Hierarchies and Relative Operators in the OLAP Environment

Elaheh Pourabbas, Maurizio Rafanelli

Istituto di Analisi dei Sistemi ed Informatica - CNR, Viale Manzoni 30, 00185 Roma, Italy
e-mail: {pourabbas, rafanelli}@iasi.rm.cnr.it

Abstract
In the last few years, numerous proposals for modelling and querying Multidimensional Databases (MDDB) are proposed. A rigorous classification of the different types of hierarchies is still an open problem. In this paper we propose and discuss some different types of hierarchies within a single dimension of a cube. These hierarchies divide in different levels of aggregation a single dimension. Depending on them, we discuss the characterization of some OLAP operators that refer to hierarchies in order to maintain the data cube consistency. Moreover, we propose a set of operators for changing the hierarchy structure. The issues discussed provide modelling flexibility during the scheme design phase and correct data analysis.

1. Introduction

Recently, the business strategies have given a strong relevance to new research areas, such as Data Warehousing and On-Line-Analytical-Processing, OLAP [2]. The OLAP concept was proposed for rendering very large, historical (statistical) databases in multidimensional perspectives, and it is oriented to decision making for business users. The connection between analyzing business data and socio-economic data (generally known as statistical data) is not obvious, but both of them deal with multidimensional data sets, and both are concerned with statistical summarizations over the dimensions of the data sets. Similarities and differences between OLAP and Statistical databases are presented in [13]. The concept of multidimensionality (or n-dimensionality) of these datasets, and in particular, of aggregate data [12], as well as the concepts of dimension (often called category attribute, descriptive variable, character, etc.) and of measure (often called summary attribute, quantitative data, variable, etc.) have been already discussed [9, 12]. Recently, in literature, many authors proposed multidimensional data models and query languages. Gray et al. in [3] proposed the data cube operator as extension to SQL which generalized the histogram, cross-tabulation, roll-up, drill-down, and sub-total constructs found in most report writers.
In [7] the authors formalized a multidimensional data model for OLAP, and developed an algebra query language called *Grouping Algebra.* The relative multidimensional cube algebra is proposed in order to facilitate the data derivation. Gyssens et al. in [4] presented a tabular database model and discussed a tabular algebra as a language for querying and restructuring tabular data. Lehner in [5] discussed the design problem that arose when the OLAP scenarios became very large and they proposed a nested multidimensional data model useful during schema designing and multidimensional data analysis phases.
In literature, multidimensional data are characterized by having two different types of attributes:

(a) one or more measured data, each representing the result of the application of an aggregation function on raw data. Their numerical values are called *measures*;
(b) a set of *dimensions*, which provide a qualitative description of the measured data and are also called *metadata*, i.e. data about data.

Since most proposed models have such constraints as "dimensions are linguistic categories corresponding to different ways of looking at the information", then each dimension is a simple concept hierarchy. A different treatment is proposed in [1] where the authors proposed a data model that provides support for multiple hierarchies along each dimension and for ad hoc aggregates, as well as a few algebraic operators. In this paper, we deal more about multiple hierarchies, and we introduce the multiplicity of a hierarchy as a semantic variant of the simple one. Sometimes, dimensions are organized in *hierarchies* in which there are different aggregate levels [11]. Some design and computation problems can arise when the mapping between different aggregate levels of a hierarchy is not complete. We distinguish this type of hierarchy from that of where the mapping between dimension levels is complete, and then accordingly, we introduce the concepts of the partial and total classification hierarchies. An OLAP system concerns mostly simple data cube, i. e., it is a simple structure to collect in a single "scheme" all the multidimensional aggregate and non-aggregate data relative to a defined event (e.g., Sales, and so on). The values in each cell of this data cube are some "measures" of interest.
In this context, through some examples, we will discuss different types of operations using the well known OLAP operators, and we propose their specialization to solve some problems which arise in particular situations.
The paper is structured as following: Section 2 gives an overview on basic concepts. Section 3 discusses the different types of hierarchies of a cube. Section 4 introduces the characterization of some OLAP operators on hierarchies. Section 5 gives a set of operators that refer to changing the hierarchy structure. Finally, Section 6 concludes.

