OLAP in Multifunction Multidimensional Databases

Ali Hassan[1], Frank Ravat[1], Olivier Teste[2], Ronan Tournier[1], and Gilles Zurfluh[1]

[1] Université Toulouse 1 Capitole, IRIT (UMR 5505), 2 Rue du Doyen Gabriel Marty, 31042 Toulouse cedex 9, France
[2] Université Toulouse 2 / IUT Blagnac, IRIT (UMR 5505), 1 Place Georges Brassens, BP 60073, 31703 Blagnac cedex, France
{hassan,ravat,teste,tournier,zurfluh}@irit.fr

Abstract. Most models proposed for modeling multidimensional data warehouses consider a same function to determine how measure values are aggregated. We provide a more flexible conceptual model allowing associating each measure with several aggregation functions according to dimensions, hierarchies, and levels of granularity. This article studies the impacts of this model on the multidimensional table (MT) and the OLAP algebra [11]. It shows how the MT can handle several aggregation functions. It also introduces the changes of the internal mechanism of OLAP operators to take into account several aggregation functions especially if these functions are non-commutative.

Keywords: Multidimensional database, OLAP analysis, OLAP operators, aggregation function, multidimensional table.

1 Introduction

Within multidimensional models, analysis indicators are analyzed according to several dimensions. Decision-makers can use OLAP operators [11] to study measures according to different levels of granularity. In this way, data is regrouped according to selected levels and aggregated using aggregation functions. Classical multidimensional databases provide the use of only the same aggregation function to aggregate a measure over all the multidimensional space. This capacity is not sufficient to face situations which require several aggregation functions to aggregate a same measure. For example, the monthly average temperatures are obtained from the calculus of the average of daily temperatures. However, it is possible to calculate the average department (a subdivision of regions in the French administrative geographical system) temperatures according to two ways. The first, i.e. **simple** way, uses the same aggregation that is used by television weather forecasts by choosing the main city ('prefecture') that is considered as representative of the considered department. The second, i.e. **scientific** way, takes into account all the temperatures of all the cities of the department.

Related Work. Most of the existing works consider that a measure is associated to one aggregation function that will be used for all the different modeled aggregation levels. [6, 13] do not specify aggregation functions at the measure level; however, they leave the possibility to use several aggregation functions during OLAP data explorations. This provides great flexibility, but allows the user to do errors by using inappropriate aggregation functions. The authors of [9] suggest linking each measure to a set of functions that contain only those which are valid for the measure. However, each function is used uniformly for all dimensions and all hierarchical levels that compose the multidimensional space. Recent works [1] allow the use of a different aggregation function for each dimension but without allowing the function to be changed according to the selected hierarchical level. This problem was solved by recent aggregation models [10, 2]. These works allow associating each measure with an aggregation function for each dimension, each hierarchy or each aggregation level. However, they do not consider non-commutative aggregation functions.

Regarding commercial tools, Business Objects uses a single aggregation function for each measure. By contrast, Microsoft Analysis Services offers the possibility of applying a 'custom rollup' in a hierarchy [7]. However aggregation functions are related to neither a specific dimension nor an aggregation level. They are related to a member (an instance) of an aggregation level in a hierarchy (i.e. a line in the dimension table). Therefore, applying this 'custom rollup' to a single aggregation level requires repeating it for all the instances of that level. This causes storage and performance problems [7].

Contribution. In order to overcome these limits, we have developed a conceptual model for representing multidimensional data [8]. This model associates a measure with different aggregation functions according to dimensions, hierarchies and aggregation levels called parameters. Moreover, the model controls the validity of the aggregated values by defining an order of execution for non-commutative functions. The model considers also the case where an aggregated measure cannot be calculated using aggregation constraints. These constraints define the starting level from which the aggregation can be calculated.

The application of this model has several impacts on the resulting multidimensional table (MT) as well as on the OLAP manipulation operators [11]. In this article, we present modifications that have to be applied to adapt the MT to present several aggregation functions and the changes on the OLAP operators to be able to deal with cases of multiple and non-commutative functions.

This paper is organized as follows: Section 2 presents our conceptual model for integrating several aggregation functions for a same measure. Section 3 discusses the changes that we make on the OLAP operators and MT to adapt to our model. Finally, Section 4 details our experiments.

