Modeling Multidimensional Databases, Cubes and Cube Operations

Panos Vassiliadis
National Technical University of Athens

Abstract

On-Line Analytical Processing (OLAP) is a trend in database technology, which was recently introduced and has attracted the interest of a lot of research work. OLAP is based on the multidimensional view of data, supported either by multidimensional databases (MOLAP) or relational engines (ROLAP).

In this paper we propose a model for multidimensional databases. Dimensions, dimension hierarchies and cubes are formally introduced. We also introduce cube operations (changing of levels in the dimension hierarchy, function application, navigation etc.). The approach is based on the notion of the base cube, which is used for the calculation of the results of cube operations. We focus our approach on the support of series of operations on cubes (i.e. the preservation of the results of previous operations and the applicability of aggregate functions in a series of operations).

Furthermore, we provide a mapping of the multidimensional model to the relational model and to multidimensional arrays.

1. Introduction

In recent database trends, *data warehouses* come to fill a gap in the field of querying large, distributed and frequently updated systems. Most researchers and developers share the same general vision of what a data warehouse is [19], [3]. Data are extracted from several data sources, cleansed, customized and inserted into the data warehouse. The logical structure and semantics of the data, or else *Enterprise Model*, is stored in an *Information Directory*. Next, the data warehouse data can be filtered, aggregated and stored in smaller specialized data stores, usually called *data marts*. Users query the data marts and/or the data warehouse, mostly through *On Line Analytical Processing* (OLAP) applications. The main characteristics of such applications are (a) multidimensional view of data, and (b) data analysis, through interactive and/or navigational querying of data [6].

The multidimensional view of data considers that information is stored in a *multi-dimensional array* (sometimes called a *Hypercube,* or *Cube*). A *Cube* is a group of data cells arranged by the dimensions of the data [13]. A *dimension* is defined in [13] as "a structural attribute of a cube that is a list of members, all of which are of a similar type in the user's perception of the data". Each dimension has an associated *hierarchy of levels* of aggregated data i.e. it can be viewed from different levels of detail (for example, *Time* can be detailed as *Year, Month, Week,* or *Day*). *Measures* (which are also known as *variables*, *metrics*, or *facts*) represent the real measured values [6].

To motivate the work describing this paper, let us use a running example of a bookstore company. When considering the sales of this company, three are the major dimensions: *Time*, *Geography* and *Item*, while we consider *Sales* as the measure of the multidimensional cube. The dimensions, along with their dimension levels are depicted in Figure 1, where the upper levels of each hierarchy point to the lower levels:

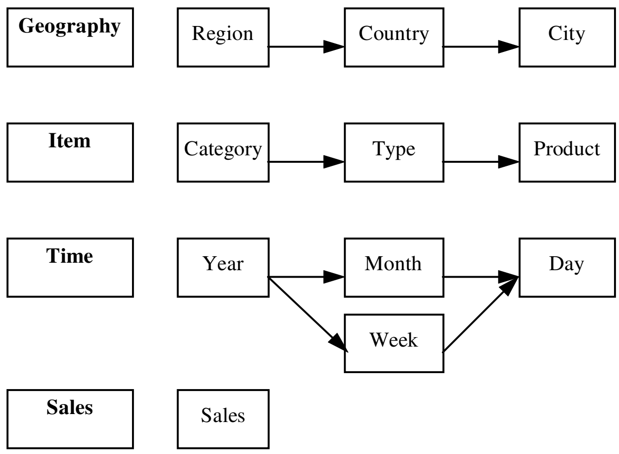

Figure 1. Dimensions and dimension levels

Consider, now, the way dimension level hierarchies are instantiated in the real world (we consider the instantiation for dimension *Time*, to be obvious):

Category	Type	Product
Books	Literature	"Report to El Greco" N. Kazantzakis
		"Karamazof brothers" F. Dostoiewsky
	Philosophy	"Zarathustra", F. W. Nietzsche
		"Symposium", Plato
Music	Heavy Metal	"Piece of Mind", Iron Maiden
		"Ace of Spades", Motorhead

Figure 2. Item dimension

Region	Country	City
Europe	Hellas	Athens
		Rhodes
	France	Paris
Asia	Israel	Tel Aviv
	Japan	Tokyo

Figure 3. Geography dimension

Navigation is a term used to describe the processes employed by users to explore a cube interactively, by manipulating the multidimensionally viewed data [6],

[13]. Possible operations which can be applied are: *Aggregation* (or *Consolidation*, or *Roll-up*) which corresponds to summarization of data for the higher level of a hierarchy, *Roll Down* (or *Drill down*, or *Drill through*) which allows for navigation among levels of data ranging from higher level summary (up) to lower level summary or detailed data (down), *Selection* (or *Screening*, or *Filtering* or *Dicing*) whereby a criterion is evaluated against the data or members of a dimension in order to restrict the set of retrieved data, *Slicing* which allows for the selection of all data satisfying a condition along a particular dimension and *Pivoting* (or *Rotation*) throughout which one can change of the dimensional orientation of the cube, e.g. swapping the rows and columns, or moving one of the row dimensions into the column dimension, etc. [6], [13].

Two are the basic architectures for storing data in an OLAP database: ROLAP and MOLAP. *ROLAP (Relational OLAP)* [3] is based on a relational database server, extended with capabilities such as extended aggregation and partitioning of data [8]. The schema of the database can be a *star, snowflake,* or *fact constellation* schema [3]. On the other hand, *MOLAP (Multidimensional OLAP)* is based on "pure" Multidimensional Databases (MDDs), which logically store data in multidimensional arrays, which are heavily compressed and indexed, in the physical level, for space and performance reasons.

The main motivation of this paper is to provide a formal model for multidimensional databases. Since multidimensional databases are defined in terms of *dimensions* (which are organized in dimension *hierarchies*), the model represents them formally. Furthermore, classical OLAP operations, such as *roll-up, slice, dice* etc. are also represented by the model. We also provide a mapping to relational databases and multidimensional arrays. We make a serious design choice: since querying is done in an interactive way, we give emphasis to the tracking of series of operations, performed in a navigational way.

The major contribution of the paper is the modeling of cubes, dimensions and cube operations, in the context of series of operations. This formalization is currently used, in this paper, for a direct modeling of the usual OLAP operations. Instead of mapping OLAP operations to complex and complicated "relational", or "calculus-like" queries, we directly model them, in a straightforward fashion. To our knowledge, the modeling of the drill-down operation is introduced for the first time in our model. Since engines are based on relational technology, or multidimensional arrays, we also provide a direct mapping of cubes and their operations for each of these formalisms, so that both data warehouse designers and the engines themselves can take advantage of it.

The rest of this paper is organized as follows: in section 2 we present related work in the fields of models and algebras for data warehouse and OLAP applications. In section 3 we provide a model for multidimensional databases and cubes. In section 4 we provide a relational mapping of the aforementioned model and a mapping to multidimensional arrays. In section 5, we present the conclusions of our work and possible future extensions.

2. Related work

Research has followed the evolution of industrial products in the field of OLAP. The *data_cube* operator was introduced in [8]. There have also been efforts to model multidimensional databases. In [1], a model for multidimensional databases is introduced. The model is characterized from its symmetric treatment of dimensions and measures. A set of minimal (but rather complicated) operators is also introduced dealing with the construction and destruction of cubes, join and restriction of cubes and merging of cubes through direct dimensions. Furthermore, an SQL mapping is presented.

