OLAP query optimization : A Framework for Combining Rule-Based and Cost-Based Approaches

Ladjel Bellatreche*, Arnaud Giacometti **, Dominique Laurent ***,
Patrick Marcel **, Hassina Mouloudi**

* LISI/ENSMA
Téléport 2, 1, ave. Clément Ader 86960 Futuroscope - France
bellatre@ensma.fr
** Université Francois-Rabelais de Tours, LI
Master SIR, Antenne Universitaire, 3 place Jean Jaurès 41000 Blois
(arnaud.giacometti, patrick.marcel, hassina.mouloudi)@univ-tours.fr
** Université de Cergy-Pontoise
2, ave. A. Chauvin - BP 222 - 95302 Cergy-Pontoise Cedex - France
dominique.laurent@dept-info.u-cergy.fr

Abstract. To optimize queries in relational databases, two categories of optimization techniques have been proposed : the Rule-Based Approach (RBA), and the Cost-Based Approach (CBA). In the RBA, the optimizer uses rule transformations using the relational algebra. In the CBA, the optimizer uses a cost model to estimate the potential cost of each operation using statistics about the database and the tables involved in the query. Usually both categories are implemented by commercial DBMSs and are often intermixed. In multidimensional databases however, most of query optimization techniques follow only the CBA to select optimization structures such as : materialized views, advanced indexing schemes and data partitioning. No approach has been proposed yet to rewrite OLAP queries using a multidimensional algebra. In this paper, we show that the RBA can be applied to multidimensional databases by rewriting each OLAP query to obtain an efficient rewritten query that can be executed using a CBA. In particular, we show that the RBA can be used to take into account one of the specificities of OLAP which is the visualization of the OLAP query result. We propose a multidimensional algebra that represents the core of our RBA optimization, and we show how rewritten queries can be processed using the CBA proposed for multidimensional databases.

1 Introduction

A data warehouse (DW) integrates massive amounts of data from multiple sources. In a DW, users access very large databases to carry out strategic analysis for maintaining business competitiveness by executing complex OLAP queries [Karloff et Mihail, 1999]. This complexity is due to the presence of join and aggregation operations. Therefore, an efficient query processing becomes a critical issue. To optimize these complex queries, several techniques were proposed that we can divide into two categories : *redundant-structures* and *non redundant-structures*. In the first category, we can find materialized views [Gupta, 1997, Kotidis et Roussopoulos, 1999, Ross *et al.*, 1996, Theodoratos et Sellis, 1997, Yang *et al.*, 1997] and indexing schemes (b-tree, bitmap, join indexes, bitmap join indexes, etc.) [Chaudhuri, 2004]. All these structures need a space storage and a maintenance overhead. Structures in the second category are those which do not need an extra storage space. For example, horizontal and vertical data partitioning [Bellatreche *et al.*, 2004, Sanjay *et al.*, 2004], parallel processing [Molina *et al.*, 1998]. All these structures are supported by commercial systems [Zilio *et al.*, 2004, Sanjay *et al.*, 2004].

We focus on relational data warehouses, where a data cube is stored using a star schema [Kimball, 1996]. The database thus consists of a huge fact table and multiple dimensions tables. Dimensions are hierarchically structured. Queries typically perform aggregations on the fact table based on selection among the available dimension levels. These queries are called star join queries which can be optimized using redundant structures as bitmap indexes. But these structures still involve substantial processing and I/O cost for high cardinality attributes and thus high storage overhead.

In the traditional databases, query optimization is done using two approaches : *rule based approach* (RBA), and *cost-based approach* (CBA). In the RBA, the optimizer uses rule transformations using the relational algebra. A set of rewriting rules can be used to generate directly the optimized form of the query. In general, these rewriting rules are based on relational algebra equivalences. For example, since a selection can commute with a join (joins are typically expensive operations [Lei et Ross, 1998]), a classical rewriting rule pushes the selection conditions ahead of the joins, or picks the most "promising" relation to join next (Oracle). In the CBA, a cost model assigns an estimated cost to any partial or complete plan in the search space. It also determines the estimated size of the data stream for output of every relational operator (selection, projection, join, etc.) in the plan. This cost model can estimate the CPU and I/O costs of query execution for every operator, by taking into account the statistical properties of its inputs data streams, or its existing access methods. The accuracy of the cost estimation depends on both the quality of the cost model and on the accuracy of the statistical information used. Most of commercial systems offer the two approaches.

In an OLAP environment, no approach has been proposed yet to rewrite OLAP queries using a multidimensional algebra. In this paper, we show that the RBA can be applied to multidimensional databases by rewriting each OLAP query to obtain an efficient plan that can be executed using a CBA (see Figure 1). In particular, we show that the RBA can be used to take into account one of the specificities of OLAP which is the visualization of the OLAP query result.

To reach this goal, we propose to describe both datacubes and their structure in a single logical model. Then, we translate the main *OLAP* operators in our model and give rewriting rules involving these operators. These rewriting rules can be used to obtain the optimized form of an *OLAP* query. For example, to optimize aggregation operation, we propose a rewriting rule that pushes the selection conditions on members ahead of the aggregates. Note that in this paper we do not take into account the physical definition of the algebraic operators.

In this framework, we also study the possibilities of optimizing *OLAP* queries based on the visualization of a cube on a screen. Our proposed optimization technique consists in determining which part of the query output will be displayed on the screen. In general, this part is the first two-dimensional slice of the cube, which is also a cube. In this article, we show that this slice can be computed by adding selection conditions to the initial $OLAP$ query. Note these conditions are mainly obtained by computing the structure of the cube to be visualized.

The intuition behind our optimization approach is as follows : given a cube C and an $OLAP$ query q over C, let us denote by $q(C)$ the answer to q. We first compute the selection conditions φ that defines the first two-dimensional slice of $C' = q(C)$. This step requires the computation of the structure of the cube C'. Then, we add the selection conditions φ to the initial $OLAP$ query and we use the rewriting rules to push them ahead of the aggregates and joins. The rewriting process is done in the main memory without accessing base tables. Finally, the rewritten query will be executed using CBA.

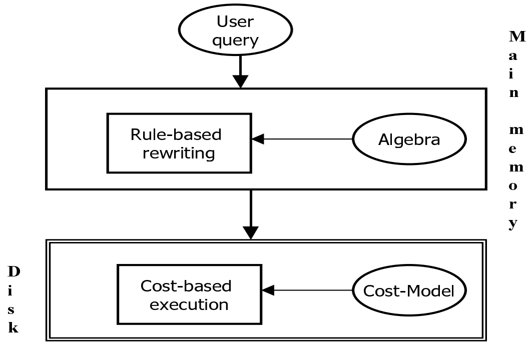

FIG. 1 – The Execution Strategy using our Approach

To the best of our knowledge, this is the first paper combining RBA and CBA approaches to optimize OLAP queries. The main contributions of this paper are that : (a) we propose a logical model to describe both datacubes and their structures, this model represents the core of our RBA approach (Section 3 and 4), (b) we propose an optimization technique of OLAP queries using rewriting rules (Section 5), and (c) we show the utility of combining the two approaches through a cost model (Section 6). We start by motivating our approach in the next section.