2 Multifunction Conceptual Data Model

Let $\mathcal{N} = \{n_1, n_2, ...\}$ be a finite set of non redundant names and $\mathcal{F} = \{f_1, f_2, ...\}$ a finite set of aggregation functions.

Definition 1. A *fact*, noted F_i, is defined by (n^{Fi}, M^i).

- $n^{Fi} \in \mathcal{N}$ is the name that identifies the fact,
- $M^i = \{m_1, ..., m_{pi}\}$ is a set of *measures.*

Definition 2. A *dimension*, noted D_i, is defined by (n^{Di}, A^i, H^i).

- $n^{Di} \in \mathcal{N}$ is the name that identifies the dimension,
- $A^i = \{a^i_1, ..., a^i_{ri}\} \bigcup \{Id^i, All^i\}$ is the set of *dimension attributes*,
- $H^i = \{H^i_1, ..., H^i_{si}\}$ is the set of *hierarchies.*

Hierarchies organize the attributes of a dimension, called parameters, from the finest granularity (root parameter noted Id^i) up to the most general granularity (the extremity parameter noted All^i).

Definition 3. A *hierarchy*, noted H_j (abusive notation of $H^i_j, \forall$ j $\in [1..s_i]$), is defined by (n^{Hj}, P^j, $\prec^{Hj}$,Weak^{Hj}).

- $n^{Hj} \in \mathcal{N}$ is the name that identifies the hierarchy,
- $P^j = \{p^j_1, ..., p^j_{qj}\}$ is the set of attributes of the dimension called *parameters*, $P^j \subseteq A^i$,
- $\prec^{Hj} = \{(p^j_x, p^j_y) \mid p^j_x \in P^j \wedge p^j_y \in P^j\}$ is a binary relation, antisymmetric and transitive,
- $\text{Weak}^{Hj} : P^j \rightarrow 2^{A^i \backslash P^j}$ is an application that associates to each parameter a set of attributes of the dimension, called *weak attributes.*

Assuming $M = \bigcup_{i=1}^{n} M^i$, $H = \bigcup_{i=1}^{m} H^i$, $P^i = \bigcup_{j=1}^{s_i} P^j$ and $P = \bigcup_{i=1}^{m} P^i$.

Definition 4. A *multidimensional schema*, noted S, is defined by (F, D, Star, Aggregate). This schema is an extension of our previous works [8] where we add a new type of aggregation (multiple hierarchical) and we revisit the execution order mechanism in order to allow our model to be more expressive.

- $F = \{F_1, ..., F_n\}$ is a set of facts, if $\mid F \mid = 1$ then the schema is called a star schema, while if $\mid F \mid > 1$ then the schema is called a constellation schema
- $D = \{D_1, ..., D_m\}$ is a finite set of dimensions
- Star : $F \rightarrow 2^D$ is a function that associates each fact to a set of dimensions according to which it can be analyzed
- Aggregate : $M \rightarrow 2^{N^* \times \mathcal{F} \times 2^D \times 2^H \times 2^P \times N^-}$ associates each measure to a set of aggregation functions [8]. It allows defining 4 types of aggregation functions:
  - General (if $2^D = \varnothing$, $2^H = \varnothing$ and $2^P = \varnothing$): aggregates measure values with any parameter. This function represents the aggregation function of the classical model,
  - Multiple dimensional (if $2^H = \varnothing$ and $2^P = \varnothing$): aggregates the measure on the whole considered dimension,
  - Multiple hierarchical (if $2^P = \varnothing$): aggregates the measure on all the considered hierarchy,
  - Differentiated (if $2^D \neq \varnothing$, $2^H \neq \varnothing$ and $2^P \neq \varnothing$): aggregates the measure between the considered parameter and the one directly above.

When considering the case of non-commutative functions, N^* associates to each function an order number that represents the execution priority. The function with the lowest order has the highest priority. Commutative functions have the same order. N^- is used to constrain aggregations by indicating a specific aggregation level from which the considered aggregation must be calculated. A non constrained aggregation will be associated to 0 while a constrained aggregation will be associated to a negative value to force the calculation from a chosen level lower than the considered level.