2. An overview on basic concepts

In literature, different sets of basic concepts and operators were proposed. In this paper, we will refer to the multidimensional data structure and to a set of minimal basic operators described in the following.
A *Cube* is "a group of data cells arranged by the dimensions of the data" [8]. It represents a logical view of multidimensional data.
A *dimension* is "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" [8]. The set of the cube dimensions represents the relative data multidimensionality.
A *hierarchy* is a set of variables which represent different levels of aggregation of the same dimension and which are linked

between them by a mapping. A typical example of hierarchy is City → State → Region → Country.
A *measure* is a particular dimension of the cube [1], which represents the extensional fashion of the phenomenon described by the cube, and which is, in general, a numeric value. Assigning a value to each dimension of a cube, the measure is obtained by a *mapping* from this assignment. In Figure 1 these concepts are graphically represented.

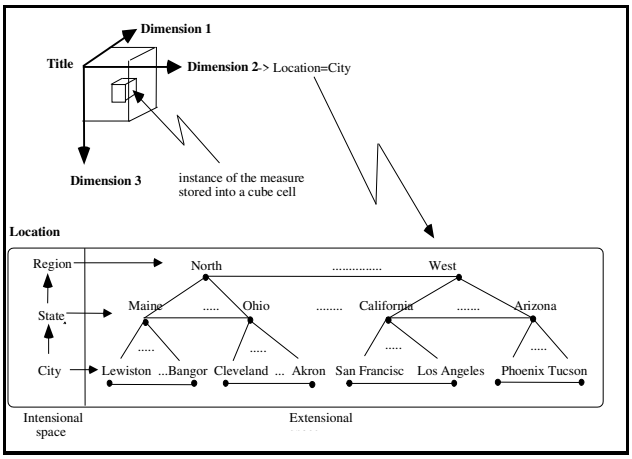

Figure 1. Example of a data cube with hierarchy.

In a multidimensional database different cubes are stored, each of which is defined with different dimensions. The domains of these dimensions consist of a set of values (instances). We define *primitive domain* of a variable the set of all the possible values that this variable can assume in the database. This means that every variable of a hierarchy has its own domain whose values are a subset of, or coincide with the values of its primitive domain. The OLAP operators defined in literature [1, 2, 3, 8] and considered in this paper are roll-up, drill-down, push, pull, slice, dice, and select. We briefly describe them in the following.
The *roll-up* operator decreases the detail of the measure, aggregating it along the dimension hierarchy [1]. It is equivalent to the classification operator of the statistical database operators [10]. "Roll-up involves computing all of the formula-based relationships of data for one or more dimension". Note that because in this paper, we consider only data obtained from count and sum function application, then this computation is a sum.

Example 2.1 Consider the hierarchy of Figure 1. The roll-up operation allows to change level from City to State, recomputing the values of the measure. ❑

We would point out that we consider a city as a municipality area. It means that each state consists of a set of municipality (form territorial, population, etc., point of view)-
The *drill-down* operator is a binary operator [1] which considers the aggregate cube joined with the cube that has more detailed information and increases the detail of the measure going to the lower level of the dimension hierarchy.

Example 2.2 Consider the hierarchy of Figure 1. The drill-down operation allows to pass from State to City, retrieving the values of the measure which were previously stored in the same cube. ❑

The *push* operator is used to convert a dimension into the relative measure in order to manipulate it or to consider it as new measure. Combining with the Pull operator, it can exchange measure and dimension and, then, allows to treat uniformly both of them.

Example 2.3 Consider the cube of Figure 1. Suppose that the phenomenon considered in it is "Cars sales" and that the measure values represent the cars sold in USA by City, Vendor (dimension 1), and Year (dimension 3). By this operation we can push, for example, Vendor instances into the cells of the cube, so that in them we will find a couple of values (in our case, for example, we will find <Smith, 2,738>, ... , etc.). It is different from the operator defined in [1]. In fact, we delete the dimension 1 (Vendor) of the cube, while in [1] it is maintained also as dimension, that is, it is duplicated into the measure of the cube.

❑

The *pull* operator is the converse of the previous one. It creates a new dimension converting the element, specified from it, which is in the measure.

