On the Need of a Reference Algebra for OLAP

Oscar Romero and Alberto Abelló

Universitat Politècnica de Catalunya

Abstract. Although multidimensionality has been widely accepted as the best solution to conceptual modeling, there is not such agreement about the set of operators to handle multidimensional data. This paper presents a comparative of the existing multidimensional algebras trying to find a common backbone, as well as it discusses about the necessity of a reference multidimensional algebra and the current state of the art.

1 Introduction

OLAP tools are conceived to exploit the Data Warehouse for analysis tasks based on *multidimensionality*, the main feature of these tools. The multidimensional conceptual view of data is distinguished by the *fact/dimension* dichotomy and it is characterized by representing data as if placed in an n-dimensional space, allowing us to easily understand and analyze data in terms of facts and dimensions showing the different points of view where a subject can be analyzed from.

Lots of efforts have been devoted to multidimensional modeling, and up to now, several models have been introduced in the literature (most of them surveyed in [1] and [2]). However, we can not yet benefit from an standard multidimensional model, and a common framework in which to translate and compare the research efforts in the area is missing. As discussed in [3] and [4], experiences in the field of databases have proved that a common framework to work with is crucial for the evolution of the area: (1) conceptual modeling is vital for the design, evolution and optimization of a data warehouse, whereas (2) a multidimensional algebra is crucial for a satisfactory navigation and analysis (i.e. querying) of data contained in the data warehouse. Specifically, a reference set of operators would help to develop design methodologies oriented to improve querying, better and accurate indexing techniques as well as to facilitate query optimization; issues even more critical than in an operational database, due to the huge amount of data stored in a data warehouse. However, although multidimensionality (i.e. to model in terms of facts and dimensions) has been widely accepted as the best solution to data warehouse modeling, there is no such agreement about the set of operators to handle multidimensional data. To our knowledge, it does not even exist a comparative of algebras in the literature.

Thus, section 3 compares existing multidimensional algebras trying to find a common backbone, whereas section 4 discusses why the relational algebra (used by ROLAP tools) does not *directly* fit to multidimensionality. Due to the lack of a reference model, section 2 presents the multidimensional framework used in this paper. Section 5 discusses about the necessity of a reference multidimensional algebra as well as the current state of the art, and section 6 concludes this paper.

2 Our Framework

Due to the lack of an standard multidimensional model, and hence, the lack of a common notation, we need a reference framework in which to translate and compare the multidimensional algebras presented in the literature. Otherwise, a comparison among all those different algebras would be rather difficult. In this section we introduce a multidimensional data structure and a set of operators (introduced in detail in [5]) used in this paper to concisely and univocally define the multidimensional concepts, as well as to provide a common notation. From here on, these concepts will be **bold faced** for the sake of comprehension.

First, we introduce our framework data structure, where a **Dimension** contains a hierarchy of **Levels** representing different granularities (or levels of detail) to study data, and a **Level** contains **Descriptors**. We differentiate between identifier **Descriptors** (univocally identifying each instance of a **Level**) and non-identifier. On the other hand, a **Fact** contains **Cells** which contain **Measures**. One **Cell** represents those individual **cells** of the same granularity that show data regarding the same **Fact** (i.e. a **Cell** is a "Class" and **cells** are its instances). We call a **Base** to those minimal set of **Levels** identifying univocally a **Cell**, that would result in "primary keys" in ROLAP tools. A set of **cells** placed in the multidimensional space with regard to the **Base** is called a **Cube**; One **Fact** and several **Dimensions** to analyze it give rise to a **Star**.

Next, we present our reference multidimensional operations set:

- **Selection**: By means of a logic clause C over a **Descriptor**, this operation allows to choose the subset of points of interest out of the whole n-dimensional space.
- **Roll-up**: It groups **cells** in the **Cube** based on an aggregation hierarchy. This operation modifies the granularity of data by means of a many-to-one relationship which relates instances of two **Levels** in the same **Dimension**, corresponding to a part-whole relationship. As argued in [6] about **Drill-down** (i.e. the inverse of **Roll-up**), it can only be applied if we previously performed a **Roll-up** and did not lose the correspondences between **cell**s.
- **ChangeBase**: This operation reallocates exactly the same instances of a **Cube** into a new n-dimensional space with exactly the same number of points, by means of a one-to-one relationship. Actually, it allows two different kinds of changes in the space **Base**: we can just rearrange the multidimensional space by reordering the **Levels** or, if exists more than one set of **Dimensions** identifying the **cells** (i.e. alternative **Bases**), by replacing the current **Base** by one of the alternatives.
- **Drill-across**: This operation changes the subject of analysis of the **Cube** by means of a one-to-one relationship. The n-dimensional space remains exactly the same, only the **cells** placed on it change.
- **Projection**: It selects a subset of **Measures** from those shown in the **Cube**.
- **Set Operations**: These operations allow to operate two **Cubes** containing the same **Cells** if both are defined over the same n-dimensional space. We consider **Union**, **Difference** and **Intersection** as the most relevant ones.