Using the 'Star' function we obtain a structural schema that visualizes structural elements (facts, dimensions and hierarchies) by hiding the aggregation mechanism, Fig. 1 (a). The graphical formalism used is inspired by [4, 11, 12]. The schema in Fig. 1 (a) corresponds to the analysis of average temperatures. The fact 'Temperature' is associated to three dimensions: 'Geography', 'Dates' and 'Time'. The 'Geography' dimension is composed of two hierarchies that correspond to different means for aggregating data: 'Simple' and 'Scientific'. The 'Time' dimension has only one hierarchy that orders the hourly granularities at which temperatures are recorded during the day. The 'Dates' dimension has several hierarchies that organize the granularity levelss of days.

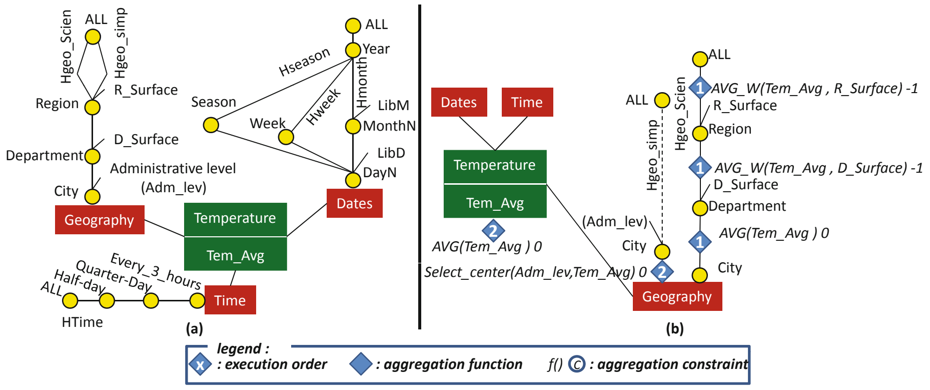


Fig. 1. Graphical representation of the multidimensional schemas

For each measure, an aggregation schema can be obtained using the 'Aggregate' function. This schema details the aggregation mechanism that applies to the measure by simplifying the structural elements as much as possible. Fig. 1 (b) shows the aggregation schema of the average temperatures measure 'Tem_Avg'. It has a general aggregation function 'Avg' and a multiple hierarchical function 'Select_centre' on the hierarchy 'Hgeo_simp'. It has also differentiated functions 'Avg' and 'Avg_W' on the hierarchy 'Hgeo_Scien'. 'Avg_W' aggregates the values from the level directly below the considered one (constraint -1).

The function Select_centre(I, M) takes two numeric inputs. It returns the value M that corresponds to Max(I). For example, if one applies Select_center (administrative_level, Tem_Avg) at the region level, it returns the temperatures of the region prefecture (the city that has the highest administrative level).

The function Avg_W(X, Y) takes two numeric inputs. It returns the average of the X values weighed by Y. In other words, a weighted average Avg_W(X, Y) = $\frac{\sum(X \times Y)}{\sum Y}$.

Within the aggregation schema the hierarchies are presented in a disjoint way (each hierarchy is represented by a complete path from the root parameter to the extremity parameter ALL) contrarily to the structural schema where they are represented in a compact way (the common parts of the hierarchies are fused, forming a tree-like structure of parameters). Aggregation functions are modeled by diamonds. Each diamond shows the execution order and the aggregation constraint. Aggregations with constraints set at -1 are calculated using the level directly below; for example, the average temperature 'Tem_Avg' for each region must be calculated from department average temperatures. In the hypothesis where we would have chosen to calculate this average temperature by region using the city temperatures, the constraint would have been set to -2.

3 OLAP Manipulation

In the next parts of this article, we concentrate our study on OLAP multidimensional analysis [3] applied to a multifunction multidimensional database.

In previous works [11], we have defined a decision-maker oriented algebra to express on-line analytical process. This algebra uses the concept of a multidimensional table (MT). An MT is a two dimensional table used as source and target for the algebraic operators, thus ensuring closure of the OLAP algebra. This choice is justified by the fact that during analyses decision-makers usually manipulate data through bi-dimensional tables (lines and columns) due to their simplicity of understanding and their precision [6]. Our multifunction model changes the MT and some of these operators. These changes do not alter their functionality; however, they may impact the definition or the internal mechanism. Before studying these changes we present our motivating example.

3.1 Motivating Example

Table 1 contains a simplified data sample consistent with our example above. In this example, temperature is recorded twice a day in the department 'Rhône' while being recorded once in the department 'Isère'.