Example 2.4 Let us consider the database described in the previous example. By this operator we can extract the element of the measure specified in the operation (for example, "Cars sales") transforming it in a dimension of the cube. The result is a new cube where the dimension 1 becomes "Cars sales" and the measure becomes "Vendor". ❑

The *slice* (or Destroy Dimension [1]) operator deletes one dimension of the cube, so that the sub-cube derived from all the remaining dimensions is the slice result that is specified. It is equivalent to the *summarization* operator of the statistical database operators [10].

Example 2.5 Let us consider the cube of Figure 1 with an additional dimension "Model". This operation allows to cut one specified dimension, recomputing all the values of the new measure in each cell of the cube. For example, *slice Model* deletes this dimension from the cube and recomputes the values of the measure in the single cell of the resulted cube that becomes, in this case, a bidimensional table. ❑

The *dice* (or Restriction [1]) operator restricts the dimension value domain of the cube removing from this domain those values of the dimension that are specified in the condition (predicate) expressed in the operation. It is equivalent to the *restriction* operator of the statistical database operators [10].

Example 2.6 Let us consider the cube of Figure 1 and suppose that the Year domain is <1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998>. This operation allows to cut the part of the domain instances of one dimension of this cube which are specified in the operation. For example, *dice Year = <1990, 1991, 1992* > carries out the removing of the above mentioned instances from the domain of the dimension Year, restricting it to the remaining values (1993, … ,1998). ❑

The *select* operator is the dual of the dice operator. It carries out the restriction operation removing from this domain those values of dimension that do not satisfy the condition (predicate) expressed in the operation.

Example 2.7 Let us consider the Example 2.6. This operation restricts the dimension value domain of the cube maintaining in this domain those values of the dimension that are specified in the condition expressed in the operation. For example, *select Year =<1990, 1991, 1992*> carries out the selection of these values into the domain, so that the new domain values consists of exactly these values. ❑

3. Characterization of hierarchies

Hierarchy is fundamental to data warehouse and OLAP environment. A hierarchy is an effective form of knowledge representation for encoding prior domain knowledge relevant to data cube. In a simple form, a hierarchy shows the relationships between domains of values. Each operation on hierarchy can be regarded as a mapping from one domain to a smaller domain.
In OLAP environment, hierarchies are used to conceptualize the process of generalizing data as a transformation of values from one domain to values of another domain by means of drill-down/roll-up operators.
In this section, we discuss the hierarchies from two different perspectives: mapping between domain values, i. e., leading to consider total and partial classification hierarchies and hierarchy structure. The later case, treats multiple and multiplicity of hierarchies.

3.1. Classification hierarchies

Dimensions have often been associated with different hierarchically organized levels. These levels correspond to different granularities of viewing data. The name of each level is expressed by the corresponding *variable* name. Generally, the shift from a lower (more detailed) level to a higher (more aggregate) level is carried out by a mapping. A mapping between two variables can be complete or incomplete. In the first case the hierarchy is called *total classification* hierarchy, and in the second case it is called *partial classification* hierarchy. We give the following definitions:

Definition 1 A mapping between two variables of a hierarchy defines a *containment function* if each variable instance of a lower level corresponds to only one variable instance of a higher level and each variable instance of a higher level corresponds to at least one variable instance of a lower level. In such case, this mapping is called *full mapping*.

Definition 2 A *total classification hierarchy* on a given dimension is a hierarchy in which between each adjacent couple of variables there is a full mapping.

The containment function respects the summarizability conditions (disjointness and completeness) of multidimensional databases described in [6] and in [11]. As known in literature, a hierarchy is intensionally represented by a partial ordered set. Then, a total classification hierarchy is any subset that defines a total order.

Example 3.1 Let us consider a nation-wide drink company that owns chain stores located in all cities. Assume that all stores in the chain sell the same beverages. Sales data are collected yearly, i. e., at the end of each year, each member store reports the total sales amount of each drink to the regional headquarters. Figure 2 shows part of the data reported in 1997 and 1998. ❑

The hierarchies along the dimensions Location, and Beverages are represented below both in intensional level and extensional level.
As shown in Figure 3, for a domain value of a level on location dimension all domain values of the lower level are defined, i. e., it is a total classification hierarchy. This is completely in accordance with the hypothesis made in Example 1, where in all cities of the given country such a drink store is located.