Algebra	Operator	Selection	Projection	Roll-up Drill-down	changeBase	Drill-across	Union Difference Intersection	Remarks
[7]	“Add Dimension”				$\checkmark_p$			
	“Transfer”				$\sim$			
	“Cube Aggr.”			$\checkmark$				
	“Rc-join”	$\checkmark$						
	“Union”						$\checkmark$	
[8]	“Push”					$\checkmark_p$		Semantic Rels.
	“Pull”		$\mathcal{D}$		$\checkmark_p$			Semantic Rels.
	“Destroy Dimension”		$\mathcal{D}$		$\checkmark_p$			
	“Restriction”	$\checkmark$						
	“Join”					$\checkmark$		
	“Merge”			$\checkmark$				
[9]	“Selection”	$\checkmark$						
	“Projection”		$\checkmark$					
	“Cartesian Product”					$\sim$		
	“Union/Diff./Inters.”						$\checkmark$	
	“Fold/Unfold”				$\checkmark_p$			
	“Classification”			$\mathcal{D}$				
	“Summarization”			$\mathcal{D}$				
[10]	“Restriction”	$\checkmark$						
	“Metric Projection”		$\checkmark$					
	“Aggregation”			$\checkmark$				
	“Cartesian Product”					$\sim$		
	“Join”					$\checkmark$		
	“Union/Diff.”						$\checkmark$	
	“Extract”				$\checkmark_p$			Semantic Rels.
	“Force”					$\checkmark_p$		Semantic Rels.
[11]	“Slicing”	$\checkmark$						
	“Roll-up/Drill-down”			$\checkmark$				
	“Split/Merge”			$\sim$				
	“Implicit/Explicit Aggr.”			$\checkmark_p$				
	“Cell Operators”							Derived Measures
[12]	“Cartesian Product”					$\sim$		
	“Natural Join”					$\checkmark$		
	“Roll-up”			$\mathcal{D}$				
	“Aggregation”			$\mathcal{D}$				
	“Level Description”				$\checkmark_p$			Semantic Rels.
	“Scalar Function App.”							Derived Measures
	“Selection”	$\checkmark$						
	“Simple Projection”		$\checkmark$		$\checkmark_p$			
	“Abstraction”		$\checkmark_+$		$\checkmark_{p+}$			
[13]	“Restrict”	$\checkmark$						
	“Destroy”				$\checkmark_p$			
	“join”					$\checkmark$		
	“Join”			$\checkmark_+$		$\checkmark_+$		
	“Aggr”			$\checkmark$				
[14]	“Selection”	$\checkmark$						
	“Projection”		$\checkmark$					
	“Union/Diff.”						$\checkmark$	
	“Identity-based Join”					$\sim$		
	“Aggregate Formation”			$\checkmark_p$				
	“Value-based Join”					$\checkmark$		
	“Duplicate Removal”							**Base** definition
	“SQL-like Aggr.”				$\checkmark_p$			
	“Star-join”	$\checkmark_+$		$\checkmark_+$				
	“Roll-up/Drill-down”			$\checkmark$				
[15]	“Navigate”			$\checkmark$				
	“Selection”	$\checkmark$						
	“Split Measure”		$\checkmark$					
[16]	“Derived Measures”							Derived Measures
	“Join”					$\checkmark_p$		
	“Slice/Multislice”	$\checkmark$						
	“Union/Diff./Inters.”						$\checkmark$	
[17]	“Selection Cube”	$\checkmark$						
	“Decoration”				$\checkmark_p$			
	“Fed. Gen. Projection”		$\checkmark_+$	$\checkmark_+$	$\checkmark_+$			

Table 1. Summary of the comparative between multidimensional algebras.

3 The Multidimensional Algebras

This section presents a thorough comparative among the multidimensional algebras presented in the literature. To the best of our knowledge, it is the first comparative about multidimensional algebras carried out. In [2], a survey describing the multidimensional algebras in the literature is presented. However, unlike us, it does not compare them.

