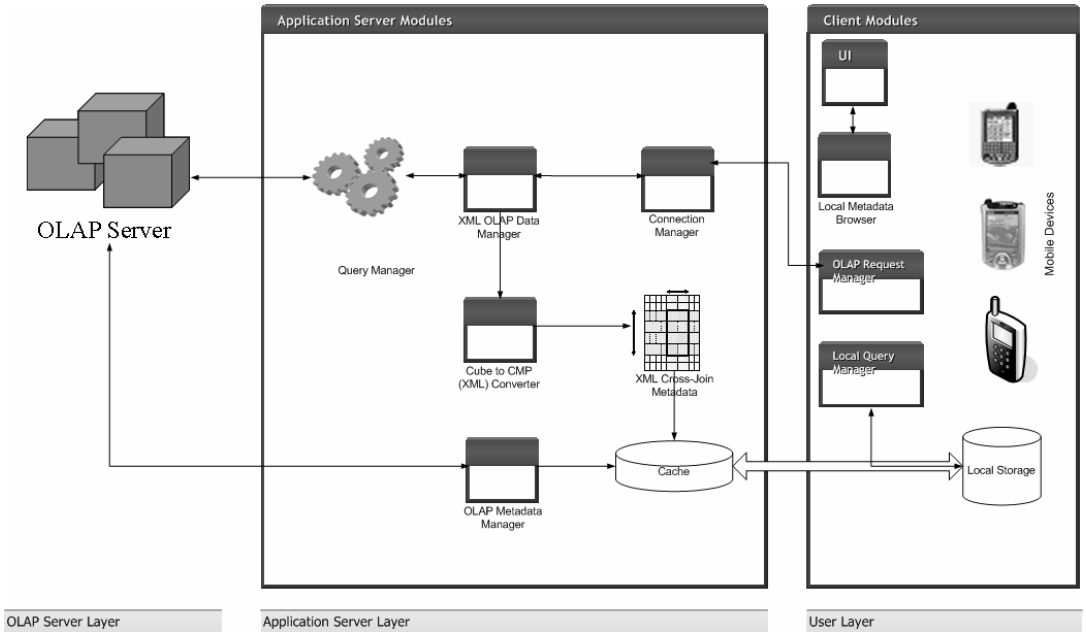

Fig. 17: The Software Architecture of *CubeView*

Using CubeView

In this section we give an illustrative example of how CubeView is designed to work and respond to queries posed by mobile users.

Fig. 18: A Prototype Front-End for CubeView on Pocket PC employing OLAP Lens.

An implementation on a Pocket PC environment of the example described in Fig. 2 is displayed in Fig. 18, where a prototype for the front end tool used is developed and displayed on the simulator provided by Microsoft (http://www.microsoft.com/windowsmobile/information/devprograms/default.mspx.). In terms of usability of the system and user interface characteristics, special features are employed. A step by step usability of the user interface controls is as follows:

- The user exploits the user interface facilities of the front end application to specify his range query, using drag and drop to select from the pop up dimensions window the actual dimension levels to form the initial screen. The first level of the highly aggregated data appears on the screen (Fig. 18).
- In a next specialization manipulation, the user selects from a pop up menu a function or formula (system or user defined) to focus on specific characteristics and attributes of the displayed data. For example, it may be that the user wants to locate the cell with the `max` value of the displayed data and drill into them to locate further details. The appearance of the display screen focuses to the desired cell, which is distorted – in accordance to the Table Lens characteristics – to guide the user to the exact results of his query (Fig. 19).