2 A Motivating Example

Let us consider the cube C_0, inspired by the example given in [Corporation, 1998], and the star schema of which is presented Figure 2. The dimensions of this cube 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), and $Salesman$ (the different salespersons).

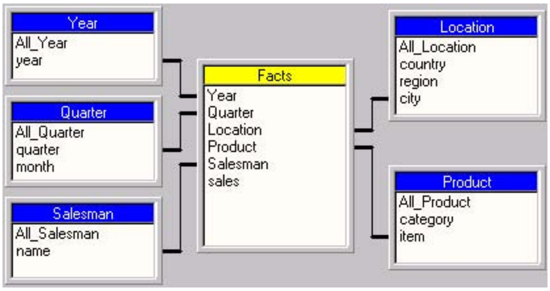

FIG. 2 – The star schema of the cube C_0

All the dimensions are hierarchically structured, for example the hierarchy of dimension *Quarter* is : *All_Quarter* $\rightarrow$ *quarter* $\rightarrow$ *month*.

Consider now the following *OLAP* query q that aims to at presenting the cumulated sales of *food* and cumulated sales of *beer* and *wine*, detailed by salespersons and quarters. Using the *MDX* language proposed by *Microsoft* [Corporation, 1998], q can be expressed by :

```
WITH MEMBER drink.Mydrink AS 'wine + beer'
  SELECT
  {[quarter].MEMBERS} ON COLUMNS,
  CROSSJOIN([Name].MEMBERS, [North].CHILDREN) ON ROWS
  {Mydrink, Food} ON PAGES
  year.MEMBERS ON SECTIONS
  FROM Co
  WHERE [sales]
```

The final output of this query is presented Figure 3. Note that in this figure, we only visualize the first two-dimensional slice of the answer of the query. Indeed, we only see the cumulated sales of drinks *beer* and *wine* for year 1988. Thus, in order to obtain the visualization presented Figure 3, it is neither necessary to compute the cumulated sales of *food*, nor the cumulated sales of drinks *beer* and *wine* for years other than year 1988. In our approach, it means that we will add to the initial query q the selection conditions ($category = drink$) and ($year = 1988$). Then, we will push these selection conditions ahead of the aggregates.

In our approach, we know precisely that the data visualized Figure 3 are the cumulated sales of drinks *beer* and *wine* for year 1988. It follows from the fact that in our model, the position of the members on the axes are explicitly represented. Thus, if the user wants to see the cumulated sales of another category of product or for another year, he has to use the restructuring operators of the language. For example, he can switch the position of year 1988 with the position of another year to see the cumulated sales for another year. Or he can nest the axes A_2 and A_4 to visualize the cumulated sales of *beer* and *wine* for every year.

A$_1$							**A$_3$**
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	**A$_4$**
		blois	20	30	20	20	**1988**
		paris	50	50	90	80	
A$_2$			q_1	q_2	q_3	q_4	

FIG. 3 – Visualization of the MDX query q

3 Cube Model

Our notion of cube extends the classical star-schema model [Kimball, 1996] by adding a structure component. This structure component defines how the cube is to be displayed to the user. The structure component models a multidimensional cross-tab, and allows to precisely describe e.g., on which axis the members of a dimension are located and in which order. In our approach, cubes are manipulated mainly by using the classical restructuring OLAP operators like nest or switch [Marcel, 1999]. We now turn to the formal definition.

Cube An N-dimensional cube C is a triple $C = \langle \mathcal{D}, F, S \rangle$ where :
- $\mathcal{D} = \{D_1, \ldots, D_N\}$ is the set of dimension tables D_i of C, $i \in [1, N]$,
- F is the fact table of C, that describes the facts at a particular level of detail,
- S is the structure of C.

The components $\mathcal{D}$, F and S of a given cube C are defined and illustrated below.

As usual in a star-schema, a dimension is a relation that represents each level of details of a hierarchy over which the facts can be aggregated.

Dimension tables A dimension $D_i, i \in [1, N]$, is a relation of schema $sch(D_i) = \{L_i^0 : dom(L_i^0), L_i^1 : dom(L_i^1), \ldots, L_i^{q_i} : dom(L_i^{q_i})\}$.

Each attribute L_i^j represents a level of detail j in the dimension i, and thus $dom(L_i^j)$ is the set of members at level j for dimension D_i. q_i is the deepest level of detail for dimension D_i. If $v \in L_i^j$ is a member, the ancestor $anc(v)$ of v is $v' \in dom(L_i^{j-1})$ such that $v' = \pi_{L_i^{j-1}}(\sigma_{L_i^j = v}(D_i))$.

Example 1 *We consider two cubes :*
- *The cube C_0 of Section 2, and the star schema of which is presented Figure 2. In our model, C_0 is a tuple $C_0 = \langle \mathcal{D}_0, F_0, S_0 \rangle$.*
- *The cube C_6 which is the result of the query q of Section 2 over the cube C_0, and which first slice can be seen Figure 5. In our model, $C_6 = \langle \mathcal{D}_6, F_6, S_6 \rangle$.*

Let us detail the dimensions of cubes C_0 and C_6.

$\mathcal{D}_0 = \{Year, Quarter, Location, Product, Salesman\}$ *with :*

- $sch(Year) = \{All\_Year, year\}$,
- $sch(Quarter) = \{All\_Quarter, quarter, month\}$,
- $sch(Location) = \{All\_Location, country, region, city\}$,
- $sch(Product) = \{All\_Product, category, item\}$,
- $sch(Salesman) = \{All\_Salesman, name\}$.

Let us detail the dimension Quarter of the cube C_0*. This dimension is a relation over the schema* $sch(Quarter) = \{All\_Quarter, quarter, month\}$*. The domains of the attributes All_Quarter, quarter and month are respectively :* $dom(All\_Quarter) = \{all\}$, $dom(quarter) = \{q_1, q_2, q_3, q_4\}$, $dom(month) = \{jan, \ldots, dec\}$*. A tuple of this dimension is* $\langle all, q_1, mar\rangle$*. Obviously we have that* $anc(mar) = q_1$*.*

The dimensions of the cube C_6 *are the following :*

$\mathcal{D}_6 = \{Year', Quarter', Location', Product', Salesman'\}$ *where :*

- $sch(Year') = \{year\}$,
- $sch(Quarter') = \{quarter\}$,
- $sch(Location') = \{region, city\}$,
- $sch(Product') = \{category, item\}$,
- $sch(Salesman') = \{name\}$.

Note that the dimension $Quarter'$ *of the cube* C_6 *is a relation over the schema* $sch(Quarter') = \{quarter\}$*, since quarter is the only attribute of the dimension Quarter the user wants to see for the cube* C_6*.*

As for the dimensions, the fact table is defined in much the same way as in a star-schema.

Fact table A fact table F is a relation of schema $sch(F) =$
$\{L_1^{d_1} : dom(L_1^{d_1}), \ldots, L_N^{d_N} : dom(L_N^{d_N}), m_1 : dom(m_1), \ldots, m_p : dom(m_p)\}$
such that, for every i, $0 \leq d_i \leq q_i$.

An instance of F is the set of facts of the cube at level d_i for dimension D_i. The m_i are attributes describing the measures.