Results presented along this section are summarized in table 1. There, rows, representing an algebraic operator, are grouped according to which algebra they belong to (also ordered chronologically), whereas columns represent multidimensional algebraic operators in our framework (notice **Roll-up** and **Drill-down** are considered together since one is the inverse of the other).

The notation used is the following: a $\checkmark$ cell means that those operations represent the same conceptual operator; a $\sim$ stands for operations with similar purpose but different proceeding making them slightly different; a $\checkmark_p$ means that the operation partially performs the same data manipulation than the reference algebra operator despite the last also embraces other functionalities, and a $\checkmark_+$ means that this operation is equal to combine the marked operators of our reference algebra, meaning it is not an atomic operator. Analogously, there are some reference operators that can be mapped to another algebra combining more than one of its operators. This case is showed in the table with a $\mathcal{D}$ (from *derived*). Keep in mind this last mark must be read vertically unlike the rest of marks. Finally, notice we have only considered those operations manipulating data and therefore, those aimed to manipulate the data structure are not included:

[7] introduces a multidimensional algebra as well as its translation to SQL. To do so, they introduce an ad hoc grouping algebra extending the relational one (i.e. with grouping and aggregation operators). Prior to present its operators, notice it was one of the first multidimensional algebras introduced, and their main effort was to construct **Cubes** from local operational databases.

More precisely, it defines five multidimensional operators representing mappings between either **Cubes** or *relations* and **Cubes**. The "Add dimension" and "Transfer" operators are aimed to rearrange the multidimensional space similar to a **changeBase**: while "Add dimension" adds a new analysis **Dimension** to the current **Cube**, "Transfer" *transfers* a **Dimension** *attribute* (i.e. a **Descriptor**) from one **Dimension** to another via a "Cartesian Product". Since multidimensional concepts are directly derived from non-multidimensional relations, **Dimensions** may be rather vaguely defined, justifying the transfer operator; the "Cube Aggregation" operator performs grouping and aggregation over data, being equivalent to a **Roll-up** and finally, the "Rc-join" operator, that allows us to join a *relational* table with a **Dimension** of the **Cube**, **Selecting** the **Dimension** values also present in the table. This low level operator is tightly related to the multidimensional model presented, and it is introduced to relate non-multidimensional relations with relations modeling **Cubes**.

[8] presents an algebra composed by six operators rather relevant, since they inspired many following algebras. First, "Push" and "Pull" transform a **Measure** into a **Dimension** and viceversa, since in their model **Measures** and **Dimen-**

sions are handled uniformly. In our framework they would be equivalent to define semantic relationships between the proper **Dimensions** and **Cells** and then, **Drill-across** and **changeBase** respectively; "Destroy Dimension" drops a **Cube Dimension** rearranging the multidimensional space and hence, being equivalent to a **changeBase**, whereas the "Restriction" operator is equivalent to a **Selection**; "Merge" to a **Roll-up** and "Join" to an unrestricted **Drill-across**. Consequently, the latter can even be performed without common **Dimensions** between two **Cubes**, giving rise to a "Cartesian Product". However, a "Cartesian Product" does not make any multidimensional sense if it is not restricted, since it would not preserve *disjointness* when aggregating data ([18]). Finally, notice we can **Project** data by means of "Pull"ing the **Measure** into a **Dimension** and performing a "Destroy Dimension" over it.

[9] presents an algebra based on the classical algebraic operations. Therefore, it includes "Selection", "Projection", "Union" / "Intersection" / "Difference" and the "Cartesian Product"; all of them being equivalent to their analogous operators in our reference algebra except for the latter; mappable to an unrestricted **Drill-across** as discussed above. The "Fold" and "Unfold" operators add / remove a **Dimension**, like in a **changeBase**; whereas **Roll-up** is decomposed in two operators: "Classification of Tables" (i.e. grouping of data) and "Summarization of Tables" (aggregation of data). Hence, this algebra proposes to differentiate *grouping* (i.e. the conceptual change of **Levels** through a part-whole relationship or in other words, the result of mapping data into groups) from *aggregation* (i.e. aggregating data according to an aggregation function).

