Formal Foundations for Multidimensional Databases[1]

Panos Vassiliadis
Computer Science Division
Department of Electrical and Computer Engineering
National Technical University of Athens
Zografou 15773, Athens, Greece
pvassil@dbnet.ece.ntua.gr
Tel: +301-772-1602, Fax: +301-772-1442

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 conceptual 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 in several occasions 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 conceptual multidimensional model to models on the logical level, such as the relational model, or multidimensional arrays. In the relational mapping, cubes and dimensions are mapped to relations and the operations of the cube are modeled using relational algebra operations. In the mapping to multidimensional arrays, stored data are enriched with an enumerating function, transforming the cube into a multidimensional array.

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 [Wido95], [CD96]. 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*.

[1] 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.

Users query the data marts and/or the data warehouse, mostly through *On Line Analytical Processing* (OLAP) applications. The high level architecture of a data warehouse can be depicted as in Figure 1.

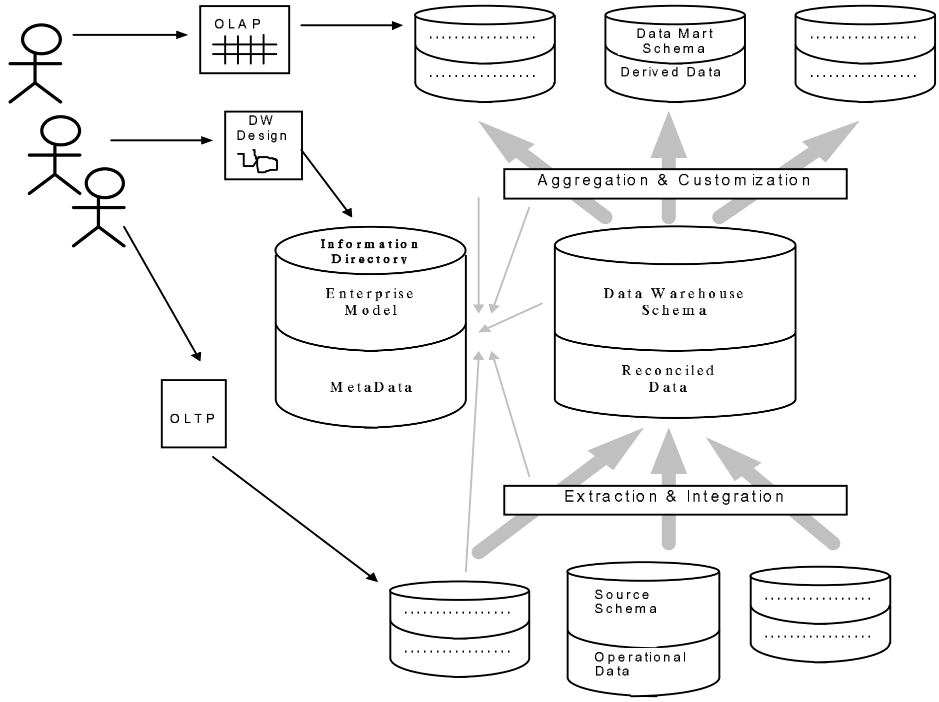

Figure 1. A generic Data Warehouse architecture

The main characteristics of such applications are [DWQ97]:

- multidimensional view of data, and
- data analysis, through interactive and/or navigational querying of data.

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 [OLAP97]. A *dimension* is defined in [OLAP97] 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 [DWQ97].

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 2, where the upper levels of each hierarchy point to the lower levels:

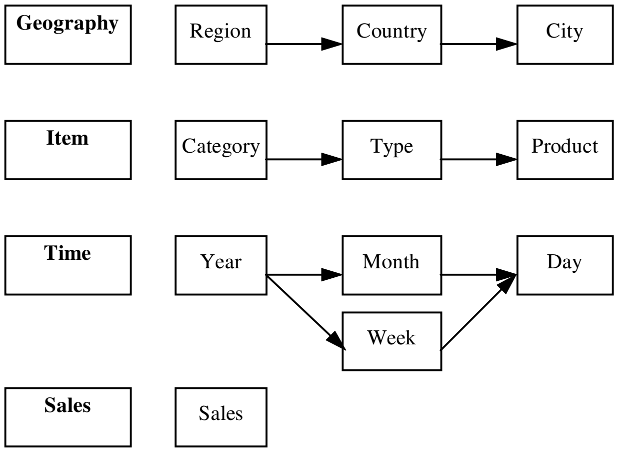

Figure 2. Dimensions and dimension levels

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 3. Item dimension

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

Figure 4. 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 [DWQ97], [OLAP97]. 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. [DWQ97], [OLAP97].

Two are the basic architectures for storing data in an OLAP database: ROLAP and MOLAP. *ROLAP (Relational OLAP*) [CD96] is based on a relational database server, extended with capabilities such as extended aggregation and partitioning of data [GBLP96]. The schema of the database can be a *star, snowflake,* or *fact constellation* schema [CD96]. 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 conceptual 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 conceptual 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 data warehouse designers can determine what kind of schemata they need for their warehouse/data marts to support cube operations, whereas the engines can map the performed cube operations to customized operations of their internal data model. Furthermore, there exist other areas of research where this formalization can be used: for example, in query optimization, the cube equivalence and reusability could be tested, using a formal framework such as the one we provide; cost models could also be based on this modeling.

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 in section 5 we provide a mapping to multidimensional arrays. In section 6, we present the conclusions of our work and possible future extensions.

2. Related Work

Research has followed the evolution of industrial products. The *data_cube* operator was introduced in [GBLP96]. There have also been efforts to model multidimensional databases. In [AGS95], 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 [LW96] 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 [GL97] *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 the data cube and monotone roll-up operators.

In [BPT97] 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, based on the star schema, for the efficient querying and update of a data warehouse, and not in the modeling of cubes or cube operations.

In [CT97], 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 [CT98] 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 [Shos97], quite a lot of similar work has been done in the past. In [Shos97] 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 [OOM85] 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 [OOM87]. Furthermore, physical organization and implementation issues are discussed. [OOM85] is very close to practical OLAP operations, although discussed in the context of summary tables.

In [RR91] 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 [AGS95], indirectly dealt through the introduction of aggregation [LW96], [GL97], or in a different context [OOM85], [RR91]. [BPT97] and [CT98] are basically dealing with the modeling of cubes. The best approach seems to be given in [CT98]; yet a direct mapping to OLAP operations is still not provided. Furthermore, apart for [RR91], series of operations are not directly dealt with. Finally, to our knowledge, no explicit modeling of the drill-down operation exists.

3. A conceptual model of multidimensional space and cubes

3.1. Multidimensional space

Let $\boldsymbol{\Omega}$ be the space of all *dimensions*. Let us suppose that we have N dimensions for a multidimensional database. Then, $\boldsymbol{\Omega} = \{D_1, D_2, ...D_N\}$. 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, i.e. $\boldsymbol{\Psi} = levels(D_1) \cup levels(D_2) \cup \ldots \cup levels(D_N) = \{DL_1, DL_2, ...DL_m\}$. 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 for each DL_i, $|h(DL_i)| = 1$, that is, 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 [OOM87], 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:

- $ancestor(x, DL) = x$, if $x \in dom(DL)$
- if $x = ancestor(y, DL)$ and $y = ancestor(x, DL)$, then $x = y$
- 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. For example, *ancestor*(1 FEB 1997, year) = 1997 regardless of which dimension path one should follow (i.e. the following hold: *ancestor*(1 FEB 1997, month) = FEB 1997, *ancestor*(FEB 1997, year) = 1997, *ancestor*(1 FEB 1997, week) = 5W 1997, *ancestor*(5W 1997, year) = 1997).

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 conceptual 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

- $\boldsymbol{D_b} = <D_1, D_2, \ldots D_n, M>$ is a list of dimensions ($D_i, M \in \boldsymbol{\Omega}$). M is a dimension that represents the measure of the cube.
- $\boldsymbol{L_b} = <DL_{b1}, DL_{b2}, \ldots DL_{bn}, *ML>$ is a list of dimension levels ($DL_{bi}, *ML \in \boldsymbol{\Psi}$. ML is the dimension level of the measure of the cube. $\forall\ DL_b \in \boldsymbol{L_b}$, $level(l) = 1$. In other words, every *basic_cube* must have all its dimension levels at the lowest level of their respective dimensions. 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 $x = [x_1, x_2, \ldots, x_n, *m]$, 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

- $\boldsymbol{D} = <D_1, D_2, \ldots D_n, M>$ 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)*.
- $\boldsymbol{L} = <DL_1, DL_2, \ldots DL_n, *ML>$ 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*.

- ***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 <***D***, *C*>. ***D*** is a set of dimensions and *C* is a basic cube, the dimensions of which belong to ***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. If we imagine the cube as a multidimensional space, then the dimensions are the coordinates of a point in it, and the measure, a value for this point. We define a cell x as a tuple $x = [x_1, x_2, ..., x_n, *m]$, and we define 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 5.

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

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:

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 5 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 [AGS95], [LW96] would not take this problem into account, or would assume that the operation will be disallowed by the system [RR91]. Since this kind of sequences of operations is typical for OLAP applications, the correctness of the result of the operations of the cube is 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 cube with the finest granularities of its dimensions, 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* [Eric95] 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. *Max* is not present in the list of the allowed functions, since it can be computed using *rank(1).* In the sequel we will suppose that the original cube $C = <\boldsymbol{D}, \boldsymbol{L}, C_b, \boldsymbol{R}>$, $\boldsymbol{D} = <D_1, D_2, ..., D_n, M>$, $\boldsymbol{L} = <DL_1, DL_2, ..., DL_n, *ML>$, $C_b = <\boldsymbol{D_b}, \boldsymbol{L_b}, C_b, \boldsymbol{R_b}>$ and that the new cube C', which is the result of the operations is $C' = <\boldsymbol{D}', \boldsymbol{L}', C_b', \boldsymbol{R}'>$.