Table 1. Average temperatures of departments by day and time

Region	Department	D_Surface	Date	Time	Tem_Avg
Rhône-Alpes	Rhône	3249	1/1/2012	00:00	-1
Rhône-Alpes	Rhône	3249	1/1/2012	12:00	1
Rhône-Alpes	Isère	7431	1/1/2012	12:00	2
Rhône-Alpes	Rhône	3249	2/1/2012	00:00	0
Rhône-Alpes	Rhône	3249	2/1/2012	12:00	2
Rhône-Alpes	Isère	7431	2/1/2012	12:00	2

If the user wishes to analyze the average temperatures of regions by month, then two aggregation functions must be used:

– 'Avg_W(Tem_Avg, D_Surface)' to calculate the average temperatures of regions from the temperatures of departments weighted by the surfaces of the departments 'D_Surface',
– 'Avg(Tem_Avg)' to calculate the average monthly temperatures from the daily temperatures.

If the function 'Avg_W(Tem_Avg, D_Surface)' is applied first, we obtain the average temperatures of regions by day and time (Table 2). If we apply the function 'Avg(Tem_Avg)' after we obtain the required average temperatures of regions by month (Table 3).

Table 2. Average temperatures of regions by day and time

Region	**Date**	**Time**	**Tem_Avg**
Rhône-Alpes	1/1/2012	00:00	-1
Rhône-Alpes	1/1/2012	12:00	1.7
Rhône-Alpes	2/1/2012	00:00	0
Rhône-Alpes	2/1/2012	12:00	2

Table 3. Average temperatures of regions by month

Region	**Month**	**Tem_Avg**
Rhône-Alpes	2012-1	**0.67**

But if the function 'Avg(Tem_Avg)' is applied first, we obtain the average temperatures of departments by month (Table 4). If we apply the function 'Avg_W(Tem_Avg, D_Surface)' afterwards, we obtain the average temperatures of regions by month (Table 5).

Table 4. Average temperatures of departments by month

Region	**Department**	**D_Surface**	**Month**	**Tem_Avg**
Rhône-Alpes	Rhône	3249	2012-1	0.5
Rhône-Alpes	Isère	7431	2012-1	2

Table 5. Average temperatures of regions by month

Region	**Month**	**Tem_Avg**
Rhône-Alpes	2012-1	**1.54**

The difference between the results obtained in Table 3 and Table 5 proves the need to use the execution order. The key role of the execution order in our model is to force the execution of aggregation functions in a specific order so as not to have an erroneous result due to non-commutativity. The choice of a valid execution order depends on the requirements of the user. It may differ from one case to another, even if the functions are the same in both cases. Our model allows setting the order which gives a valid result for the user. In this example, the result that correspond to the user's requirements are in Table 3. This justifies why we give a value of 1 for the execution order of the function 'Avg_W' while we give 2 for the execution order of the function 'Avg' Fig. 1 (c).

We conclude from the previous example that the application of an aggregation function with an execution order lower after an aggregation function with an execution upper give an erroneous result.

For RollUp[1] and Rotate[2] operators, the aggregation level has to be changed from the current level to another higher level.

– For the RollUp operator: from the current level to the targeted level
– For the Rotate operator: from the current level to the ALL level of the dimension that will be removed from the MT

In order to do that, it is necessary to apply new aggregation functions after having applied the previous functions that built the current MT (before applying the RollUp and Rotate operators). But, the execution orders of the previous functions can be higher than those of the new functions, i.e. it is not possible to apply the new functions after having applied the previous functions. This requires setting up costly internal mechanisms for these RollUp and Rotate operators. We study these mechanisms in the following section.

3.2 Extended OLAP

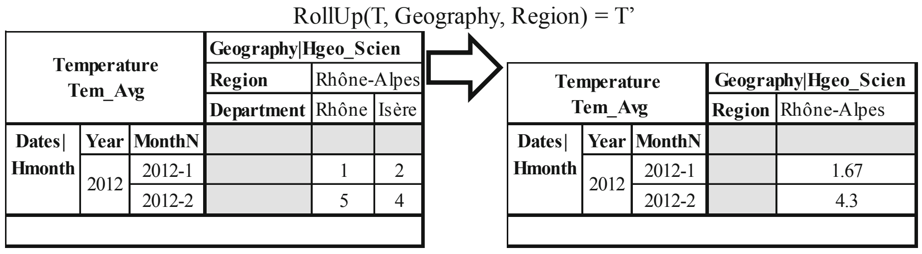
Within this section, we study the RollUp operator as it is not only one of the most emblematic OLAP manipulation operator and but also one of the most used. First the classical RollUp operator is presented, then changes required for this operator to work within our multifunction model are described.

