A new way of optimizing OLAP queries

Arnaud Giacometti* — **Dominique Laurent**** — **Patrick Marcel*** — **Hassina Mouloudi***

* *LI, Université François Rabelais de Blois-Tours-Chinon*
IUP GEII, Antenne Universitaire, 3 place Jean Jaurès
41000 Blois
{arnaud.giacometti,patrick.marcel}@univ-tours.fr
Hassina.Mouloudi@etu.univ-tours.fr

** *LICP, Université de Cergy-Pontoise*
dominique.laurent@dept-info.u-cergy.fr

ABSTRACT. *For around 10 years, the academic research in database has attempted to define a commonly agreed logical modeling for the multidimensional and hierarchical nature of data manipulated with OLAP treatments (called datacube, or cube for short). But only recently has the concept of representation of a cube on a screen, or the optimization of OLAP queries at a logical level, been taken into account in this study. As many others, we believe that these two concepts are essential for the definition of a multidimensional query language. In this article, we propose to consider representations of cubes as first class citizens for query optimisation at the logical level. To reach this goal, we formally define the concept of representation by using the model of complex values [ABI 95]. This allows to have a single model for manipulating both cubes and their representations through typical OLAP operations. These typical operations are studied to propose rewrite rules in order to optimize OLAP queries.*

RÉSUMÉ. *Depuis environ 10 ans, la définition d'un modèle concensuel englobant la nature multidimensionnelle et hiérarchisée des données manipulées par les traitements OLAP (appelées cube de données) est à l'étude. Mais c'est seulement récemment que les concepts de représentation d'un cube à l'écran et d'optimisation de requêtes OLAP à un niveau logique ont été pris en compte dans cette étude. Nous pensons, comme beaucoup, que ces concepts sont essentiels à la définition d'un langage de requêtes pour OLAP. Dans cet article, nous proposons de considérer les représentations de cubes de données comme une base pour l'optimisation de requêtes au niveau logique. Pour ce faire, nous définissons formellement le concept de représentation en utilisant le modèle des valeurs complexes [ABI 95]. Cela permet d'avoir un modèle unique pour manipuler un cube et ses représentations via les opérations OLAP usuelles. L'étude de ces opérations nous permet de donner des règles de réécriture pour optimiser les requêtes OLAP.*

KEYWORDS: *OLAP, query language, logical modeling, optimisation*

MOTS-CLÉS : *OLAP, langage de requêtes, modélisation logique, optimisation*

1. Introduction

Even if OLAP (On-line Analytical Processing, [CHA 97]) logical modeling has been deeply studied in the database community [GYS 97, AGR 97, VAS 00, CAB 97, HAC 97, MEN 04], it is still too early for a commonly agreed logical model to arise. As noticed in [JAR 02], it seems that the practical value of such studies involves both efficient query execution and multidimensional visualization. To the best of our knowledge, there is no work that studies the possibilities of optimizing queries based on the representation of a cube on a screen.

In this paper we concentrate on:

– The formal definition of representation of a cube, i.e., the logical counterpart of what is displayed on a screen.

– The definition of a language for manipulating both cubes and representations.

– The optimization of OLAP queries at a logical level, i.e., based on the properties of the operators.

We propose to describe in a single logical model both datacubes and their representations. We show that the model of complex values [ABI 95] is a good candidate to achieve this goal. We translate into this model the most typical basic OLAP operators [MAR 99]. The language so defined allows the user to precisely:

– Describe what the output of a query is, in terms of representation on a screen.

– Know how the facts displayed have been computed, in particular aggregations are computed on the basis of the actual hierarchies the user sees.

This framework allows us to study the optimization of OLAP queries at a logical level in the following way:

– We give rewriting rules involving the OLAP operators that are used to generate the optimized form of the query.

– Based on the rewriting rules, we identify a canonical form for OLAP queries that can be obtained from any query.

– We rewrite the query so that evaluating this query only deals with the data that will be displayed on the screen as a result.

A lot of work on the optimisation of OLAP queries has been done by studying how to rewrite OLAP queries in the presence of precomputed aggregates ([PAR 01] is a recent work in this area and contains a short overview). This approach does not take into account how the answers to the queries are represented on the screen. Because we concentrate on determining the part of the answer that is to be displayed, our approach can be seen as orthogonal to this one.

To our knowledge, there is no study of the properties of OLAP operations, whereas this study exists for the traditional relational operations [ABI 95]. Only [VAS 00] proposes a logical model and uses it for the optimization of OLAP queries at a logical level. In this work, the authors adapt the classical view subsumption techniques to the

OLAP framework. In that case again our technique can be seen as orthogonal to this one.

In the context of OLAP, the concept of representation of a cube (what [KAR 03] calls the presentation level) attracted very little attention. Many of the logical models and languages proposed in the literature confuse logical level and presentation level. To our knowledge, only two proposals have been made:

– The MDX API proposed in [Mic 98]. This complex SQL-like language, where the intuitions behind the SELECT and WHERE clauses do not really correspond to those of SQL, mixes the presentation and logical level and lacks of theoretical foundations.

– The CPM model [MAN 03], that seems to be an attempt for giving MDX expressions a clear logical foundation. In this model, the logical and presentation levels are separated. Our work can be seen as an extension of this research, towards the use of the presentation level as a basis for query optimisation.

The rest of the paper is structured as follows: In Section 2, we propose a logical definition of a cube and a representation in terms of complex values. In Section 3, we define an algebra by translating the basic OLAP operators into the algebra for complex values. Section 4 deals with the optimization process, and Section 5 draws conclusions and future work.

2. Cubes and Representations

In this work, we make a clear distinction between a cube, a representation of a cube and the visualization of a representation. Intuitively, we propose to consider that logically:

– A cube is the set of all basic facts that can be presented to the user, including the most detailed data and the precomputed aggregates. A cube is composed of a fact table and a set of dimension tables, as in a star schema [JAR 02]. This corresponds to what [VAS 00] calls the detailed data set.

– The representation of a cube is what the user wants to see, under the form of a multidimensional cross-tab. A representation corresponds to the result of a query over a cube or a representation. It is composed of the set of facts the user wants to see and of the description of the axes of the cross-tab. If the representation the user wants to see has more than two axes, as it cannot be easily displayed on a two-dimensional screen, only the first two-dimensional slice of the representation is displayed. Note that in our formalism, this two-dimensional slice is also a representation of a cube.

To take the concept of representation into account, we propose to use the model of complex values [ABI 95] since this model is very well adapted for describing the nesting of attributes[1]. Note that a first investigation of this idea occurs in [DEK 98].

1. For the sake of space we refer the reader to [ABI 95] for a presentation of this model.

In what follows, we distinguish two categories of attributes, as usual in the OLAP context: the member attributes, having names like L or F, and the measure attribute, named m. Without loss of generality, we consider only one measure attribute. In our model, both cubes and representations are complex values $\langle I_A, I_F, I_P \rangle$ having sort the same generic form $\langle A : sort(A), F : sort(F), P : sort(P) \rangle$ where:

– I_A is a complex value that describes:

- In the case of a cube: The dimensions of the cube and their hierarchies[2].

- In the case of a representation: The axes of the representation.

– I_F is a complex value that describes:

- In the case of a cube: The complete set of facts of the datacube.

- In the case of a representation: The facts that the user wants to see.

– I_P is a relation that describes:

- In the case of a cube: a default position for each member of the cube on an axis of a representation built from this cube.

- In the case of a representation: The position of each displayed member on an axis.

We now turn to the formal definitions.

2.1. *Cube*

An instance I_C of an N-dimensional cube C is a tuple $I_C = \langle I_{A_{All}}, I_{F_{All}}, I_{P_{All}} \rangle$ that is a complex value having sort $sort(C) = \langle A_{All} : sort(A_{All}), F_{All} : sort(F_{All}), P_{All} : sort(P_{All}) \rangle$ where:

– $I_{A_{All}}$ is a tuple $\langle I_{D_1}, \ldots, I_{D_N} \rangle$ that describes the dimensions of the cube. It is a complex value having sort $sort(A_{All}) = \langle D_1 : sort(D_1), \ldots, D_N : sort(D_N) \rangle$. For all $i \in [1, N]$, I_{D_i} is a complex value describing a particular dimension i of the cube. The nested structure of this complex value allows to describe the hierarchy associated with the dimension in a straightforward manner. It has sort: $sort(D_i) = \{\langle L_i^0 : dom(L_i^0), D_i^1 : \{\langle L_i^1 : dom(L_i^1), \ldots, D_i^{q_i} : \{\langle L_i^{q_i} : dom(L_i^{q_i}) \rangle\} \ldots \rangle\}\rangle\}$.