In [12] a multidimensional data model is introduced based on relational elements. Dimensions are modeled as "dimension relations", practically annotating attributes with dimension names. The cubes are modeled as functions from the Cartesian product of the dimensions to the measure and are mapped to "grouping relations" through an applicability definition. A grouping algebra is presented, extending existing relational operators and introducing new ones, such as ordering and grouping to prepare cubes for aggregations. Furthermore, a multidimensional algebra is presented, dealing with the construction and modification of cubes as well as with aggregations and joins.

In [9] *n-dimensional* tables are defined and a relational mapping is provided through the notion of *completion*. An algebra (and an equivalent calculus) is defined with classical relational operators as well as restructuring, classification and summarization operators. The expressive power of the algebra is demonstrated through the expression of operators like the data cube operator and monotone roll-up.

In [2] multidimensional databases are considered to be composed from sets of tables forming denormalized star schemata. Attribute hierarchies are modeled through the introduction of functional dependencies in the attributes of the dimension tables. Nevertheless, this work is focused on the selection of an optimal set of materialized views, for the efficient querying and update of a data warehouse, and not in the modeling of cubes or cube operations.

In [4], a multidimensional database is modeled through the notions of dimensions and *f-tables*. Dimensions are constructed from hierarchies of dimension levels, whereas f-tables are repositories for the factual data. Data are characterized from a set of *roll-up* functions, mapping the instances of a dimension level to instances of another dimension level. A query language is the focus of this work: a calculus for f-tables along with scalar and aggregate functions is presented, basically oriented to the formulation of aggregate queries. In [5] the focus is on the modeling of multidimensional databases: the basic model remains practically the same, whereas ER modeling techniques are given for the conceptual modeling of the multidimensional database.

In statistical databases [17], quite a lot of similar work has been done in the past. In [17] a comparison of work done in statistical and multidimensional databases is presented. The comparison is made with respect to application areas, conceptual modeling, data structure representation, operations, physical organization aspects and privacy issues. The basic conclusion of this comparison is that the two areas have a lot of overlap, with statistical databases emphasizing on conceptual modeling and OLAP emphasizing on physical organization and efficient access.

In [14] a data model for statistical databases is introduced. The model is based on "summary tables" and operators defined on them such as construction/destruction, concatenation/extraction, attribute splitting/merging and aggregation operators. The underlying algebra is a subset of the algebra described in [15]. Furthermore, physical organization and implementation issues are discussed. [14] is very close to practical OLAP operations, although discussed in the context of summary tables.

In [16] a functional model ("Mefisto") is presented. Mefisto is based on the definition of a data structure, called "statistical entity" and on operations defined on it like summarization, classification, restriction and enlargement.

In all of the aforementioned approaches the relationship of the proposed operators to real OLAP operations, such as roll-up, drill-down, slice and dice seems to be weak: it is either discussed informally for a subset of operators [1], indirectly dealt through the introduction of aggregation [12], [9], or in a different context [14], [16]. [2] and [5] are basically dealing with the modeling of cubes. The best approach seems to be given in [5]; yet a direct mapping to OLAP operations is still not provided. Furthermore, apart for [16], series of operations are not directly dealt with. Finally, to our knowledge, no explicit modeling of the drill-down operation exists.

3. A model of multidimensional space and cubes

3.1. Multidimensional space

Let $\boldsymbol{\Omega}$ be the space of all *dimensions*. For each dimension D_i there exist a set of *levels*, denoted as $levels(D_i)$. A *dimension* is a lattice $(\boldsymbol{H}, \leq)$ of levels. Each path in the lattice of a dimension hierarchy, beginning from its least upper bound and ending in its greatest lower bound is called a *dimension path*. Each dimension path is a linear, totally ordered list of levels. We extend the notion of the function *levels*, for dimension paths: $levels(D_{pi})$ is a list, where the higher a level semantically is, the higher its rank is in the dimension path. The total order allows us to use comparison operators for the dimension levels. For instance, if we consider the dimension path [year, month, day], then $day \leq month \leq year$, whereas for the dimension path [year, week, day], $day \leq week \leq year$ holds. A dimension D consists of a set of dimension paths, $paths(D)$. In the case of linear dimensions, where there is a single dimension path in the dimension, we will use the terms *dimension* and *dimension path* interchangeably.

Let $\boldsymbol{\Psi}$ be the space of all dimension levels. We can find the dimension where a dimension level belongs to, through the operator h: $h(DL_i) = D$ if $DL_i \in levels(D)$. We impose the restriction that a dimension level belongs to exactly one dimension. Furthermore, we can find the rank of a dimension level in a dimension path, through the function $level(DL_i)$. $level(DL_i) = k$, when $DL_i = levels(D_{pi})[k]$ (in other words, DL_i is the k-th level of the dimension path D_{pi}, starting the enumeration from the lowest levels).

For each dimension level there is a set of *values* belonging to it (e.g. dimension level "city" has "Athens", "Paris", "Rome"... as values). We define $dom(DL_i)$ as the set of all the values of a dimension level DL_i. Let $\boldsymbol{V}$ be the space of all values. A dimension level is *atomic* if its domain is a subset of $\boldsymbol{V}$. If the domain of a dimension level is a subset of $\boldsymbol{P(V)}$ (the power set of $\boldsymbol{V}$) then the dimension level is *multi-valued*. We use bag semantics for multi-valued dimension levels. As in [15], we use the prefix "*" for multi-valued attributes.

A value x, can have *ancestors* and *descendants*. Let x belong to a specific dimension level L_0; then, there are specific instances related to x, at higher (lower) dimension levels, corresponding to more general (detailed) terms, that is

$ancestor(x, DL) = y$, $y \in dom(DL)$, $DL_0 \leq DL$ and

$descendants(x, DL) = \{x_1, x_2, ..., x_k\}$, $x_1, x_2, ..., x_k \in dom(DL)$, $DL \leq DL_0$.

For example, if we consider the dimension path [year, month, day] then *ancestor*(FEB 1997, year) = 1997 and *descendants*(FEB 1997, day) = {1 FEB 1997, 2 FEB 1997, ..., 28 FEB 1997}. We will assume the following properties for the *ancestor* relationship:

1. $ancestor(x, DL) = x$, if $x \in dom(DL)$
2. if $x = ancestor(y, DL)$ and $y = ancestor(x, DL)$, then $x = y$
3. if $x = ancestor(y, DL_1)$ and $y = ancestor(z, DL_2)$, then $x = ancestor(z, DL_1)$

The third property guarantees that when more than possible paths exist from z to x, in the dimension level lattice, then all these paths are consistent.

3.2. Cubes

In this section we shall introduce the notion of *cubes*, *basic cubes* and *multidimensional databases*. The *cubes* are the basic entities of the model, whereas *basic cubes* are cubes with the most detailed data. A *multidimensional database* is a set of dimensions, dimension levels and a basic cube.