Level_Climbing

We define *Level_Climbing* as follows:

$$C' = Level\_Climbing(C, \underline{d}, \underline{dl}) = LC(C, \underline{d}, \underline{dl})$$

where $\underline{d}$ is a set of dimensions belonging to C and $\underline{dl}$ is the set of the corresponding dimension levels of C. For reasons of expressiveness, and without loss of generality we will make the assumption 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 $\underline{d}$: $\underline{dl}_{old} = \{DL_{n-k+1}, \ldots, DL_n\}$. Then, we define C' to be as follows:

$\boldsymbol{D}' = \boldsymbol{D}$

$\boldsymbol{L}' = \boldsymbol{L} - \underline{dl}_{old} \cup \underline{dl}$

$C_b' = C_b$

$\boldsymbol{R}' = \{x \mid \exists\, y \in \boldsymbol{R}$:

$\quad dimensions(x)(D_i) = dimensions(y)(D_i)\ \forall\, D_i \notin \underline{d} \wedge$

$\quad dimensions(x)(D_i) = ancestor(dimensions(y)(D_i), dl_j),\ \forall\, D_i \in \underline{d}, dl_j \in \underline{dl}, dl_j \in levels(D_j) \wedge$

$\quad measure(x) = measure(y)$, if $M \notin \underline{d}$

}

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 6, 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 6. 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 *Packing* as follows:

$$C' = Packing(C) = P(C)$$

where C' has the following elements:

$\boldsymbol{D}' = \boldsymbol{D}$

$\boldsymbol{L}' = \boldsymbol{L}$

$C_b' = C_b$

$\boldsymbol{R}' = \{ x \mid \exists\, y \in \boldsymbol{R}$:

$\quad dimensions(x)(D_i) = dimensions(y)(D_i)\ \forall\, i \in 1, \ldots, n \wedge$

$\quad measure(x) = \{ l \mid \exists\, t \in \boldsymbol{R}, dimensions(y) = dimensions(t) \wedge l = measure(t) \}$

$\}$

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 7, 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 7. 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

We define *Function_Application* as follows:

$$C' = Function\_Application(C, f) = F(C, f)$$

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

$\boldsymbol{D}' = \boldsymbol{D}$

$\boldsymbol{L}' = \boldsymbol{L}$

$C_b' = C_b$

$\boldsymbol{R}' = \{x \mid \exists\, y \in \boldsymbol{R}$:

$\quad$ *dimensions*(x) = *dimensions*(y) $\wedge$

$\quad$ *measure*(x) = f(*measure*(y))

}

Intuitively, *Function_application* is the application of a specific function to the measure of a cube. In Figure 8, an example of the *Function_Application* operation is presented:

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. C3 = F(C2,sum) C3 = <*D3, L3*, C_b3, *R3*>,
***D3* = <Time, Item, Geography, Sales>, *L3* = <Year, Product, Region, Sales>, C_b3 = Basic_Cube, *R3* is shown in the above table**

Projection

We define *Projection* as follows:

$$C' = Projection(C, d) = \pi(C, d)$$

where d is the projected dimension. C' is then defined, using

$\boldsymbol{D}' = \boldsymbol{D} - d$

$\boldsymbol{L}' = \boldsymbol{L} - DL,\ DL \in levels(d),\ DL \in \boldsymbol{L}$

$C_b' = \langle \boldsymbol{D_b}', \boldsymbol{L_b}', C_b', \boldsymbol{R_b}' \rangle$, where, supposing that $C_b = \langle \boldsymbol{D_b}, \boldsymbol{L_b}, C_b, \boldsymbol{R_b} \rangle$,

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

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

$\quad \boldsymbol{R_b}' = \{x \mid \forall\, y \in \boldsymbol{R_b}$,

$\quad\quad$ *dimensions*(x)(D_i) = *dimensions*(y)(D_i), $\forall\, D_i \neq d,\ i \in 1, \ldots, n \wedge$

$\quad\quad$ *measure*(x) = *measure*(y)

$\quad\quad$ }

$\boldsymbol{R}' = \{x \mid \exists\, y \in \boldsymbol{R}$:

$\quad$ *dimensions*(x)(D_i) = *dimensions*(y)(D_i), $\forall\, D_i \neq d,\ i \in 1, \ldots, n \wedge$

$\quad$ *measure*(x) = *measure*(y)

}

One can prove easily that the definition of C' with C_b' as its *base_cube* is legal -i.e. C_b' fulfills the requirements to be defined as the *base_cube* of C'. This is formally proved as follows:

(1) $\forall\, l \in C_b'.L_b'$, level($l$) = 1. (We know that $\forall\, l \in L_b$, level(l) = 1. Since $L_b' = L - levels(d)(1) \Rightarrow \neg\exists\, l' \in L_b' : l' \in L_b \Rightarrow \forall\, l \in C_b'.L_b'$, level($l$) = 1)

(2) $\forall$ d $\in$ C'.D' $\exists$ d' $\in$ C_b'.D_b' : d = d'. (We know that $\forall$ d $\in$C.D, $\exists$ d_b $\in$ C_b.D_b : d = d_b. Since D' = D - d and C_b'.D_b' = D_b - d $\Rightarrow$ d $\in$ C'.D' $\exists$ d' $\in$ C_b'.D_b' : d = d')

From (1), and the fact that C_b' = <D_b', L_b', C_b', R_b'>, it follows that C_b' is a *basic_cube*. Furthermore, from (2) it follows that the definition of C' with C_b' as its *base_cube* is legal.

Intuitively, *projection* is the deletion of a dimension both from the cube and its *base_cube*. In Figures 9, 10 an example of the *Projection* operation is presented:

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

Figure 9. C4 = π(C3, time) C4 = <*D4*, *L4*, C_b4, *R4*>,
***D4* = <Item, Geography, Sales>, *L4* = < Product, Region, Sales>, *R4* 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 10. C4 = π(C3, time) ,C4 = <*D4*, *L4*, C_b4, *R4*>, C_b4 = <D_b*4*, L_b*4*, C_b4, R_b*4*>,
D_b*4* = <Item, Geography, Sales>, L_b*4* = < Product, City, Sales>, R_b*4* is shown in the above table

Navigation

We define *Navigation* as follows:

$$C' = Navigation(C, d, dl, f),$$

where *d* is the dimension over which we navigate, *dl* is the target level of the navigation and *f* is the applied aggregate function. Suppose that the dimension d is the *i-th* element of D. Then, we define C' to be 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:

C5 = Navigate(Basic_Cube, Geography, Region, no_operation)

C6 = Navigate(C5, Time, Year, sum)

C7= Navigate(C6, 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 11. C5 = Navigation(Basic_Cube, Geography, Region, no_operation), C5 = <*D5*, *L5*, C_b5, *R5*>, *D5* = <Time, Item, Geography, Sales>, *L5* = <Day, Product, Region, Sales>, C_b5 = Basic_Cube, *R5* 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 12. C6 = Navigation(C5, Time, Year, sum), C6 = <*D6*, *L6*, C_b6, *R6*>, *D6* = <Time, Item, Geography, Sales>, *L6* = <Year, Product, Region, Sales>, C_b6 = Basic_Cube, *R6* 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 13. C7= Navigation(C6, Time, Month, avg), C7 = <*D7*, *L7*, C_b7, *R7*>, *D7* = <Time, Item, Geography, Sales>, *L7* = <Month, Product, Region, Sales>, C_b7 = Basic_Cube, *R7* is shown in the above table

As the previous example shows, the basic contribution of the navigation operator is that it can allow any sequence of operations along the dimension hierarchies. For example, the navigation from the *Basic_Cube* to cube *C7*, is characterized by three features:

- it preserved the previous navigations (e.g. the navigation to the dimension level of Geography (Region))
- 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.
- 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

We define *Dicing* as follows:

$$C' = Dicing(C, d, \sigma(v)),$$

where d is the dimension over which we perform the dicing, σ is a formula consisting of a dimension, an operator and v, which is a value used as a parameter for the formula σ. 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 [OOM87]) -i.e. that {<, =} are applied to atomic dimension levels and {≡, ⊂, ∈} to multi-valued ones). Consequently, let $\sigma(v)$ be of the form d *op* v. Then, C' is defined as follows:

$\boldsymbol{D}' = \boldsymbol{D}$

$\boldsymbol{L} = \boldsymbol{L}'$

$C_b' = \langle \boldsymbol{D_b}', \boldsymbol{L_b}', C_b', \boldsymbol{R_b}' \rangle$, where $\boldsymbol{D_b}' = C_b.\boldsymbol{D_b}$, $\boldsymbol{L_b}' = C_b.\boldsymbol{L_b}$, and

$\boldsymbol{R_b}' = \{x \mid x \in C_b.\boldsymbol{R_b},\ x[d] \text{ op } y = \text{true},\ y \in \textit{descendants}(v, \textit{levels}(d)(1))\}$

$\boldsymbol{R}' = \{x \mid \exists\, x \in \boldsymbol{R},\ x[d] \text{ op } v = \text{true}\}$

One can prove easily that the definition of C' with C_b' as its *base_cube* is legal -i.e. C_b' fulfills the requirements to be defined as the *base_cube* of C'. This is formally proved as follows:

(1) $\forall\, l \in C_b'.L_b'$, level($l$) = 1. (We know that $\forall\, l \in L_b$, level(l) = 1. Since $L_b' = L - \textit{levels}(d)(1) \Rightarrow \neg\exists\, l' \in L_b' : l' \in L_b \Rightarrow \forall\, l \in C_b'.L_b'$, level($l$) = 1)

(2) $\forall\, d \in C'.D'\ \exists\, d' \in C_b'.D_b' : d = d'$. (We know that $\forall\, d \in C.D,\ \exists\, d_b \in C_b.D_b : d = d_b$. Since $D' = D - d$ and $C_b'.D_b' = D_b - d \Rightarrow d \in C'.D'\ \exists\, d' \in C_b'.D_b' : d = d'$)

From (1), and the fact that $C_b' = \langle D_b', L_b', C_b', R_b' \rangle$, it follows that C_b' is a *basic_cube*. Furthermore, from (2) it follows that the definition of C' with C_b' as its *base_cube* is legal.

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. In Figures 14, 15, an example of the *Dicing* operation is presented:

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
1996-09	"Piece of Mind"	Europe	7

Figure 14. C8 = Dicing(C7, Geography, Geography = "Europe"), C8 = <*D8*, *L8*, C_b8, *R8*>, *D8* = <Time, Item, Geography, Sales>, *L8* = <Month, Product, Region, Sales>, *R8* is shown in the above table

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
1996-09-07	"Piece of Mind"	Rhodes	7

Figure 15. C8 = Dicing(C7, Geography, Geography = "Europe"), C_b8 = <D_b8, L_b8, C_b8, R_b8>, D_b8 = <Time, Item, Geography, Sales>, *L8* = <Day, Product, City, Sales>, R_b8 is shown in the above table

Slicing

We define *Slicing* as follows:

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

where *d* is the dimension which we slice and *f* is the applied aggregate function. 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 16, 17, an example of the *Slicing* operation is presented:

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 16. C9 = Slicing(C6, Time, avg), C9 = <*D9*, *L9*, C_b9, *R9*>, *D9* = <Item, Geography, Sales>, *L9* = < Product, Region, Sales>, *R9* 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 17. C9 = Slicing(C6, Time, avg), C_b9 = <D_b9, L_b9, C_b9, R_b9>, D_b9 = <Item, Geography, Sales>, L_b9 = < Product, City, Sales>, R_b9 is shown in the above table

In this section we have defined *Cubes* and cube operations for a conceptual multidimensional model. The 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). It is worth noting that these characteristics would not exist if we had not introduced the notion of the *base_cube* of a cube. By defining operations on a cube with respect to the data of its base cube we are able to both keep track of previous modifications of the data and allow the applicability of all the aggregate functions in the context of series of operations.