For a dimension i, each attribute L_i^j describes a level of the hierarchy, j being the depth of this level in the dimension. In the following, we denote by $\mathcal{L}_i(C)$ the set of all attributes L_i^j in dimension i of cube C, i.e. $\mathcal{L}_i(C) = \{L_i^j \mid 0 \leq j \leq q_i\}$, and by $\mathcal{L}(C)$ the set of all attributes L_i^j of cube C, i.e. $\mathcal{L}(C) = \bigcup_{i \in [1,N]} \mathcal{L}_i(C)$. Finally, let $v \in dom(L_i^j)$ and $k \in [0, j-1]$. We define the classical notion of ancestors of v at level k by: $anc(v, L_i^k) = \pi_{L_i^k}(\sigma_{L_i^j = v}(unnest_{D_i^j}(\ldots(unnest_{D_i^1}(I_{D_i}))\ldots)))$.

– $I_{F_{All}}$ is the set of all facts of the cube. It is a complex value having sort $sort(F_{All}) = \{\langle F_1^{all} : dom(F_1^{all}), \ldots, F_N^{all} : dom(F_N^{all}), m : dom(m) \rangle\}$. $I_{F_{All}}$

2. For the sake of readability, the model is restricted to only one hierarchy per dimension.

represents the fact table of the cube. It describes every fact at every level of detail. Hence the following equalities hold: $\forall i \in [1, N], dom(F_i^{all}) = \bigcup_{j \in [0,q_i]} dom(L_i^j)$.

– $I_{P_{All}}$ is an instance of relation of sort $\{\langle member : \bigcup_i (\bigcup_j dom(L_i^j)), position : \mathbb{N}\rangle\}$ that associates each member of the cube with a distinct integer. This integer corresponds to the default position of a member on an axis when a representation is built from a cube. This relation defines a total ordering over $dom(L_i^j)$, for all i and j.

Example 2.1 As an example, we consider a 5 dimensional cube named $sales$, inspired by the example used in [Mic 98]. The dimensions are:

– Year: the different years.
– Quarter: the months grouped in quarters.
– Location: the cities grouped in regions and countries.
– Product: the items grouped in categories.
– Salesman: the different salespersons.

The sort of the cube $sales$ is:

$$sort(C_{sales}) = \langle A_{All} : sort(A_{All}), F_{All} : sort(F_{All}), P_{All} : sort(P_{All})\rangle,$$

$$sort(A_{All}) = \langle Year : sort(Year), Quarter : sort(Quarter),$$
$$Location : sort(Location), Product : sort(Product),$$
$$Salesman : sort(Salesman)\rangle$$

$$sort(F_{All}) = \{\langle F_1^{all} : \{all_{year}, 1990, \ldots, 2004\}, F_2^{all} : \{all_{quarter}, q_1, \ldots, dec\},$$
$$F_3^{all} : \{all_{location}, france, \ldots, paris\},$$
$$F_4^{all} : \{all_{product}, drink, \ldots, wine\},$$
$$F_5^{all} : \{all_{salesman}, john, \ldots, bill\}, sales : \mathbb{N}\rangle\}$$

$$sort(P_{All}) = \{\langle member : \{1990, john, \ldots\}, position : \mathbb{N}\rangle\}.$$

The dimension $Quarter$, an instance of F_{All} and an instance of P_{All} are described in Figure 1. In what follows, we consider an instance I_{sales} of the cube $sales$.

2.2. *Constructing a representation of a cube*

A representation of a cube is constructed by using the $Navigate$ operation. Intuitively, this operation is a selection in the fact table of the cube in order to present the data at a given level for each dimension[3]. For example, the query $Navigate_{year,quarter,city,item,name}(I_{sales}) = I_1$ allows to obtain the representation displayed in Figure 2 (a). Note that only one 2-dimensional slice of this representation is displayed (on the left-hand side of the figure), i.e., the facts concerning the cities

3. This operator is close to the navigate operator proposed in [VAS 00]. It is formally defined in the Appendix.

Quarter	top_Q	quarter	month
	$all_{quarter}$	q_1	jan
			feb
			mar
		...	...
		q_4	oct
			nov
			dec

(a) An instance of the dimension $Quarter$

$I_{F_{All}}$	Year	Quarter	Location	Product	Salesman	sales
	1988	q_1	paris	drink	john	90
	1988	q_1	paris	beer	john	30
	1988	q_1	paris	wine	john	20
	1988	q_1	paris	milk	john	20
	1988	q_1	paris	$all_{product}$	john	1000
	⋮	⋮	⋮	⋮	⋮	⋮
	all_{year}	$all_{quarter}$	$all_{location}$	$all_{product}$	$all_{salesman}$	500000

(b) An instance of F_{All}

$I_{P_{All}}$	member	position
	1988	1
	drink	1
	food	2
	milk	1
	beer	2
	john	1
	france	1
	paris	1
	⋮	⋮

(c) An instance of P_{All}

$$sort(Quarter) = \{\langle top_Q : \{all_{quarter}\}, D_2^1 : \{\langle quarter : \{q_1, q_2, q_3, q_4\}, D_2^2 : \{\langle month : \{jan, \ldots, dec\}\rangle\}\rangle\}\rangle\}.$$

(d) The sort of the dimension $Quarter$

Figure 1. *The dimension $Quarter$, the facts and default positions of the cube $sales$*

other than $paris$, the items other than $milk$ and the salesmen other than $john$ (these values appear on the right-hand side of the figure) are hidden. Indeed, a representation is always displayed so that the user can see only the members of the first two axes, and the member at the lowest position on every remaining axis. When there is more than one attribute on an axis, the attributes are displayed according to the order imposed by their level and by their position, and the measures are displayed accordingly.

Formally, an instance of a representation R of a cube C is a tuple $I_R = \langle I_A, I_F, I_P \rangle$ of sort $sort(R) = \langle A : sort(A), F : sort(F), P : sort(P) \rangle$ where:

– I_A is a tuple $\langle I_{A_1}, \ldots, I_{A_K} \rangle$ of axes, of sort $sort(A) = \langle A_1 : sort(A_1), \ldots, A_K : sort(A_K) \rangle$. For all $k \in [1, K]$, I_{A_k} is a complex value de-

scribing a particular axis of the representation. I_{A_k} is of sort $sort(A_k) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_k^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_k^S : \{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\}\ldots\rangle\}\rangle\}$. We denote by $\mathcal{A}_k(R)$ the set of attributes of $\mathcal{L}(C)$ that occur on A_k, i.e. $\mathcal{A}_k(R) = \{L_{i_0}^{j_0}, L_{i_1}^{j_1}, \ldots, L_{i_S}^{j_S}\}$, and by $\mathcal{A}(R)$ the set of all attributes of $\mathcal{L}(C)$ that occur on an axis of R, i.e. $\mathcal{A}(R) = \bigcup_{k=1}^{K} \mathcal{A}_k(R)$.

– I_F is a complex value describing the facts depicted by the representation. I_F is of sort $sort(F) = \{\langle L_1^{d_1} : dom(L_1^{d_1}), \ldots,\ L_N^{d_N} : dom(L_N^{d_N}), m : dom(m)\rangle\}$. We denote by $\mathcal{F}(R)$ the set of attributes of $\mathcal{L}(C)$ that occur in $sort(F)$, i.e. $\mathcal{F}(R) = \{L_1^{d_1}, \ldots, L_N^{d_N}\}$.

– I_P is a complex value of sort $\{\langle member : \bigcup_i(\bigcup_j dom(L_i^j)), position : \mathbb{N}\rangle\}$ that associates each member of R with an integer. This integer corresponds to the relative position of the member on an axis of the representation.