[10] and [19] present an algebra with eight operators based on the algebra presented in [8]. Therefore, the "Restriction" operator is equivalent to a **Selection**; the "Metric Projection" to a **Projection**; the "Aggregation" to a **Roll-up** and the "Union" / "Difference" operators to those with the same name in our reference algebra. Moreover, like in [8], **Measures** can be transformed into **Dimensions** and viceversa. Hence, the "Force" and "Extract" operators are equivalent to the "Push" and "Pull" operators. Finally, the "Cubic Product" is equivalent to the "Join" operator in [8]. However, since, in general, a "Cartesian Product" does not make multidimensional sense, they also remark the specific case of a "Cubic Product" over two **Cubes** with common **Dimensions** (i.e. preserving disjointness if joined through their shared **Dimensions**). They call "Join" to this restricted "Cubic Product".

[11] presents an algebra composed by five operators. "Slicing" reduces the multidimensional space in the same sense than **Selection**, whereas "Roll-up" and "Drill-down" and the "Split" and "Merge" operators are equivalent to **Roll-up** and **Drill-down**. Despite they represent the same conceptual operators, its model data structure, that differentiates two analysis phases of data, justifies them: while "Roll-up" and "Drill-down" find and interesting context in a first phase, "Split" and "Merge" modify the data granularity *dynamically* along the "dimensional attributes" (non-identifiers **Descriptors**) defined in the "classification hierarchies" nodes of the data structure. It also introduces two operators to aggregate data: the "Implicit" and the "Explicit" aggregation. The first one is

implicitly used when navigating by means of “Roll-up”s, whereas the second one can be *explicitly* stated by the end-user. Since they are equivalent, these operators are just differentiated because of the conceptual presentation followed in the paper. Finally, the “Cell-oriented operator” derives new data preserving the same multidimensional space by means of “unary operators” (-, *abs* and *sign*) or “binary operators” (***, *+*, -, */*, *min* and *max*). “Binary operators” ask for two multidimensional objects aligned (i.e. over the same multidimensional space). In our framework it is obtained defining **Derived Measures** in design time.

[20], [21] and [12] present an algebra with nine operators. Similar to [9], **Roll-up** is decomposed into “Roll-up” (i.e. grouping) and “Aggregation”; “Level description” is equivalent to an specific **changeBase**: it changes a **Level** by another one related through a one-to-one relation to it. In our framework we should define a semantic relationship among **Levels** involved and perform a **changeBase**; “Simple projection” projects out selected **Measures** and reduce the multidimensional space dropping **Dimensions**: it can just drop **Measures** (equivalent to a **Projection**), **Dimensions** (to a **changeBase**) or combine both. Finally, “Abstraction” is equivalent to the “Pull” operator in [8] and “Selection”, “Cartesian Product” and “Natural Join” to those already discussed along this section.

[13] presents a Description Logics based algebra developed from that presented in [8]. Therefore, it also introduces the “Restrict” operator; the “Destroy” one equivalent to the “Destroy Dimension” and the “Aggr” operator equivalent to a “Merge”. Furthermore, the “join” and “Join” operators can be considered an extension of the “Join” operator in [8]: both operators restrict it to make multidimensional sense and consequently, being equivalent to a **Drill-across**; despite the second one also allows to group and aggregate data before showing it (i.e. being equivalent to a **Drill-across** plus a **Roll-up**).

[14] presents an algebra where “Selection”, “Projection”, “Union” / “Difference” and **Roll-up** and **Drill-down** are equivalent to those with the same name presented in our framework, whereas the “Value-based join” is equivalent to a **Drill-across** and the “Identity-based join” to a “Cartesian product”. Moreover, it also differentiates the “Aggregate operation” from the “Roll-up” (i.e. grouping); the “Duplicate Removal” operator is aimed to remove **cells** characterized by the same combination of dimensional values. In our framework it can never happen because of the **Base** definition introduced. Finally, it presents a set of non-atomic operators; the “star-join” operator combines a **Selection** with a **Roll-up**, by the same aggregation function, over a set of **Dimensions**, and the “SQL-like aggregation” applies the “Aggregate operation” to a certain **Dimensions** and projects out the rest (that is, performs a **changeBase**).

[15] presents an algebra with three operators focusing on the most common multidimensional operators: “Navigation” allows us to **Roll-up**, and according to [22] it is performed by means of “Level-Climbing” -reducing the granularity of data-, “Packing” -grouping data- and “Function Application” -aggregating by means of an aggregation function-. Finally, “Split a Measure” is equivalent to a **Projection** and a “Selection” to the reference **Selection**.