Sales						
	Class	*City*	*Vendor*	*Year*	1997	1998
	Alcoholic	Los Angles	Smith		10000	12000
		New York	Wong		20000	16000
		Washington	Mc Donald		23000	17000
		Atlanta	Laurent		50000	60000
		…	…	....	…	…
		Dallas	Backer		21000	32000
		Detroit	Clifford		90000	18000
	Non alcoholic	Los Angles	Smith		20900	14500
		New York	Wong		12300	32009
		Washington	Mc Donald		87000	23890
		Atlanta	Laurent		23100	49000
		.....	.....		.....	.....
		Dallas	Backer		56000	34500
		Detroit	Clifford		21000	30000

Figure 2. Example of a data cube

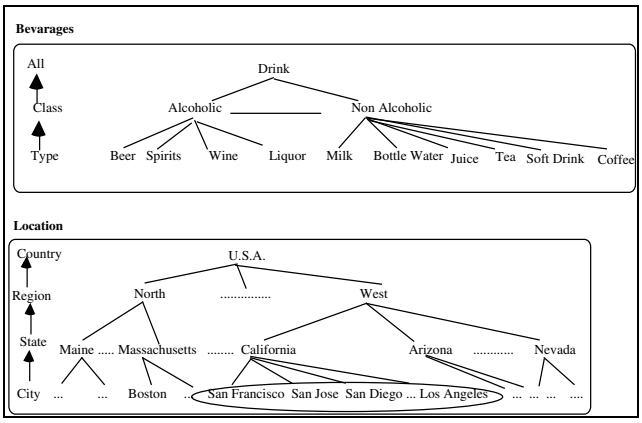

Figure 3. The hierarchies along the dimensions Beverages, and Location (at left) and the relative domain value (at right)

Definition 3 A *partial classification hierarchy* on a given dimension is a hierarchy in which between at least one adjacent couple of variables there is no full mapping.

Example 3.2 Consider the chain store example we gave in Example 3.1. Suppose that the chain stores of the above mentioned company in the state of California are located only in some of cities of this state (see Figure 4).

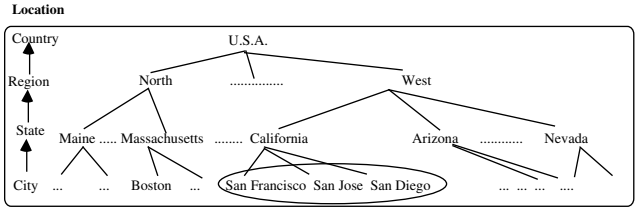

Figure 4. Domain values of the *City* level

Then, the domain value of *City* level along the *Location* dimension is restricted with respect to that shown in Figure 3. Accordingly, these cities are not listed in the table of *Drink sales*. ❑

These two types of hierarchies will influence the result of queries for which the summarization operations will be needed. Details of this fact are discussed in a further section.
Note that, in this paper, we consider only hierarchies in which no overlapping exists among domain instances of each variable.

3.2. Multiplicity of a hierarchy and multiple hierarchies

One of the more important problems regarding the hierarchies refers to their definition. In this section we propose a set of definitions in order to fix a reference point in their study.

First of all, we distinguish between multiplicity of a hierarchy and multiple hierarchy.

Definition 4 Let H and H_1 be two hierarchies. H_1 is a *multiplicity* of H if its level domains are the same as the H level domains and the variable name associated to each level of H_1 is a specialization of the variable name associated to the corresponding levels of H.

Example 3.3 Let us consider a location hierarchy defined as: City → Province → Region. A possible multiplicity of this hierarchy is City of residence → Province of residence → Region of residence. ❑

Definition 5 Let H_1, H_2, ... , H_n be a set of hierarchies. This set forms a *multiple hierarchy* if each of them has at least one variable in common with another hierarchy of the same set.