Classical RollUp. In order to study this operator, let us have an MT that analyses average temperatures of departments by months (Fig. 2(a)). From this MT, a RollUp operator will be used to analyze the average temperatures by regions instead of departments (Fig. 2(b)).

To study the classical RollUp operator, we take into account the classical multidimensional model where there is only one function for aggregating the

[1] RollUp(T_{SRC}, D, p_i) [11]: modifies the granularity level on the current hierarchy of the dimension (D) towards an upper level or less detailed level (p_i) by removing one (or several) parameters in line or column of an MT (T_{SRC}).

[2] Rotate(T_{SRC}, D_{old}, D_{new} [,H_{new}]) [11]: allows changing an analysis axis (D_{old}) by another one (D_{new}) in an MT (T_{SRC}). It is possible to specify the hierarchy (H_{new}) of the new dimension to use it in the resulting MT.


RollUp(T, Geography, Region) = T'

Temperature Tem_Avg			Geography|Hgeo_Scien		
			Region	Rhône-Alpes	
			Department	Rhône	Isère
Dates| Hmonth	Year	MonthN			
	2012	2012-1		1	2
		2012-2		5	4

(a) T

Temperature Tem_Avg			Geography|Hgeo_Scien	
			Region	Rhône-Alpes
Dates| Hmonth	Year	MonthN		
	2012	2012-1		1.67
		2012-2		4.3

(b) T'

Fig. 2. RollUp

measure. The SQL query that performs the analysis of the average temperatures by departments and by months (Fig. 2(a)) is the following:

```
R1:Result1 =
SELECT G.REGION, G.DEPARTMENT, D.YEAR, D.MonthN,
       AVG(TT.TEM_AVG) AS TEM_AVG, SUM(TT.TEM_AVG) AS sum_Tem_Avg,
       COUNT(TT.TEM_AVG) AS count_Tem_Avg
FROM DATES D, GEOGRAPHY G, TEMPERATURE TT
WHERE TT.ID_CITY = G.ID_CITY AND TT.ID_DATE = D.ID_DATE
GROUP BY G.REGION, G.DEPARTMENT, D.YEAR, D.MonthN;
```

To execute the desired drilling operation (RollUp), one may profit from the results of the previous MT, if the aggregation function is distributive or algebraic (in this later case, intermediate values have to be stored) [5]. In our example, the aggregation function (Avg) is algebraic. Intermediate values required are the sum of the temperatures of cities (sum_Tem_Avg) for each department and the number of occurrences (count_Tem_Avg). Thus, the R2 query that performs the RollUp and corresponds to the MT in Fig. 2(b), benefits from the intermediate values of the results (Result1) of the previous query R1. In other words, in order to perform a RollUp operation, it is not necessary to load the original base measure values, as the MT values can be used as an intermediate values.

```
R2:
SELECT REGION, YEAR, MonthN,
       SUM(sum_Tem_Avg)/SUM(count_Tem_Avg) AS TEM_AVG
FROM Result1
GROUP BY REGION, YEAR, MonthN;
```

Extended RollUp. The changes of the RollUp are due to the difference between the execution orders of the aggregation functions of a measure. In this multifunction model, as there are several functions that aggregate the measure in the multidimensional space, the SQL query that performs the analysis of the average temperatures by departments (on the hierarchy 'Hgeo_scien') by month (Fig. 2 (a)) becomes more complex:

```
R3: Result2 =
SELECT REGION, DEPARTEMENT, YEAR, MonthN, AVG(TEM_AVG) AS TEM_AVG
FROM (SELECT G.REGION, G.DEPARTEMENT, D.YEAR, D.MonthN, D.DayN,
             T.EVERY_3_HOURS, AVG(TT.TEM_AVG) AS TEM_AVG
FROM DATES D, GEOGRAPHY G, TEMPERATURE TT, TIME T
WHERE TT.ID_TIME = T.ID_TIME AND TT.ID_CITY = G.ID_CITY
  AND TT.ID_DATE = D.ID_DATE
GROUP BY G.REGION, G.DEPARTEMENT, D.YEAR, D.MonthN, D.DayN,
         T.EVERY_3_HOURS)
GROUP BY REGION, DEPARTEMENT, YEAR, MonthN;
```

To perform a RollUp in a multifunction model, we can distinguish two cases according to aggregation functions that correspond to the requeste RollUp:

Case 1. All the execution orders of the aggregation functions that aggregate the measure between the current parameters and the requested parameters are higher or equal to the execution orders of the aggregation functions that aggregate the measure between the base parameters and the current parameters (including the ALL levels of the non-shown dimensions in the MT).