[17] presents an algebra over an XML and OLAP federation: "Selection Cube" allows us to **Select** data, while "Decoration" adds new **Dimension**s to the **Cube** (i.e. mappable to a **changeBase**) and a "Federation Generalized Projection" **Roll-ups** the **Cube** and removes unspecified **Dimensions** (**changeBase**) and **Measures** (**Projection**). Notice despite **Roll-up** is mandatory, this operator can combine it with a **Projection** or/and a **changeBase**.

An algebra with four operations is presented in [16]. "Slice" and "Multislice" **Select** a single or a range of dimensional values; "Union" / "Intersection" / "Difference" combine two aligned **Cubes** according to their semantics, whereas "Join" is rather close to **Drill-across** but in a more restrictive way, forcing both **Cube**s to share the same multidimensional space. Finally, "Derived Measures" derives new **Measures** from already existent. In our framework, as already said, derived **Measures** should be defined in design time. Finally, notice that **Roll-up** is not included in their set of operators, since it is considered in their model data structure.

Finally, to conclude our comparative, we would like to remark that some of these approaches have also presented an equivalent calculus besides the algebra introduced above (like [9] and [12]). Moreover, [23] presents a query language to define the expected workload for the Data Warehouse. We have not included it in table 1 since it can not be smoothly compared to algebraic operators one per one. Anyway, analyzing it, we can deduce many of our reference algebraic operators are also supported by their model like **Selection**, **Projection**, **Roll-up**, **Union** and even a partial **Drill-across** as they allow to overlap *fact schemes*.

4 Multidimensional Algebra Vs. Relational Algebra

In our study, we also need to place the relational algebra in our framework since, nowadays, ROLAP tools are the most widely spread approach to model multidimensionality and therefore, multidimensional queries are being translated to SQL and (eventually) to the relational algebra.

This section aims to justify the necessity of a semantic layer (the multidimensional algebra) on the top of a RDBMS (i.e. the relational algebra). Despite we believe ROLAP tools are a good choice to implement multidimensionality, we present, by means of a conceptual comparative between the multidimensional and the relational algebra operators, why the relational algebra (and therefore SQL) does not directly fit properly to multidimensionality. Furthermore, we emphasize in those restrictions and considerations needed to be made over the relational algebra with regard to multidimensionality.

In this comparative we consider the relational algebra presented in [24]. Thus, we consider "Selection" (σ), "Projection" (π), "Union" ($\cup$), "Difference" ($-$) and "Natural Join" ($\bowtie$) as the relational algebra operators. We talk about "Natural Join", or simply "Join", instead of the "Cartesian Product" (the one presented in [24] and where "Join" can be derived from) since a "Cartesian Product" without restrictions is meaningless in the multidimensional model, as discussed in [18].

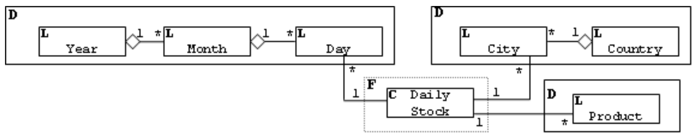

Fig. 1. Schema of a multidimensional Cube

For the sake of comprehension, since we focus on a conceptual comparison, and to avoid messing results with considerations about the Data Warehouse implementation, we can consider, without loss of generality, that each multidimensional **Cube** is implemented as a single relation (i.e. a denormalized relational table). So that, considering the **Cube** depicted in figure 1 we would get the following relation: $\{\underline{\texttt{City}, \texttt{Day}, \texttt{Product}}, \texttt{Daily Stock}, \texttt{Country}, \texttt{Month}, \texttt{Year}\}$. Being the underlined fields the multidimensional **Base** and therefore, the relation "primary key". Along this section, we will refer to this kind of denormalized relation as the *multidimensional table*.

Table 2 summarizes the mapping between both set of algebraic operators. Notice we are considering the "group by" and "aggregation" as relational operators, and both will be justified consequently below. Since *multidimensional tables* contain (1) identifier fields (i.e. identifier **Descriptors**) identifying data -for instance: `City`, `Day` and `Product` in the above example-; (2) numerical fields -`Daily Stock`- representing multidimensional data (i.e. **Measures**) and (3) descriptive fields -`Country`, `Month` and `Year`- (i.e. non-identifier **Descriptors**), we use the following notation in the table: $\checkmark_{Measures}$ if the multidimensional operator is equivalent to the relational one but it can be only applied over relation fields representing **Measures**, $\checkmark_{Descs}$ if the multidimensional operator must be applied over **Descriptors** fields and finally, $\checkmark_{Descs_{id}}$ if it can be only applied over identifier **Descriptors** fields. Consequently, a ✓ without restrictions means both operators are equivalent, without additional restrictions. If the translation of a multidimensional operator combines more than one relational operator, the subscript + is added. Next, we clearly define the relational algebra *proper subset* mappable to the multidimensional algebra:

– The multidimensional **Selection** operator is equivalent to a restricted relational "Selection". It can only be applied over **Descriptors** and then, it is equivalent to restrict the relational "Selection" just over **Level** data. Accord-

Reference Operator		"Selection"	"Projection"	"Join"	"Union"/"Diff."	"Group by"	"Aggregation"
Selection		$\checkmark_{Descs}$					
Projection			$\checkmark_{Measures}$				
Roll-up						$\checkmark_{Descs_{id}+}$	$\checkmark_{Measures+}$
Drill-across			$\checkmark_{Descs_{id}+}$	$\checkmark_{Descs_{id}+}$			
changeBase	**Add Dim.**			$\checkmark_{Descs_{id}}$			
	Remove Dim.		$\checkmark_{Descs_{id}}$				
	Alt. Base		$\checkmark_{Descs_{id}+}$	$\checkmark_{Descs_{id}+}$			
Union/Difference					✓		

Table 2. Comparative table between the relational and the multidimensional algebras.

ing to our notation, we express the multidimensional **Selection** in terms of the relational algebra as $\sigma_{Descriptors}$.

– Similarly, the multidimensional **Projection** operator is equivalent to the relational one restricted to **Measures**; that is, specific **Cell** data. In terms of the relational algebra we could express it as $\pi_{Measures}$.

– OLAP tools emphasize on flexible data grouping and efficient aggregation evaluation over groups and it is the multidimensional **Roll-up** operator the one aimed to provide us with powerful grouping and aggregation of data. In order to support it, we need to extend the relational algebra to provide grouping and aggregation mechanisms. This topic have already been studied and previous works like [25], [7] and [26] have already presented extensions of the relational algebra to what is also called the *grouping algebra*. All of them introduce two new operators; one to group data and apply a simple addition, counting or maximization of a collection of domain values and the other one to compute the aggregation of a given attribute over a given *nested* relation. Following the [26] grouping algebra, we will refer to them as the "group by" and the "aggregation" operators. In terms of this grouping algebra, a **Roll-up** operator consists of a proper "group by" operation along with an "aggregation" of data. Keep in mind this operation must perform a proper aggregation of data if we want it to be consistent.

– A consistent **Drill-across** typically consists on a "Join" between two *multidimensional tables* sharing the same multidimensional space. Notice that to "Join" both tables it must be performed over their common **Level** identifiers that must univocally identify each **cell** in the multidimensional space (the **Cube Base**). Moreover, once "joined", we must "project" out the columns in the *multidimensional table drill-acrossed* to, except for its **Measures**. Formally, Let $\mathcal{A}$ and $\mathcal{B}$ be the *multidimensional tables* implementing, respectively, the origin and the destination **Cells** involved. In the relational algebra it can be expressed as:

$$\pi_{Descriptors_{\mathcal{A}},Measures_{\mathcal{A}},Measures_{\mathcal{B}}}(\mathcal{A} \bowtie \mathcal{B})$$

– As stated in section 2, **changeBase** allows us to rearrange our current multidimensional space either by changing to an alternative **Base** (adding / removing a **Dimension** or replacing **Dimensions**) or reordering the space (i.e. "pivoting" as presented in [3]).
When changing to an alternative **Base** we must assure it does not affect the functional dependency of data with regard to the **Cube Base**. Hence:
  - To add a **Dimension** it must be done through its `All` **Level** or fixing just one value at any other **Level** by means of a **Selection**, to not lose **cells** (i.e. representing the whole **Dimension** as a unique instance as discussed in 2). Therefore, in the relational algebra adding a **Dimension** is achieved through a "cartesian product" between the *multidimensional table* and the **Dimension** table (that would contain a unique instance). Specifically, being $\mathcal{C}$ the initial *multidimensional table* and $\mathcal{D}$ the relation implementing the added **Dimension**, it can be expressed as:

$$\mathcal{C} \times \mathcal{D}, \quad where\ |\mathcal{D}| = 1$$

- To remove a **Dimension** it is just the opposite, and we need to get rid of the proper **Level** identifier projecting it out in the *multidimensional table.*
- To change the set of **Dimensions** identifying each **cell**, that is, choosing an alternative **Base** to display the data, we must perform a "join" between both **Bases** and project out the replaced **Levels Descriptors** in the *multidimensional table.* In this case, the "join" must be performed through the identifier **Descriptors** of **Levels** replaced and **Levels** introduced. Formally, let $\mathcal{A}$ be the *multidimensional table*, $\mathcal{B}$ the table showing the correspondence between both **Bases** and $\mathtt{d_1}, ..., \mathtt{d_n}$ the identifier **Descriptors** of those **Dimensions** introduced. In the relational algebra, it is equivalent to:

$$\pi_{Descriptors_{\mathcal{B}(\mathtt{d_1},\ldots,\mathtt{d_n})}, Measures_{\mathcal{A}}}(\mathcal{A} \bowtie \mathcal{B})$$

- Finally, pivoting just asks to reorder the **Levels** identifiers using the SQL "order by" operator, not mappable to the relational algebra. For that reason, it is not included in table 2.

– The multidimensional **Union** (**Difference**) unites (differences) two **Cubes** defined over the same multidimensional space. In terms of the relational algebra, it is equivalent to "Union" ("Difference") two *multidimensional tables.*

5 Discussion

By means of a comparative of the multidimensional algebras introduced in the literature, section 3 has been able to identify a *multidimensional backbone* shared by all the algebras. Firstly, **Selection**, **Roll-up** and **Drill-down** operators are considered in every algebra. It is quite reasonable since **Roll-up** is the main operator of multidimensionality and **Selection** is a basic one, allowing us to select a subset of multidimensional points of interest out of the whole n-dimensional space. **Projection**, **Drill-across** and **Set Operations** are included in most of the algebras. In fact, along the time, just two of the first algebras presented did not include **Projection** and **Drill-across**. About **Set Operations**, it depends on the transformations that the model allows us to perform over data and indeed, it is a personal decision to make. However, we do believe that to unite, intersect or difference two **Cubes** is a kind of navigation desirable. Finally, **changeBase** is also partially considered in most of the algebras. Specifically, they agree on the necessity of modifying the n-multidimensional space adding / removing **Dimensions**, and they include it as a first class citizen operator. Moreover, our framework provides additional alternatives to rearrange the multidimensional space (i.e. to change the multidimensional space **Base** and "pivoting"). In general, we can always rearrange the multidimensional space in any way, if we preserve the functional dependencies of the **cells** with regard to the **Levels** conforming the **Cube Base**; that is, if the replaced **Dimension**(s) and the new one(s) are related through a one-to-one relationship.

Finally, we would like to underline the need to work in terms of a multidimensional algebra. As shown in section 4, the multidimensional data manipulation should be performed by a restricted subset of the relational algebraic operators (used by ROLAP tools). Otherwise, the results of the operations performed either would not conform a **Cube** (since the whole relational algebra is not closed with regard to the multidimensional model) or would introduce aggregation problems ([18]). In other words, the multidimensional algebra represents the relational algebra *proper subset* applicable to multidimensionality.

Summing up, all the algebras surveyed are subsumed by our framework; that is also strictly subsumed by the relational algebra. Thus, we have been able to (1) identify an implicit agreement about how multidimensional data should be handled and to (2) show that this common set of multidimensional operators can be expressed as a subset of the relational algebra.

6 Conclusions

The comparative of algebras presented in this paper has revealed many implicit agreements about how multidimensional data should be handled. We strongly believe that the *multidimensional backbone* identified in our study could be used to develop design methodologies oriented to improve querying, better and accurate indexing techniques and to facilitate query optimization. That is, provide us with all the benefits of a reference framework. Moreover, we have shown that this common set of multidimensional operators can be expressed as a *proper subset* of the relational algebra; essential to give support to ROLAP tools.