Example 2 *The fact table of the cube* C_0 *is a relation over the schema* $sch(F_0) = \{year, month, city, item, name, sales\}$*, where sales is a measure attribute and the facts are presented at the deepest level of details in each dimension. An example of fact in* F_0 *is the tuple* $\langle 1988, jan, paris, milk, john, 07\rangle$*.*

The fact table of the cube C_6 *is a relation over the schema* $sch(F_6) = \{year, quarter, city, category, name, sales\}$*. An example of fact in* F_6 *is the tuple* $\langle 1988, q_1, paris, drink, john, 50\rangle$*.*

To precisely describe the cross-tab used to display a cube, we need to know the following information : the number of axes of the cross-tab, what are the dimensions on the different axes, what are the positions of the members on an axis, and at which level of details the measures are represented.

The structure The structure S of the cube is a 4-tuple $S = \langle K, axis, pos, depth\rangle$ where :

- K is the number of axes.
- $axis$ is a function that defines which dimensions appear on which axis, and specifies the order of the dimensions on each axis. It mapps each integer i in $[1, K]$ to a totally ordered set $(E_i, \prec_i)$ where $E_i \subseteq \mathcal{D}$.
- pos is a function that associates to every attribute L_i^j of each dimension D_i a total order on it. $pos(L_i^j)$ specifies the order of the members on an axis.
- $depth$ indicates the level of detail of a fact for a given dimension. It is a function defined from $\mathcal{D}$ to $\bigcup_{D_i \in \mathcal{D}} sch(D_i)$. We can note that $depth(D_i) = L_i^{d_i}$ if $sch(F) = \{L_1^{d_1}, \ldots, L_N^{d_N}\}$.

Example 3 *The structure of the cube C_0 is $S_0 = \langle 5, axis, pos, depth\rangle$ where :*

- *The function axis is defined by :*
  - $axis(1) = (\{Year\}, \prec_1)$,
  - $axis(2) = (\{Quarter\}, \prec_2)$,
  - $axis(3) = (\{Location\}, \prec_3)$,
  - $axis(4) = (\{Product\}, \prec_4)$ *and*
  - $axis(5) = (\{Salesman\}, \prec_5)$.

  Note that since the ordered sets $axis(k), k \in [1, 5]$ are singletons, the orders $\prec_k$ are trivial.
- *Let $<_{year}$, $<_{category}$ and $<_{item}$ be the total orders associated by function pos to attributes year, category and item, respectively. They are such that :*
  - $1988 <_{year} \ldots <_{year} 2004$,
  - $drink <_{category} \ldots <_{category} food$,
  - $milk <_{item} \ldots <_{item} beer$.
- *The function depth is defined by $depth(Year) = year$, $depth(Quarter) = month$, $depth(Location) = city$, $depth(Product) = item$ and $depth(Salesman) = name$.*

The visualization of C_6 of Figure 5(c) shows a case of nesting several dimensions on the same axis. The structure of the cube C_6 is $S_6 = \langle 4, axis', pos', depth'\rangle$ where :

- *The function $axis'$ is defined by :*
  - $axis'(1) = (\{Salesman, Location\}, \prec'_1)$ *with* $Salesman \prec'_1 Location$,
  - $axis'(2) = (\{Quarter\}, \prec'_2)$,
  - $axis'(3) = (\{Product\}, \prec'_3)$,
  - $axis'(4) = (\{Year\}, \prec'_4)$.
- *For every attribute L_i^j, $pos'(L_i^j)$ is the restriction of $pos(L_i^j)$ to $adom(L_i^j)$.*
- *The function $depth'$ is defined by $depth'(Year) = year$, $depth'(Quarter) = quarter$, $depth'(Location) = city$, $depth'(Product) = category$ and $depth'(Salesman) = name$.*

4 Operations

In this section, we translate the most typical OLAP operators [Marcel, 1999] into our model. We consider the following OLAP operators, that are classified according to 3 categories :

	$\mathbf{A_1}$	jan	...	dec
	2004	04		02
	2003	30		30
	2002	20		70
	2001	10		60
	2000	10		40
	1999	20		02
	1998	15		06
all_y	1997	20		30
	1996	02		05
	1995	20		60
	1994	10		40
	1993	01		40
	1992	02		20
	1991	60		02
	1990	30		40
	1989	10		08
	1988	07		70
		q_1	...	
	$\mathbf{A_2}$		all_q	

$\mathbf{A_3}$
all_l
↑
france
↑
north
↑
paris
$\mathbf{A_4}$
all_p
↑
drink
↑
milk
$\mathbf{A_5}$
all_s
↑
john

The first 2-dimensional slice of C_0

	$\mathbf{A_1}$	q_1	q_2	q_3	q_4
	2004	10	20	20	10
	2003	50	70	50	60
	2002	60	60	80	80
	2001	70	50	40	70
	2000	40	80	10	80
	1999	50	30	10	10
	1998	50	30	70	10
all_y	1997	40	50	30	70
	1996	10	40	50	10
	1995	40	50	80	70
	1994	50	80	30	50
	1993	10	70	40	50
	1992	10	80	40	30
	1991	80	40	50	10
	1990	70	70	10	50
	1989	40	50	50	10
	1988	20	50	70	80
	$\mathbf{A_2}$		all_q		

$\mathbf{A_3}$
all_l
↑
france
↑
north
↑
paris

$\mathbf{A_4}$
all_p
↑
drink
↑
milk

$\mathbf{A_5}$
all_s
↑
john

(a) $C_1 = Aggregate_{month \rightarrow quarter, sum(sales)}(C_0)$

$\mathbf{A_1}$				
2004	10	20	20	10
2003	50	70	50	60
2002	60	60	80	80
⋮				
1995	40	50	80	70
1994	50	80	30	50
1993	10	70	40	50
1992	10	80	40	30
⋮				
1989	40	50	50	10
1988	20	50	70	80
$\mathbf{A_2}$	q_1	q_2	q_3	q_4

$\mathbf{A_3}$
north
↑
paris

$\mathbf{A_4}$
drink
↑
milk

$\mathbf{A_5}$
john

(b) $C_2 = \pi_{year}(\pi_{quarter}(\pi_{region,city}(\pi_{category,item}(name(C_1)))))$

$\mathbf{A_1}$				
2004	40	30	20	40
2003	50	50	40	50
2002	60	40	30	50
⋮				
1992	70	80	50	70
1991	60	40	60	80
1990	60	70	50	40
1989	50	50	30	10
1988	30	40	70	70
$\mathbf{A_2}$	q_1	q_2	q_3	q_4

$\mathbf{A_3}$
north
↑
paris

$\mathbf{A_4}$
drink
↑
beer

$\mathbf{A_5}$
john

(c) $C_3 = \sigma^{member}_{item=beer \vee item=wine \vee category=food}(C_2)$

FIG. 4 – Outputs of steps 1-3. On each figure, the left-hand side displays the first 2-dimensional slice of the cube, and the right-hand side displays the member at the lowest position for each hidden axis.