We define a *basic_cube* C_b as a 3-tuple $<\boldsymbol{D_b}, \boldsymbol{L_b}, \boldsymbol{R_b}>$, where

- $\boldsymbol{D_b}$ = <D_1, D_2, ... D_n, M> is a list of dimensions (D_i, M $\in$ $\boldsymbol{\Omega}$). M is a dimension that represents the measure of the cube.
- $\boldsymbol{L_b}$ = <DL_{b1}, DL_{b2}, ... DL_{bn}, *ML> is a list of dimension levels (DL_{bi}, *ML $\in$ $\boldsymbol{\Psi}$). ML is the dimension level of the measure of the cube. We demand that all the dimension levels are at the lowest level of their respective dimensions ($\forall$ $DL_b \in \boldsymbol{L_b}$, $level(l)$ = 1). We also demand that ML is multi-valued.
- $\boldsymbol{R_b}$ is a set of *cell data* -i.e. a set of tuples of the form $x = [x_1, x_2, ..., x_n, *m]$, where $\forall$ *i in [1, ..n]*, $x_i \in dom(DL_{bi})$ and $*m \in dom(*ML)$.

We define a *Cube* C as a 4-tuple <$\boldsymbol{D}$, $\boldsymbol{L}$, C_b, $\boldsymbol{R}$>, where

- $\boldsymbol{D}$ = <D_1, D_2, ... D_n, M> is a list of dimensions (D_i, M $\in$ $\boldsymbol{\Omega}$). M is a dimension that represents the measure of the cube. We will denote M as *measure_dimension(C)*.
- $\boldsymbol{L}$ = <DL_1, DL_2, ... DL_n, *ML> is a list of dimension levels (DL_i, *ML $\in$ $\boldsymbol{\Psi}$). *ML is the dimension level of the measure of the cube. We will denote *ML as *measure_dimension_level(C)*. We demand that $\forall$ $DL_i \in \boldsymbol{L}$, $DL_i \in levels(D_i)$. As it will be shown from the cube operations, we also demand that *ML is multi-valued.
- C_b is a *basic_cube*. We will call C_b, the *base_cube* of C (C_b = *base_cube(C)*). The data of C_b can be used for the calculation of the contents of C. Furthermore, we impose the restriction, that $\forall$ d $\in$ C.$\boldsymbol{D}$ $\exists$ d' $\in$ C_b.$\boldsymbol{D}$: d = d'. In other words, all the dimensions of a cube must exist in its *base_cube*.
- $\boldsymbol{R}$ is a set of cell data -i.e. a set of tuples of the form as a tuple $x = [x_1, x_2, ..., x_n, *m]$, where $\forall$ *i in [1, ..n]*, $x_i \in dom(DL_i)$ and $*m \in dom(*ML)$.

We can consider basic cubes as cubes. We extend the definition of a basic cube C_b to be a 4-tuple <$\boldsymbol{D_b}$, $\boldsymbol{L_b}$, C_b, $\boldsymbol{R_b}$> -i.e. we define a basic cube to be the *base_cube* of itself.

We define a *Multidimensional Database* as a couple <$\boldsymbol{D}$, *C*>. $\boldsymbol{D}$ is a set of dimensions and *C* is a basic cube, the dimensions of which belong to $\boldsymbol{D}$.

Cell data are the data of a cube. Each cell is defined by a set of values and a measure, which is also a value. Thus, a cell *x* is a tuple $x = [x_1, x_2, ..., x_n, *m]$. We introduce the following shortcut notations:

$dimensions(x) = <x_1, x_2, ..., x_n>$,

$measure(x) = *m$,

$dimensions(x)(i) = x_i$, where $C = <\boldsymbol{D}, \boldsymbol{L}, C_b, \boldsymbol{R}> \wedge (x \in R)$,

$dimensions(x)(d) = x_i$, where $C = <\boldsymbol{D}, \boldsymbol{L}, C_b, \boldsymbol{R}> \wedge d \in D \wedge d = D(i) \wedge (x \in R)$.

In our running example, let us consider that a *basic_cube* for the bookstore company is instantiated as shown in Figure 4.

Intuitively, it might strike the reader as strange the fact that we define a cube in terms of another cube and that we practically provide two data sets (R and $C_b.R_b$) for the instantiation of a single cube. Nevertheless, there are two major reasons for which we choose to follow this specific approach:

Time	Item	Geography	Sales
1997-01-01	"Report to El Greco"	Rhodes	15
1997-01-01	"Ace of Spades"	Paris	8
1997-01-01	"Report to El Greco"	Athens	11
1997-02-06	"Symposium"	Rhodes	7
1997-02-18	"Karamazof brothers"	Paris	5
1997-02-18	"Report to El Greco"	Athens	2
1997-03-03	"Karamazof brothers"	Rhodes	4
1997-03-03	"Karamazof brothers"	Athens	10
1997-03-28	"Symposium"	Rhodes	5
1996-10-12	"Report to El Greco"	Paris	7
1996-05-06	"Piece of Mind"	Tokyo	10
1996-09-07	"Piece of Mind"	Rhodes	7
1996-03-28	"Karamazof brothers"	Tel Aviv	12
1996-01-01	"Karamazof brothers"	Tel Aviv	40

Figure 4. Basic_Cube = <*D0, L0*, Basic_Cube, *R0*>, *D0* = <Time, Item, Geography, Sales>, *L0* = <Day, Product, Region, Sales>, *R0* is shown in the above table

First, the definition of the data of a cube in terms of its *base_cube* enables the direct and correct evaluation of its contents. A specific example will help us clarify this statement. Suppose, that we summarize the sales of Figure 4 at the month level. Suppose then, that we would like to see the average sales at the year level. This result cannot be directly calculated from the result of the previous cube. The existing algebras that we know of [1], [12], [LR97] would not take this problem into account, or would assume that the operation will be disallowed by the system [16]. Since this kind of sequences of operations is typical for OLAP applications, the correctness of the result of the operations of the cube can be guaranteed, by referring to the relevant data of the most basic granularity.

Secondly, all the aforementioned algebras cannot deal directly with *drill-down* operations (i.e. with navigation to lower levels of dimension hierarchies). This is obvious, since a sum cannot be analyzed to its components unless a join operation with a cube of the required granularity takes place. As it can easily be anticipated, the definition of a cube in terms of a basic cube enables the drilling-down without possibly costly join operations with other cubes. As it will be shown in the sequel, in the case of the relational mapping of our model (which can be used for ROLAP), joins actually take place; yet they are made between a fact table and the tables representing the dimensions of the cube. Techniques like *star-join* [7] can be employed to optimize this kind of operations.

3.3. Cube operations

The definition of a cube is accompanied with the definition of cube operations. We categorize cube operations into simple ones, such as *level_climbing, packing, function_application, projection*, *dicing* and complex ones, such as *navigation* and *slicing*, which are defined on top of the simple ones. We do not deal with *pivoting* since we consider it to be just a reorganization of the presentation of the data, rather than a modification of their value or structure. Each one of the operations results in a new cube, when applied to an existing cube. *Slicing* and *navigation* apply aggregate functions to the data of the cube. The set of allowed aggregate functions is {*sum, avg, count, min, rank(n), no-operation*}. All of them are the well known relational aggregate functions, except for *no-operation* which means that no function is applied on the data of the cube and *rank(n)* which returns the first n-components of an aggregated set of values which can be ordered. In the sequel we will suppose that the original cube C = <***D***, ***L***, C_b, ***R***>, ***D*** = <D_1, D_2, ..., D_n, M>, ***L*** = <DL_1, DL_2, ..., DL_n, *ML>, C_b = <$\boldsymbol{D_b}$, $\boldsymbol{L_b}$, C_b, $\boldsymbol{R_b}$> and that the new cube C', which is the result of the operations is C' = <***D***', ***L***', C_b', ***R***'>.