Our model deals with the conceptual level of multidimensional databases. Yet, if we would like engines performing ROLAP and MOLAP to take advantage of it, we need a mapping of the model in the logical level. Therefore, we provide a relational mapping of both the structures and the operations of the multidimensional model, in the following section. A mapping to multidimensional arrays is also provided.

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

In this section we will try to the 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 [OOM87] to represent relational databases and tables. We keep the basic characteristics introduced in [OOM87]; yet we modify the application of aggregate functions to relations with a simpler approach.

A relation *r* is a set of n-tuples for some fixed integer n > 0. Each component of a tuple in *r* has a name, which is called an *attribute* of *r.* A *relation scheme R* is a set of attributes. *Deg*(*R*) denotes the number of attributes in *R*. Each attribute of a relation has an associated domain of values. Let ***V*** be the space of all values (as defined in section 3). *Attr*(*r*) denotes the set {1, 2, …,*Deg*(*R*)} of attributes of relation *r* whose relation scheme is *R*.

An attribute is *atomic* or *simple-valued* if its domain is a subset of ***V***. If the domain of an attribute is a subset of P (***V***) (i.e., power set of ***V***), the attribute is called a *set-valued attribute*[2]. As with dimension levels, we use bag semantics for multi-valued attributes. The symbol "*" is used as a prefix to the symbolic name of a set-valued attribute for distinguishing it from a atomic attributes. A *database scheme* is a sequence $<R_1,\dots, R_n>$ of relation schemes. An instance *I* of a database scheme $<R_1,\dots, R_n>$ is a sequence of *n* relations such that relation r_i has degree $Deg(R_i)$, i =1, …, n. *F = {sum, avg, count, min, rank(n), no-operation}* is the set of aggregate functions which can be applied.

Let t_1 and t_2 be two tuples having components for a set of attributes X. Then $t_1[X] = t_2[X]$ denotes $t_1[X_i] = t_2[X_i]$ for all $X_i \in X$.

As in [OOM87], the set **E** of extended relation algebra (ERA) expressions is defined. A fixed scheme $<R_1, \dots, R_n>$ is considered. Then, the following are defined:

Literals. For any constant $c \in N$ or $c \in P(N)$, $\{c\} \in \mathbf{E}$ and {c} has degree 1. Formally the instance of {c} is {c}(I) = {c}.

Relations. For each R_j in the scheme $<R_1, \dots, R_n>$, $R_j \in \mathbf{E}$ and $R_j(I) = r_j$.

Furthermore, we adopt several algebraic operators also defined in [OOM87]: projection, cross-product, restriction, set union, set difference, pack, unpack and ϑ-join. We will mention here the definition of pack (which groups tuples with equal values for a set of attributes into a new tuple) and projection.

---

[2] 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 [KMN 1997].

Projection. Let $e \in \mathbf{E}$, X be a subset of *Attr*(e), and *Deg*(e) = k. Then $e[X] \in E$, with *Deg*(e[X]) being equal to length (X), where length (X) is a function that returns the number of elements in X, and $e[X](I) = \{t[X] \mid t \in e(I)\}$.

Pack. Let $e \in \mathbf{E}$, *Deg(e)* = n, $X \in$ *Attr (e)* and CX = *Attr(e)* - {X}. Then $P_X(e) \in \mathbf{E}$ and has degree n. For each tuple $g \in e[CX](I)$ where *Deg*(e[CX]) = n-1, tuple w_g is defined as follows

$$w_g[CX] = g$$

$$w_g[X] = \begin{cases} \{t[x] \mid t \in e(I) \text{ and } t[CX] = g\}, \text{ if X is atomic} \\ \{y \mid (\exists (t \in e(I) \text{ and } (t[CX] = g) \text{ and } y \in)\}, \text{ otherwise} \end{cases}$$

Then, $P_X(e)(I) = \{w_g \mid g \in [CX] (I)\}$.

Finally, let us introduce the *Selection* and the *Aggregate_Function* application operators for set-valued attributes:

Selection. Let $e \in \mathbf{E}$, *Deg*(e) = n, X be a subset of *Attr*(e), f a formula involving:

- operands which are constants or attributes belonging to X
- applicable operators -i.e. $\{<, =\}$ are applied to atomic attributes and $\{\equiv, \subset, \in\}$ to set-valued ones
- logical operators such as *and, or, not*

Then, $e[X] \in \mathbf{E}$, with *Deg*(e[X]) = n and $e[f, X](I) = \{t \mid t \in (I) \text{ and } f(t[X]) \text{ is true}\}$.

Function_application. Let $e \in \mathbf{E}$, $*X \subseteq$ *Attr*(e), CX = *Attr*(e) - {*X}, $f_i \in F$. Then $e[*X, f_i] \in \mathbf{E}$ and has degree equal to *Deg*(e) = n. For each tuple $t \in e(I)$

$$e[*X, f_i] = \{y \mid \exists t \in e(I), y[CX] = t[CX] \text{ and } y[*X] = f(t[*X])\}$$

The motivation for the relational mapping is double: on the one hand, the engine performing ROLAP must be able to map conceptual to logical level 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 in the logical level and vice versa -what kind of database one needs to construct in order to be able to map a cube in the conceptual level to relational tables in the logical levels.

4.1. Mapping of cubes and tables

In this subsection, we will describe the mapping of basic entities of the conceptual model, such as cubes and dimensions to relational constructs. 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 $R(A_1, A_2, \ldots, A_k)$, *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, $t \in r$:
   $t[A_i] = ancestor(\delta,\lambda(A_i))$,
4. $\forall t \in r$, $\forall A_i \in R$, $\exists \delta$, $\delta \in dom(DL_s)$: $t[A_i] = ancestor(\delta,\lambda(A_i))$,

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 the 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 18.

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

Figure 18. Geography dimension as a table

From the definition of the *ancestor* operator, and its transitivity property it follows easily that $\forall A_i, A_j \in R$, $\exists DL_i, DL_j \in D$: $\alpha(DL_i) = A_i \wedge \alpha(DL_j) = A_j \wedge DL_i > DL_j$. Then, $\forall x \in r$, $x = \langle x_1, x_2, \ldots, x_k \rangle$, $x[A_i] = ancestor(x[A_j], DL_i)$. In other words, 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 $C = \langle \boldsymbol{D}, \boldsymbol{L}, C_{base}, \boldsymbol{R} \rangle$ (denoted also as $r = \boldsymbol{R_B}(C)$) iff

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

1. $\forall$ $d_i \in$ ***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. $\exists$ $r_B \in \mathbb{d}$: $r_B = \boldsymbol{R_B}(C)$

Definition 5. A database $\mathbb{d}$ defined over a database scheme ***S*** *represents* a cube C = <***D***, ***L***, C_{base}, ***R***> iff:

1. $\mathbb{d}$ covers C
2. $\exists$ $r_C \in \mathbb{d}$: $r_C = \boldsymbol{R_C}(C)$

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. In that case we say that the database *covers* the 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* (definition 5). 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*. If there existed a table *YEAR_REGION_SALES* under the schema

```
YEAR_REGION_SALES(YEAR, PRODUCT, REGION, SUM_SALES)
```

being the *cube_table* of cube *C6* (Figure 12), then the table *DETAILED_SALES* would be its *base_cube_table.* The database *TIME_M, TIME_W, GEOGRAPHY, ITEM*, *DETAILED_SALES* would then *cover C6* whereas the database *TIME_M, TIME_W, GEOGRAPHY, ITEM*, *DETAILED_SALES, YEAR_REGION_SALES* would *represent* it.

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 Appendix A. 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.

We will prove now, that for a basic cube C_{base}, a table *representing* it can also *cover* a cube based on C_{base}.

Theorem 1. Let C = <D, L, C_{base}, R>, C_{base} = <D', L', C_{base}, R'>, $r_1 = \boldsymbol{R_C}(C_{base})$ and $r_2 = \boldsymbol{R_B}(C)$, where r_1 is defined over R_1 ($A_{11}, A_{12}, ..., A_{1k}$) and r_2 is defined over R_2 ($A_{21}, A_{22}, ..., A_{2k}$). Then $r_1 = r_2$.

Proof. Since $r_1 = \boldsymbol{R_C}(C_{base})$ then

$\forall$ DL $\in C_{base}$.L', $\exists A_i \in R_1$: DL = $\lambda(A_i)$ (1)

$\forall x = <x_1, x_2, \ldots, x_{k-1}, *x_m> \in C_{base}.R', \exists t \in r_1: x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$ (2)

$\forall t \in r_1, t = <a_1, a_2, \ldots, a_{k-1}, *a_m>, \exists x, x \in C_{base}.R', : t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$. (3)

If we apply the definition of $\boldsymbol{R_b}$ to $r_2 = \boldsymbol{R_B}(C)$, then the resulting relations are:

$\forall$ DL $\in C_{base}$.L', $\exists A_i \in R_2$: DL = $\lambda(A_i)$ (4)

$\forall x = <x_1, x_2, \ldots, x_{k-1}, *x_m> \in C_{base}.R', \exists t \in r_2: x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$ (5)

$\forall t \in r_2, t = <a_1, a_2, \ldots, a_{k-1}, *a_m>, \exists x, x \in C_{base}.R', : t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$. (6)

For each dimension level DL_i in L', there is an attribute A_{1i} *representing* it in R_1 and an attribute A_{2i} *representing* it in R_2. Since, $\lambda(A_{1i}) = \lambda(A_{2i}) = dl_i$ then $dom(A_{1i}) = dom(A_{2i})$. Furthermore, since R_1, R_2 have the same number of attributes, there is a one to one mapping of the domains of the attributes of R_1 and R_2. So, $R_1 = dom(A_{11}) \times dom(A_{12}) \times \ldots \times dom(A_{1k}) = dom(A_{21}) \times dom(A_{22}) \times \ldots \times dom(A_{2k}) = R_2$.