Example 3.4 Let us suppose that we have four hierarchies, labelled (a), (b), (c), and (d), as illustrated in Figure 5. The hierarchy labeled (d) is a multiple hierarchy, where the level *Province* is the same for (a) and (b) and the level *Region* is the same for (a) and (c). ❑

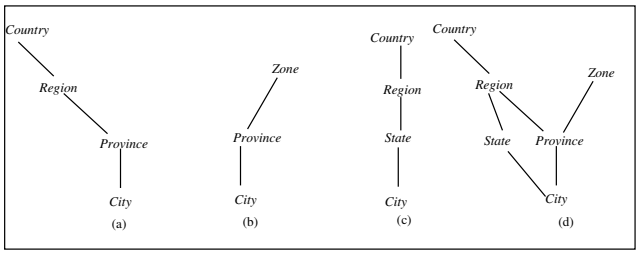

Figure 5. Example of a multiple hierarchy

Definition 6 Let H be a hierarchy. The hierarchy obtained from deleting one or more non terminal variable or level of H is a *derived hierarchy*.

Example 3.5 From the multiple hierarchy (d) shown in Figure 4, we obtain the following derived hierarchies: *City* → *Province* → *Country, City* → *Region* → *Country, City* → *Country, City* → *Zone, City* → *Region* → *Country, City* → *Province* → *Country, and City* → *State* → *Country*. ❑

Specifically, in the case of partial classification hierarchies, the variable instances of derived hierarchies are the instances of variables that are adjacent to the instances of deleted levels and between which a connected path can be defined. For example, in Figure 6-(b) are reported the variable instances of derived hierarchies obtained from variable instances of the partial classification hierarchy illustrated in Figure 6-(a) that satisfy the above mentioned condition.

4. Characterization of OLAP operators on hierarchies

Recently different authors proposed a set of OLAP operators, which are defined on data cube and which produce as output a new cube [1, 2, 9]. In this section, we discuss the operators involved in manipulating dimensions with hierarchies in order to introduce some important modifications and specializations.

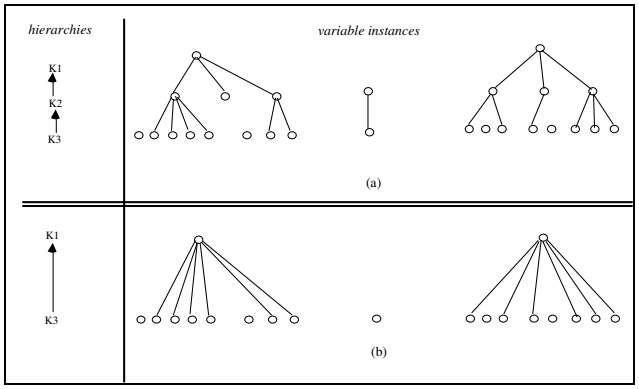

Figure 6. Example of path generation between levels

4.1. Case of the Roll-up operator

As mentioned above, this operator decreases the detail of the measure, aggregating it along the dimension hierarchy. A problem arises when a variable relative to a level of the hierarchy is not complete (i.e., case of partial classification hierarchy).
In the following we consider what happens when this operator is applied to a total classification hierarchy and, then, to a partial classification hierarchy.

Example 4.1 Let us consider the data cube represented in Figure 2. In Figure 7, an its "multidimensional" view is illustrated. ❑

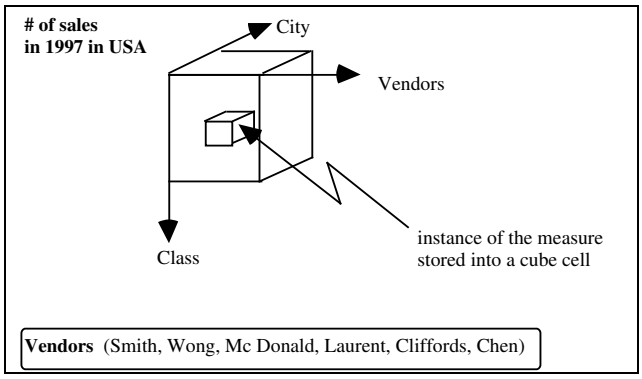

Figure 7. Multidimensional view of Drink Sales data cube

Let us suppose to formulate a query defined as below:

"Select Vendors for which the total Sales is >10000 units in each State of the West"

This query is solved in the following way:

Roll-up from City to Region, *Select* Region = West, *Drill-down* from Region to State, *Push* Vendors, *Pull* # of sales, *Dice* # of sales "≤10000". ❑

Note that in some cells of the resulted cube (see Figure 8) null values can appear. This demonstrates that some instances of "Vendors" are not defined.
Let us suppose, now, that the classification hierarchy relative to "City → State → Region" has, as domains the cities of California, with only the instances "San Francisco, San Jose, San Diego". This one is a subset of the primitive domain of City, in which all the cities of California (San Francisco, San Jose, San Diego, Los Angeles, etc.) are stored.