$\mathbf{A_1}$				
2004	60	60	40	50
2003	80	90	90	90
2002	70	50	70	80
⋮				
1992	80	90	60	80
1991	80	50	70	90
1990	70	80	60	50
1989	90	90	50	20
1988	50	50	90	80
$\mathbf{A_2}$	q_1	q_2	q_3	q_4

$\mathbf{A_3}$
north
↑
paris

$\mathbf{A_4}$
drink
↑
beer,wine

$\mathbf{A_5}$
john

(a) $C_4 = Aggregate_{item \rightarrow category; sum(sales)}(C_3)$

$\mathbf{A_1}$				
bill	30	90	50	60
rose	30	10	90	90
irma	70	60	70	10
kate	90	60	50	10
lara	90	90	70	10
averell	10	30	30	90
jack	10	60	50	60
joe	90	70	30	10
john	50	50	90	80
$\mathbf{A_2}$	q_1	q_2	q_3	q_4

$\mathbf{A_3}$
north
↑
paris

$\mathbf{A_4}$
drink
↑
beer,wine

$\mathbf{A_5}$
1988

(b) $C_5 = Permute_{5,1}(C_4)$

$\mathbf{A_1}$						
bill	north	lille	40	60	60	70
		blois	10	20	30	20
		paris	30	90	50	60
⋮						
john	north	lille	70	70	50	50
		blois	20	30	20	20
		paris	50	50	90	80
$\mathbf{A_2}$			q_1	q_2	q_3	q_4

$\mathbf{A_3}$
drink
↑
beer,wine

$\mathbf{A_4}$
1988

(c) $C_6 = Nest_{1(3)}(\sigma_{region=north}(C_5))$

FIG. 5 – Outputs of steps 4-6

- Restructuring operators that change the viewpoint on data. Operators in this category are *Permute, Switch, Nest, Order by.*
- Operator that change the level of detail. We consider the *Aggregate* operator that groups the members of a dimension and then aggregates the measures accordingly.
- Filtering operators, that are mainly the extension of classical selection and projection to cubes.

We introduce the different operators on an example, the formal definitions are given in the Appendix.

Example 4 *We use as an example the following query over C_0, which is the translation of the query q of Section 2 into our algebra :*

$$Nest_{1(3)}(\sigma^{member}_{region=north}(Permute_{1,5}(Aggregate_{item \rightarrow category; sum(sales)}(\sigma^{member}_{item=beer \vee item=wine \vee category=food}(\pi_{year}(\pi_{quarter}(\pi_{region,city}(\pi_{category,item}(\pi_{name}(Aggregate_{month \rightarrow quarter; sum(sales)}(C_0)))))))))))$$

We decompose the query and examine the different steps independently. The first step, i.e. :
$C_1 = Aggregate_{month \rightarrow quarter; sum(sales)}(C_0)$*, allows to present the data at level quarter by rolling up from the current level month of the C_0 cube.*

We now illustrate the filtering operations in step 2 and 3 of the query :

– *step 2 :* $C_2 = \pi_{year}(\pi_{quarter}(\pi_{region,city}(\pi_{category,item}(\pi_{name}(C_1)))))$

– *step 3 :* $C_3 = \sigma^{member}_{item=beer \vee item=wine \vee category=food}(C_2)$

The outputs of these two steps are given in Figure 4(b) and 4(c), respectively, where we see the first slice of the resulting cubes. These two steps reduce the amount of displayed facts (for selection) and displayed hierarchies (for projection).

Remark 1 *Note that after step 3, the displayed slice has changed. This is due to the fact that the item milk, that was at the lowest position on axis A_4 in C_2 is no more selected in cube C_3. Then among the selected items, the one at the lowest position, beer, is displayed first. Note also that in our formalism, a selection on members changes the dimension part of the cube in such a way that a dimension represents the hierarchy that has been used to aggregate the data. Thus, for a given dimension, selection on members cannot be applied on a member attribute that identifies a level deeper than the level currently displayed (e.g., selection cannot be applied on the item level if the facts are depicted at the category level). Otherwise the measures displayed would no more correspond to the hierarchy displayed. The same remark holds for the projection operation. Finally, note that the projection operation allows to choose among all the levels of a hierarchy, the ones that the user wants to see, since by default all the hierarchical levels are displayed.*

Step 4, i.e. $C_4 = Aggregate_{item \rightarrow category; sum(sales)}(C_3)$ *illustrates a way of changing the level of detail of the cube. Intuitively, the Aggregate operation groups and aggregates the data according to the groupings defined by the dimensions. The output of step 4 is given in Figure 5(a).*

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), nest two axes (Nest) or order the members on an axis (Orderby). Permute and Nest are illustrated by the following steps :

– *step 5 :* $C_5 = Permute_{5,1}(C_4)$ – *step 6 :* $C_6 = Nest_{1(3)}(\sigma_{region=north}(C_5))$

The outputs of these two steps are given in Figure 5. Note that these operations do not change the facts of the cubes, but only the way they are displayed.

Summary of the operators The table of Figure 6 summarizes the influence of the operators over the three components $\mathcal{D}$, F and, S of a cube $C = \langle \mathcal{D}, F, S \rangle$.

Note that q can be decomposed in q_1, q_2, q_3 such that : $C' = \langle q_1(\mathcal{D}), q_2(F, \mathcal{D}), q_3(S) \rangle$. This follows from the definition of the algebraic operators and the definition of a cube. For example the structure of a cube is independent from both its facts and its dimensions, and thus the structure of the result of a query can be computed independently from its facts and dimensions.

Rewriting rules Figure 7 shows the rules we defined for transforming any OLAP query. Note that the last rule r_{13} is valid since in our formalism, selection cannot be applied on an attribute that identifies a level deeper than the displayed level (see Remark 1). For example, on cube C_6 displayed in Figure 5(d), selection on item, e.g., *item* = *beer* is not allowed, because facts are displayed at the category level. To express

Changes	$\mathcal{D}$	F	S
Permute			$\surd$
Switch			$\surd$
Nest			$\surd$
Order by			$\surd$
Project members	$\surd$		$\surd$
Project measures		$\surd$	
Aggregate		$\surd$	$\surd$
Select members	$\surd$	$\surd$	$\surd$
Select measures		$\surd$	