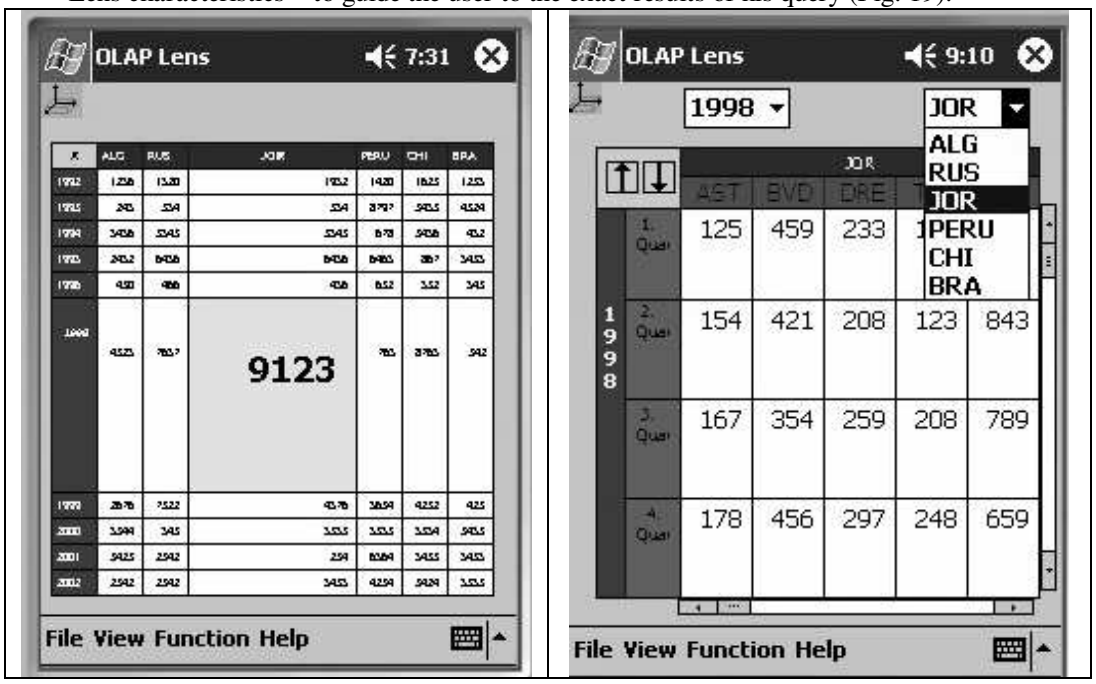

Fig. 19: OLAP Lens User Interface details.

- By double clicking on the focused cell, the user drills in to the details of the cell. The cross tab displayed is replaced by the next level of the dimension hierarchies, both for rows and columns. The previous level of the dimension hierarchies are displayed as pull down menus on the top of the screen.
- They user may continue performing the previous steps to perform further analysis. Special controls (such as ↓↑ for drill in and drill out) and menu options assist the user in performing OLAP analysis on his mobile device.

Finally, we would also like to point out that the architecture implemented in CubeView is flexible enough to provide a framework suitable for both desktop and mobile OLAP visualization applications.

RELATED WORK

In this section we will present related work on the topics covered by our research. This includes existing presentation models for databases and multidimensional data, implementations of visualization tools and a discussion of related efforts in the field of OLAP for mobile environments.

Furthermore, we should mention that preliminary results of the work analysed herein can be found in (Maniatis et al., 2003, Maniatis et al., 2003a, Maniatis, 2003, Maniatis, 2004). Still, in this paper, (a) we provide the big picture for these works, (b) we have further decoupled the logical and presentational models from the work of (Maniatis et al., 2003) and (c) we provide more examples and details on the proofs.

Presentational Models

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. The first effort is from the industrial field, where Microsoft has issued a commercial standard for multidimensional databases and where the presentational issues form a major part (Microsoft, 1998). 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). However, this standard suffers from several problems, with two of them being the most prominent: First, the logical and presentational models are mixed, resulting in a complex language which we personally found hard to use (although powerful enough). Secondly, the model is formalized but not thoroughly: for instance, we did not really see a definition for the schema of a multicube. Also, there are specific axes that are predefined, namely "rows", "columns", "pages", "sections" and "chapters"; no other axes are supposed.

The second proposal is an academic approach, the Tape Model (Gebhardt et al., 1997), based on the notion of “Tapes”, called thus 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 model offers the possibility of defining *hierarchical structures* within a tape. 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*.

Compared to CPM, the aforementioned models can be characterized as follows: CPM is a formal approach, with a rigorous formal background. The tape model seems to be limited in its expressive power (with respect to the Microsoft proposal) and, to our knowledge, its formal aspects are not yet publicly available. The Microsoft proposal, on the other hand, appears to be too complicated, without a clean cut separation of the underlying concepts. Also, its coupling to the underlying logical structures is not clear.