Example 2.2 As an example, we consider the representation I_7 of Figure 3 (c). $I_7 = \langle I_A, I_F, I_P\rangle$ of sort $sort(I_7) = \langle A : sort(A), F : sort(F), P : sort(P)\rangle$ where $I_A = \langle I_{A_5}, I_{A_2}, I_{A_4}, I_{A_1}\rangle$ and:

– $sort(A) = \langle A_5 : sort(A_5), A_2 : sort(A_2), A_4 : sort(A_4), A_1 : sort(A_1)\rangle$, with, for instance:

$$sort(A_5) = \{\langle name : dom(name),\ U_5^1 : \{\langle region : dom(region),\ U_5^2 : \{\langle city : dom(city)\rangle\}\rangle\}\rangle\}$$

$$I_{A_5} = \{\langle john,\ U_5^1 : \{\langle north,\ U_5^2 : \{\langle paris\rangle, \langle blois\rangle, \langle lille\rangle\}\rangle, \langle south,\ U_5^2 : \{\langle lyon\rangle, \langle marseille\rangle\}\rangle, \ldots\}\rangle, \ldots,\ \langle bill,\ U_5^1 : \{\langle north,\ U_5^2 : \{\langle paris\rangle, \langle blois\rangle, \langle lille\rangle\}\rangle \ldots\}\rangle\}$$

– $sort(F) = \{\langle year : dom(year),\ quarter : dom(quarter), city : dom(city), category : dom(category),\ name : dom(name), quantity : dom(quantity)\rangle\}$

$$I_F = \{\langle 2000, q_1, paris, drink, john,\ 10\rangle, \langle 2000, q_1, blois, drink, john,\ 20\rangle, \langle 2000, q_1, lille, drink, john,\ 70\rangle \ldots\}$$

– $sort(P) = \{\langle member : \bigcup_i(\bigcup_j dom(L_i^j)), position : \mathbb{N}\rangle\}$

$$I_P = \{\langle john, 1\rangle, \langle north, 1\rangle, \langle paris, 1\rangle, \ldots, \langle q_1, 1\rangle, \ldots, \langle drink, 1\rangle, \ldots, \langle 2000, 1\rangle\}$$

3. OLAP operations

In this section, we describe how cubes and representations are manipulated using the most typical OLAP operators [MAR 99]. Basically, using the definition of cubes and representations above, we translate the OLAP operators into the algebra for complex values (as defined in [ABI 95]). We consider the following OLAP operators, that are classified according to 3 categories:

– Restructuring operators that change the viewpoint on data. Operators in this category are *Permute, Switch, Nest*.

	$\mathbf{A_1}$					$\mathbf{A_3}$
	2004	10	20	20	10	$all_{location}$
	2003	50	70	50	60	↑
	2002	60	60	80	80	france
a	2001	70	50	40	70	↑
l	2000	40	80	10	80	north
l	1999	50	30	10	10	↑
	1998	50	30	70	10	**paris**
y	1997	40	50	30	70	$\mathbf{A_4}$
e	1996	10	40	50	10	$all_{product}$
a	1995	40	50	80	70	↑
r	1994	50	80	30	50	drink
	1993	10	70	40	50	↑
	1992	10	80	40	30	**milk**
	1991	80	40	50	10	$\mathbf{A_5}$
	1990	70	70	10	50	$all_{salesman}$
	1989	40	50	50	10	↑
	1988	20	50	70	80	**john**
		q_1	q_2	q_3	q_4	
	$\mathbf{A_2}$		$all_{quarter}$			

(a) $I_1 = Navigate_{year,quarter,city,item,name}(I_{sales})$

$\mathbf{A_1}$					$\mathbf{A_3}$
2004	10	20	20	10	north
2003	50	70	50	60	↑
2002	60	60	80	80	**paris**
⋮					
1995	40	50	80	70	$\mathbf{A_4}$
1994	50	80	30	50	drink
1993	10	70	40	50	↑
1992	10	80	40	30	**milk**
⋮					
1989	40	50	50	10	$\mathbf{A_5}$
1988	20	50	70	80	**john**
$\mathbf{A_2}$	q_1	q_2	q_3	q_4	

(b) $I_2 = \pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}(I_1)))))$

$\mathbf{A_1}$					$\mathbf{A_3}$
2004	40	30	20	40	north
2003	50	50	40	50	↑
2002	60	40	30	50	**paris**
⋮					$\mathbf{A_4}$
1992	70	80	50	70	drink
1991	60	40	60	80	↑
1990	60	70	50	40	**beer**
1989	50	50	30	10	$\mathbf{A_5}$
1988	30	40	70	70	**john**
$\mathbf{A_2}$	q_1	q_2	q_3	q_4	

(c) $I_3 = \sigma^{member}_{item=beer \vee item=wine \vee category=food}(I_2)$

$\mathbf{A_1}$					$\mathbf{A_3}$
2004	60	60	40	50	north
2003	80	90	90	90	↑
2002	70	50	70	80	**paris**
⋮					$\mathbf{A_4}$
1992	80	90	60	80	**drink**
1991	80	50	70	90	↑
1990	70	80	60	50	beer,wine
1989	90	90	50	20	$\mathbf{A_5}$
1988	50	50	90	80	**john**
$\mathbf{A_2}$	q_1	q_2	q_3	q_4	

(d) $I_4 = Aggregate_{item \rightarrow category; sum(sales)}(I_3)$

Figure 2. *Outputs of steps 1-4. On each figure, the left-hand side displays the first 2-dimensional slice of the representation, and the right-hand side displays the member at the lowest position for each hidden axis.*

$\mathbf{A_5}$					$\mathbf{A_3}$
bill	30	90	50	60	north
rose	30	10	90	90	↑
irma	70	60	70	10	**paris**
kate	90	60	50	10	$\mathbf{A_4}$
lara	90	90	70	10	**drink**
averell	10	30	30	90	↑
jack	10	60	50	60	beer,wine
joe	90	70	30	10	$\mathbf{A_1}$
john	50	50	90	80	**1988**
$\mathbf{A_2}$	q_1	q_2	q_3	q_4	

(a) $I_5 = Permute_{A_5,A_1}(I_4)$

$\mathbf{A_5}$							$\mathbf{A_4}$
bill	north	lille	40	60	60	70	**drink**
		blois	10	20	30	20	↑
		paris	30	90	50	60	beer,wine
⋮							
john	north	lille	70	70	50	50	$\mathbf{A_1}$
		blois	20	30	20	20	**1988**
		paris	50	50	90	80	
$\mathbf{A_2}$			q_1	q_2	q_3	q_4	

(b) $I_6 = Nest_{A_5(A_3)}(\sigma_{region=north}(I_5))$

$\mathbf{A_5}$							$\mathbf{A_4}$
bill	north	lille	20	30	40	30	**drink**
		blois	20	10	10	20	↑
		paris	10	40	30	50	beer,wine
⋮							
john	north	lille	70	60	50	50	$\mathbf{A_1}$
		blois	20	20	30	30	**2000**
		paris	10	20	10	10	
$\mathbf{A_2}$			q_1	q_2	q_3	q_4	

(c) $I_7 = Switch_{year;1988,2000}(I_6)$

Figure 3. *Outputs of steps 5-7*

– Operators that change the level of detail. We consider two different operators in this category:

- The $Navigate$ operator, introduced in the previous section, defines for each dimension of the cube at which level the data are to be represented.

- the $Aggregate$ operator groups the members of an axis according to the part of the hierarchy displayed in the representation, and then aggregates the measures accordingly.

– Filtering operators, that are the extension of classical selection and projection to representations.

Note that the classical roll-up, drill-down or slice&dice OLAP queries can be expressed in our framework by using this set of operators. For instance, as the $Navigate$ operator selects from the fact table the data at a given level for each dimension, it can be used to express drill-down or roll-up queries.