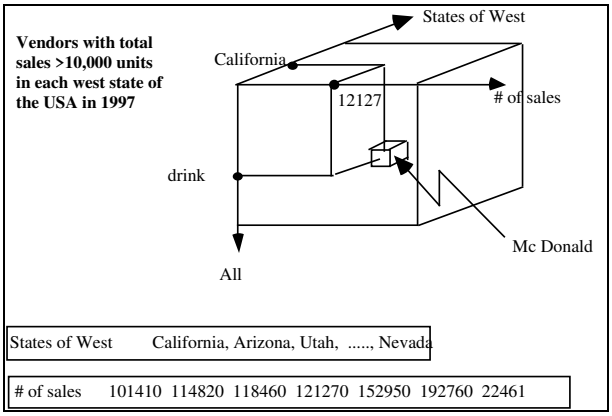

Figure 8. The result of the query

In particular, when the operator *Roll-up* from City to Region is applied, no information is stored about the non completeness of the domain of City relative to California. This means that for California the number of vendors for which the total Drinks sold in 1990 is >10,000 units refers only to the cities of San Francisco, San Jose, and San Diego and not to all the cities of California. Then, since this information is not specified anywhere, the answer for this state is wrong.
A solution to that is to save the information about the domain values that cause the non-completenesses of the hierarchy. This can be obtained in two different ways. The former consists of adding a *Note* (the clause *where <variable name> is-a subset of the primitive domain*) to the title of the cube. In the case of Figure 8, the title becomes "Vendors with total Drinks sales >10,000 units in each state of the West in 1997 in USA, *where city of California is-a subset of the primitive domain*". The later consists of adding the same *Note* to each variable of the hierarchy whose level is higher with respect to the level of the variable with the incomplete domain. In the same Figure 8, we have to add the clause *where city of California is-a subset of the primitive domain* to the variables State, Region, and Country.

4.2. Case of the Slice operator

As mentioned above, the *slice* operator reduces the dimensionality (or cardinality) of a cube eliminating one dimension through its multidimensional space. This fact is not always true because, if we delete a dimension whose domain is a subset of the primitive domain, we lose information and the resulting cube of this operation contains incorrect data.
Before discussing this situation, we need to introduce the *implicit dimension* definition.

> **Definition 7** We call *implicit dimension* any dimension of a data cube which has only one instance in its definition domain. This instance can be one value or multi-valued.

Example 4.2 Let us consider the cube of Figure 2, where the primitive domain of the dimension Year assumes the values <1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998>. This means that they are all the possible values that this dimension can assume in the database. Instead, the value domain of Year in the considered cube is <97, 98>.
Let us suppose that the following query is carried out:
"Give me the drink sales in all Cities by Class and Vendor"
It is solved in the following way:
Slice Year.

In this case if the slice operator deletes the dimension Year, the result seems to refer to the whole primitive domain of Year. This means that we lose the exact information on the real period to which the result should refer. To overcome this mistake we introduce a specialization of the *Slice* operator, called *Partial Slice* (or *P-Slice*) and defined below. ❑

> **Definition 8** The *P-Slice* operator removes the dimension on which it is applied transforming it in an implicit dimension. The only value of the implicit dimension domain is the set valued of all the values that formed the domain of the removed dimension.

Similarly to the solution proposed for the Roll-up operator, the same *Note* is added to the title of the cube.
According to this definition, the above query is now solved in the following way:
P-Slice Year.