FIG. 6 – Influence of the algabraic operators

$$
\begin{array}{rcll}
Nest_{i(j)}(Permute_{k,l}(C)) & = & Permute_{k',l'}(Nest_{i(j)}(C)) \text{ if } i,j \notin \{k,l\} & r_1 \\
Nest_{i(j)}(Permute_{i,l}(C)) & = & Permute_{i',l'}(Nest_{l(j)}(C)) \text{ if } l \notin \{i,j\} & r_2 \\
Nest_{i(j)}(Permute_{j,l}(C)) & = & Rotate_{[l',j']}(Nest_{i(l)}(C)) \text{ if } l \notin \{i,j\} & r_3 \\
Nest_{i(j)}(Permute_{k,j}(C)) & = & Rotate_{[k',j']}(Nest_{i(k)}(C)) \text{ if } k \notin \{i,j\} & r_4 \\
Nest_{i(j)}(Permute_{i,j}(C)) & = & Rotate_{[i',j']}(Nest_{j(i)}(C)) & r_5 \\
Nest_{i(j)}(Switch_{L_k^l;v,v'}(C)) & = & Switch_{L_k^l;v,v'}(Nest_{i(j)}(C)) & r_6 \\
Nest_{i(j)}(\pi_{L_i^{j_1},\dots,\ L_i^{j_p}}(C)) & = & \pi_{L_i^{j_1},\dots,\ L_i^{j_p}}(Nest_{i(j)}(C)) & r_7 \\
Permute_{i,j}(Switch_{L_k^l;v,v'}(C)) & = & Switch_{L_k^l;v,v'}(Permute_{i,j}(C)) & r_8 \\
Permute_{i,j}(\pi_{L_i^{j_1},\dots,L_i^{j_p}}(C)) & = & \pi_{L_i^{j_1},\dots,L_i^{j_p}}(Permute_{i,j}(C)) & r_9 \\
Switch_{L_i^j;v,v'}(\pi_{L_i^{j_1},\dots,L_i^{j_p}}(C)) & = & \pi_{L_i^{j_1},\dots,L_i^{j_p}}(Switch_{L_i^j;v,v'}(C)) & r_{10} \\
\sigma_\varphi(Op(C)) & = & Op(\sigma_\varphi(C)) & r_{11} \\
Aggregate_{L_i^{d_i}\rightarrow L_i^j;\ f(m)}(Op(C)) & = & Op(Aggregate_{L_i^{d_i}\rightarrow L_i^j;\ f(m)}(C)) & r_{12} \\
\sigma_\varphi^{members}(Aggregate_{L_i^{d_i}\rightarrow L_i^j;\ f(m)}(C))) & = & Aggregate_{L_i^{d_i}\rightarrow L_i^j;\ f(m)}(\sigma_\varphi^{member}(C))) & r_{13}
\end{array}
$$

FIG. 7 – Rewriting rules for OLAP algebra. In r_1, r_4 : if $j < k$ then $k' = k - 1$ else $k' = k$. In $r_1, r_2, r3$: if $j < l$ then $l' = l - 1$ else $l' = l$. In r_2, r_5 : if $j < i$ then $i' = i - 1$ else $i' = i$. In r_3, r_4, r_5 : if $l < j$, $k < j$ or $i < j$ then $j' = j - 1$ else $j' = j$.

the rules in an user-friendly way, we define the *Rotate* operation as follows ($\circ$ denotes the composition of operations) :

(a) $Rotate_{[i,j]}(C) = (\circ_{k=i}^{j-1} Permute_{k,k+1})(C))$ if $i < j$, and

(b) $Rotate_{[i,j]}(C) = (\circ_{k=i-1}^{j} Permute_{k,k+1})(C))$ if $i > j$.

In this figure, $Op \in \{Permute, Switch, Nest, \pi\}$, σ denotes a selection on measures or members and π denotes a projection on measures or members.

Example 5 *As an example, consider the translation of query q into our algebra (see Example 4). This query can be rewritten using rule r_{11} and r_{13} to push selections ahead of aggregations, projections and permute. The rewritten query is :*

$Nest_{1(3)}(Permute_{1,5}(Aggregate_{item\rightarrow category;sum(sales)}(\pi_{year}(\pi_{quarter}($
$\pi_{region,city}(\pi_{category,item}(\pi_{name}(Aggregate_{month\rightarrow quarter;sum(sales)}($
$\sigma^{member}_{region=north}(\sigma^{member}_{item=beer\vee item=wine\vee category=food}(C_0)))))))))))$

5 Optimization

In this section, we propose an optimization technique at the logical level for OLAP queries. We first introduce this technique informally.

5.1 Intuition

Our optimization technique consists in the following two steps :

1. Determine which part of the query output will be displayed on the screen. Assuming that we can visualize data in V dimensions, this part is the first V-dimensional slice of the cube, which is also a cube. This slice can be computed by adding selection conditions to the initial query. These conditions are obtained by only computing the dimensions and structure of the answer to the query.
2. Once the selection conditions are added to the query, the rewriting rules are used to push the selections ahead of the *Aggregate* operations, assuming this operation is the most costly.

We now illustrate these two steps.

Step 1 : Let $C = \langle \mathcal{D}, F, S \rangle$ be a N-dimensional cube, q be a query over C and $C' = q(C) = \langle \mathcal{D}', F', S' \rangle$ be the results of the query. Evaluating q means computing the different parts $\mathcal{D}', F'$ and S' of C', which can be written $C' = \langle q_1(\mathcal{D}), q_2(F, \mathcal{D}), q_3(S) \rangle$. To find the first V-dimensional slice of C', we only need to know what are the members on the axes of C' and what are the positions of these members, so computing $\mathcal{D}' = q_1(\mathcal{D})$ and $S' = q_3(S)$ is sufficient. If only V dimensions are used to display the cube C', V axes are fully displayed, and the facts displayed are the facts for the members at the lowest positions on the remaining axes (called the hidden axes). This defines the first V-dimensional slice of C'. Hence, this slice can be seen as the cube C' on which we have selected the members at the lowest positions on the hidden axes. Determining these selection conditions is done by the function $FirstSliceSelection$, presented Figure 9.

Step 2 : The selection conditions computed in Step 1, referred to as φ, are added to the selection conditions of the original query q, resulting in a new query $\sigma_{\varphi}^{member}(q(C))$. The rewriting rules are then used to obtain a query q' where the selections are pushed ahead of aggregations. This query q' is such that $q' = \langle q_1'(\mathcal{D}), q_2'(F, \mathcal{D}), q_3'(S) \rangle$. It is then sufficient to compute $q_2'(F, \mathcal{D})$ since :

1. $q_1'(\mathcal{D}) \subseteq q_1(\mathcal{D})$ and $q_1(\mathcal{D})$ has already been computed in step 1,
2. $q_3(S) = \langle K_3, axis_3, pos_3, depth_3 \rangle$ has already been computed in step 1, and $q_3'(S)$ is $\langle K, axis_3, pos_3', depth_3 \rangle$ where, for all attributes L_i^j, $pos_3'(L_i^j)$ is the restriction of pos_3 to the active domains of the attributes.

The principle of our optimization technique is summarized in Figure 8.

5.2 The optimization algorithm

The function $FirstSliceSelection$ is presented Figure 9. Let us examine each step of the function :

$$
\begin{array}{cr}
q(C) & \\
\downarrow & \text{FirstSliceSelection} \\
\sigma_{\varphi}^{member}(q(C)) & \\
\downarrow & \text{rewriting rules} \\
q'(C) = \langle q'_1(\mathcal{D}), q'_2(F,D), q_3(S) \rangle & \\
\downarrow & \text{evaluation of } q'_2 \\
F' = q'_2(F,D) &
\end{array}
$$

FIG. 8 – Summary of the optimization technique

Function *FirstSliceSelection*

Input :	A cube C and an OLAP query q
Output :	A selection condition φ such that $\sigma_{\varphi}^{members}(q(C))$ is the first V-dimensional slice of $q(C)$
Use :	The size V of the slice to be visualized

Compute the structure S' of cube $C' = q(C)$
Compute the dimensions $\mathcal{D}'$ of cube $C' = q(C)$
Let $\mathcal{D}' = \{D'_1, \ldots, D'_N\}$
Let $S' = \langle K', axis', pos', depth' \rangle$ and $\varphi = true$
For $l = V + 1$ **to** K' **do**
    Let $(E_l, \prec_l) = axis'(l)$
    For every $D'_i \in E_l$ **do**
        Let $v = min_{pos'(depth'(D'_i))}(\pi_{depth'(D'_i)}(D'_i))$
        $\varphi = \varphi \wedge (depth'(D'_i) = v)$
    End for
End for
Return φ

FIG. 9 – Identification of the first V-dimensional slice of a cube

Optimization Algorithm

Input :	A cube C and an OLAP query q
Output :	An optimized query q'

1. **Let** $\varphi = FirstSliceSelection(C, q)$
2. **Rewrite** $\sigma_{\varphi}^{member}(q(C))$ using the rewriting rules
3. **Return** q' the rewritten query

FIG. 10 – The optimization algorithm

1 This step computes the structure of the result.

2 This step computes the dimensions of the result.

4 This steps initialises the selection condition.

5 This loop explores each hidden axis.

7 This loop explores every nested dimension on the hidden axis.

8 This step finds the member at lowest position for the level of the dimension at which the facts are displayed.

The full algorithm is presented in Figure 10. We illustrate the different steps of the algorithm on an example.