We introduce the different operators in an example, the formal definitions are given in the Appendix. The example is based on the following query:

$$Switch_{year;1988,2000}(Nest_{A_5(A_3)}(\sigma^{member}_{region=north}(Permute_{A_1,A_5}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$\pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

This query aims at presenting the sales of food and the cumulated sales of beer and wine for year 2000 in the north, detailed by salespersons and quarters. The final output of this query is given in Figure 3 (c). We examine the different steps independently. The first step, which involves the $Navigate$ operation, is illustrated in the previous section. We now illustrate the filtering operations.

3.1. *Filtering*

We illustrate step 2 and step 3 of the query:

– step 2: $I_2 = \pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}(I_1)))))$

– step 3: $I_3 = \sigma^{member}_{item=beer\vee item=wine\vee category=food}(I_2)$

The outputs are given in Figure 2 (b) and (c). These two steps reduce the amount of displayed facts (for selection) and displayed levels (for projection).

Note that after step 3, the displayed slice has changed. This is due to the fact that the item $milk$, that was at position 1 of axis A_4 in I_2 is no more selected in representation I_3. Then among the selected items, the one at the lowest position, $beer$, is displayed first. Note also that in our formalism, a representation must always depict the hierarchies that have been used to aggregate the data. Thus, for a given dimension, selection on members can not be applied on a member attribute that identifies a level deeper than the level currently displayed. Otherwise the measures displayed would no more correspond to the displayed hierarchy. The same remark holds for the projection operation.

3.2. *Changing the level of detail*

Step 4, i.e., $I_4 = Aggregate_{item \rightarrow category; sum(sales)}(I_3)$ illustrates a way of changing the level of detail of the representation. Note that we distinguish two different ways of changing the level of detail: by using the $Navigate$ operation to present the data at a given level for each dimension, or by using the $Aggregate$ operation. Intuitively, the $Aggregate$ operation groups and aggregates the data according to the groupings $depicted$ on a particular axis. This operation does not change the sort of the axes.

The output of step 4 is given in Figure 2 (d). Note that the facts that are displayed are no more the facts in the fact table of the cube $sales$. This is so because the $Aggregate$ operator has been used to compute the sales of $drink$, but only for the drinks that were selected in step 3, i.e., $beer$ and $wine$.

3.3. *Restructuring*

The last operations to be considered are the restructuring operations. These operations allow to exchange the positions of two axes ($Permute$), exchange the positions of two members ($Switch$) or nest two axes together ($Nest$). They are illustrated by the following steps:

– step 5: $I_5 = Permute_{A_5, A_1}(I_4)$
– step 6: $I_6 = Nest_{A_5(A_3)}(\sigma_{region=north}(I_5))$
– step 7: $I_7 = Switch_{year; 1988, 2000}(I_6)$

The outputs of these three steps are given in Figure 3. Note that these operations do not change the facts of the representation, but only the way they are displayed.

4. OLAP query optimisation

In this section, we propose an optimisation technique for OLAP queries. We first introduce this technique informally.

4.1. *Intuitions*

This optimisation technique consists in determining which part of the query output will be displayed on the screen. This part is the first 2-dimensional slice of the representation, which is also a representation. This slice can be computed by adding selection conditions to the initial query. Intuitively, these conditions are obtained by inspecting the parameters of the restructuring and filtering operations in the query. As the costly operation is $Aggregate$, the optimization consists in pushing the conditions

before the $Aggregate$. We first explain how selection conditions are computed, and then we present how these conditions are added to the query.

When statically inspecting a query, the $Nest$ and projection operations are used to identify which attributes are displayed and on which axis. The $Permute$ operation is used to identify the first 2 axes of the representation, i.e., the axes that are fully displayed. The remaining hidden axes contain attributes for which only values at the lowest position are visible. These values can be deduced from the $Switch$ operation and the default positions of the members.

As an example, we consider the query presented in Section 3. It can be found by inspecting the parameters of the $Permute$, $Nest$, $Switch$ and projection operations that:

– The final representation has 4 axes.

– The attributes on the two hidden axes are $year$, $item$ and $category$.

– The values of these attributes at the lowest position are respectively 2000, $beer$ and $drink$.

It can be found by inspecting the parameters of the $Aggregate$ and selection operations that:

– The facts displayed concern levels $year$, $quarter$, $category$, $city$ and $salesman$.

– The measures for $drink$ are computed only from the measures for $beer$ and $wine$.

Hence the selection condition that is generated is $year = 2000 \wedge category = drink$.

4.2. *Canonical form of an OLAP query*

Our optimisation technique is based on a particular form of queries, which we call *canonical form*. We define the canonical form of an OLAP query, that complies with the standard intuition behind OLAP queries: A $Navigate$ identifies the levels at which the facts are to be represented, some ad-hoc aggregations, i.e., sequences of selections and aggregations, can be performed, and finally the restructuring operations construct the viewpoint. The canonical form of a query is given below ($\circ$ denotes the composition of operations):

$$Q = \circ_{k=1}^{W} Switch_{L_{z_k}^{w_k}:v_k,v'_k} \circ_{k=1}^{P} Permute_{A_{\alpha_k},A_{\beta_k}} \circ_{k=1}^{S} Nest_{A_{x_k}(A_{y_k})}$$

$$\circ_{i=1}^{N} \pi_{X_i}^{A_i} \circ_{k=1}^{G} [\sigma_{\varphi_k}^{measure} \circ Aggregate_{L_{i_k}^{j_k} \to L_{i_k}^{l_k};\, f(m)} \circ \sigma_{\psi_k}^{member}$$

$$\circ\, \sigma^{measure}_{\varphi'_k}] \circ Navigate_{L_1^{d_1},\ldots,L_N^{d_N}}(I)$$

where for all k, $\sigma^{member}_{\psi_k} = \circ_{i=1}^{N} \sigma^{member}_{\psi_k^{D_i}}$

The following lemma, which proof can be found in [GIA 04], states that all queries have an equivalent canonical form.

Lemma 4.1 *Every OLAP query can be put under the canonical form.*

Moreover, the rewriting rules presented in Figure 4 allow to transform any query into its canonical form. Note that the last rule is valid since in our formalism, selection cannot be applied on an attribute that identifies a level deeper than the displayed level. For example, on representation I_7 displayed in Figure 3 (c), selection on item, e.g., $item = beer$ is not allowed, because facts are displayed at the category level.

Among the algebraic operators, the $Aggregate$ is the most costly since it is used to compute new facts. So the principle of our technique is to propagate the selection conditions before the aggregate operation. Getting back to our example, the rewritten query is:

$$Permute_{A_1,A_5}(Nest_{A_5(A_3)}(\pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{S}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

where $S = (region = north \wedge (item = beer \vee item = wine \vee category = food)) \wedge (year = 2000 \wedge category = drink)$. Moreover it is easy to see that condition $category = drink$ is redundant in S, thus we can consider $S = (region = north \wedge (item = beer \vee item = wine) \wedge year = 2000)$.

4.3. *Optimisation algorithm*

We are now ready to present the algorithm for optimizing OLAP queries. It consists of one function, named $Compute\_First\_Slice\_Selection$, given Figure 5.

Let us examine each step of the function. Recall that $Navigate_{L_1^{d_1},\ldots,L_N^{d_N}}$ outputs a representation having N axes, with attributes of dimension D_i on axis A_i.

1) This step only looks for the axes of the representation.