From 2, 3 we can deduce that there is a one to one mapping between the tuples of r_1 and the tuples of R'. The same holds for the tuples of r_2 and R', as it can be deduced from 5, 6. Consequently, there is a one to one mapping of the tuples of r_1 and r_2.

Consequently, tables r_1 and r_2 have the same domains for their attributes and the same tuples and as a result, we can consider them to be equal. ■

For the rest of this paper, we assume that we have a cube C = <$\boldsymbol{D}$, $\boldsymbol{L}$, C_{base}, $\boldsymbol{R}$>, a database $\mathbb{d}$ defined over the database scheme $\boldsymbol{S}$ = ($R_C, R_B, R_{D1}, R_{D2}, ..., R_{Dn}$), an instantiation of $\boldsymbol{S}$, s = ($r_C, r_B, r_{D1}, r_{D2}, ..., r_{Dn}$) and that the following hold:

$\boldsymbol{D}$ = <$d_1, d_2, ..., d_n$, M> (I)

$\boldsymbol{L}$ = <$DL_1, DL_2, ..., DL_n$, *ML> (II)

$r_C = \boldsymbol{R_c}(C)$, where r_C is defined over $R_C = (A_{C1}, A_{C2}, ..., A_{Cn}, A_{CM})$ (III)

$r_B = \boldsymbol{R_B}(C)$, where r_B is defined over $R_B = (A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM})$ (IV)

$\forall d_i \in \boldsymbol{D}$, $r_{Di} = \boldsymbol{R_D}(d_i)$, where r_{Di} is defined over $R_{Di} = (A_{i1}, A_{i2}, ..., A_{ik})$. (V)

4.2. Overview of the 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.

In Table 1, one can see the relation definitions for the *base_cube_table* for the results of each cube operation. Remember that the base_cube is needed for the correct calculation of the results of cube operations. The same holds for the relational mapping of cubes: the operations must be performed with respect to the *base_cube_table* of the cube.

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* 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*. In the following theorem we present this approach formally.

When performing *Slicing*, the *base_cube* changes because of the presence of the projection in the sequence of applied operations.

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*, let r_{Di} be defined over $R_{Di} = (A_{i1}, A_{i2}, ..., A_{ij}, \ldots, A_{ik})$, representing dimension path D_{pi}. D_{pi} must be a dimension path, such that, for each dimension level $DL \in \underline{dl}$, a dimension level $DL_{old} \in \underline{dl}_{old}$ exists, -where $\underline{dl}_{old}$ are the respective levels of $\underline{d}$, in C.***L***- such that $DL_{old} \leq DL$. Obviously DL, DL_{old} must both belong to *levels(D_{pi})*. Let A_{ij} be the corresponding attribute of A_{Ci}, both representing DL_i, $DL_i \in \underline{dl}_{old}$. We rename A_{ij} as A_{Ci}. Let $r_{Di}' = (r_{Di})[\alpha(DLi), A_{Ci}]$, $\forall$ i, $k \leq i \leq n$, defined over $R_{di}' = (A_{Ci}', A_{ci})$. Furthermore, since r_{Di} represents D_i, $x[A_{Ci}] =$ *ancestor*$(x[A_{Ci}'], DL_i)$. Intuitively, 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* in a cube, 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' = *Level_Climbing*(C, $\underline{d}$, $\underline{dl}$)	$\boldsymbol{R_B}$(C') = r_B
C' = *Pack*(C)	$\boldsymbol{R_B}$(C') = r_B

C' = *Function_Application*(C, f)	$\boldsymbol{R_B}$(C') = r_B
C' = *Projection*(C, d)	$\boldsymbol{R_B}$(C') = $r_B' = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$ 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' = *Navigation(C, d, dl, f)*	$\boldsymbol{R_B}$(C') = r_B
C' = *Dicing*(C, d, σ(v))	$\boldsymbol{R_B}$(C') = $r_B' = ((r_B \bowtie_{AD1=AD1} r_D')[\sigma(v), A_D])[A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM}]$ 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 $A_{D1} = \alpha(\mathit{levels(d)}(1))$
C' = *Slicing(C, d, f)*	$\boldsymbol{R_B}$(C') = $r_B' = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$ 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.

In our running example, the *cube_table* for cube *C9 = Slicing(C6, Time, avg)* (Figure 16) would be the table *CITY_PRODUCT_SALES* under the schema

CITY_PRODUCT_SALES(PRODUCT, REGION, AVG_SALES)

which would be instantiated from the result of the following operation:

$(P_{SALES}($ *(DETAILED_SALES* $\bowtie_{DAY\ =\ DAY}$ *TIME_M[DAY, YEAR]* $\bowtie_{CITY\ =\ CITY}$ *GEOGRAPHY [CITY, REGION]* $\bowtie_{PRODUCT\ =\ PRODUCT}$ *ITEM) [REGION, PRODUCT, SALES])) [SALES, avg]*

Notice that the consistency of values of the dimension tables would allow us to use table *TIME_W* instead if table *TIME_M*. The *base_cube_table* for cube *C9* would be determined from the following schema:

CBT9 = DETAILED_SALES(DAY, PRODUCT, CITY, SALES)[PRODUCT, CITY, SALES]

C' = *Level_Climbing*(C, $\underline{d}$, $\underline{dl}$)	$\boldsymbol{R_c}(C') = r_C' = (r_C \bowtie_{A_{Cn-k+1} = A_{Cn-k+1}} r_{DCn-k+1}' \bowtie_{A_{Cn-k+2} = A_{Cn-k+2}} r_{DCn-k+2}' \bowtie \ldots \bowtie_{A_{Cn} = A_{Cn}} r_{Dn}')[A_{C1}, A_{C2}, \ldots A_{Cn-k}, A'_{Cn-k+1}, A'_{Cn}, A_{CM}]$ 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)	$\boldsymbol{R_c}(C') = r_C' = r_C[A_{C1}, A_{C2}, \ldots, A_{Ck-1}, A_{Ck+1}, \ldots, A_{Cn}, A_{CM}]$ defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Ck-1}, A_{Ck+1}, \ldots, A_{Cn}, A_{CM})$, where $A_{Ck} = \alpha(DL_k)$, $DL_k \in \boldsymbol{L}$, $DL_k \in$ *levels*(d).
C' = *Navigation(C, d, dl, f)*	$\boldsymbol{R_c}(C') = r_C' = (P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \ldots \bowtie_{ABn = ABn} r_{Dn}')[A_{C1}, A_{C2}, \ldots, A_{cn}, A_{BM}]))[{*}A_{BM}, f]$, 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 $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM})$, and $A_D = \alpha(DL_k)$, $DL_k \in C.\boldsymbol{L}$, $DL_k \in$ *levels(d).*
C' = *Slicing(C, d, f)*	$\boldsymbol{R_c}(C') = r_C' = (P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \ldots \bowtie_{ABn = ABn} r_{Dn}')[A_{C1}, A_{C2}, \ldots, A_{Ck-1}, A_{Ck+1}, \ldots, A_{Cn}, A_{BM}]))[{*}A_{BM}, f]$, 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

An interesting point for the relational mapping provided in this section is that the results of the cube operations do not always have to be stored. The definition of a cube in terms of its based cube can always be considered as a relational expression; in other words, as a view. This is especially crucial, since our approach focuses on the support of series of operations: the combination of relational expressions, produced from a series of cube operations can be optimized through classical optimization techniques for relational databases.

In our running example, suppose the *base_cube_table* for the *Slicing* operation, *CBT9*, does not necessarily have to be stored: it can be kept as a relational expression to be applied in the sequel for each new operation. Consequently, if a *Dicing* operation was to take place on top of cube *C9*, the *base cube table* for the *Dicing* operation can also be considered to be a new view, over the *base_cube_table CBT9*, which can obviously be reduced to a view over the *base_cube_table DETAILED_SALES*.

4.3. Relational Mapping of Cube Operations in Detail

In the rest of this paragraph we will detail 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.

Level_Climbing

Level_Climbing does not change the *base_cube* of a cube. Consequently, one would normally expect that the *base_cube_table* will not change either. This is presented formally in the following theorem.

Theorem 2. The set of tables (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Level_Climbing*(C, $\underline{d}$, $\underline{dl}$).

Proof. We know from (V) that $\forall$ $d_i \in \boldsymbol{D}$, $\exists$ $r_{Di} = \boldsymbol{R_D}(d_i)$. From the definition of *Level_Climbing*, *D*' = *D*; consequently $\forall$ $d_i \in D'$, $\exists$ $r_{Di} = \boldsymbol{R_D}(d_i)$ -i.e. that the dimension tables of C' are the same with the ones of C. We also know (IV) that $r_B = \boldsymbol{R_B}(C)$, where r_B is defined over $R_B = (A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM})$. Consequently, the following hold:

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

From the definition of *Level_Climbing* $C_{base}' = C_{base}$. Consequently, the following hold:

4. $\forall$ DL $\in C_b'.\boldsymbol{L_b}'$, $\exists$ $A_i \in R_B$: DL = $\lambda(A_i)$
5. $\forall$ x = <$x_1, x_2, \ldots, x_{k-1}, *x_m$> $\in C_b'.\boldsymbol{R_b}'$, $\exists$ t $\in r_B$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall$ t $\in r_B$, t = <$a_1, a_2, \ldots, a_{k-1}, *a_m$>, $\exists$ x, x $\in C_b'.\boldsymbol{R_b}'$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

From the definition of $\boldsymbol{R_B}$ and 4, 5, 6 it can follow easily, that $r_B = \boldsymbol{R_B}(C')$.

Consequently, the set of tables (R_B, R_{D1}, R_{D2}, …, R_{Dn}) *covers* C'. ■

We will use the following shortcut notations and the assumption that $\underline{d}$ consists of the k last dimensions of $\boldsymbol{D}$:

$\underline{d'}$ = D - $\underline{d}$, $\underline{d'}$ = {$D_1, D_2, \ldots, D_{n-k}$}

$\underline{d}$ = {$D_{n-k+1}, \ldots, D_n$}

$\underline{dl}_{old}$ = {$DL_{n-k+1}, \ldots, DL_n$}

$\underline{dl'}$ = L - $\underline{dl}_{old}$ = {$DL_1, DL_2, \ldots, DL_{n-k}$}

$\underline{dl}$ = {$DL_{n-k+1}', \ldots, DL_n'$}

Let r_{Di} be defined over $R_{Di} = (A_{i1}, A_{i2}, ..., A_{ik})$. Let A_{ij} be the corresponding attribute of A_{Ci}, both representing DL_i, $DL_i \in \underline{dl}_{old}$. We rename A_{ij} as A_{Ci}.