References

1. Abelló, A., Samos, J., Saltor, F.: A Framework for the Classification and Description of Multidimensional Data Models. In: Proc. of 12th Int. Workshop on Database and Expert Systems Applications (DEXA 2001), Springer (2001) 668–677
2. Vassiliadis, P., Sellis, T.: A Survey of Logical Models for OLAP Databases. SIGMOD Record **28**(4) (ACM, 1999) 64–69
3. Franconi, E., Baader, F., Sattler, U., Vassiliadis, P.: Multidimensional Data Models and Aggregation. In: Fundamentals of Data Warehousing. Springer (2000) M. Jarke, M. Lenzerini, Y. Vassilious and P. Vassiliadis editors.
4. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in Data Warehouse Modeling and Design: Dead or Alive? In: Proc. of 9th Int. Workshop on Data Warehousing and OLAP (DOLAP 2006). (2006) 3–10
5. Abelló, A., Samos, J., Saltor, F.: **YAM**^{2} (Yet Another Multidimensional Model): An extension of UML. Information Systems **31**(6) (2006) 541–567
6. Hacid, M.S., Sattler, U.: An Object-Centered Multi-dimensional Data Model with Hierarchically Structured Dimensions. In: Proc. of IEEE Knowledge and Data Engineering Exchange Workshop (KDEX 1997), IEEE (1997)
7. Li, C., Wang, X.: A Data Model for Supporting On-Line Analytical Processing. In: Proc. of 5th Int. Conf. on Information and Knowledge Management (CIKM 1996), ACM (1996) 81–88

8. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling Multidimensional Databases. In: Proc. of the 13th Int. Conf. on Data Engineering (ICDE'97), IEEE (1997) 232–243
9. Gyssens, M., Lakshmanan, L.: A Foundation for Multi-dimensional Databases. In: Proc. of 23rd Int. Conf. on Very Large Data Bases (VLDB 1997), Morgan Kaufmann (1997) 106–115
10. Thomas, H., Datta, A.: A Conceptual Model and Algebra for On-Line Analytical Processing in Data Warehouses. In: Proc. of the 7th Workshop on Information Technologies and Systems (WITS 1997). (1997) 91–100
11. Lehner, W.: Modelling Large Scale OLAP Scenarios. In: Proc. of 6th Int. Conf. on Extending Database Technology (EDBT 1998). Volume 1377 of LNCS., Springer (1998) 153–167
12. Cabibbo, L., Torlone, R.: From a Procedural to a Visual Query Language for OLAP. In: Proc. of the 10th Int. Conf. on Scientific and Statistical Database Management (SSDBM 1998), IEEE (1998) 74–83
13. Hacid, M.S., Sattler, U.: Modeling Multidimensional Database: A formal object-centered approach. In: Proc. of the 6th European Conference on Information Systems (ECIS 1998). (1998)
14. Pedersen, T.: Aspects of Data Modeling and Query Processing for Complex Multidimensional Data. PhD thesis, Faculty of Engineering and Science (2000)
15. Vassiliadis, P.: Data Warehouse Modeling and Quality Issues. PhD thesis, Dept. of Electrical and Computer Engineering (Nat. Tech. University of Athens) (2000)
16. Franconi, E., Kamble, A.: The GMD Data Model and Algebra for Multidimensional Information. In: Proc. of the 16th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2004). Volume 3084 of LNCS., Springer (2004) 446–462
17. Yin, X., Pedersen, T.: Evaluating XML-extended OLAP queries based on a physical algebra. In: Proc. of 7th Int. Workshop on Data Warehousing and OLAP (DOLAP 2004), ACM (2004)
18. Romero, O., Abelló, A.: Improving Automatic SQL Translation for ROLAP Tools. Proc. of 9th Jornadas en Ingeniería del Software y Bases de Datos (JISBD 2005) **284**(5) (2005) 123–130
19. Thomas, H., Datta, A.: A Conceptual Model and Algebra for On-Line Analytical Procesing in Decision Suport Databases. Information Systems **12**(1) (2001) 83–102
20. Cabibbo, L., Torlone, R.: Querying Multidimensional Databases. In: Proc. of the 6th International Workshop on Database Programming Languages (DBPL 1997). Volume 1369 of LCNS., Springer (1997) 319–335
21. Cabibbo, L., Torlone, R.: A Logical Approach to Multidimensional Databases. In: Proc. of 6th Int. Conf. on Extending Database Technology (EDBT 1998). Volume 1377 of LNCS., Springer (1998) 183–197
22. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube operations. In: Proc. of the 10th Statistical and Scientific Database Management (SSDBM 1998), IEEE (1998) 53–62
23. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. Int. Journal of Cooperative Information Systems (IJCIS) **7**(2-3) (1998) 215–247
24. Codd, E.F.: Relational Completeness of Data Base Sublanguages. Database Systems (1972) 65–98
25. Klug, A.: Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. Journal of the Association for Computing Machinery. **29**(3) (1982) 699–717
26. Larsen, K.: On Grouping in Relational Algebra. Int. Journal of Foundations of Computer Science. **10**(3) (1999) 301–311