Visualization in Contemporary OLAP Tools

Most vendors offering data warehousing and OLAP tools and platforms have included in their products special modules running on mobile devices and offering OLAP analysis possibilities to

mobile decision makers. Vendors such as MicroStrategy Inc. (http://www.microstrategy.com) and Business Objects (http://www.businessobjects.com) have gone a great deal towards implementing dedicated broadcast servers that provide OLAP specific information to users in numerous typical formats, such as emails, beeps on pagers or specifically designed web pages, using WAP and WML and employing a specific but typical server based architecture to offer this functionality.

With respect to academic pilot visualization tools and platforms, numerous have been developed, mainly in the area of general area of information visualization. Many proposals focused in more specific areas, such as statistical & scientific databases, data mining and multidimensional data visualization. In the last area, we can mention VisDB (Keim & Kriegel, 1994), HD-Eye (Hinneburg et al., 2002) and Polaris (Stolte & Hanrahan, 2000).

VisDB (Keim & Kriegel, 1994) and its more recent sibling *HD – Eye* (Hinneburg et al., 2002), originated from the area of general database exploration techniques, with specialization in multidimensional visualization. It cannot be considered as an OLAP visualization platform instead it is a platform for the exploration of large multidimensional data sets using techniques such as the mapping of two dimensions to axes, parallel coordinates etc, all integrated into an interactive graphical environment. In a sense, ViSDB can be viewed as being closer to data mining than OLAP. *HD-Eye* is a more recent version oriented towards visual clustering of large data sets containing high-dimensional data.

Polaris (Stolte & Hanrahan, 2000), finally, is one of the most recently designed and implemented visual interfaces, designed to explore large multi-dimensional databases that extends the well-known Pivot Table interface. The features of Polaris include an interface for constructing visual specifications of table-based graphical displays, and the ability to generate a precise set of relational queries from the visual specification. The visual specifications can be incrementally developed, giving the analyst visual feedback as they construct complex queries and visualizations.

Applications for Mobile OLAP

OLAP specific functionality for mobile devices provided by the vendors does very little towards exploiting the specific characteristics and power of the mobile devices on which these applications run. Rather, they base their solutions in migrating the desktop OLAP interface of their tools to the mobile device, employing WAP and the WML but fail to take into account recent, improved facts about mobile devices, such as increased system memory, clearer color screens, increased processing power or their limitations, such as small screen size, different usability and user interface requirements, common off line work etc.

To fill this gap, numerous approaches coming mainly from the academia and from various research areas have been proposing solutions and frameworks that address this problem. Many of them propose novel approaches to cope with the case of mobile OLAP and – what is more important – many of them have been actually implemented and used in real case scenarios. We will briefly present some of the most notable – to our judgment – approaches.

MOCHA (Rodriguez-Martinez & Rossopoulos, 2000), was an early, more generic approach for a database middleware for distributed data sources which, although not specifically addressing the case of mobile devices and OLAP, it incorporates many of the notions present in "*Mobile OLAP*", such as the distributed nature of the system, the scaling to a larger environment along with the novel approach of deploying application specific functionality from one point of the system to all the others through the middleware itself. The system was implemented in Java – which allowed for the shipping of Java code to implement either advanced data types or tailored remote operators to remote data sources and have it executed remotely – and was actually put to work effectively on a large aerospace organization.

A more specific approach is presented in (Cuzzocrea et al., 2003), namely Hand-OLAP, a system specifically designed for bringing OLAP functionality to users of mobile devices. This proposal focuses mainly on a number of the drawbacks of handhelds devices, with emphasis in the small storage space and the usual discontinuance of the connection to the Wireless LAN, as opposed to the user needs for querying and browsing information extracted from enormous amounts of data, accessible through the network. To cope with this issue, this approach focuses in presenting a solution for storing locally, in the mobile device, a compressed and highly summarized view of the data that can be more efficiently transmitted from the OLAP server than the original ones. Hand-OLAP is a prototype system, with a suitable architecture that seamlessly supports the interaction between mobile device and OLAP server, stores data locally on the mobile device in a compressed format – based on Quadtree representation – and always providing the user with a specific bi-dimensional (tabular) view of the data, even when the connection to the WLAN is off.