Let $r_{Di}' = (r_{Di})[\alpha(DLi), A_{Ci}]$, $\forall$ i, k $\leq$ i $\leq$ n, defined over $R_{di}' = (A_{Ci}', A_{Ci})$. Intuitively we project the dimension tables to the columns corresponding to the dimension levels of the new cube and the columns of the old cube. Furthermore, since r_{Di} *represents* D_i, $x[A_{Ci}]$ = *ancestor*($x[A_{Ci}']$, DL_i).

The relational mapping of the result of *Level_Climbing* in a cube, 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. The equivalence of this series of operations with the result of *Level_Climbing* is formally proved in the following theorem.

Theorem 3. The set of tables (r_C', r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C', R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Level_Climbing*(C, d, dl), where

$r_C' = (r_C \bowtie_{A_{Cn-k+1} = A_{Cn-k+1}} r_{DCn-k+1}' \bowtie_{A_{Cn-k+2} = A_{Cn-k+2}} r_{DCn-k+2}' \bowtie \ldots \bowtie_{A_{Cn} = A_{Cn}} r_{Dn}')[A_{C1}, A_{C2}, \ldots A_{Cn-k}, A'_{Cn-k+1}, A'_{Cn}, A_{CM}]$ defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn-k}, A_{Cn-k+1}', \ldots, A_{Cn}', A_{CM})$.

Proof. Since, (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that $r_C' = \boldsymbol{R_C}$ (C').

We know that $r_c = \boldsymbol{R_C}$ (C). This means that

1. $\forall$ DL $\in$ C.$\boldsymbol{L}$, $\exists$ $A_i \in R_C$: DL = $\lambda(A_i)$
2. $\forall$ x = <x_1, x_2, …, x_n, *x_m> $\in$ C.$\boldsymbol{R}$, $\exists$ t $\in$ r_C: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
3. $\forall$ t $\in$ r_C, t = <a_1, a_2, …, a_n, *a_m>, $\exists$ x, x $\in$ C.$\boldsymbol{R}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

We want to prove that

4. $\forall$ DL $\in$ C'.$\boldsymbol{L'}$, $\exists$ $A_i \in R_C'$: DL = $\lambda(A_i)$
5. $\forall$ x = <x_1, x_2, …, x_n, *x_m> $\in$ C'.$\boldsymbol{R'}$, $\exists$ t $\in$ r_C': $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall$ t $\in$ r_C', t = <a_1, a_2, …, a_n, *a_m>, $\exists$ x, x $\in$ C'.$\boldsymbol{R'}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

We can see that

(i) Since $AC_i' = \alpha(DLi)$, and 1. holds, then one can prove easily that 4. holds.

(ii) x $\in$ $\boldsymbol{R'}$, y $\in$ $\boldsymbol{R}$, $D_i \in$ d', i = 1, ... n-k, $x[D_i] = y[D_i]$ =>(from 2) $x[x_i] = t[\alpha(DL_i)]$.

(iii) x $\in$ $\boldsymbol{R'}$, y $\in$ $\boldsymbol{R}$, $D_i \in$ d, i = n-k+1, ..., n, $x[D_i] \in$ *dom*(DL_i), $x[D_i]$ = *ancestor*($y[D_i]$, DL_i) $\Rightarrow x[x_i] = t[\alpha(DL_i')]$.

(iv) From (ii) and (iii) 5. holds.

(v) t $\in$ r_C', y $\in$ r_C, $D_i \in$ d', $A_{Ci} = \alpha(DL_i)$, $t[A_{Ci}] = y[A_{Ci}] \Rightarrow$ (from 3) $t[A_{Ci}] = x[\lambda(A_{Ci})]$.

(vi) t $\in$ r_C', y $\in$ r_C, $D_i \in$ d, $A_{Ci}' = \alpha(DL_i)$, $t[A_{Ci}']$ = *ancestor*($y[A_{Ci}]$, DL_i) $\Rightarrow t[A_{Ci}'] = x[\lambda(A_{Ci}')]$.

(vii) From (v), (vi), 6. holds.

Since 4, 5, 6 hold then $r_C' = \boldsymbol{R_C}$ (C'). ■

Packing

Packing, just like *Level_Climbing,* does not change the *base_cube* of a cube. Consequently, the *base_cube_table* will not change either. This is presented formally in the following theorem.

Theorem 4. The set of tables (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Pack*(C).

Proof. From the definition of *Packing*, $\boldsymbol{D' = D}$. It can be easily proved from (V) (cf. Theorem 2) that the dimension tables of C' are the same with the ones of C. From the definition of *Packing* $C_{base}' = C_{base}$. From the definition of $\boldsymbol{R_B}$ it can follow easily, that since $r_B = \boldsymbol{R_B}$(C) then $r_B = \boldsymbol{R_B}$(C').
Consequently, the set of tables (R_B, R_{D1}, R_{D2}, …, R_{Dn}) *covers* C'. ■

The relational mapping of the result of *Packing* in a cube, is the performance of a *pack* operation (as defined in section 4.1) on its *cube_table*, on the attribute representing the measure of the cube. The equivalence of this operation with the result of *Packing* is formally proved in the following theorem.

Theorem 5. The set of tables (r_C', r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C', R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Packing*(C, $\underline{d}$, $\underline{dl}$), where $r_C' = P_{ACM}(r_c)$, defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM}')$, where we assume that $A_{CM}' = \alpha($*measure_dimension_level*(C')).

Proof. Since, (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that $r_C' = \boldsymbol{R_C}$ (C').

We know that $r_c = \boldsymbol{R_C}$ (C). This means that

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

We want to prove that

4. $\forall$ DL $\in$ C'.***L'***, $\exists\, A_i \in R_C'$: DL = $\lambda(A_i)$
5. $\forall\, x = <x_1, x_2, \ldots, x_{k-1}, *x_m> \in$ C'.***R'***, $\exists\, t \in r_C'$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall\, t \in r_C'$, $t = <a_1, a_2, \ldots, a_{k-1}, *a_m>$, $\exists$ x, x $\in$ C'.***R'***, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

4. obviously holds for all attributes, since all attributes except for A_{CM} remain the same and we have already assumed A_{CM}' to represent *measure_dimension_level*(C'). We will also prove that the values of the respective data for C' and r_C' are equal.

(i) Let x, $x_1, x_2, \ldots, x_k \in$ C.***R***: $x[D_j] = x_i[D_j]$, $\forall$ i = 1,…, k, j = 1, … n. Let $x_1, x_2, \ldots, x_k$ be the only cells with the previous property in C.***R***.

(ii) Let x': $y[D_j] = x[D_j]$, j = 1, … n and $y[*M] = \{x[M] \cup x_1[M] \cup \ldots \cup x_k[M]\}$. If we perform C' = *P(C)* then obviously x' $\in$ C'.***R'***.

(iii) Let also $t \in r_C$: $x[D_j] = t[\alpha(DL_j)]$, where $DL_j \in$ *levels*(D_j), $DL_j \in$ L, j = 1, … n.

(iv) Let $t_1, t_2, \ldots, t_k \in r_C$: $x_i[D_j] = t_i[\alpha(DL_j)]$ $\forall$ i = 1,…, k, j = 1,2, … n, m. (the number of t_j is equal to the number of x_j from 2., 3.). Note that x_i and t_i are equal for all the dimensions and the measure. Obviously $t[A_{Cj}] = t_i[A_{Cj}]$, $\forall$ i = 1,…, k, j = 1, … n. From 2,3 one can also prove that $t_1, t_2, \ldots, t_k$ are the only tuples with the previous property in r_C.

(v) Let t': $t'[A_{Cj}] = t[A_{Cj}] = t_i[A_{Cj}]$, $\forall$ i = 1,…, k, j = 1, … n and $t'[*A_{CM}] = \{t[*A_{CM}] \cup t_1[*A_{CM}] \cup \ldots \cup t_k[*A_{CM}]\}$

(vi) If we perform $r_C' = P_{*ACM}[r_C]$, then obviously $t' \in r_C'$.

5,6 hold for y, t since

(vii) $x'[D_j] = x[D_j] = t[\alpha(DL_j)] = t[A_{Cj}] = t'[A_{Cj}]$, j = 1, … n, and

(viii) $x'[*x_m'] = \{x[M] \cup x_1[M] \cup \ldots \cup x_k[M]\} = \{t[*A_{CM}] \cup t_1[*A_{CM}] \cup \ldots \cup t_k[*A_{CM}]\} = t'[*A_{CM}]$

Since 4, 5, 6 hold then $r_C' = \boldsymbol{R_C}$ (C') ■

Function_Application

Function_Application, as with *Packing* and *Level_Climbing,* does not change the *base_cube_table* of a cube, since the *base_cube* remains the same. This is presented formally in the following theorem.

Theorem 6. The set of tables (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Function_Application*(C, f).

Proof. From the definition of *Function_Application*, $\boldsymbol{D'} = \boldsymbol{D}$. It can be easily proved from (V) (cf. Theorem 2) that the dimension tables of C' are the same with the ones of C. From the definition of *Function_Application*, $C_{base}' = C_{base}$. From the definition of $\boldsymbol{R_B}$ it can follow easily, that since $r_B = \boldsymbol{R_B}(C)$ then $r_B = \boldsymbol{R_B}(C')$.

Consequently, the set of tables (R_B, R_{D1}, R_{D2}, …, R_{Dn}) *covers* C'. ■

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. The equivalence of this operation with the result of *Function_Application* is formally proved in the following theorem.

Theorem 7. The set of tables (r_C', r_B, r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C', R_B, R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Function_Application*(C, f), where $r_C' = r_c[{*}ACM, f]$ defined over $R_C' = (A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{CM}')$, where we assume that $A_{CM}' = \alpha($*measure_dimension_level*$(C'))$.

Proof. Since, (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that $r_C' = \boldsymbol{R_C}(C')$.