The result is now a cube with the same dimensions of the primary one, where the title becomes "Drink sales by Class, Vendor, and Year *where Year is a subset of the primitive domain*".
For symmetric reasoning of terminology we use the term *Total Slice* (or *T-Slice*) for the well known *Slice* operator
If, instead, the Year domain in the considered cube coincided with its primitive domain, then the previous query would be solved in the following way:
T-Slice Year.

The cardinality of the resulting cube is now decreased of one dimension, since, by removing Year no information is lost.

5. An enlargement of the operator set referring to hierarchies

In this section we propose a set of operators able to change the primary configuration of a hierarchy extending or reducing its level number, adding a new multiplicity, and creating a multiple hierarchy. For formalizing them, let us consider l_1 and l_2 be two adjacent levels of a given hierarchy defined as $l_1 \rightarrow l_2$. Let us discuss them.

5.1 Insert level

The *Insert level* operator allows to add a new level to a hierarchy, (giving the variable name, the domain instances, and the relationships between this level and, respectively, the higher and the lower adjacent levels in the hierarchy). The insertion of a new level denoted by l_i between the above mentioned levels is represented through the symbol $Insertlevel\ {}^{l_2}_{l_1} l_i\,(I_{i,1},\ldots,I_{i,n};R_i)$, where $I_{i,1},\ldots,I_{i,n}$ represents the inserted level domain instances and

$$R_i = (I_{l_2,1}(I_{l_i,1}(I_{l_1,1},\ldots,I_{l_1,q_1}),\ldots,I_{l_i,p_1}(I_{l_1,q_{1+1}},\ldots,I_{l_1,q_2})),$$
$$\ldots,I_{l_i,q_k=h}(I_{l_1,q_{p_{h-1}+1}},\ldots,I_{l_1,q_n=n})))$$

where the instances of levels l_2, l_i, l_1 are divided in respectively, k, h, and n subsets and p_j, q_v represent a generic set of l_i, and l_1 levels instances where $j=1,\ldots,h$ and $v=1,\ldots,n$.

Example 5.1 Let us consider the Location dimension shown in Figure 2. Let us suppose to insert the variable *County* between the variables *City* and *State*. We have to define its domain values, as well as the mapping relationship between *City* and *County*, and between *County* and *State* (see Figure 9). This is obtained by the following formula:

$Insertlevel\ {}^{State}_{City}\ County\ (Green, Orange, \ldots; R_{County})$ where

$R_{County} = (\ldots, (California\ (Green\ (San\ Francisco, \ldots, Richmond), \ldots, Orange\ (Los\ Angels, \ldots, Oxnard))), \ldots)$ ❑

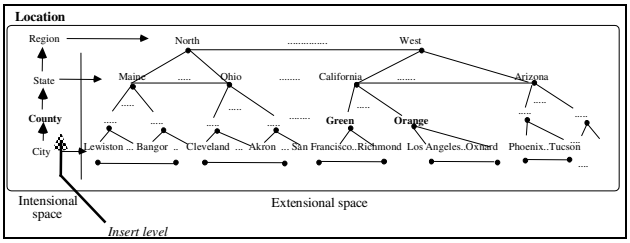

Figure 9. Example of Insert level operator

5.2 Delete level

The *Delete level* operator redefines a hierarchy as a subset of the existing one, deleting a variable with its relative domain. This operation re-creates the mapping relationships between the two levels (higher and the lower) adjacent to the deleted variable. The deletion of a level l_i is denoted by $Deletelevel\ {}^{l_2}_{l_1} l_i(I_{i,1}, \ldots, I_{i,n})$ where l_1 and l_2 are, respectively, the relative lower and higher level in the given hierarchy. This is the converse operation of the insert level operation. Note that, in this case the relationships between level are defined automatically by the system.

5.3. Add multiplicity

The *Add multiplicity* operator duplicates a given hierarchy, changing the name of the variables in order to specialize them, as already discussed in section 3.2. Let H be a hierarchy. This operator is represented through

$Addmultiplicity\ H\ (additionalname)$ where *additionalname* is a string to be added to each variable names of the hierarchy in order to specialzing the last one.