Level_Climbing. Let $\underline{d}$ be a set of dimensions belonging to C and $\underline{dl}$ the set of the corresponding dimension levels of C. Without loss of generality we assume that $\underline{d}$ consists of the last k dimensions of D. Let also $\underline{dl}_{old}$ be the original dimension levels of C, belonging to $\underline{d}$: $\underline{dl}_{old}$ = {DL_{n-k+1}, …, DL_n}. Then, *C' = Level_Climbing(C,* $\underline{d}$*,* $\underline{dl}$*) = LC(C,* $\underline{d}$*,* $\underline{dl}$*)* is defined as follows:

D' = ***D***, ***L***' = ***L*** - $\underline{dl}_{old}$ ∪ $\underline{dl}$, C_b' = C_b and

R' ={x | ∃ y ∈ ***R***: *dimensions*(x)(D_i) = *dimensions*(y)(D_i) ∀ D_i ∉ $\underline{d}$ ∧ *dimensions*(x)(D_i) = *ancestor*(*dimensions*(y)(D_i), dl_j), ∀ D_i ∈ $\underline{d}$, dl_j ∈ $\underline{dl}$, dl_j ∈ *levels*(D_j) ∧ *measure*(x) = *measure*(y), if M ∉ $\underline{d}$ }

We impose the restrictions that $\underline{d}$, $\underline{dl}$ are consistent with each other and that for all the dimension levels of $\underline{dl}$, the respective dimension levels of $\underline{dl}_{old}$ belong to the same dimension path and are of lower or equal level (for example, one cannot perform *Level_Climbing* between months and weeks). Intuitively, *Level_Climbing* is the replacement of all values of a set of dimensions with values of dimension levels of higher level. In Figure 5, an example of the *Level_Climbing* operation is presented:

Time	Item	Geography	Sales
1997	"Report to El Greco"	Europe	15
1997	"Ace of Spades"	Europe	8
1997	"Report to El Greco"	Europe	11
1997	"Symposium"	Europe	7
1997	"Karamazof brothers"	Europe	5
1997	"Report to El Greco"	Europe	2
1997	"Karamazof brothers"	Europe	4
1997	"Karamazof brothers"	Europe	10
1997	"Symposium"	Europe	5
1996	"Report to El Greco"	Europe	7
1996	"Piece of Mind"	Asia	10
1996	"Piece of Mind"	Europe	7
1996	"Karamazof brothers"	Asia	12
1996	"Karamazof brothers"	Asia	40

Figure 5. C1 = LC(Basic_Cube, {Geography, Time}, {Region, Year}), C1 = <*D1, L1*, C_b1, *R1*>, *D1* = <Time, Item, Geography, Sales>, *L1* = <Year, Product, Region, Sales>, C_b1 = Basic_Cube, *R1* is shown in the above table

Packing. We define *C' = Packing(C) = P(C)* as follows:

D' = ***D***, ***L***' = ***L***, C_b' = C_b and

R' ={x | ∃ y ∈ ***R***: *dimensions*(x)(D_i) = *dimensions*(y)(D_i) ∀ i ∈ 1, …, n ∧ *measure*(x) = {*l* | ∃ t ∈ ***R***, *dimensions*(y) = *dimensions*(t) ∧ *l* = *measure*(t)}}

Intuitively, *packing* is the consolidation of the cube, through the merging of multiple instances having the same dimension values into one. *Packing* has bag semantics. In Figure 6, an example of the *Packing* operation is presented:

Time	Item	Geography	Sales
1997	"Report to El Greco"	Europe	15, 11, 2
1997	"Ace of Spades"	Europe	8
1997	"Symposium"	Europe	7, 5
1997	"Karamazof brothers"	Europe	5, 4, 10
1996	"Report to El Greco"	Europe	7
1996	"Piece of Mind"	Asia	10
1996	"Piece of Mind"	Europe	7
1996	"Karamazof brothers"	Asia	12, 40

Figure 6. C2 = P(C1), C2 = < *D2, L2*, C_b2, *R2*>, *D2* = <Time, Item, Geography, Sales>, *L2* = <Year, Product, Region, Sales>, C_b2 = Basic_Cube, *R2* is shown in the above table

Function_Application. Let *f* be a function belonging to {*sum, avg, count, min, rank(n), no-operation*}. Then, *C' = Function_Application(C, f) = F(C, f)* is defined as follows:

D' = ***D***, ***L***' = ***L***, C_b' = C_b and

R' ={x | ∃ y ∈ ***R***: *dimensions*(x) = *dimensions*(y) ∧ *measure*(x) = f(*measure*(y)) }

Intuitively, *Function_application* is the application of a specific function to the measure of a cube.

Projection. Let d be a projected dimension. *C' = Projection(C, d) = π(C, d)* is then defined, as follows:

D' = ***D*** - d, ***L***' = ***L*** - DL, DL ∈ *levels*(d), DL ∈ ***L***,

C_b' = <$\boldsymbol{D_b}$', $\boldsymbol{L_b}$', C_b', $\boldsymbol{R_b}$'>, where,

$\boldsymbol{D_b}$' = $\boldsymbol{D_b}$ - d,

$\boldsymbol{L_b}$' = $\boldsymbol{L_b}$ - *levels*(d)(1), and

$\boldsymbol{R_b}$' = {x | ∀ y ∈ $\boldsymbol{R_b}$, *dimensions*(x)(D_i) = *dimensions*(y)(D_i), ∀ D_i ≠d, i ∈ 1, …, n ∧ *measure*(x) = *measure*(y)}

R' ={x | ∃ y ∈ ***R***: *dimensions*(x)(D_i) = *dimensions*(y)(D_i), ∀ D_i ≠d, i ∈ 1, …, n ∧ *measure*(x) = *measure*(y) }

Intuitively, *projection* is the deletion of a dimension both from the cube and its *base_cube*.

Navigation. Let *d* be the dimension over which we navigate, *dl* the target level of the navigation and *f* the applied aggregate function. Suppose that the dimension d is the *i-th* element of D. Then, we define *C' = Navigation(C, d, dl, f)* as follows:

C' = Navigation(C, d, dl, f) = F(P(LC(C_b, {D_1, D_2, ..., d, ..., D_n}, {DL_1, DL_2, ..., dl, ..., DL_n})),f)

The purpose of the navigation operator is to take a cube from a specific state, change the level of a specific dimension, pack the result and produce a new cube with a new state, through the use of an aggregate function. The dimensions of the new cube are the dimensions of the old one. The dimension levels are also the same, except for the one of the dimension where we change level. Notice that the restrictions imposed by *Level_Climbing*, regarding the position of the respective dimension levels in the dimension lattice, still hold. Furthermore, the *base_cube* remains the same. The *Navigation* is performed at the level of the *base_cube*, for reasons that will be best illustrated in the following example:

C3 = Navigate(Basic_Cube, Geography, Region, no_operation)
C4 = Navigate(C3, Time, Year, sum)
C5= Navigate(C4, Time, Month, avg)