Finally, the work of (Sharaf & Chrysanthis, 2002, Sharaf & Chrysanthis, 2002a) focused more on matters of wireless network and power consumption on the mobile devices, proposing a suitable mobile OLAP model, along with an on-demand scheduling algorithm to minimize access time and energy consumption on mobile agents. This approach in based on summary tables too, along with the functionality of simple OLAP front-end tools to execute simple SQL queries. What is more, this proposal maximizes the aggregated data sharing between clients and reduces the broadcast length. Finally, the proposed on-demand scheduling algorithm employs user parameters to fine tune the degree of aggregation of data so as to control the tradeoff between access time and energy consumption and adapts to different request rates, access patterns and data distributions.

As a whole, comparing CubeView with all the previously described tools, both the ones used for visualization and those reviewed for Mobile OLAP, we stress the fact that our prototype is the only one that supports the full cycle, starting from a formal and rigorous theory background, depicted in CPM itself, and reaching a full fledged implementation covering both worlds, the traditional desktop environments and the mobile devices. All the other paradigms are departmental in the sense that the they tamper only portions of the big picture, this being either the information visualization area [VisDB (Keim & Kriegel, 1994), HD-Eye (Hinneburg et al., 2002), Polaris (Stolte & Hanrahan, 2000)], or specific approaches and implementations for mobile devices [Hand-OLAP (Cuzzocrea et al., 2003)] or simply middleware, like MOCHA (Rodriguez-Martinez & Rossopoulos, 2000) or – finally – a framework for a wireless OLAP model (Sharaf & Chrysanthis, 2002, Sharaf & Chrysanthis, 2002a).

CONCLUSIONS AND FUTURE WORK

So far, visualization has not been fully incorporated in the abstraction levels of DBMS architecture (conceptual, logical, or physical). In this paper we have discussed the separation of the aforementioned abstraction levels to take visualization into consideration. In this context, we have presented the Cube Presentation Model (CPM), a formal presentation model for OLAP data. Our contributions can be listed as follows: (a) we have presented an extension of a previous logical model for cubes, to handle more complex cases; (b) we have introduced a novel presentational model for OLAP screens, intuitively based on the geometrical representation of a cube and its human perception in the space; and (c) we have discussed how these two models can be smoothly integrated. Moreover, we have demonstrated how CPM can be naturally mapped into an advanced visualization technique (Table Lens) and we discussed suitable algorithms for proactive automated support of the user towards the highlighting of interesting areas of a report.

Finally, we have discussed implementation and usage issues in the context of an academic prototype system (CubeView) that we have implemented.

Obviously, we do not claim that this is the ultimate solution to the problem of OLAP data visualization, 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.

An obvious particularity of our approach is that is mostly crafted for tabular data in the context of OLAP. Should we wish to differentiate the context of data utilization or the data themselves (e.g., perform OLAP over spatial or biological data), the presentation and visualization techniques would have been different. In general, it is an interesting research challenge to discuss the integration of different models in the presence of different contexts, either in terms of the data or their usage.

At the same time, new hardware developments pose new requirements for our visualization techniques. One of our goals has been 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 have chosen 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. Naturally, as part of future research, different implementation issues (e.g., caching schemes or visualization techniques) can be applied in this context.

Finally, coming back to the visualization issue, we have brought up Table Lens to highlight the possibility of facilitating 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 (Miller, 1956), 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, our contribution is related to a broader line of research (Han, 1998, Sarawagi et al., 1998), which is obviously open to a wide range of different possibilities.

REFERENCES

Cuzzocrea, A., & Furfaro, F., & Sacca, D. (2003). *Hand-OLAP: a System for Delivering OLAP Services on Handheld Devices*. In Proceedings of ISADS 2003, pp. 213 – 224, Pisa, Italy, April 9 – 11.

Gebhardt, M. , & Jarke, M., & Jacobs, S. (1997). *A Toolkit for Negotiation Support Interfaces to Multi-Dimensional Data*. In Proceedings of ACM SIGMOD 1997, pp. 348 – 356.

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

Hinneburg, A., & Keim, D.A., & Wawryniuk, M. (2002). *HD-Eye: Visual Clustering of High-Dimensional Data*, In Proceedings of the 2002 ACM SIGMOD 2002, pp. 629 – 629, Madison, Wisconsin.