2) For every axis, this step performs the following four actions:

a) It finds the depth at which the data will be presented. For axis A_i, this depth is either d_i if A_i is not one of the parameters of $Aggregate$ (recall that d_i is the depth on axis A_i according to which the facts of I are depicted), or it is the smallest l_k such that $Aggregate_{L_{i_k}^{j_k} \rightarrow L_{i_k}^{l_k};\, f(m)} \in Q$ otherwise.

r_1	$Nest_{A_i(A_j)}(Permute_{A_k,A_l}(I))$	$= Permute_{A_k,A_l}(Nest_{A_i(A_j)}(I))$ if $j \notin \{k,l\}$
r_2	$Nest_{A_i(A_j)}(Permute_{A_k,A_j}(I))$	$= (\circ_{l=j+1}^{k-1} Permute_{A_k,A_l})(Nest_{A_i(A_j)}(I))$
r_3	$Nest_{A_i(A_j)}(Switch_{L_k^l;v,v'}(I))$	$= Switch_{L_k^l;v,v'}(Nest_{A_i(A_j)}(I))$
r_4	$Nest_{A_i(A_j)}(\pi^{A_k}_{L_{x1}^{y1},\ldots,\ L_{xp}^{yp}}(I))$	$= \pi^{A_k}_{L_{x1}^{y1},\ldots,\ L_{xp}^{yp}}(Nest_{A_i(A_j)}(I))$
r_5	$Permute_{A_i,A_j}(Switch_{L_k^l;v,v'}(I))$	$= Switch_{L_k^l;v,v'}(Permute_{A_i,A_j}(I))$
r_6	$Permute_{A_i,A_j}(\pi^{A_k}_{L_{x1}^{y1},\ldots,L_{xp}^{yp}}(I))$	$= \pi^{A_k}_{L_{x1}^{y1},\ldots,L_{xp}^{yp}}(Permute_{A_i,A_j}(I))$
r_7	$Switch_{L_i^j;v,v'}(\pi^{A_k}_{L_{x1}^{y1},\ldots,L_{xp}^{yp}}(I))$	$= \pi^{A_k}_{L_{x1}^{y1},\ldots,L_{xp}^{yp}}(Switch_{L_i^j;v,v'}(I))$
r_8	$\sigma_\varphi(Op(I))$	$= Op(\sigma_\varphi(I))$
r_9	$Aggregate_{L_{ix}^{jx}\to L_{iy}^{jy};\ f(m)}(Op(I))$	$= Op(Aggregate_{L_{ix}^{jx}\to L_{iy}^{jy};\ f(m)}(I))$
r_{10}	$\sigma_\varphi^{mb}(Aggregate_{L_{ix}^{jx}\to L_{iy}^{jy};\ f(m)}(I)))$	$= Aggregate_{L_{ix}^{jx}\to L_{iy}^{jy};\ f(m)}(\sigma_\varphi^{mb}(I)))$

Figure 4. *Rewriting rules for OLAP algebra. In this figure, $\circ$ denotes the composition of operations, $Op \in \{Permute, Switch, Nest, \pi\}$, σ^{mb} denotes a selection on members, and σ denotes a selection on measures or members.*

b) It assigns to the axis a location number in $[1, N]$, which is i for axis A_i.

c) It finds the set of attributes appearing on the axis, by using the parameters of the projection operation. These attributes are simply X_i for axis A_i.

d) It finds the initial position of every member appearing on the axis.

3) For each axis being the result of $Nest$ operations, this step computes the set of attributes appearing on this axis. For each $Nest_{A_{x_k}(A_{y_k})}$ operation, the set of attributes on axis A_{x_k} is the union of the sets of attributes of A_{x_k} and A_{y_k}. Note that when two axes of a representation are nested, the resulting representation has one axis less. This means that the location number of the axes that are located after A_{y_k} has to be updated. After this step, only $N - S$ axes remain. The location number of each of these $N - S$ axes remains in $[1, N - S]$. The set $\mathcal{A}'$ contains these $N - S$ axes.

4) This steps computes the new location number of two axes if these two axes are permuted.

5) This steps computes the new position of two members if these two members are switched. Note that the members that are not switched keep their original position, i.e., the ones that are given by the cube I.

6) This step computes the selection conditions. Recall that the idea is to find the members at the lowest position on the axes that are hidden. These axes are the ones of $\mathcal{A}'$ having a location number greater than 2. For each such axis i, we consider each attribute appearing on this axis. For each such attribute L_i^k , we identify the members that will belong to the resulting representation, i.e., the members that appear in the selection condition $\psi_k^{D_i}$ (or all the members of these attributes if this condition is *true*). Then the member v having the lowest position is found, and the condition $L_i^k = v$ is generated.

Function $Compute\_First\_Slice\_Selection[Q]$

Input: A query under canonical form
$Q = \circ_{k=1}^{W} Switch_{L_{z_k}^{w_k};v_k,v'_k} \circ_{k=1}^{P} Permute_{A_{\alpha_k},A_{\beta_k}} \circ_{k=1}^{S} Nest_{A_{x_k}(A_{y_k})}$
$\circ_{i=1}^{N} \pi_{X_i}^{Ai} \circ_{k=1}^{G} [\sigma_{\varphi_k}^{measure} \circ Aggregate_{L_{i_k}^{j_k} \to L_{i_k}^{l_k};\, f(m)} \circ \sigma_{\psi_k}^{member}$
$\circ \sigma_{\varphi'_k}^{measure}] \circ Navigate_{L_1^{d_1},\ldots,L_N^{d_N}}(I)$
where for all k, $\sigma_{\psi_k}^{member} = \circ_{i=1}^{N} \sigma_{\psi_k^{D_i}}^{member}$
An instance $I = \langle I_{A_{All}}, I_{F_{All}}, I_{P_{All}} \rangle$ of a cube
Output: A condition φ of selection
Local: $att(A_i)$ is the set of attributes on axis A_i
$loc(A_i)$ is the location of axis A_i
$pos(v_k)$ is the position of a member v_k

1. Let $\mathcal{A} = \{A_1, \ldots, A_N\}$
2. *// Aggregate and Projection operators*
   for $i = 1$ *to* N **do begin**
   $k_i = min(\{l_k \mid Aggregate_{L_{i_k}^{j_k} \to L_{i_k}^{l_k};\, f(m)} \in Q \wedge i_k = i\} \cup \{d_i\})$
   $loc(A_i) = i$
   $att(A_i) = X_i$
   for every $v \in \bigcup_i (\bigcup_j dom(L_i^j))$ such that $L_i^j \in att(A_i)$ **do**
   $pos(v) = \pi_{position}(\sigma_{L_i^j = v}(I_{P_{All}}))$
   end for
3. *// Nest operators*
   for $k = S$ *to* 1 **do begin**
   $att(A_{x_k}) = att(A_{x_k}) \cup att(A_{y_k})$;
   for $A_l \in \mathcal{A}$ such that $loc(A_l) > loc(A_{y_k})$ **do** $loc(A_l) = loc(A_l) - 1$
   end for
4. *// Permute operators*
   for $k = P$ *to* 1 **do** $x = loc(A_{\beta_k})$; $loc(A_{\beta_k}) = loc(A_{\alpha_k})$; $loc(A_{\alpha_k}) = x$ **end for**
5. *// Switch operators*
   for $k = W$ *to* 1 **do** $x = pos(v_k)$; $pos(v_k) = pos(v'_k)$; $pos(v'_k) = x$ **end for**
6. *// Selection conditions on the hidden axes*
   $\varphi = true$
   $\mathcal{A}' = \{A_j \in \mathcal{A} \mid \nexists\, Nest_{A_i(A_j)} \in Q\}$
   for $l = 3$ *to* $|\mathcal{A}'|$ **do begin**
   Let $A_k \in \mathcal{A}'$ such that $loc(A_k) = l$;
   for every i such that $L_i^{k_i} \in att(A_k)$ **do begin**
   $\varphi = \varphi \wedge (L_i^{k_i} = v)$ where $v \in dom(L_i^{k_i})$ and
   $pos(v) = min(\{pos(v') \mid v' \in \pi_{L_i^{k_i}}(\sigma_{\phi_i}(unnest_{D_i^{k_i}}(\ldots(unnest_{D_i^1}(I_{D_i})))\})$
   where $\sigma_{\phi_i} = \circ_{k=1}^{G} \sigma_{\psi_k^{D_i}}$
   end for
   end for
7. **Return** φ

Figure 5. *The function* $Compute\_First\_Slice\_Selection$

Example 4.1 Consider the OLAP query used throughout the paper:

$$q = Switch_{year;1988,2000}(Nest_{A_5(A_3)}(\sigma^{member}_{region=north}(Permute_{A_1,A_5}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$\pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

First, we illustrate how the rewriting rules are used to put this query under canonical form. The rules can be applied as follows:

– $q \xrightarrow{r_8} q_1$ (selection can be performed before restructuring)

$$q_1 = Switch_{year;1988,2000}(Nest_{A_5(A_3)}(Permute_{A_1,A_5}(\sigma^{member}_{region=north}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}($$
$$\pi^{A_5}_{name}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