Time	Item	Geography	Sales
1997-01-01	"Report to El Greco"	Europe	15, 11
1997-01-01	"Ace of Spades"	Europe	8
1997-02-06	"Symposium"	Europe	7
1997-02-18	"Karamazof brothers"	Europe	5
1997-02-18	"Report to El Greco"	Europe	2
1997-03-03	"Karamazof brothers"	Europe	4, 10
1997-03-28	"Symposium"	Europe	5
1996-10-12	"Report to El Greco"	Europe	7
1996-05-06	"Piece of Mind"	Asia	10
1996-09-07	"Piece of Mind"	Europe	7
1996-03-28	"Karamazof brothers"	Asia	12
1996-01-01	"Karamazof brothers"	Asia	40

Figure 7. C3 = Navigation(Basic_Cube, Geography, Region, no_operation), C3 = <*D3*, *L3*, C_b3, *R3*>, *D3* = <Time, Item, Geography, Sales>, *L3* = <Day, Product, Region, Sales>, C_b3 = Basic_Cube, *R3* is shown in the above table

Time	Item	Geography	Sales
1997	"Report to El Greco"	Europe	28
1997	"Ace of Spades"	Europe	8
1997	"Symposium"	Europe	12
1997	"Karamazof brothers"	Europe	19
1996	"Report to El Greco"	Europe	7
1996	"Piece of Mind"	Asia	10
1996	"Piece of Mind"	Europe	7
1996	"Karamazof brothers"	Asia	52

Figure 8. C4 = Navigation(C3, Time, Year, sum), C4 = <*D4*, *L4*, C_b4, *R4*>, *D4* = <Time, Item, Geography, Sales>, *L4* = <Year, Product, Region, Sales>, C_b4 = Basic_Cube, *R4* is shown in the above table

Time	Item	Geography	Sales
1997-01	"Report to El Greco"	Europe	13
1997-01	"Ace of Spades"	Europe	8
1997-02	"Symposium"	Europe	7
1997-02	"Karamazof brothers"	Europe	5
1997-02	"Report to El Greco"	Europe	2
1997-03	"Karamazof brothers"	Europe	7
1997-03	"Symposium"	Europe	5
1996-10	"Report to El Greco"	Europe	7
1996-05	"Piece of Mind"	Asia	10
1996-09	"Piece of Mind"	Europe	7
1996-03	"Karamazof brothers"	Asia	12
1996-01	"Karamazof brothers"	Asia	40

Figure 9. C5= Navigation(C4, Time, Month, avg), C5 = <*D5*, *L5*, C_b5, *R5*>, *D5* = <Time, Item, Geography, Sales>, *L5* = <Month, Product, Region, Sales>, C_b5 = Basic_Cube, *R5* is shown in the above table

This example shows that the basic contribution of the navigation operator is that it can allow any sequence of operations along the dimension hierarchies. The navigation from the *Basic_Cube* to cube *C5*, is characterized by three features:

1. it preserved the previous navigations -e.g. the navigation to the dimension level of Geography (Region),
2. it allowed the application of the average function over a cube whose data was previously produced through the application of a sum function. If the definition of the navigation was done on the result of the actual cube, the correct calculation of the result would not be possible,
3. it allowed the *drilling down* at the Time dimension (i.e. moving directly from "Year" to "Month" level) without having to join cubes directly. The drill-down operation was mapped to *Level_Climbing* upwards in the Time dimension. The consinstency of the values between different levels in the dimension lattice guarantees a correct result.

Dicing. Let d be the dimension over which we perform the dicing, σ a formula consisting of a dimension, an operator and a value v. We assume that v belongs to the values of the dimension level of d in C and that σ is applicable to d (in the sense presented in [15]) - i.e. that {<, =} are applied to atomic dimension levels and {≡, ⊂, ∈} to multi-valued ones). Let $\sigma(v)$ be of the form d *op* v. Then, $C' = Dicing(C, d, \sigma(v))$ is defined as follows:

$\boldsymbol{D}' = \boldsymbol{D}$, $\boldsymbol{L} = \boldsymbol{L}'$,
$C_b' = <\boldsymbol{D_b}', \boldsymbol{L_b}', C_b', \boldsymbol{R_b}'>$, where
$\boldsymbol{D_b}' = C_b.\boldsymbol{D_b}$, $\boldsymbol{L_b}' = C_b.\boldsymbol{L_b}$, and
$\boldsymbol{R_b}' = \{x \mid x \in C_b.\boldsymbol{R_b}, x[d] \text{ op } y = \text{true}, y \in \textit{descendants}(v, \textit{levels}(d)(1))\}$
$\boldsymbol{R}' = \{x \mid \exists\, x \in \boldsymbol{R}, x[d] \text{ op } v = \text{true}\}$

Intuitively, *dicing* is a simple form of selection. Yet, it has its impact both on the cube itself and its *base_cube*. We are allowed to check for descendants of v in the *base_cube*, since each dimension path ends at a dimension level of the lowest granularity and the *base_cube* is in the lowest possible granularity for all levels.

Slicing. Let d be the dimension which we slice and f the applied aggregate function. We define *Slicing* as follows:

$C' = Slicing(C, d, f) = F(P(\pi(LC(C_b, \{D_1, D_2, ..., d, ..., D_n\}, \{DL_1, DL_2, ..., dl, ..., DL_n\}), d)), f)$

The purpose of the slicing operator is to take a cube from a specific state, cut out a specified dimension and aggregate over the rest of the dimensions, using an aggregation function. Notice that all the restrictions of *Level_Climbing* implicitly hold, without realy affecting the *Slicing* operation. In Figures 10, 11, an example of the *Slicing* operation is presented.

In this section we have defined cubes and cube operations for a multidimensional model. Since in practice, the multidimensional view of data is supported from multidimensional (MOLAP) or relational (ROLAP) engines, in the following section we will provide a mapping of the structures and the operations of the multidimensional model, to the relational model and to multidimensional arrays.

Item	Geography	Sales
"Ace of Spades"	Europe	8
"Karamazof brothers"	Asia	26
"Karamazof brothers"	Europe	6.3
"Piece of Mind"	Asia	10
"Piece of Mind"	Europe	7
"Report to El Greco"	Europe	8.75
"Symposium"	Europe	6

Figure 10. C6 = Slicing(C4, Time, avg), C6 = <*D6*, *L6*, C_b6, *R6*>, *D6* = <Item, Geography, Sales>, *L6* = < Product, Region, Sales>, *R6* is shown in the above table

Item	Geography	Sales
"Report to El Greco"	Rhodes	15
"Ace of Spades"	Paris	8
"Report to El Greco"	Athens	11
"Symposium"	Rhodes	7
"Karamazof brothers"	Paris	5
"Report to El Greco"	Athens	2
"Karamazof brothers"	Rhodes	4
"Karamazof brothers"	Athens	10
"Symposium"	Rhodes	5
"Report to El Greco"	Paris	7
"Piece of Mind"	Tokyo	10
"Piece of Mind"	Rhodes	7
"Karamazof brothers"	Tel Aviv	12
"Karamazof brothers"	Tel Aviv	40

Figure 11. C6 = Slicing(C4, Time, avg), C_b6 = <*D_b6*, *L_b6*, C_b6, *R_b6*>, *D_b6* = <Item, Geography, Sales>, *L_b6* = < Product, City, Sales>, *R_b6* is shown in the above table

4. A mapping of the multidimensional model to an extended relational data model

In this section we map multidimensional cubes, defined in Section 3, to relational tables. For this purpose we will base our approach on the extended relational model and algebra proposed in [15]. *Atomic* vs. *set-valued attributes*[1] (with bag semantics) are introduced. Apart from the classical relational operations, operations such as *packing* ($P_X(r)$) (merging tuples with the same values for several attributes into one tuple) and *function_application* ($r[*X, f_i]$) (application of a function f_i to a multi-valued attribute $*X$) are introduced. A more detailed presentation for the employed model can be found in [18].