Example 5.2 Let us consider the example 3.3. The hierarchy City of residence → Province of residence → Region of residence is obtained by

$Addmultiplicity\ Location\ (of\ residence)$ ❑

5.3. Add level

The *Add level* operator creates a new hierarchy starting from a given hierarchy. It creates a new variable and its relative domain, and defines the mapping relationship between this and the variable of the starting hierarchy. These two hierarchies define, in this way, a multiple hierarchy. Let l_n be a level to be added to a hierarchy defined as $l_1 \to l_2 \to \ldots \to l_{n-1}$. The *Add level* operator is represented through

$$Addlevel_{l_1 \to l_2 \to \ldots \to l_{n-1}}\ l_n(I_{i1}, \ldots, I_{in}, R_i)$$

gives as result a hierarchy $l_1 \to l_2 \to \ldots \to l_{n-1} \to l_n$.

Example 5.3 An example of multiple hierarchy has been already shown in Figure 5-(d). ❑

6. Conclusions

In this paper, after a brief overview of basic concepts relative to the multidimensional data structures and to a set of OLAP operators, which are object of the discussions that followed, a characteriazation of hierarchy is proposed. We distinguished between hierarchies in which the mapping between different levels is full and hierarchies in which this condition is not satisfied. Then, we defined the concepts of multiplicity of a given hierarchy and of multiple hierarchies.

Based on the definitions and concepts proposed in this paper, we discussed the characterization of the OLAP operators involved in the hierarchy manipulation. In particular, depending on the type of hierarchy, we studied the different behaviour of the Roll-up and the drill-down operators in order to keep the consistency of data that is the result of the queries. We also characterize the slice operator, defining the implicit dimension concept and specializing the operator in two different types: P-slice and T-slice. Finally, we proposed an enlargement of the operator set, specific for the hierarchy manipulation, which are the Insert level, the Delete level, the Add multiplicity, and the Add level operators. For each situation discussed, clarifying examples are given.

References

[1] Agrawal R., Gupta A., Sarawagi S., Modelling Multidimensional Databases, in 13th International Conference on Data Engineering, (ICDE'97, Birmingham, U. K., April 7-11, 1997), 232-243.

[2] Codd E.F., Codd S.B., Salley C.T., Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical report, 1993.

[3] Gray J., Bosworth A., Layman A., Pirahesh H., Data cube: a relational aggregation operator generalizing group-by, cross-tabs and subtotals. in 12th IEEE International Conference on Data Engineering (ICDE'96, New Orleans, Louisiana, Feb. 26-Mar 1, 1995), 152-159.

[4] Gyssens M., Lakshmanan L.V.S., Subramanian I.N., Tables as a paradigm for querying and restructuring, ACM-PODS 1996, Montreal, Canada, 93-103.

[5] Lehner W., Modeling large scale OLAP scenarios, Advances in Database Technology-EDBT'98, Valencia, Spain, Lecture Notes in Computer Science, Springer Verlag Pub., Vol. 1377, 53-167.

[6] Lenz H-J., Shoshani A., Summarizzability in OLAP and statistical data bases, in 9th International Conference on Scientific and Statistical Data Management (SSDBM'97), 1997.

[7] Li C., Wang X.S., A data model for supporting on-line analytical processing, in Proceedings of Conference on Information and Knowledge Management, November, 1996, 81-88.

[8] OLAP Council, The OLAP glossary. http://www.olapcouncil.org. The OLAP Council 1997.

[9] Rafanelli M., Ricci F. L., Proposal of a logical model for statistical database, Proceedings 2nd International. Workshop on SDBM'83, Los Altos, CA, 264-272.

[10] Rafanelli M., Ricci F.L., Mefisto: a functional model for statistical entities, IEEE Transactions on Knowledge and Data Engineering, August 1993, Vol.5, No.4.

[11] Rafanelli M., Shoshani A., STORM: A STatistical Object Representation Model, Proceedings of 5th Intern. Confer. on SSDBM, Charlotte, NC, April 1990, Lecture Notes in Computer Science, Springer Verlag Pub., Vol. 420.

[12] Shoshani A., Wong H.K.T., Statistical and Scientific Database Issues" IEEE Transactions on Software Engineering, October 1985, Vol.SD-11, No.10.

[13] Shoshani A., OLAP and Statistical Databases: Similarities and Differences, in Proceedings of ACM PODS '97, 1997, 185-196.