We know that $r_c = \boldsymbol{R_C}(C)$. This means that

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

We want to prove that

4. $\forall\, DL \in C'.\boldsymbol{L'}, \exists\, A_i \in R_C': DL = \lambda(A_i)$
5. $\forall\, x = <x_1, x_2, \ldots, x_n, {*}x_m> \in C'.\boldsymbol{R'}, \exists\, t \in r_C': x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$
6. $\forall\, t \in r_C', t = <a_1, a_2, \ldots, a_n, {*}a_m>, \exists\, x, x \in C'.\boldsymbol{R'}, : t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

4. obviously holds for all attributes, since all attributes except for A_{CM} remain the same and we have already set A_{CM}' to represent *measure_dimension_level*(C'). We will also prove that the values of the respective data for C' and r_C' are equal.

(i) Let $x \in C.\boldsymbol{R}$, $x = <x_1, x_2, \ldots, x_n, {*}x_m>$. Let also $t \in r_C$, $t = <a_1, a_2, \ldots, a_n, {*}a_m>: x[x_i] = t[\alpha(DL_i)]$, where $x_i \in dom(DL_i)$, i = 1, …, n, m

(ii) Let us also perform C' = *F(C, f)* and $r_C' = r_C[{*}A_{CM}, f]$. Then from the definition of these operations, $\exists\, x' \in C'.\boldsymbol{R'}, t \in r_C'$:

$x' = <x_1, x_2, \ldots, x_n, f({*}x_m)>$ and $t' = <a_1, a_2, \ldots, a_n, f({*}a_m)>$

5,6 hold for y, t since

(iii) $x'[D_j] = x[D_j] = t[\alpha(DL_j)] = t[A_{Cj}] = t'[A_{Cj}]$, j = 1, … n, and

(iv) $x'[{*}x_m'] = f({*}x_m) = f({*}a_m) = t'[{*}A_{CM}]$

Since 4, 5, 6 hold then $r_C' = \boldsymbol{R_C}(C')$ ■

Projection

The relational mapping of the result with respect to its *base_cube* of *Projection* on a cube, is the performance of a *projection* operation on the relevant attribute of its *base_cube_table*. The equivalence of this operation with the result of *Projection* for the *base_cube_table* is shown in the following theorem. We will assume that the k-th position of the cube, and respectively of the *cube_table* r_B, represents the dimension level for dimension d.

Theorem 8. The set of tables (r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Projection*(C, d), where $r_B' = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$ defined over $R_B' = (A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM})$, where we assume that $A_{Bk} = \alpha(DL_k)$, $DL_k \in C_{base}.L_{base}$, $DL_k \in$ *levels*(d).

Proof. From the definition of *Projection*, D' = D - d. We know that $r_B = \boldsymbol{R_B}$ (C). This means that

1. $\forall\ DL \in C_{base}.\boldsymbol{L_{base}},\ \exists\ A_i \in R_B$: $DL = \lambda(A_i)$
2. $\forall\ x = <x_1, x_2, \ldots, x_n, *x_m> \in C_{base}.\boldsymbol{R_{base}},\ \exists\ t \in r_B$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
3. $\forall\ t \in r_B$, $t = <a_1, a_2, \ldots, a_n, *a_m>$, $\exists\ x, x \in C_{base}.\boldsymbol{R_{base}}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

We want to prove that

4. $\forall\ DL \in C_{base}'.\boldsymbol{L'},\ \exists\ A_i \in R_B'$: $DL = \lambda(A_i)$
5. $\forall\ x = <x_1, x_2, \ldots, x_n, *x_m> \in C_{base}'.\boldsymbol{R_{base}'},\ \exists\ t \in r_B'$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall\ t \in r_B'$, $t = <a_1, a_2, \ldots, a_n, *a_m>$, $\exists\ x, x \in C_{base}'.\boldsymbol{R_{base}'}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

4\. obviously holds for all attributes, since all attributes remain the same (except for the one which is in any case projected). Furthermore,

(i) Let $x = <x_1, x_2, \ldots, x_n, *x_m>$, $x \in C_{base}.\boldsymbol{R_{base}}$. Let also $t \in r_B$, $t = <a_1, a_2, \ldots, a_n, *a_m>$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i), i = 1, …, n, m

(ii) Let us also perform C' = *Projection*(C, d), and $r_B' = r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn}, A_{BM}]$. Then from the definition of these operations, $\exists\ x' \in C_{base}'.\boldsymbol{R_{base}'}$, $t \in r_B'$:

$x' = <x_1, x_2, \ldots, x_{k-1}, x_{k+1}, ..., x_n, *x_m>$ and $t' = <a_1, a_2, \ldots, a_{k-1}, a_{k+1}, ..., a_n, *a_m>$

5,6 hold for y, t since

(iii) $x'[D_j] = x[D_j] = t[\alpha(DL_j)] = t[A_{Cj}] = t'\ [A_{Cj}]$, j = 1, … n,m, $j \neq k$.

Since 4, 5, 6 hold, then $r_B' = \boldsymbol{R_B}$ (C'). ■

In theorem 8 we have proved that a *projection* on the *base_cube_table* models the result of a *Projection* operation on a cube with respect to its *base_cube*. In the following theorem we will prove that a projection on the *cube_table* can model the results of the *Projection* of a cube with respect to its cell data.

Theorem 9. The set of tables (r_C', r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C',R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Projection*(C, d), where $r_C' = r_C[A_{C1}, A_{C2}, ..., A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{CM}]$ defined over $R_C' = (A_{C1}, A_{C2}, ..., A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{CM})$, where we assume that $A_{Ck} = \alpha(DL_k)$, $DL_k \in L$, $DL_k \in$ *levels*(d).

Proof. Since, (r_B', r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that $r_C' = \boldsymbol{R_C}$ (C').

We know that $r_C = \boldsymbol{R_C}$ (C). This means that

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

We want to prove that

4. $\forall$ DL $\in$ C'.$\boldsymbol{L'}$, $\exists$ $A_i \in R_C'$: DL = $\lambda(A_i)$

5. $\forall$ x = <$x_1, x_2, \ldots, x_n, *x_m$> $\in$ C'.$\boldsymbol{R'}$, $\exists$ t $\in$ r_C': x[x_i] = t[α(DL_i)], where $x_i \in$ *dom*(DL_i)

6. $\forall$ t $\in$ r_C', t = <$a_1, a_2, \ldots, a_n, *a_m$>, $\exists$ x, x $\in$ C'.$\boldsymbol{R'}$, : t[A_i] = x[$\lambda(A_i)$], where $a_i \in A_i$.

4. obviously holds for all attributes, since all attributes remain the same (except for the one which is in any case projected). Furthermore,

(i) Let x = <$x_1, x_2, \ldots, x_n, *x_m$>, x $\in$ C.$\boldsymbol{R}$. Let also t $\in$ r_C, t = <$a_1, a_2, \ldots, a_n, *a_m$>: x[$x_i$] = t[$\alpha$($DL_i$)], where $x_i \in$ *dom*(DL_i), i = 1, …, n, m

(ii) Let us also perform C' = *Projection*(C, d), and r_C' = r_C[$A_{C1}, A_{C2}, ..., A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{CM}$]. Then from the definition of these operations, $\exists$ x' $\in$ C'.$\boldsymbol{R'}$, t $\in$ r_C': x' = <$x_1, x_2, \ldots, x_{k-1}, x_{k+1}, ..., x_n, *x_m$> and t' = <$a_1, a_2, \ldots, a_{k-1}, a_{k+1}, ..., a_n, *a_m$>

5,6 hold for y, t since

(iii) x'[D_j] = x[D_j] = t[α(DL_j)] = t[A_{Cj}] = t' [A_{Cj}], j = 1, … n,m, j $\neq$ k

Since 4, 5, 6 hold then r_C' = $\boldsymbol{R_C}$ (C') ■

Navigation

In Section 3, *Navigation* has been defined as:

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

If we apply the relational mappings of the cube operations for the *cube_table*, it can be easily proved that the dimension tables of C' are the same with the ones of C. Furthermore, the *base_cube* does not change for any of the operations applied to C_{base}. Consequently, the following theorem holds:

Theorem 10. The set of tables ($r_B, r_{D1}, r_{D2}, ..., r_{Dn}$) defined over ($R_B, R_{D1}, R_{D2}, ..., R_{Dn}$) *covers* C' = *Navigation*(C, d, dl,f).

Proof. If we apply the relational mappings of the cube operations for the *cube_table*, it can be easily proved that the dimension tables of C' are the same with the ones of C. Furthermore, the *base_cube* does not change for any of the operations applied to C_{base}. Consequently, since r_B = $\boldsymbol{R_B}$(C) then r_B = $\boldsymbol{R_B}$(C') and the set of tables ($R_B, R_{D1}, R_{D2}, \ldots, R_{Dn}$) *covers* C'. ■

Furthermore, the application of the relational mappings of the cube operations which participate at the definition of *Navigation*, produces the following theorem for the calculation of the *cube_table* of the product of *Navigation*.

Theorem 11. The set of tables ($r_C', r_B, r_{D1}, r_{D2}, ..., r_{Dn}$) defined over ($R_C', R_B, R_{D1}, R_{D2}, ..., R_{Dn}$) *represents* C' = *Navigation*(C, d, dl, f), where r_C' is defined over R_C' = ($A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{BM}'$) and has the following form:

$(P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \ldots \bowtie_{ABn = ABn} r_{Dn}')[A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{BM}]))[*A_{BM}, f]$,

$r_{Di}' = (r_{Di})[A_{Bi}, A_{Ci}] \forall$ i, $1 \leq i \leq n$.

We also assume that A_{BM}' = α(*measure_dimension_level*(C')).

Proof. Since, ($r_B, r_{D1}, r_{D2}, ..., r_{Dn}$) *covers* C', we just have to prove that r_C' = $\boldsymbol{R_C}$ (C').

If we apply the relational mappings of the cube operations for the *cube_table*, the aforementioned formula is produced for r_C', which has to be defined over R_C' = ($A_{C1}, A_{C2}, \ldots, A_{Cn}, A_{BM}'$). Furthermore, the relational mappings impose the assumption about the mapping of A_{BM}' to

α(*measure_dimension_level*(C'). ■

Dicing

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 respective dimension table, perform the *selection* at the result and then *project* the attributes of the *base_cube_table*. In the following theorem we present this approach formally.

Theorem 12. The set of tables (r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Dicing*(C, d, σ(v)), where $r_B' = ((r_B \bowtie_{AD1=AD1} r_D')[\sigma(v), A_D])[A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM}]$ defined over $R_B' = (A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM})$, and $A_D = \alpha(DL_k)$, $DL_k \in$ C.L, $DL_k \in$ *levels(d)*, r_D *represents* d, $r_D' = (r_D)[A_{D1}, A_D]$ and $A_{D1} = \alpha($*levels(d)*$(1))$.