– $q_1 \xrightarrow{r_9} q_2$ (aggregation can be performed before restructuring)

$$q_2 = Switch_{year;1988,2000}(Nest_{A_5(A_3)}(Permute_{A_1,A_5}(\sigma^{member}_{region=north}($$
$$\pi^{A_1}_{year}(\pi^{A_2}_{quarter}(\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

– $q_2 \xrightarrow{r_8} q_3$ (selection can be performed before restructuring)

$$q_3 = Switch_{year;1988,2000}(Nest_{A_5(A_3)}(Permute_{A_1,A_5}(\pi^{A_1}_{year}(\pi^{A_2}_{quarter}($$
$$\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}(\sigma^{member}_{region=north}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

– $q_3 \xrightarrow{r_1} q_4$ ($Nest$ can be performed before $Permute$)

$$q_4 = Switch_{year;1988,2000}(Permute_{A_1,A_5}(Nest_{A_5(A_3)}(\pi^{A_1}_{year}(\pi^{A_2}_{quarter}($$
$$\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}(\sigma^{member}_{region=north}($$
$$Aggregate_{item\rightarrow category;sum(sales)}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

– $q_4 \xrightarrow{r_{10}} q_{cf}$ (selection on members can be performed before $Aggregate$)

$$q_{cf} = Switch_{year;1988,2000}(Permute_{A_1,A_5}(Nest_{A_5(A_3)}(\pi^{A_1}_{year}(\pi^{A_2}_{quarter}($$
$$\pi^{A_3}_{region,city}(\pi^{A_4}_{category,item}(\pi^{A_5}_{name}(Aggregate_{item\rightarrow category;sum(sales)}($$
$$\sigma^{member}_{(region=north)\wedge(item=beer\vee item=wine\vee category=food)}($$
$$Navigate_{year,quarter,city,item,name}(I_{sales}))))))))))$$

The query q_{cf} can now be used as input of the function $Compute\_First\_Slice\_Selection$. Note that for notational purpose, the selection $\sigma^{member}_{(region=north)\wedge(item=beer\vee item=wine\vee category=food)}$ is rewritten $\circ_{i=1}^{5}\sigma^{member}_{\psi^{D_i}}$ with:

– $\psi^{D_1} = \psi^{D_2} = \psi^{D_5} = true$
– $\psi^{D_3} = (region = north)$
– $\psi^{D_4} = (item = beer \vee item = wine \vee category = food)$

We illustrate each step of the algorithm:

1) $\mathcal{A} = \{A_1, A_2, A_3, A_4, A_5\}$

2) For $i = 1$ to 5 we have:

i	k_i	$att(A_i)$	$loc(A_i)$
1	1	year	1
2	1	quarter	2
3	3	region, city	3
4	1	category, item	4
5	1	name	5

For instance, when $i = 4$, $d_i = 2$ since $item$, the level at depth 2, appears in the parameter list of the $Navigate$ operation. $l_k = 1$, since $category$, the level at depth 1, appears in the parameter list of the $Aggregate$ operation, and thus $k_4 = min(\{1\} \cup \{2\}) = 1$. $att(A_4)$ is the list of parameter of the π^{A_4} operation, and $loc(A_4)$ is set to 4.

3) The only Nest operation is $Nest_{A_5(A_3)}$. Hence $att(A_5) = \{name, region, city\}$, and $loc(A_4) = 3, loc(A_5) = 4$.

4) The only Permute operation is $Permute_{A_1,A_5}$. Hence $loc(A_1) = 4$ and $loc(A_5) = 1$.

5) The only Switch operation is $Switch_{year;1988,2000}$. Hence $pos(1988) = 13$ and $pos(2000) = 1$.

6) As we have $\mathcal{A}' = \{A_1, A_2, A_4, A_5\}$, l ranges from 3 to 4.

- When $l = 3$, $A_k = A_4$ and $att(A_4) = \{category, item\}$. Moreover, $k_4 = 1$, $L_4^1 = category \in att(A_4)$ and $\sigma_{\phi_4} = \sigma_{\psi^{D_4}} = (item = beer \vee item = wine) \vee (category = food)$. Thus:

$\pi_{category}(\sigma_{\phi_4}(unnest_{D_4^2}(unnest_{D_4^1}((I_{Product})))))) = \{drink, food\}$.

As $pos(drink) = 1 < pos(food) = 2$ (see Figure 1 (b)), we have $v = drink$ and $\varphi = true \wedge (category = drink)$.

- When $l = 4$, $A_k = A_1$ and $att(A_1) = \{year\}$. Moreover $k_1 = 1$, $L_1^1 = year \in att(A_1)$ and $\sigma_{\phi_1} = \sigma_{\psi^{D_1}} = true$. Thus:

$\pi_{year}(\sigma_{\phi_1}(unnest_{D_1^1}(I_{Year}))) = \{1988, \ldots, 2004\}$. As $pos(2000) = 1$, we have $v = 2000$ and $\varphi = (category = drink) \wedge (year = 2000)$.

7) The function returns the selection condition $\varphi = (category = drink) \wedge (year = 2000)$.

4.4. *Discussion*

It is important to note that our optimisation technique consists in a syntactical exploration of the query, and thus it does not require accessing the data of the cube from which the representation is computed. In that respect the cost of the optimisation can be seen as not significant compared to the global cost of processing the query.

Note also that, as mentioned in the introduction, our optimisation technique can be used in conjunction with other optimisation techniques, such as for instance techniques taking advantage of materialized aggregates. Optimising an OLAP query by combining the two techniques could be done, in a naive way, by first applying our technique to determine what is the first slice to be displayed, and then determine what are the materialized aggregates to be queried.

5. Conclusion

In this paper we have proposed a new way of optimizing OLAP queries, based on the output of a query on a screen. We have used the model of complex values to define a cube and a representation of a cube at a logical level. We defined an algebra for manipulating cubes and representations and we showed how the properties of the algebraic operators can be used to optimize queries.

We are currently working on the following two open issues. First, we are investigating a way for optimizing sequences of OLAP queries. For example, if a query $I' = Q_1(I)$ has been optimized, we can distinguish two cases for optimizing $Q_2(I')$:

– Take advantage of what has been done for optimizing Q_1, if it can be found that Q_2 can be computed only by looking at the first slice of I', or

– Use the rewriting rules for optimizing $Q_2(Q_1(I))$ otherwise.

Second, we are extending both the language and the definition of representations. In this respect, typical restructuring operators like $Push$, $Pull$, or $Unnest$ are under consideration, and we are thinking of defining more sophisticated representations, e.g., with nested cells. Finally we are currently implementing our approach.

6. References

[ABI 95] ABITEBOUL S., HULL R., VIANU V., *Foundations of Databases*, 1995.

[AGR 97] AGRAWAL R., GUPTA A., SARAWAGI S., "Modeling Multidimensional Databases", *Proc. ICDE*, 1997.

[CAB 97] CABIBBO L., TORLONE R., "Querying Multidimensional Databases", *Proc. DBPL*, vol. 1369 of *LNCS*, 1997.

[CHA 97] CHAUDHURI S., DAYAL U., "An Overview of Data Warehousing and OLAP Technology", *SIGMOD Record*, vol. 26, num. 1, 1997.

[DEK 98] DEKEYSER S., KUIJPERS B., PAREDAENS J., WIJSEN J., "Nested Data Cubes for OLAP", *Advances in Database Technologies*, vol. 1552 of *LNCS*, 1998.

[GIA 04] GIACOMETTI A., LAURENT D., MARCEL P., MOULOUDI H., "A new way of optimizing OLAP queries", Full version, 2004.

[GYS 97] GYSSENS M., LAKSHMANAN L. V. S., "A Foundation for Multi-dimensional Databases", *Proc. VLDB*, 1997.

[HAC 97] HACID M.-S., MARCEL P., RIGOTTI C., "A rule-based data manipulation language for OLAP systems", *Proc. DOOD*, vol. 1341 of *LNCS*, 1997.

[JAR 02] JARKE M., LENZERINI M., VASSILIOU Y., VASSILIADIS P., *Fundamentals of Data Warehouses*, Springer-Verlag, 2002.