Example 6 *Consider the cube* $C_6 = \langle \mathcal{D}_6, F_6, S_6 \rangle$*, and recall that* $S_6 = \langle 4, axis', pos', depth' \rangle$*. Suppose that* $V = 2$*, meaning that we can only visualize the first two-dimensional slice of* C_6*. Thus there are two hidden axes, and* l *varies from 3 to 4. Let us detail the execution of the body of the loop of step 5 in function* $FirstSliceSelection$*.*

When $l = 3$*, we have* $axis'(3) = (\{Product\}, \leq'_3)$ *and* $category = depth'(Product')$*. Let* $<'_{category}$ *be the total order associated by* pos' *with attribute category. We have* $drink = min_{<'_{category}}(\pi_{category}(Product')\})$*, and thus* $\varphi = \varphi \wedge (category = drink)$*.*

When $l = 4$*, we have* $axis'(4) = (\{Year\}, \leq'_4)$ *and* $year = depth'(Year')$*. Let* $<'_{year}$ *be the total order associated by* pos' *with attribute year.*
Since $1988 = min_{<_{year}}(\pi_{year}(Year')\})$*, we have* $\varphi = \varphi \wedge (year = 1988) = (category = drink) \wedge (year = 1988)$ *which concludes the execution of function* $FirstSliceSelection$*.*
After application of the rewriting rules of Figure 7, the optimized query is :

$$Permute_{1,5}(Nest_{1(3)}(\pi_{year}(\pi_{quarter}(\pi_{region,city}(\pi_{category,item}(\pi_{name}(Aggregate_{item \rightarrow category;sum(sales)}(Aggregate_{month \rightarrow quarter,sum(sales)}(\sigma_{\varphi}^{member}(C_0))))))))))$$

where $\varphi = (region = north \wedge (item = beer \vee item = wine \vee category = food)) \wedge (year = 1988 \wedge category = drink)$*. Moreover it is easy to see that* φ *can be simplified as* $(region = north \wedge (item = beer \vee item = wine) \wedge year = 1988)$*.*

6 Translation into the relational algebra

In our cube model, the parts $\mathcal{D}$ and F are regarded as the components of a star schema. Thus a straightforward translation of the optimized query into the relational algebra can be given by considering only the algebraic operators that operate on these parts (see the table of Figure 6). We illustrate this translation on an example.

Example 7 *Let F_0 be the fact table of cube C_0. The fact table F' of the result of q', the optimized version of q is :*
$F_1 = \pi_{year,quarter,city,item,name,sum(sales)}(\pi_{sch(F_0)}(F_0 \bowtie_{city} \sigma_{region=north}(Location)$
$\bowtie_{item} \sigma_{item=beer \vee item=wine}(Product) \bowtie_{year} \sigma_{year=1988}(Year)) \bowtie_{month} Quarter)$
$F' = \pi_{year,quarter,city,category,name,sum(sales)}(F_1 \bowtie_{item} Product)$

The dimensions of the result are :
– $Year' = \pi_{year}(\sigma_{year=1988}(Year))$,
– $Quarter' = \pi_{quarter}(Quarter))$,
– $Location' = \pi_{region,city}(\sigma_{region=north}(Location))$,
– $Salesman' = \pi_{name}(Salesman)$,
– $Product' = \pi_{category,item}(\sigma_{item=beer \vee item=wine}(Product))$.

Thus, if the query q' is evaluated on the star schema of cube C_0, the join sequence of q' can be noted $F_0 \bowtie_{city} \sigma_{region=north}(Location) \bowtie_{item} \sigma_{item=beer \vee item=wine}(Product) \bowtie_{year} \sigma_{year=1988}(Year)) \bowtie_{month} \sigma_{true}(Quarter) \bowtie_{name} \sigma_{true}(Salesman)$.

7 Cost-based Approach

We consider a DW modeled by a star schema. Let q be a star join query. Let $D^{sel} = \{D_1^{sel}, ..., D_k^{sel}\}$ be the set of dimension tables having selection predicates. Each selection predicate p_j has two selectivity factors, one defined on a dimension table (D_i) used by this predicate and denoted by $Sel_{D_i}^{p_j}$ ($Sel_{D_i}^{p_j} \in [0,1]$) and another defined on the fact table denoted by $Sel_F^{p_j}$. Note that $Sel_{D_i}^{p_j} \neq Sel_{p_j}^F$, as it is shown in the following example :

Example 8 *Let us consider the selection predicate* quarter=q_1 *defined on the dimension table Quarter (Figure 2). Its selectivity factor is 0.25. But sales for quater q_1 may represent 70% of sale activities.*

Now we present a cost model to show the utility of our approach by considering a *Large Memory Hypothesis (LMH)* : all dimension tables are in the main memory because their sizes are very small [Corp., 1997]. This assumption becomes more and more realistic as the size of main memory keeps increasing because of fall in main memory prices. The selection conditions are always pushed down onto the dimension tables like in [Labio *et al.*, 1997]. This model computes the Inputs/outputs cost for reading and writing data between disk and main memory. The notations used by this cost model are summarized in Table 1.

Let M be the number of selection predicates defined on dimension tables. The cost of executing the query q without using our RBA is given by :

$$JCW = \prod_{i=1}^{M} Sel_F^{p_i} \times |F| \qquad (1)$$

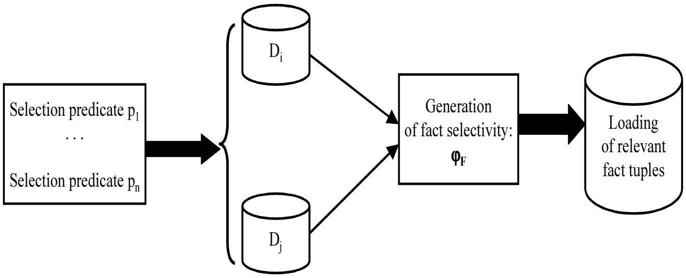

FIG. 11 – Execution strategy using RBA

Symbol	Meaning
PS	Page size of the file system (in bytes)
$w(T_j)$	Width, in bytes, of a tuple of a table T_j
$\|\|T_j\|\|$	Number of tuples present in a table T_j
$\|T_j\|$	Total number of pages occupied by a table T_j
$Sel_{D_i}^{p_j}$	Selection factor of the predicate p_j defined on dimension table D_i
$Sel_F^{p_j}$	Selection factor of the predicate p_j defined on the fact table F

TAB. 1 – Symbols and their Meanings

where $|F| = \left\lceil \frac{||F|| \times w(F)}{PS} \right\rceil$ (representing the number of pages occupied by the fact table).