The motivation for the relational mapping is double: on the one hand, the engine performing ROLAP must be able to map multidimensional to relational entities and on the other hand, the data warehouse administrator can be helped to check out whether a relational database fulfills the requirements to model a cube (and vice versa - what kind of database one needs to construct in order to be able to map a cube to relational tables).

[1] This requirement does not constraint the applicability of the algebra, since existing DBMSs already support NF[2] characteristics. The object extensions of the upcoming SQL3 standard will formalize this kind of support [10].

At the end of the section a mapping of our multidimensional model to multidimensional arrays (used as logical structures in engines performing MOLAP) is also presented.

4.1. Mapping of cubes to relations

To map multidimensional cubes to relations we need as prerequisite, the existence of two mapping functions α and λ. The function α maps a dimension level to an attribute of a relation, whereas λ is its inverse and maps an attribute to a dimension level. We say that a dimension level DL *represents* an attribute A, and vice versa, if α(DL) = A**,** and consequently λ(A) = DL.

A dimension level can be mapped to more than one attributes. The reason for this is that in both *star* and *snowflake* schemata, which are common for data warehousing and ROLAP applications, two columns - possibly related by foreign key constraints- in two different tables, may represent the same entity, due to normalization. Furthermore, we make the assumption that an attribute and a dimension level which can be mapped to one another, have the same structure (simple vs. set-valued) and domain.

Definition 1. A relation r, defined over a relation scheme R(A_1, A_2, …, A_k), *represents* a dimension path D_p (denoted also as r = $\boldsymbol{R_D}(D_p)$) iff

1. $\forall\ DL_i \in$ *levels*(D_p) $\exists\ A_j \in$ R: $\alpha(DL_i) = A_j$
2. $\forall\ A_j \in R\ \exists\ DL_i \in$ *levels*(D_p): $\lambda(A_j) = DL_i$
3. If DL_s is the lowest level of D_p, $\forall\ \delta \in$ *dom*(DL_s), $\forall\ A_i \in$ R, $\exists$ exactly one t, t $\in$ r: t[A_i] = *ancestor*(δ, $\lambda(A_i)$),
4. $\forall$ t $\in$ r, $\forall\ A_i \in$ R, $\exists\ \delta$, $\delta \in$ *dom*(DL_s): t[A_i] = *ancestor*(δ, $\lambda(A_i)$),

Intuitively, for a table to represent a dimension path, there must be a one to one mapping between the table columns and the dimension levels of the dimension path (items (1), (2) in definition 1). The instantiation of the table is such, so that for every value of the lowest granularity there is a tuple with all its ancestors (item 3). Furthermore, we require that the table contains no more tuples than those needed to represent the values (item 4). The tables representing dimension paths are denormalized structures, commonly employed in *star schemata* in data warehouses; they are usually encountered with the name *dimension tables*. For example, the dimension *Geography*, which comprises of a single dimension path, can be represented using the table in Figure 12.

Region	Country	City
Europe	Hellas	Athens
Europe	Hellas	Rhodes
Europe	France	Paris
Asia	Israel	Tel Aviv
Asia	Japan	Tokyo

Figure 12. Geography dimension as a table

From the definition of the *ancestor* operator, and its transitivity property it follows easily that if we consider the values of two attributes of the same tuple, they are characterized from an *ancestor* relationship between them.

Definition 2. A relation r, defined over a relation scheme $R = (A_1, A_2, \ldots, A_k)$, is the *base_cube_table* of a cube $C = <\boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R}>$ (denoted also as $r = \boldsymbol{R_B}(C)$) iff

1. $\forall\, DL \in C_{base}.\boldsymbol{L}, \exists\, A_i \in R: DL = \lambda(A_i)$
2. $\forall\, x = <x_1, x_2, \ldots, x_{k-1}, *x_m> \in C_{base}.\boldsymbol{R}_{base}, \exists\, t \in r: x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$
3. $\forall\, t \in r, t = <a_1, a_2, \ldots, a_{k-1}, *a_m>, \exists\, x, x \in C_{base}.\boldsymbol{R}_{base}, : t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

Definition 3. A relation r defined over a relation scheme $R = (A_1, A_2, \ldots, A_k)$ is *the cube_table* of a cube $C = <\boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R}>$ (denoted also as $r = \boldsymbol{R_C}(C)$) iff

1. $\forall\, DL \in C.\boldsymbol{L}, \exists\, A_i \in R: DL = \lambda(A_i)$
2. $\forall\, x = <x_1, x_2, \ldots, x_{k-1}, *x_m> \in C.\boldsymbol{R}, \exists\, t \in r: x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$
3. $\forall\, t \in r, t = <a_1, a_2, \ldots, a_{k-1}, *a_m>, \exists\, x, x \in C.\boldsymbol{R}, : t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

Intuitively, we define a table to be a *cube_table* of a cube if the dimension levels of the cube can be mapped to attributes of the table. The measure -which is also a dimension- is included in this definition (item 1 in definition 3). The contents of a table should be such, that all cells in the result of the cube have an equivalent tuple in the table (item 2 in definition 3). Furthermore, no tuples should exist in the table, where no equivalent cell exists in the result of the cube (item 3 in definition 3). A *base_cube_table* differs from a *cube_table* in the fact that its attributes and data can be mapped to the *base_cube* of a specific cube.

Definition 4. A database $\mathbb{d}$ defined over a database scheme $\boldsymbol{S}$ *represents* a cube $C = <\boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R}>$ iff:

1. $\forall\, d_i \in \boldsymbol{D}$ - *measure_dimension*(C), $\forall\, d_{pi} \in paths(d_i)$, $\exists\, r_i \in \mathbb{d} : r_i = \boldsymbol{R_D}(d_{pi})$
2. $\exists\, r_B \in \mathbb{d} : r_B = \boldsymbol{R_B}(C)$
3. $\exists\, r_C \in \mathbb{d} : r_C = \boldsymbol{R_C}(C)$

A set of relations is the *dimension tables* of a cube, if for every cube dimension and for every dimension path of these dimensions (except for its measure) there is a relevant table in this set, *representing* the dimension path (item 1 in definition 4). If the *base_cube_table* of the cube also exists, then all the cube operations can be applied, by using the *base_cube_table* (item 2 in definition 4); remember that several operations in the multidimensional model have been defined with respect to the base cube. Furthermore, if there is a table in the set, being the *cube_table* of the specific cube, then the data of the cube can be directly accessed through the *cube_table* (item 3 in definition 4). In that case we say that the database *represents* the cube. Since we have required that the values of the dimension paths of different paths in the same dimension, are consistent with each other, then the consistency between the values of the dimension tables for the same dimension, comes natural.

The full schema for the bookstore database of our running example would be:

```
TIME_M(YEAR, MONTH, DAY)
TIME_W(YEAR, WEEK, DAY)
GEOGRAPHY(REGION, COUNTRY, CITY)
ITEM(CATEGORY, TYPE, PRODUCT)
DETAILED_SALES(DAY, PRODUCT,
CITY, SALES)
```

Supposing that the instantiantions are performed correctly, the *TIME_M, TIME_W, GEOGRAPHY, ITEM* relations are the *dimension tables*, whereas the *DETAILED_SALES* relation is the *cube_table* for the *Basic_Cube*.