[KAR 03] KARAYANNIDIS N., TSOIS A., VASSILIADIS P., SELLIS T., "Design of the ER-ATOSTHENES OLAP Server", *Advances in Informatics*, vol. 2563 of *LNCS*, 2003.

[MAN 03] MANIATIS A., VASSILIADIS P., SKIADOPOULOS S., VASSILIOU Y., "CPM: A Cube Presentation Model for OLAP", *Proc. DaWaK*, vol. 2737 of *LNCS*, 2003.

[MAR 99] MARCEL P., "Modeling and querying multidimensional databases: An overview.", *Networking and Information Systems Journal*, vol. 2, num. 5-6, 1999, p. 515–548.

[MEN 04] MENDELZON A., HURTADO C., LEMIRE D., "Data Warehousing and OLAP: A Research-Oriented Bibliography", Available at http://www.ondelette.com/OLAP/dwbib.html, 2004.

[Mic 98] MICROSOFT CORPORATION, "OLEDB for OLAP", Available at http://www.microsoft.com/data/oledb/olap, 1998.

[PAR 01] PARK C.-S., KIM M. H., LEE Y.-J., "Rewriting OLAP Queries Using Materialized Views and Dimension Hierarchies in Data Warehouses", *ICDE*, 2001, p. 512–523.

[VAS 00] VASSILIADIS P., SKIADOPOULOS S., "Modelling and Optimization Issues for Multidimensional Databases", *Proc. CAISE*, vol. 1789 of *LNCS*, 2000.

Appendix

In this appendix, we give the formal definitions of the algebraic operators of the language. Note that for this language, we consider only atomic operators, i.e., *Permute* only exchanges two axes, *Switch* only exchanges two members. In that sense, the *Rotate* operation cannot be considered as atomic and can be expressed as a combination of *Permutes* and *Switches*. All operators are defined on representations, except the $Navigate$ operator which is defined on a cube, and allows to construct an initial representation. The operators defined on representations are applied on an instance $I_R = \langle I_A, I_F, I_P \rangle$ of a representation R having sort $sort(R) = \langle A : sort(A), F : sort(F), P : sort(P) \rangle$, where $I_A = \langle I_{A_1}, \ldots, I_{A_K} \rangle$ and $sort(A) = \langle A_1 : sort(A_1), \ldots, A_K : sort(A_K) \rangle$.

Restructuring

Permute: The Permute operation consists of interchanging two axes of a representation. Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R. Given two axes A_i and A_j of R, $Permute_{A_i,A_j}(I_R)$ is an instance $I_{R'} = \langle I_{A'}, I_F, I_P \rangle$ of a representation R' having sort $sort(R') = \langle A' : sort(A'), F : sort(F), P : sort(P) \rangle$ where:

– $sort(A') = \langle A_1 : sort(A_1), \ldots, A_{i-1} : sort(A_{i-1}), A_j : sort(A_j), A_{i+1} : sort(A_{i+1}), \ldots, A_{j-1} : sort(A_{j-1}), A_i : sort(A_i), A_{j+1} : sort(A_{j+1}), \ldots, A_K : sort(A_K) \rangle$,

– $I_{A'} = \langle I_{A_1}, \ldots, I_{A_{i-1}}, I_{A_j}, I_{A_{i+1}}, \ldots, I_{A_{j-1}}, I_{A_i}, I_{A_{j+1}}, \ldots, I_{A_K} \rangle$.

Note that $Permute_{A_k,A_k}(I_R) = I_R$ for every $k \in [1, K]$.

Switch: Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R. Let A_i be an axis of R with $sort(A_i) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_i^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_i^S : \{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\} \ldots\rangle\}\rangle\}$.

The Switch operation consists of interchanging the positions p and p' of two values v and v' of an attribute $L_{i_k}^{j_k}$ on axis A_i. It is defined if $k = 0$, or if $i_k \neq i_{k-1}$, or if $i_k = i_{k-1}$ and $anc(v, X) = anc(v', X)$ where $X = L_{i_k}^{j_{k-1}}$.

$Switch_{L_{i_k}^{j_k};v,v'}(I_R)$ is an instance $I_{R'} = \langle I_A, I_F, I'_P \rangle$ of a representation R' having sort $sort(R') = sort(R)$, where:

– $I'_P = (I_P - I) \cup (\pi_{member,position}(\rho_{position \to P}(I) \times \rho_{member \to M}(I)) - I)$, with:

– $I = \sigma_{member=v \vee member=v'}(I_P)$

Nest: Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R. Let A_i and A_j be two axes of R with:

– $sort(A_i) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_i^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_i^S : \{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\} \ldots\rangle\}\rangle\}$.

– $sort(A_j) = \{\langle L_{k_0}^{l_0} : dom(L_{k_0}^{l_0}),\ U_j^1 : \{\langle L_{k_1}^{l_1} : dom(L_{k_1}^{l_1}), \ldots,\ U_j^T : \{\langle L_{k_T}^{l_T} : dom(L_{k_T}^{l_T})\rangle\} \ldots\rangle\}\rangle\}$.

The Nest[4] operation allows to nest the attributes initially on axis I_{A_i} with the attributes on axis I_{A_j}. $Nest_{A_i(A_j)}(I_R)$ is an instance $I_{R'} = \langle I_{A'}, I_F, I_P \rangle$ of a representation R' having sort $sort(R') = \langle A' : sort(A'), F : sort(F), P : sort(P)\rangle$ where:

– $sort(A') = \langle A_1 : sort(A_1), \ldots, A_i : \tau_{A_i}, \ldots, A_{j-1} : sort(A_{j-1}), A_{j+1} : sort(A_{j+1}),\ \ldots, A_K : sort(A_K)\rangle$, with $\tau_{A_i} = \{\langle L_{i_0}^{j_0},\ U_i^1 : \{\langle L_{i_1}^{j_1}, \ldots,\ U_i^S : \{\langle L_{i_S}^{j_S},\ U_i^{S+1} : \{\langle L_{k_0}^{l_0}, \ldots,\ U_i^{S+T+1} : \{\langle L_{k_T}^{l_T}\rangle\} \ldots\rangle\}\rangle\}\rangle\}\rangle\}$,

– $I_{A'} = tup\_create_{A_1,\ldots,A_i,\ldots,A_{j-1},A_{j+1},\ldots,A_K}(\pi_{A_1}(I_A), \ldots, I_{A'_i}, \ldots, \pi_{A_{j-1}}(I_A), \pi_{A_{j+1}}(I_A), \ldots, \pi_{A_K}(I_A))$, with $I_{A'_i}$ being defined by:

$I_{A'_i} = nest_{U_i^1=(L_{i_1}^{j_1},U_i^2)}(\ldots(nest_{U_i^S=(L_{i_S}^{j_S},U_i^{S+1})}(I_{ij}))\ldots)$, where:

- $I_{ij} = nest_{U_i^{S+1}=(L_{k_0}^{l_0},U_i^{S+2})}(\ldots(nest_{U_i^{S+T+1}=(L_{k_T}^{l_T})}(I_i \times I_j))\ldots)$, and
- $I_i = unnest_{U_i^S}(\ldots(unnest_{U_i^1}(\pi_{A_i}(I_A))))$,
- $I_j = unnest_{U_j^T}(\ldots(unnest_{U_j^1}(\pi_{A_j}(I_A))))$.

Filtering

Selection on measure: Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R. Let c be a constant in $dom(m)$. $\sigma_{m=c}^{measure}(I_R)$ is an instance $I_{R'} = \langle I_A, I'_F, I_P \rangle$ of a representation R' having sort $sort(R') = sort(R)$ and where $I'_F = \sigma_{m=c}(I_F)$.