Proof. From the definition of *Dicing*, ***D' = D.*** It can be easily proved from (V) that the dimension tables of C' are the same with the ones of C.

We know that $r_B = \boldsymbol{R_B}$ (C). This means that

1. $\forall$ DL $\in C_{base}.\boldsymbol{L_{base}}$, $\exists A_i \in R_B$: DL = $\lambda(A_i)$
2. $\forall x = <x_1, x_2, \ldots, x_n, *x_m> \in C_{base}.\boldsymbol{R_{base}}$, $\exists t \in r_B$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
3. $\forall t \in r_B$, $t = <a_1, a_2, \ldots, a_n, *a_m>$, $\exists x$, $x \in C_{base}.\boldsymbol{R_{base}}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

We want to prove that

4. $\forall$ DL $\in C_{base}'.\boldsymbol{L_{base}'}$, $\exists A_i \in R_B'$: DL = $\lambda(A_i)$
5. $\forall x = <x_1, x_2, \ldots, x_n, *x_m> \in C_{base}'.\boldsymbol{R_{base}'}$, $\exists t \in r_B'$: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall t \in r_B'$, $t = <a_1, a_2, \ldots, a_n, *a_m>$, $\exists x$, $x \in C_{base}'.\boldsymbol{R_{base}'}$, : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

4. obviously holds for all attributes. Furthermore,

(i) Let $x = <x_1, x_2, \ldots, x_D, ..., x_n, *x_m>$, $x \in C_{base}.\boldsymbol{R_{base}}$. Let also $t \in r_B$, $t = <a_1, a_2, \ldots, a_D, ..., a_n, *a_m>$: $x[x_i] = t[\alpha(DL_i)] = t[A_i]$, and where $x_i \in$ *dom*(DL_i), i = 1, …, n.

(ii) Let us consider $x' = x \circ <x_D'>$, $x_D' =$ *ancestor*(x_D, DL_k) and $t' = t \circ [a_D']$, $t' \in r_B''$, r_B'' defined on ($r_B \bowtie_{AD1=AD1} r_D'$). Then, $x[x_i] = t[\alpha(DL_i)]$, and $x'[x_D'] = t'[a_D']$. Furthermore, let r_B''' be defined as ($r_B \bowtie_{AD1=AD1} r_D'$) [$A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM}$]. Then $t \in r_B'''$.

(iii) If we apply σ(v) to both $x'[x_D']$ and $t'[a_D']$, then it is true either for both or for none, since their values are equal. Let us suppose it is true. Then $t \in r_B'''$ also belongs to $r_B' = ((r_B \bowtie_{AD1=AD1} r_D')[\sigma(v), A_D])[A_{B1}, A_{B2}, ..., A_{Bn}, A_{BM}]$, whereas x belongs to $C_{base}'.\boldsymbol{R_{base}'}$. If we suppose that σ(v) is false, then none of x' and t' belongs to r_B' or $C_{base}'.\boldsymbol{R_{base}'}$ respectively.

(iv) From (i), (iii) 5,6 hold.

Since 4, 5, 6 hold, then $r_B' = \boldsymbol{R_B}$ (C'). ■

Although the mapping of *Dicing* is rather complex for the *base_cube_table*, it is much simpler for the *cube_table*: just the performance of a *selection* on its *cube_table*. This is formally proved in the following theorem.

Theorem 13. The set of tables (r_C', r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C', R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Dicing*(C, d, σ(v)), where 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.L, $DL_k \in$ *levels(d).*

Proof. Since, (r_B, r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that r_C' = $\boldsymbol{R_C}$ (C').

We know that $r_C = \boldsymbol{R_C}$ (C). This means that

1. $\forall$ DL $\in$ C.***L,*** $\exists$ $A_i \in R_C$: DL = $\lambda(A_i)$
2. $\forall$ x = <x_1, x_2, …, x_n, *x_m> $\in$ C.***R,*** $\exists$ t $\in$ r: $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
3. $\forall$ t $\in$ r_C, t = <a_1, a_2, …, a_n, *a_m>, $\exists$ x, x $\in$ C.***R,*** : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

We want to prove that

4. $\forall$ DL $\in$ C'.***L',*** $\exists$ $A_i \in R_C$': DL = $\lambda(A_i)$
5. $\forall$ x = <x_1, x_2, …, x_n, *x_m> $\in$ C'.***R',*** $\exists$ t $\in$ r_C': $x[x_i] = t[\alpha(DL_i)]$, where $x_i \in$ *dom*(DL_i)
6. $\forall$ t $\in$ r_C', t = <a_1, a_2, …, a_n, *a_m>, $\exists$ x, x $\in$ C'.***R',*** : $t[A_i] = x[\lambda(A_i)]$, where $a_i \in A_i$.

4. obviously holds for all attributes. Furthermore,

(i) Let x = <x_1, x_2, …, x_D, ..., x_n, *x_m>, x $\in$ C.***R.*** Let also t $\in$ r_C, t = <a_1, a_2, …, a_D, ..., a_n, *a_m>: $x[x_i] = t[\alpha(DL_i)] = t[A_i]$, and where $x_i \in$ *dom*(DL_i), i = 1, …, n.

(ii) If we apply σ(v) to both $x[x_D]$ and $t[a_D]$, then it is either true for both or for none, since their values are equal. Let us suppose it is true. Then t $\in$ r_C also belongs to r_C' = $r_C[\sigma(v), A_D]$ whereas x belongs to C'.***R'.*** If we suppose that σ(v) is false, then none of x' and t' belongs to r_C' or C'.R' respectively.

(iii) From (i), (ii) 5,6 hold.

Since 4, 5, 6 hold then r_C' = $\boldsymbol{R_C}$ (C'). ■

<u>Slicing</u>

In Section 3, *Slicing* is defined as follows:

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

If we apply the relational mappings of the cube operations for the *cube_table*, it can be easily proved that the dimension tables of C' are the same with the ones of C. Furthermore, the *base_cube* changes because of the presence of the projection in the sequence of applied operations, whereas the rest of the operations leave the *base_cube* unchanged. The following theorem formalizes this approach.

Theorem 14. The set of tables (r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C' = *Slicing*(C, d, f), where r_B' = $r_B[A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., A_{Bn,} A_{BM}]$ defined over R_B' = (A_{B1}, A_{B2}, ..., A_{Bk-1}, A_{Bk+1}, ..., $A_{Bn,}$ A_{BM}), and $A_{Bk} = \alpha(DL_k)$, $DL_k \in C_b.L_b$, $DL_k \in$ *levels(d).*

Proof. If we apply the relational mappings of the cube operations for the *cube_table*, it can be easily proved that the dimension tables of C' are the same with the ones of C. Furthermore, the *base_cube* changes because of the presence of the projection in the sequence of applied operations, whereas the rest of the operations leave the *base_cube* unchanged.

Consequently, r_B' = $\boldsymbol{R_B}$(C') and the set of tables (R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *covers* C'. ■

Furthermore, the application of the relational mappings of the cube operations for the *cube_table* produces the following theorem:

Theorem 15. The set of tables (r_C', r_B', r_{D1}, r_{D2}, ..., r_{Dn}) defined over (R_C', R_B', R_{D1}, R_{D2}, ..., R_{Dn}) *represents* C' = *Slicing*(C, d, f), where r_C' is defined over R_C' = (A_{C1}, A_{C2}, …, A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{BM}') and has the following form:

$(P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \dots \bowtie_{ABn = ABn} r_{Dn}')\ [A_{C1}, A_{C2}, \dots, A_{Ck-1}, A_{Ck+1}, \dots, A_{Cn}, A_{BM}]))[*A_{BM}, f]$,

$r_{Di}' = (r_{Di})[A_{Bi}, A_{Ci}] \forall\ i,\ 1 \leq i \leq n$.

We assume that the k-th position of the cube, and respectively of the *base_cube_table* r_B represents the dimension level for dimension d. We also assume that $A_{BM}' = \alpha$(*measure_dimension_level*(C')).

Proof. Since, (r_B', r_{D1}, r_{D2}, ..., r_{Dn}) *covers* C', we just have to prove that $r_C' = \boldsymbol{R_C}$ (C').

If we apply the relational mappings of the cube operations for the *cube_table*, the following formula is produced for r_C', which has to be defined over R_C' = (A_{C1}, A_{C2}, …, A_{Ck-1}, A_{Ck+1}, ..., A_{Cn}, A_{BM}'):

$(P_{ABM}(((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \dots \bowtie_{ABn = ABn} r_{Dn}')[A_{C1}, A_{C2}, \dots, A_{Cn}, A_{BM}])[A_{C1}, A_{C2}, \dots, A_{Ck-1}, A_{Ck+1}, \dots, A_{Cn}, A_{BM}]))[*A_{BM}, f] =$

$(P_{ABM}((r_B \bowtie_{AB1 = AB1} r_{D1}' \bowtie_{AB2 = AB2} r_{D2}' \bowtie \dots \bowtie_{ABn = ABn} r_{Dn}')\ [A_{C1}, A_{C2}, \dots, A_{Ck-1}, A_{Ck+1}, \dots, A_{Cn}, A_{BM}]))[*A_{BM}, f]$

Furthermore, the relational mappings impose the assumption about the mapping of A_{BM}' to α(*measure_dimension_level*(C')). ■

In the following section, we will provide a mapping of the conceptual multidimensional model to multidimensional arrays.

5. A mapping of the conceptual multidimensional model to multidimensional arrays

The conceptual multidimensional model can trivially be mapped to multidimensional arrays, practically in the same way it is done in [CT98]. 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 assignement is done in a contiguous fashion.

As a result, each value $x = [x_1, x_2, \dots, x_n, *m]$, belonging to the cell data of a cube can be considered to be as the conjuction of coordinates $[enum(x_1), enum(x_2), \dots, enum(x_n)]$ with value $*m$. We consider that the multidimensional array takes a NULL value in the conjuction of coordinates when the corresponding tuple of tis conjunction does not exist in the cell data of the original cube. Let us consider the example of figures 3, 4, 5. We denote each value with its mapping to integers.