An interesting issue is that although our definition of dimension tables is based on the notion of denormalized *star* schemata our mapping is also applicable to fully normalized *snowflake* schemata, since that the dimension table of a star schema can be considered as a view defined on the respective tables of the snowflake schema. This is formally proved in [18]. The result is dual: one can map snowflake schemata to cubes and vice versa. Furthermore, cube operations can be mapped to relational operations for a snowflake schema.

For the rest of this paper, we assume that we have a cube $C = <\boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R}>$, $\boldsymbol{D} = <d_1, d_2, \ldots, d_n, M>$, $\boldsymbol{L} = <DL_1, DL_2, \ldots, DL_n, *ML>$. We also assume a database $\mathbb{d}$ defined over the database scheme $\boldsymbol{S} = (R_C, R_B, R_{D1}, R_{D2}, \ldots, R_{Dn})$, an instantiation of $\boldsymbol{S}$, $s = (r_C, r_B, r_{D1}, r_{D2}, \ldots, r_{Dn})$, where $r_C = \boldsymbol{R_c}(C)$, where r_C is defined over $R_C = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM})$, $r_B = \boldsymbol{R_B}(C)$, defined over $R_B = (A_{B1}, A_{B2}, \ldots, A_{Bn}, A_{BM})$ and $\forall\, d_i \in \boldsymbol{D}$, $r_{Di} = \boldsymbol{R_D}(d_i)$, defined over $R_{Di} = (A_{i1}, A_{i2}, \ldots, A_{ik})$.

4.2. Relational mapping of cube operations

In this subsection we will provide the relational mappings for the cube operations which were introduced in Section 3. For each operation we will provide a relational expression for both the *cube_table* and the *base_cube_table* of the resulting cube. In other words, we examine the impact a cube operation has on the cell data of both the *base_cube* and the cube itself and present tables that represent them. All formulas are fully proved in [18].

In Table 1, one can see the relation definitions for the *base_cube_table* for the results of the cube operations, where the *base_cube_table* changes. *Level_Climbing, Packing, Function_Application* and *Navigation* do not change the *base_cube* of a cube. Consequently, one would normally expect that the *base_cube_table* will not change either.

The relational mapping of the result of *Projection* and *Slicing* with respect to the *base_cube* of a cube, is the performance of a *projection* operation on the relevant attribute of its *base_cube_table*.

The mapping of *Dicing* is somewhat more complex than the mappings of other operations. With respect to the *base_cube,* what must be done is the mapping of the parameter value *v* to its descendants, which are found at the *base_cube_table*. Consequently, we *join* the *base_cube* with the proper *dimension table*, representing a dimension path which includes the respective dimension level of the diced cube, perform the *selection* at the result and then *project* the attributes of the *base_cube_table*.

As far as the *cube_tables* are concerned, we also provide a set of formulas, one for each operation. The *cube_tables* represent the actual result of an operation, expressed in a relation instance. For *Level_Climbing*, first we project the dimension tables to the columns corresponding to the dimension levels of the new cube and the columns of the old cube. The relational mapping of the result of *Level_Climbing* is the join of its *cube_table* with all the *dimension tables* involved in the changing of levels and the performance of a projection, in order to keep just the attributes representing the correct dimension levels.

C' = *Projection*(C, d)	$\boldsymbol{R_B}(C') = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$, defined over $R_B' = (A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM})$, where $A_{Bk} = \alpha(DL_k)$, $DL_k \in C_{base}.\boldsymbol{L}_{base}$, $DL_k \in$ *levels*(d), d is the k-th dimension of C.
C' = *Dicing*(C, d, σ(v))	$\boldsymbol{R_B}(C') = ((r_B \bowtie_{AD1=AD1} r_D')[\sigma(v), A_D])[A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM}]$ defined over $R_B' = (A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM})$, and $A_D = \alpha(DL_k)$, $DL_k \in C.\boldsymbol{L}$, $DL_k \in$ *levels(d)*, r_D *represents* d_p, $d_p \in$ *paths(d)*, $DL_k \in d_p$, $r_D' = (r_D)[A_{D1}, A_D]$ and $A_{D1} = \alpha($*levels(d)*$(1))$
C' = *Slicing(C, d, f)*	$\boldsymbol{R_B}(C') = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$, defined over $R_B' = (A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM})$, and $A_{Bk} = \alpha(DL_k)$, $DL_k \in C_b.\boldsymbol{L}_b$, $DL_k \in$ *levels(d).*

Table 1. *Base_cube_table* for the results of cube operations

The relational mapping of the result of *Packing* in a cube, is the performance of a *packing* operation on its *cube_table*, on the attribute representing the measure of the cube. The relational mapping of the result of *Function_Application*, is the performance of a *function_application* operation on the attribute of its *cube_table* representing the measure of the cube. A *projection* on the *cube_table* can model the results of the *Projection* of a cube with respect to its cell data.

Since *Navigation* and *Slicing* have been defined as complex operations, based on other atomic operations, the application of the relational mappings of the cube operations which participate at their definition, produces the formula for the calculation of the *cube_table* of the product of these operations. Notice that the restrictions imposed by *Level_Climbing* still hold. The mapping of *Dicing* is just the performance of a *selection* on its *cube_table*. All formulas are presented in table 2.

4.3 A mapping of the multidimensional model to multidimensional arrays

The multidimensional model can trivially be mapped to multidimensional arrays, practically in the same way it is done in [5]. We assume that there exists a mapping function *enum(d)* between a value *d* of a dimension level *l* and the set of integers. In other words, for each dimension level, we assign a unique integer to each one of its values. The assignment is done in a contiguous fashion. As a result, each value $x = [x_1, x_2, ..., x_n, *m]$, belonging to the cell data of a cube can be considered to be as the conjunction of coordinates $[enum(x_1), enum(x_2), ..., enum(x_n)]$ with value $*m$.

The cube can still be considered to be a 4-tuple C = $<\boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R}>$. We do not need to change the cube operations either: the only thing that changes is that we now have an additional way to refer to the cell data of the cube.