Selection on members: Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R and A_k be an axis of R with $sort(A_k) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_k^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_k^S :$

4. This OLAP operation should not be mistaken with the nest operation of the algebra for complex values [ABI 95].

$\{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\}\ldots\rangle\}\rangle\}$. Given an attribute $L_{i_x}^{j_x}$ in $\mathcal{A}_k(R)$ and a constant $v \in dom(L_{i_x}^{j_x})$, the selection $\sigma^{members}_{L_{i_x}^{j_x}=v}(I_R)$ is defined if there exists an attribute $L_i^{d_i} \in \mathcal{F}(R)$ such that $i_x = i$ and $j_x \leq d_i$. In that case, $\sigma^{members}_{L_{i_x}^{j_x}=v}(I_R)$ is an instance $I_{R'} = \langle I'_A, I'_F, I_P\rangle$ of a representation R' having sort $sort(R') = sort(R)$ where:

– $I'_A = \langle I_{A_1}, \ldots, I_{A'_k}, \ldots, I_{A_K}\rangle$, with:

- $I'_{A_k} = nest_{U_k^1=(L_{i_1}^{j_1},U_k^2)}(\ldots nest_{U_k^S=(L_{i_S}^{j_S})}(\sigma_{L_{i_x}^{j_x}=v}(unnest_{U_k^S}(\ldots (unnest_{U_k^1}(I_{A_k}))\ldots)$,

– $I'_F = \sigma_\varphi(I_F)$ where:

- $\varphi = \bigvee_{v'\in X}(L_i^{d_i} = v')$
- $X = \{v' \in dom(L_i^{d_i}) \mid anc(v', L_{i_x}^{j_x}) = v\}$.

Projection: Let I_R be an instance of a representation R and A_i be an axis of R with $sort(A_i) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_i^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_i^S : \{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\}\ldots\rangle\}\rangle\}$. Given a subset $\{L_{x_0}^{y_0}, \ldots, L_{x_p}^{y_p}\}$ of $\mathcal{A}_i(R)$ such that for every $i, j \in [1, p]$, if $x_i = x_j$, then $y_i \leq y_j$, the projection $\pi^{A_i}_{L_{x_0}^{y_0},\ldots,L_{x_p}^{y_p}}(I_R)$ is defined if we have: $\{L_{i_k}^{j_k} \in \mathcal{A}_i(R) \mid (\exists L_i^{d_i} \in \mathcal{F}(R))(i = i_k \wedge j_k \geq d_i)\} \subseteq \{L_{x_0}^{y_0}, \ldots, L_{x_p}^{y_p}\}$. In that case, $\pi^{A_i}_{L_{x_0}^{y_0},\ldots,L_{x_p}^{y_p}}(I_R)$ is an instance $I_{R'} = \langle I_{A'}, I_F, I_P\rangle$ of a representation R' having sort $sort(R') = \langle A' : sort(A'), F : sort(F), P : sort(P)\rangle$ where:

– $sort(A') = \langle A_1 : sort(A_1), \ldots, A_i : \tau_{A_i}, \ldots A_K : sort(A_K)\rangle$, with:

$\tau_{A_i} = \{\langle L_{x_0}^{y_0} : dom(L_{x_0}^{y_0}),\ U_i^1 : \{\langle L_{x_1}^{y_1} : dom(L_{x_1}^{y_1}), \ldots, U_i^p : \{\langle L_{x_p}^{y_p} : dom(L_{x_p}^{y_p})\rangle\}\ldots\rangle\}\rangle\}$.

– $I_{A'} = \langle I_{A_1}, \ldots, I_{A'_i}, \ldots, I_{A_p}\rangle$, with:

- $I_{A'_i} = nest_{U_i^1=(L_{x_1}^{y_1},U_i^2)}(\ldots (nest_{U_i^p=(L_{x_p}^{y_p})}(I))\ldots)$,
- $I = \pi_{L_{x_0}^{y_0},\ldots,L_{x_p}^{y_p}}(unnest_{U_i^S}(\ldots (unnest_{U_i^1}(\pi_{A_i}(I_A)))))$.

Changing the level of detail

Navigate: Let $I_C = \langle I_{A_{All}}, I_{F_{All}}, I_{P_{All}}\rangle$ be an instance of a N-dimensional cube C. Let $L_1^{d_1}, \ldots, L_N^{d_N}$ be such that for every $i \in [1, N]$, $L_i^{d_i} \in \mathcal{L}_i(R)$. $Navigate_{L_1^{d_1},\ldots,L_N^{d_N}}(I_C)$ is an instance $I_R = \langle I_A, I_F, I_P\rangle$ of a representation R having sort $sort(R) = \langle A : sort(A), F : sort(F), P : sort(P)\rangle$, where:

– $sort(A) = \langle A_1 : sort(A_1), \ldots, A_N : sort(A_N)\rangle$, where for every $i \in [1, N]$, $sort(A_i) = \{\langle L_i^0 : dom(L_i^0), U_i^1 : \{\langle L_i^1 : dom(L_i^1), \ldots, U_i^{d_i} : \{\langle L_i^{d_i} : dom(L_i^{d_i})\rangle\}\ldots\rangle\}\rangle\}$,

– $sort(F) = \{\langle L_1^{d_1} : dom(L_1^{d_1}), \ldots, L_N^{d_N} : dom(L_N^{d_N}), m : dom(m)\rangle\}$,

– $I_A = \langle I_{A_1}, \ldots, I_{A_N}\rangle$ where for all $i \in [1, N]$, I_{A_i} is defined by: $I_{A_i} = nest_{U_i^1=(L_i^1,U_i^2)}(\ldots nest_{U_i^{d_i}=(L_i^{d_i})}(\pi_{L_i^0,\ldots,L_i^{d_i}}(unnest_{D_i^{q_i}}(\ldots (unnest_{D_i^1}(I_{D_i})))))$,

– $I_F = \pi_{L_1^{d_1},\ldots,L_N^{d_N},m}(I_{F_{All}} \bowtie_{L_1^{d_1}=F_1^{all}} Level_1 \ldots \bowtie_{L_N^{d_N}=F_N^{all}} Level_N)$, with $Level_i = \pi_{L_i^{d_i}}(unnest_{D_i^{q_i}}(\ldots (unnest_{D_i^1}(I_{D_i})))))$, for every $i \in [1, N]$,

– $I_P = I_{P_{All}}$.

Aggregate: The aggregate operator allows to apply an aggregate function on the facts of a representation. This operator relies on the extended projection operation as defined in [ABI 95].

Let $I_R = \langle I_A, I_F, I_P \rangle$ be an instance of a representation R and A_i an axis of R with $sort(A_i) = \{\langle L_{i_0}^{j_0} : dom(L_{i_0}^{j_0}),\ U_i^1 : \{\langle L_{i_1}^{j_1} : dom(L_{i_1}^{j_1}), \ldots,\ U_i^S : \{\langle L_{i_S}^{j_S} : dom(L_{i_S}^{j_S})\rangle\}\ldots\rangle\}\rangle\}$.

Let f be an aggregate functions defined on $dom(m)$. Given two attributes $L_{i_x}^{j_x}$ and $L_{i_y}^{j_y}$ in $\mathcal{A}_i(R)$ such that $L_{i_x}^{j_x} \in \mathcal{F}(R)$, $k = i_x = i_y$ and $d_k = j_x > j_y$, the aggregation $Aggregate_{L_{i_x}^{j_x} \rightarrow L_{i_y}^{j_y};\ f(m)}(I_R)$ is an instance $I_{R'} = \langle I_A, I_{F'}, I_P \rangle$ of a representation R' having sort $sort(R') = \langle A : sort(A), F' : sort(F'), P : sort(P)\rangle$, where:

– $sort(F') = \{\langle L_1^{d_1} : dom(L_1^{d_1}), \ldots, L_{k-1}^{d_{k-1}} : dom(L_{k-1}^{d_{k-1}}), L_k^{j_y} : dom(L_k^{j_y}), L_{k+1}^{d_{k+1}} : dom(L_{k+1}^{d_{k+1}}), \ldots, L_N^{d_N} : dom(L_N^{d_N}), m : dom(m)\rangle\}$,

– $I_{F'} = \pi_{L_1^{d_1},\ldots,L_{k-1}^{d_{k-1}},L_k^{j_y},L_{k+1}^{d_{k+1}},\ldots,L_N^{d_N};\ f(m)}(I)$, with $I = I_F \bowtie_{L_{i_x}^{j_x}} unnest_{U_i^S}(\ldots(unnest_{U_i^1}(I_{A_i})))$.