By using our RBA, we get a new query q' with M' ($M' \geq 1$) new selection predicates (see Figure 11). Therefore, the cost of executing q' is given by :

$$JC = \prod_{j=1}^{M+M'} Sel_F^{p_j} \times |F| \tag{2}$$

where M' represents the number of new predicates added by the RBA.

Our optimization technique reduces the I/O cost of the query q if and only if : $\frac{JC}{JCW} < 1 \Longrightarrow \frac{\prod_{i=1}^{M+M'} Sel_F^{p_i} \times |F|}{\prod_{j=1}^{M} Sel_F^{p_j} \times |F|} < 1 \Longrightarrow \prod_{i=M'+1}^{M+M'} Sel_F^{p_i} < 1$.

In the reality, this can be always true because a selectivity factor of a fact table is always less than 1. If we consider the example of section 2, the new added selection predicates were *year = 1988* and *category = drink*. The product of the selectivity factors of these predicates is usually less than 1.

To execute the rewritten query q', we can force the optimizer to use bitmap join indexes already defined on the new added predicate selection attributes using *hints* available on commercial systems (like Oracle).

Example 9 *Let consider the query defined in the motivating example section. The new selection attributes are : year and category. To extract the relevant tuples of the fact tables, the query can use the two bitmap join indexes, defined as follows :*

```
CREATE BITMAP INDEX Year_sales_bji ON Sales(Year.year)
FROM Sales, Year
WHERE Sales.year = Year.year;

CREATE BITMAP INDEX Product_sales_bji ON Sales(Product.category)
FROM Sales, Product
WHERE Sales.product = Product.product;
```

This example shows clearly the benefit of the combination of RBA and CBA approaches in optimizing OLAP queries. The new added selection predicates can also be used to provide a fragmentation schema (since the data partitioning process is based on selection predicates defined on OLAP queries) of the star schema of the data warehouse and also in selecting a set of materialized views and indexing schemes (as in previous example).

8 Conclusion

In this paper, we addressed a framework for combining Rule-Based Approache (RBA) and Cost-Based Approach (CBA) to optimize OLAP queries. In order to use the RBA in the data warehousing environment, we developed a logical model to describe both datacubes and their structures. The main OLAP operators are translated into this model, and a set of rewriting rules involving these operators are proposed. These rules are similar to those defined on relational algebra (e.g., pushing down selection operations in query trees). An optimization algorithm is proposed based on the visualization of a cube on a screen. This algorithm is implemented using Java and generates new selection predicates over the dimension tables of a given OLAP query and therefore a new rewritten query is generated. These new predicates contribute in optimizing queries. Finally, the rewritten query can be executed by CBA. To show the utility of our framework, we developed a simple cost model for reading and writing data between disk and main memory. Our proposed approaches can be incorporated in the existing systems without modifying the existing optimization techniques.

To show the effectiveness of our approach, we are now evaluating the performance of our approach using a benchmark (TPC-H or APB-1) with a large set of queries. Our future works include : the study in more details of the combination of our RBA and materialized views, index and partitioning schema selections algorithms, the study of the optimization of sequences of OLAP queries, and the extension of the algebra towards more sophisticated OLAP operators.

Références

[Bellatreche *et al.*, 2004] L. Bellatreche, M. Schneider, H. Lorinquer, et M. Mohania. Bringing together partitioning, materialized views and indexes to optimize performance of relational data warehouses. *Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2004)*, pages 15–25, September 2004.

[Chaudhuri, 2004] S. Chaudhuri. Index selection for databases : A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 16(11) :1313–1323, November 2004.

[Corp., 1997] Oracle Corp. Star queries in oracle8. *White Paper*, June 1997.

[Corporation, 1998] Microsoft Corporation. OLEDB for OLAP. Available at http ://www.microsoft.com/data/oledb/olap, 1998.

[Gupta, 1997] H. Gupta. Selection of views to materialize in a data warehouse. *Proceedings of the 6th International Conference on Database Theory (ICDT '97)*, pages 98–112, 1997.

[Karloff et Mihail, 1999] H. Karloff et M. Mihail. On the complexity of the view-selection problem. *Proceedings of the Symposium on Principles of Databases Systems (PODS'99)*, pages 167–173, 1999.

[Kimball, 1996] R. Kimball. *The Data Warehouse Toolkit.* John Wiley & Sons, 1996.

[Kotidis et Roussopoulos, 1999] Y. Kotidis et N. Roussopoulos. Dynamat : A dynamic view management system for data warehouses. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 371–382, June 1999.

[Labio *et al.*, 1997] W. Labio, D. Quass, et B. Adelberg. Physical database design for data warehouses. *Proceedings of the International Conference on Data Engineering (ICDE)*, 1997.

[Lei et Ross, 1998] H. Lei et K. A. Ross. Faster joins, self-joins and multi-way joins using join indices. *Data and Knowledge Engineering*, 28(3) :277–298, November 1998.

[Marcel, 1999] P. Marcel. Modeling and querying multidimensional databases : An overview. *Networking and Information Systems Journal*, 2(5-6) :515–548, 1999.

[Molina *et al.*, 1998] H. Molina, W. J. Labio, J. L. Wiener, et Y. Zhuge. Distributed and parallel computing issues in data warehousing. *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, page 7, June 1998.

[Ross *et al.*, 1996] K. Ross, D. Srivastava, et S. Sudarshan. Materialized view maintenance and integrity constraint checking : Trading space for time. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 447–458, 1996.

[Sanjay *et al.*, 2004] A. Sanjay, V. R. Narasayya, et B. Yang. Integrating vertical and horizontal partitioning into automated physical database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 359–370, 2004.

[Theodoratos et Sellis, 1997] D. Theodoratos et T. K. Sellis. Data warehouse configuration. *Proceedings of the International Conference on Very Large Databases*, pages 126–135, August 1997.

[Yang *et al.*, 1997] J. Yang, K. Karlapalem, et Q. Li. Algorithms for materialized view design in data warehousing environment. *Proceedings of the International Conference on Very Large Databases*, pages 136–145, August 1997.

[Zilio *et al.*, 2004] D. C. Zilio, J. Rao, S. Lightstone, G. M Lohman, A. Storm, C. Garcia-Arellano, et S. Fadden. Db2 design advisor : Integrated automatic physical database design. *Proceedings of the International Conference on Very Large Databases*, pages 1087–1097, August 2004.

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 this framework, the non atomic *Rotate* can be expressed as a combination of *Permutes* and *Switches.*

In what follows, let $C = \langle \mathcal{D}, F, S \rangle$ be an N-dimensional cube with $\mathcal{D} = \{D_1, \ldots, D_N\}$ and $S = \langle K, axis, pos, depth \rangle$. We note $adom(L)$ the active domain of attribute L.