C' = *Level_Climbing*(C, $\underline{d}$, $\underline{dl}$)	$\boldsymbol{R_c}(C') = r_C' = (r_C \bowtie_{ACn-k+1 = ACn-k+1} r_{DCn-k+1}' \bowtie_{ACn-k+2 = ACn-k+2} r_{DCn-k+2}' \bowtie ... \bowtie_{ACn = ACn} r_{Dn}')[A_{C1}, A_{C2}, \ldots A_{Cn-k}, A'_{Cn-k+1}, A'_{Cn}, A_{CM}]$ defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn-k}, A_{Cn-k+1}', \ldots, A_{Cn}', A_{CM})$, where $\underline{d}$ consists of the k last dimensions of $\boldsymbol{D}$, $r_{Di}' = (r_{Di})[\alpha(DLi), A_{Ci}]$, $\forall$ i, $k \leq i \leq n$, defined over $R_{di}' = (A_{Ci}', A_{ci})$.
C' = *Pack*(C)	$\boldsymbol{R_c}(C') = r_C' = P_{ACM}(r_c)$, defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM}')$, where $A_{CM}' = \alpha($*measure_dimension_level*$(C'))$
C' = *Function_Application*(C, f)	$\boldsymbol{R_c}(C') = r_C' = r_c[*ACM, f]$ defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM}')$, where $A_{CM}' = \alpha($*measure_dimension_level*$(C'))$.
C' = *Projection*(C, d)	$\boldsymbol{R_c}(C') = r_C' = r_C[A_{C1}, A_{C2}, ..., A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{CM}]$ defined over $R_C' = (A_{C1}, A_{C2}, ..., A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{CM})$, where $A_{Ck} = \alpha(DL_k)$, $DL_k \in \boldsymbol{L}$, $DL_k \in$ *levels*(d).
C' = *Navigation(C, d, dl, f)*	$\boldsymbol{R_c}(C') = r_C' = (P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie ... \bowtie_{ABn = ABn} r_{Dn}')[A_{C1}, A_{C2}, \ldots, A_{cn}, A_{BM}]))[*A_{BM}, f]$, where $r_{Di}' = (r_{Di})[A_{Bi}, A_{Ci}] \forall$ i, $1 \leq i \leq n$, $A_{BM}' = \alpha($*measure_dimension_level*(C').
C' = *Dicing*(C, d, σ(v))	$\boldsymbol{R_c}(C') = r_C' = r_c[\sigma(v), A_D]$ defined over $R_C' = (A_{C1}, A_{C2}, ..., A_{Cn}, A_{CM})$, and $A_D = \alpha(DL_k)$, $DL_k \in C.\boldsymbol{L}$, $DL_k \in$ *levels(d).*
C' = *Slicing(C, d, f)*	$\boldsymbol{R_c}(C') = r_C' = (P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie ... \bowtie_{ABn = ABn} r_{Dn}') [A_{C1}, A_{C2}, \ldots, A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{BM}]))[*A_{BM}, f]$, where $r_{Di}' = (r_{Di})[A_{Bi}, A_{Ci}] \forall$ i, $1 \leq i \leq n$, d is in k-th position of the cube, and $A_{BM}' = \alpha($*measure_dimension_level*(C').

Table 2. *Cube_table* for the results of cube operations

In the following section, we will conclude our results and present topics for future work.

5. Conclusions and future work

In this paper we have proposed a model for multidimensional databases. *Dimensions*, *dimension hierarchies* and *cubes* are formally introduced in our model. We have also introduced simple cube operations, such as *level_climbing, packing, function_application, projection*, *dicing* and complex ones, such as *navigation* and *slicing*. Our approach is based on the notion of the *base_cube*, which can be used in the complex operations for the calculation of the results of the cube operations. A major motivation for our approach was the support of series of operations on the cubes (for example, the preservation of the results of previous operations and the applicability of aggregate functions in a series of operations). Efficiency is also targeted, so that information refinement operations (such as drill-down) are directly performed.

Furthermore, we have provided mappings of the multidimensional model (a) to the relational model, where cubes and dimensions are mapped to relations and cube operations to relational algebra operations and (b) to multidimensional arrays, through a mapping function.

Apart from the applicability to both MOLAP and ROLAP engines, a basic contribution of our approach for ROLAP engines is that although a cube is defined in terms of another cube, in its relational mapping, only the relational expressions are necessary. For example, if an OLAP tool is to perform a navigation operation, it is not obligatory that the result is always temporarily stored; the definition of a view over the *base_cube_table* is sufficient.

Yet, there are still issues which have not been dealt with. The relaxation of several constraints imposed throughout the definitions of the paper is a possible topic of future research (for example, the relaxation of the constraint that the dimension levels of the *base_cube* must be of level 1). The applicability of existing results of research on view usability [11] can also be investigated in the framework we have set (especially since a relational mapping is provided), in order to optimize the execution of the operations. For example, if Navigation is to be performed in a *roll-up* fashion, one could possibly use the cell data of the cube itself, rather than calculating the new result from the basic cube. Finally, it is not at all certain, that the set of cube operations that we provide is exhaustive, so extensions and new operators are a topic of future research.

Acknowledgment

The author wishes to thank Prof. Timos Sellis for many helpful and detailed comments which enabled the improvement of this paper. This research was partially supported by the European Commission funded LTR ESPRIT project "DWQ: Foundations on Data Warehouse Quality", Project No. 22469 and by the General Secretariat of Research and Technology (Greece) under the PENED program.

6. References

[1] R. Agrawal, A. Gupta, S. Sarawagi, *"Modeling Multidimensional Databases"*, IBM Research Report, IBM Almaden Research Center, September 1995.

[2] E. Baralis, S. Paraboschi, E. Teniente, *"Materialized View Selection in a Multidimensional Database"*, Proceedings of the 23rd VLDB Conference, 1997.

[3] S. Chaudhuri, U.Dayal, *"Data warehousing and OLAP for Decision Support"*, Tutorials of 22nd VLDB Conference, 1996.

[4] L. Cabbibo, R. Torlone, *"Querying Multidimesional Databases"*, 6th International Workshop on Database Programming Languages (DBPL6), 1997.

[5] L. Cabbibo, R. Torlone, *"A Logical Approach to Multidimensional Databases"*, EDBT 1998.

[6] DWQ, *"Deliverable D1.1, Data Warehouse Quality Requirements and Framework"*, NTUA, RWTH, INRIA, DFKI, UNIROMA, IRST, DWQ TR DWQ - NTUA - 1001, 1997, available at http://www.dbnet.ece.ntua.gr/~dwq/

[7] C.G. Erickson, *"Multidimensionalism and the data warehouse"*, in the Data Warehousing Conference (Orlando FL, February 1995).

[8] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. *"Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tabs, and Sub-Totals"*, Proceedings of ICDE '96, New Orleans, February 1996.

[9] M. Gyssens, L.V.S. Lakshmanan, *"A Foundation for Multi-Dimensional Databases"*, Proceedings of the 23rd VLDB Conference, 1997.

[10] K. Kulkarni, N. Mattos, A. Nori, *"Object-Relational Database Systems - Principles, Products and Challenges"*, Tutorials of the 23rd International VLDB Conference, 1997.

[11] A. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava, *"Answering Queries using Views"*, In. PODS, 1995.

[12] C. Li, X. Sean Wang, *"A Data Model for Supporting On-Line Analytical Processing"*, CIKM 1996.

[13] OLAP Council, *"The OLAP glossary"*. http://www.olapcouncil.org, The *OLAP Council*, 1997.

[14] G. Ozsoyoglu, M. Ozsoyoglu, F. Mata, *"A Language and a Physical Organization Technique for Summary Tables"*, Proceedings of the ACM SIGMOD Conference, 1985.

[15] G. Ozsoyoglu, M. Ozsoyoglu, V. Matos, *"Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregation Functions"*, ACM TODS 12(4), 1987.

[16] M. Rafanelli, F.L. Ricci, *"A functional model for macro-databases"*, SIGMOD Record, 20(1), March 1991.
[17] A. Shoshani, *"OLAP and Statistical Databases: Similarities and Differences"*, Tutorials of PODS 1997.
[18] P. Vassiliadis, *"Formal Foundations for Multidimensional Databases"* (extended version) - NTUA Technical Report, January 1998.
[19] J. Widom, "*Research Problems in Data Warehousing"*, Proc. of 4th CIKM Conference, 1995