Category	Type	Product
[1] Books	[1] Literature	[1] "Report to El Greco" N. Kazantzakis
		[2] "Karamazof brothers" F. Dostoiewsky
	[2] Philosophy	[3] "Zarathustra", F. W. Nietzsche
		[4] "Symposium", Plato

Region	Country	City
[1] Europe	[1] Hellas	[1] Athens
		[2] Rhodes
	[2] France	[3] Paris
[2] Asia	[3] Israel	[4] Tel Aviv

[2] Music	[3] Heavy Metal	[5] "Piece of Mind", Iron Maiden
		[6] "Ace of Spades", Motorhead

Figure 19. Item dimension

	[4] Japan	[5] Tokyo

Figure 20. Geography dimension

We assume that a mapping of integers is also made for dates (as a concatenation of the numbers of the date).

Time	Item	Geography	Sales
[19960101] 1996-01-01	[2] "Karamazof brothers"	[4] Tel Aviv	40
[19960328] 1996-03-28	[2] "Karamazof brothers"	[4] Tel Aviv	12
[19960506] 1996-05-06	[5] "Piece of Mind"	[5] Tokyo	10
[19960907] 1996-09-07	[5] "Piece of Mind"	[2] Rhodes	7
[19961012] 1996-10-12	[1] "Report to El Greco"	[3] Paris	7
[19970101] 1997-01-01	[1] "Report to El Greco"	[1] Athens	11
[19970101] 1997-01-01	[1] "Report to El Greco"	[2] Rhodes	15
[19970101] 1997-01-01	[6] "Ace of Spades"	[3] Paris	8
[19970206] 1997-02-06	[4] "Symposium"	[2] Rhodes	7
[19970218] 1997-02-18	[1] "Report to El Greco"	[1] Athens	2
[19970218] 1997-02-18	[2] "Karamazof brothers"	[3] Paris	5
[19970303] 1997-03-03	[2] "Karamazof brothers"	[1] Athens	10
[19970303] 1997-03-03	[2] "Karamazof brothers"	[2] Rhodes	4
[19970328] 1997-03-28	[4] "Symposium"	[2] Rhodes	5

Figure 21. Multidimensional array R, representing the cell data of the Basic Cube

From the mapping above we can consider that R[19960907][5][2] = 7, whereas R[19960907][5][1] = NULL. Furthermore, if several operations are performed and there is a change in the dimension levels of the cube, the mapping can still be made. For example, for the cell data of cube C5 (figure 11) we have the following mapping:

Time	Item	Geography	Sales
[19960101] 1996-01-01	[2] "Karamazof brothers"	[2] Asia	40
[19960328] 1996-03-28	[2] "Karamazof brothers"	[2] Asia	12
[19960506] 1996-05-06	[5] "Piece of Mind"	[2] Asia	10
[19960907] 1996-09-07	[5] "Piece of Mind"	[1] Europe	7
[19961012] 1996-10-12	[1] "Report to El Greco"	[1] Europe	7
[19970101] 1997-01-01	[1] "Report to El Greco"	[1] Europe	15, 11
[19970101] 1997-01-01	[6] "Ace of Spades"	[1] Europe	8
[19970206] 1997-02-06	[4] "Symposium"	[1] Europe	7
[19970218] 1997-02-18	[1] "Report to El Greco"	[1] Europe	2
[19970218] 1997-02-18	[2] "Karamazof brothers"	[1] Europe	5
[19970303] 1997-03-03	[2] "Karamazof brothers"	[1] Europe	4, 10
[19970328] 1997-03-28	[4] "Symposium"	[1] Europe	5

Figure 22. Multidimensional array R', representing the cell data of the cube C5

In the array depicted in figure 22, the mapping is still made, yet the values obviously change. Now, we have that R[19960907][5][2] = NULL, whereas R[19960907][5][1] = 7.

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

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

6. Conclusions - Future work

In this paper we have proposed a conceptual 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 conceptual multidimensional model to the logical level. In the relational mapping, the cubes and the dimensions are mapped to relations. The mapping of cube operations to relational algebra operations has also been devised. In the mapping to multidimensional arrays, a mapping function is employed so as to view the data of a cube as a multidimensional array.

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). Furthermore, the notion of the *base_cube*, although it provides "correct"[3] results, can be optimized. The applicability of existing results of research on view usability [LMSS95] can also be investigated in the framework we have set (especially since a relational mapping is provided). 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.

7. References

[AGS95] R. Agrawal, A. Gupta, S. Sarawagi, *"Modeling Multidimensional Databases"*, IBM Research Report, IBM Almaden Research Center, September 1995

[BPT97] E. Baralis, S. Paraboschi, E. Teniente, *"Materialized View Selection in a Multidimensional Database"*, Proceedings of the 23rd International Conference on Very Large Databases (VLDB), August 1997

[CD96] S. Chaudhuri, U.Dayal, *"Data warehousing and OLAP for Decision Support"*, Tutorials of 22nd International Conference on Very large Data Bases, Mumbai, India,

[3] according to our own notion of correctness, in the context of series of operations, as presented in Section 3.

September 1996

[Coll96] George Colliat. *"OLAP, Relational, and Multidimensional Database Systems", SIGMOD Record*, Vol. 25, No. 3, September 1996.

[CT97] L. Cabbibo, R. Torlone, *"Querying Multidimesional Databases"*, 6th International Workshop on Database Programming Languages (DBPL6), 1997

[CT98] L. Cabbibo, R. Torlone, *"A Logical Approach to Multidimensional Databases"*, EDBT 1998

[DWQ97] 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/

[Eric95] C.G. Erickson, *"Multidimensionalism and the data warehouse"*, in the Data Warehousing Conference (Orlando FL, February 1995)

[GBLP96] 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

[GL 1997] M. Gyssens, L.V.S. Lakshmanan, *"A Foundation for Multi-Dimensional Databases"*, Proceedings of the 23rd International Conference on Very Large Databases (VLDB), August 1997

[KMN97] K. Kulkarni, N. Mattos, A. Nori, *"Object-Relational Database Systems - Principles, Products and Challenges"*, Tutorials of the 23rd International Conference on Very Large Databases (VLDB), August 1997

[LMSS95] A. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava, *"Answering Queries using Views"*, In. Proc. of the ACM Symposium on Principles of Database Systems, 1995

[LW96] C. Li, X. Sean Wang, *"A Data Model for Supporting On-Line Analytical Processing"*, CIKM 1996

[OLAP97] OLAP Council, *"The OLAP glossary"*. http://www.olapcouncil.org, The *OLAP Council*, 1997.

[OOM85] G. Ozsoyoglu, M. Ozsoyoglu, F. Mata, *"A Language and a Physical Organization Technique for Summary Tables"*, Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, Austin, Texas, May 1985

[OOM87] G. Ozsoyoglu, M. Ozsoyoglu, V. Matos, *"Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregation Functions"*, ACM TODS 12(4), 1987

[RR91] M. Rafanelli, F.L. Ricci, *"A functional model for macro-databases"*, SIGMOD Record, March 1991, 20(1)

[Shos97] A. Shoshani, *"OLAP and Statistical Databases: Similarities and Differences"*, Tutorials of PODS 1997

[Wido95] J. Widom, *"Research Problems in Data Warehousing"*, Proceedings of the 4th Int'l

APPENDIX A: Star and Snowflake Schema Mapping

A denormalized *star* schema can be directly mapped to its respecting fully normalized *snowflake* schema. The difference in the two schemata lies in the way they represent the *dimension* tables (the *fact* table -*base cube*, or *cube table* in our case- remains the same). We will prove that the dimension table of a star schema can be considered as a view defined on the respective tables of the snowflake schema. In this paper we provide a mapping of cubes and cube operations to the star schema - nevertheless, since a direct mapping exists, both cubes and operations can be represented from snowflake schemata.

We will first show how the cubes can be mapped to snowflake schemata, by decomposing a dimension table to its respective normalized relations. Consider a dimension table r defined on a relation scheme $R(A_1, A_2, \ldots, A_k)$ and representing dimension path D_p. We will suppose an ordering of attributes, in the sense that the lower the subscript of an attribute is, the lower its granularity. Formally, let

1. $levels(D_p) = \{DL_1, DL_2, \ldots, DL_k\}$,
2. $\lambda(A_i) = DL_i, \forall\ i \in 1 .. k$, and
3. $DL_i < DL_j\ \forall\ i, j \in 1 .. k, i < j$.

We have made the assumption than for each value there exists exactly one ancestor. From the definition of *dimension tables*, the dimension table contains exactly one tuple per lowest level value. Consequently, we can consider the value of this instance to be the key of the tuple and A_1 is the key for table R. Furthermore, as in [BPT97], for each A_i, A_j, $i < j$, we can define a functional dependency $A_i \rightarrow A_j$ (since the value of A_j is an ancestor of the value of A_i and each value can have only one ancestor). Consequently, for the relation R we have the following functional dependencies:

$A_1 \rightarrow A_2$ (1)

$A_1 \rightarrow A_3$ (2)

…

$A_1 \rightarrow A_k$ (k - 1)

$A_2 \rightarrow A_3$ (k)

$A_3 \rightarrow A_4$ (k + 1)

… …

$A_{k-1} \rightarrow A_k$ (2k - 2)

The set of the above functional dependencies is not minimal (e.g. the functional dependency (2) can be produced from the combination of (1) and (k)). If we remove the functional dependencies (2) … (k - 1), then the resulting set of functional dependencies is minimal since:

1. all the right parts of the rules are single attributes

2. we obviously cannot remove any functional dependency from this set. Suppose that we remove $A_i \rightarrow A_j$. Then the original functional dependency $A_1 \rightarrow A_j$ cannot be derived from the new set of functional dependencies.
3. we cannot replace any functional dependency $X \rightarrow Y$ with another one $Z \rightarrow Y$, $Z \subset X$, since all left parts are single attributes

Based on this set of functional dependencies we normalize the *dimension table* R, to a set of 3NF relations. The algorithm is classic and straightforward [EN89] and gives us the set of the following tables:

$R_1(A_1, A_2)$, $R_2(A_2, A_3)$, …, $R_{k-1}(A_{k-1}, A_k)$

which are the tables representing dimension path D_p in a *snowflake* schema.

The basic advantage of this approach is that the lossless join property is retained and consequently the original table R can be reconstructed from the normalized tables as follows:

$R = R_1 \bowtie_{A2 = A2} R_2 \bowtie_{A3 = A3} R_3 \bowtie \ldots \bowtie_{Ak-1 = Ak-1} R_{k-1}$

As a consequence, we can map a cube to a snowflake schema, by mapping it to its respective star schema and normalizing this star schema. We can also perform the cube operations when the underlying schema is a snowflake one, since:

- the operations use the star schema dimension tables, which can be considered to be views on the snowflake schema
- all operations are performed on the *cube_table* and the *base_cube_table*; hence, dimension tables are not affected from them.