Permute : Let $i, j \in \{1, \ldots, K\}$. $Permute_{i,j}(C)$ is a cube $C' = \langle \mathcal{D}, F, S' \rangle$ where $S' = \langle K, axis', pos, depth \rangle$ such that :

– $axis'(k) = axis(k)$ for all $k \in [1, K] \setminus \{i, j\}$.

– $axis'(i) = axis(j)$. – $axis'(j) = axis(i)$.

Switch : Let $L_i^j \in sch(D_i)$ and let $v, v' \in adom(L_i^j)$ such that $anc(v) = anc(v')$. $Switch_{L_i^j;v,v'}(C)$ is a cube $C' = \langle \mathcal{D}, F, S' \rangle$ where $S' = \langle K, axis, pos', depth \rangle$ and pos' is defined by :

- For every $L_k^l \in \bigcup_{D \in \mathcal{D}} sch(D)$ such that $L_k^l \neq L_i^j$, $pos'(L_k^l) = pos(L_k^l)$.
- Let $<$ and $<'$ be the total orders associated with attribute L_i^j by the functions pos and pos' respectively. If $v < v'$, then $v' <' v$, else $v <' v'$. Moreover, for every $x \in adom(L_i^j) \setminus \{v, v'\}$, we have :
  – If $x < v$, then $x <' v'$, else $v' <' x$. – If $x < v'$, then $x <' v$, else $v <' x$.

Nest : Let $i, j \in \{1, \ldots, K\}$ and such that $i \neq j$. $Nest_{i(j)}(C)$ is a cube $C' = \langle \mathcal{D}, F, S' \rangle$ where $S' = \langle K-1, axis', pos, depth \rangle$ is defined by :

1. $axis'(k) = axis(k)$ for all k such that $1 \leq k \leq K$
2. $axis'(i) = (E_{ij}, \prec_{ij})$ where $E_{ij} = E_i \cup E_j$ if $axis(i) = (E_i, \prec_i)$ and $axis(j) = (E_j, \prec_j)$, and $\prec_{ij}$ is defined by :
   – For every $D, D' \in E_i$, if $D \prec_i D'$, then $D \prec_{ij} D'$.
   – For every $D, D' \in E_j$, if $D \prec_j D'$, then $D \prec_{ij} D'$.
   – For every $(D, D') \in E_i \times E_j$, we have $D \prec_{ij} D'$.
3. $axis'(k) = axis(k+1)$ for all k such that $j \leq k \leq K-1$.

Order by Let $L_i^j \in sch(D_i)$, and let $<$ be an order on $dom(L_i^j)$. $orderby_{L_i^j,<}(C)$ is a cube $C' = \langle \mathcal{D}, F, S' \rangle$ where $S' = \langle K, axis, pos', depth \rangle$ and pos' is defined by :

- $\forall (k, l) \neq (i, j), pos'(L_k^l) = pos(L_k^l)$,
- $pos'(L_i^j)$ is the restriction of $<$ to $adom(L_i^j)$.

Projection on members : Let $X = \{L_i^{j_1}, \ldots, L_i^{j_p}\}$ such that $X \subseteq sch(D_i)$ and $depth(D_i) \in X$. $\pi_X^{members}(C)$ is a cube $C' = \langle \mathcal{D}', F, S' \rangle$ where $\mathcal{D}' = (\mathcal{D} \setminus \{D_i\}) \cup \{D_i'\}$ with $D_i' = \pi_X(D_i)$ and $S' = \langle K, axis', pos', depth \rangle$ is defined by :

- Let $k \in \{1, \ldots, K\}$ such that $axis(k) = (E_k, \prec_k)$ and $D_i \in E_k$. For every $l \in \{1, \ldots, K\}$, if $l \neq k$, then $axis'(l) = axis(l)$. On the other hand, $axis'(k) = (E_k', \prec_k')$ where $E_k' = (E_k \setminus \{D_i\}) \cup \{D_i'\}$ and $\prec_k'$ is defined by :
  - For every $D, D' \in E_k' \setminus \{D_i'\}$, if $D \prec_k D'$, then $D \prec_k' D'$.
  - For every $D \in E_k' \setminus \{D_i'\}$, if $D \prec_k D_i$, then $D \prec_k' D_i'$.
- pos' is the restriction of pos to $\bigcup_{D \in \mathcal{D}'} sch(D)$.

Projection on measures Let $sch(F) = \{L_1^{d_1} : dom(L_1^{d_1}), \ldots, L_N^{d_N} : dom(L_N^{d_N}), m_1 : dom(m_1), \ldots, m_p : dom(m_p)\}$ and let $\{m_{j_1}, \ldots, m_{j_p}\} \subseteq \{m_1, \ldots, m_p\}$. $\pi_{m_{j_1}, \ldots, m_{j_p}}^{measures}(C)$ is a cube $C' = \langle \mathcal{D}, F', S \rangle$ where $F' = \pi_{L_1^{d_1}, \ldots, L_N^{d_N}, m_{j_1}, \ldots, m_{j_p}}(F)$.

Aggregate : Let $L_i^{d_i} \in sch(F)$, $L_i^j \in sch(D_i)$ such that $d_i > j$. Let $f_1, \ldots, f_p$ be aggregate functions.

$Aggregate_{L_i^{d_i} \to L_i^j;\ f_1(m_1), \ldots, f_p(m_p)}(C)$ is a cube $C' = \langle \mathcal{D}, F', S' \rangle$ such that :

- $sch(F') = \{L_1^{d_1}, \ldots, L_{i-1}^{d_{i-1}}, L_i^j, L_{i+1}^{d_{i+1}}, \ldots, L_N^{d_N}, m_1, \ldots, m_p\}$
  $F' = \pi_{L_1^{d_1}, \ldots, L_{i-1}^{d_{i-1}}, L_i^j, L_{i+1}^{d_{i+1}}, \ldots, L_N^{d_N};\ f(m)}(F \bowtie_{L_i^{d_i}} D_i)$
- $S' = \langle K, axis, pos, depth' \rangle$ with :
  - $depth'(D_k) = depth(D_k)$, for all $k \in [1, N] \setminus \{i\}$.
  - $depth'(D_i) = L_i^j$.

Selection on members : Let $L_i^j \in sch(D_i)$ be such that $j \leq d_i$ where $L_i^{d_i} = depth(D_i)$. Let $v \in dom(L_i^j)$. $\sigma_{L_i^j = v}^{members}(C)$ is a cube $C' = \langle \mathcal{D}', F', S' \rangle$ where

- $\mathcal{D}' = \{D_1, \ldots, D_{i-1}, D_i', D_{i+1}, \ldots, D_N\}$ with $D_i' = \sigma_{L_i^j = v}(D_i)$
- $F' = \pi_{sch(F)}(F \bowtie_{L_i^{d_i}} \sigma_{L_i^j = v}(D_i))$
- $S' = \langle K, axis, pos', depth \rangle$ where $pos'(L_k^l) = pos(L_k^l)$ if $L_k^l \neq L_i^j$ and $pos'(L_i^j)$ is the restriction of $pos(L_i^j)$ to $adom(L_i^j)$.

Selection on measures : Let $m \in \{m_1, \ldots, m_p\}$ and $c \in dom(m)$. $\sigma_{m=c}^{measures}(C)$ is a cube $C' = \langle \mathcal{D}, F', S \rangle$ where $F' = \sigma_{m=c}(F)$.