Inmon, W.H. (1996). *Building the Data Warehouse*. John Wiley & Sons.

Inselberg, A. (2001). *Visualization and Knowledge Discovery for High Dimensional Data*. 2nd Workshop Proceedings of UIDIS, IEEE press.

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

Keim, D.A., & Kriegel, H.–P. (1994): *VisDB: Database Exploration Using Multidimensional Visualization,* IEEE Computer Graphics and Applications, September 1994.

Kimbal, R., & Reeves, L., & Ross, M., & Thornthwaite, W. (1998). *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*. John Wiley & Sons.

Lenz, H.-J., & Thalheim, B. (2001). *OLAP Databases and Aggregation Functions*. In Proceedings of the SSDBM 2001, pp. 91 – 100, Fairfax, USA.

Abiteboul, S., et al. (2003). *The Lowell Database Research Self Assessment*. Lowell, Massachusetts USA, May 4-6, Available at: http://research.microsoft.com/~Gray/lowell/.

Maniatis, A. (2003). *OLAP Presentation Modeling with UML and XML.* In Proceedings of BCI 2003, pp. 232 - 241, Thessaloniki, Greece.

Maniatis, A. (2004). *The Case for Mobile OLAP*. Proceedings of the 1st International Workshop on Pervasive Information Management (in conjunction with EDBT '04), pp. 103 – 114, Heraklion, Greece.

Rodriguez-Martinez, M., & Rossopoulos, N. (2000). *MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources*. In Proceedings of ACM SIGMOD 2000, pp 213 – 224, Dallas, USA.

Microsoft (1998). *OLEDB for OLAP*, Available at: http://www.microsoft.com/data/oledb/olap/.

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

Maniatis, A., & Vassiliadis, P., & Skiadopoulos, S., & Vassiliou, Y. (2003). *CPM: A Cube Presentation Model for OLAP*. In Proceedings of DaWaK 2003, pp. 4 – 13, Prague, Czech Republic.

Maniatis, A., & Vassiliadis, P., & Skiadopoulos, S., & Vassiliou, Y. (2003a). *Advanced Visualization for OLAP*. In Proceedings of DOLAP 2003, pp. 9 – 16, New Orleans, USA.

Pirollo, P., & Rao, R. (1996). *Table Lens as a Tool for Making Sense of Data*. In Proceedings of the AVI 1996 Workshop, pp. 67 – 80, Gubbio, Italy.

Rao, R., & Card, S.K. (1994). *The Table Lens: Merging Graphical and Symbolic Representations in an effective Focus + Context Visualization for Tabular Information.* In Proceedings of ACM SIGCHI 1994, Boston, USA.

Sarawagi, S., & Agrawal, R., & Megiddo, N. (1998). *Discovery-Driven Exploration of OLAP data Cubes*. In Proceedings of EDBT 1998, pp. 168 – 182, Valencia, Spain.

Sharaf, M.A., & Chrysanthis, P.K. (2002). *On-Demand Broadcast: New Challenges and Algorithms*. In Proceedings of HDMS 2002, Athens, Greece.

Sharaf, M.A., & Chrysanthis, P.K. (2002a). *Semantic-based delivery of OLAP summary tables in wireless environments*. In Proceedings of CIKM 2002, pp. 84 – 92, McLean, USA.

Stolte, C., & Hanrahan, P. (2000). *Polaris: A System for Query, Analysis and Visualization of Multidimensional Relational Databases*. In Proceedings of InfoVis 2000, pp. 5 – 14, Salt Lake City, USA.

Tsois, A., & Karayannidis, N., & Sellis, T. (2001). *MAC: Conceptual Data Modeling for OLAP*. In Proceedings of DMDW 2001, pp. 5.1 – 5.11, Interlaken, Switzerland.

Vassiliadis, P., & Sellis, T. (1999). A Survey on Logical Models for OLAP Databases. *SIGMOD Record,* 28 (4), 64 – 69.

Vassiliadis, P., & Skiadopoulos, S. (2000). *Modeling and Optimization Issues for Multidimensional Databases*. In Proceedings of CaiSE 2000, pp. 482 - 497, Stockholm, Sweden.