For example, if we want to perform a RollUp to analyze the average temperatures of departments by year, the function that aggregates the average temperatures between the level 'MonthN' and 'Year' is the general function 'Avg(Tem_Avg)'. This function has an execution order of 2 that is greater or equal to the execution orders of aggregation functions that aggregate the average temperatures between the base levels and the levels 'Department', 'MonthN' and 'ALL' of the 'Time' dimension. In this case, in the same way as the classic RollUp, we can benefit from the values already in the MT, as in query R4.

```
R4:
SELECT REGION, DEPARTEMENT, YEAR,
       SUM(sum_Tem_Avg)/SUM(count_Tem_Avg)AS TEM_AVG
FROM Result2
GROUP BY REGION, DEPARTEMENT, YEAR;
```

Case 2. The execution order of an aggregation function that aggregates the measure between a current parameter and a requested parameter is less than an execution order of an aggregation function that aggregates the measure between a base parameter and a current parameter.

For example, if we want to perform a RollUp to analyze the average temperatures of regions by months from the average temperatures of departments by months, the function that aggregates the average temperatures between the levels 'Department' and 'Region' on the hierarchy 'Hgeo_Scien' is the function 'Avg_W(Tem_Avg, D_Surface)'. This function has an execution order of 1 that is less than the execution order of the general function that aggregates the average temperatures between the base level and the 'MonthN' level. This means that it is necessary to calculate the average temperature by region before calculating the average temperatures by months. In this case, it is not possible to benefit from the values of the measure displayed within the MT. Base values of the measure have to be used to get the aggregated values as in query R5[3].

```
R5:
SELECT REGION, YEAR, MonthN, AVG(TEM_AVG) AS TEM_AVG
FROM (SELECT REGION, YEAR, MonthN, DayN, EVERY_3_HOURS,
      AVG_W(DATA_WEIGHTED(TEM_AVG, D_SURFACE)) AS TEM_AVG
FROM (SELECT G.REGION, G.DEPARTEMENT, D.YEAR, D.MonthN, D.DayN,
          T.EVERY_3_HOURS, G.D_SURFACE, AVG(TT.TEM_AVG) AS TEM_AVG
FROM DATES D, GEOGRAPHY G, TEMPERATURE TT, TIME T
WHERE TT.ID_TIME = T.ID_TIME AND TT.ID_CITY = G.ID_CITY
  AND TT.ID_DATE = D.ID_DATE
GROUP BY G.REGION, G.DEPARTEMENT, D.YEAR, D.MonthN, D.DayN,
         T.EVERY_3_HOURS, G.D_SURFACE)
GROUP BY REGION, YEAR, MonthN, DayN, EVERY_3_HOURS)
GROUP BY REGION, YEAR, MonthN;
```

3.3 Extended Multidimensional Table

Within the scope of this article, we extend the concept of the MT in order to support the multifunction principles of our multidimensional model, especially by integrating the associated aggregation functions within its definition. An extended MT is thus defined as follows:

TM = (F, <(D_L, h_L, <p_{L1}, p_{L2}, ...>), (D_C, h_C, <p_{C1}, p_{C2}, ...>)>,
<{Aggregate(m_1)}, {Aggregate(m_2)}, ...>, Pred)

– F: analyzed fact,
– D_L, D_C: dimensions displayed in lines and columns respectively,
– h_L, h_C: hierarchies, used to respectively navigate in lines or columns,
– p_{L1}, p_{L2}..., p_{C1}, p_{C2}, ...: displayed parameters,
– m_1, m_2...: displayed measures,
– Aggregate(m_1), Aggregate(m_2)...: aggregation functions respectively associated to measure m_1, m_2...
– Pred: selection predicate on the fact and/or dimension(s) to limit the set of analyzed values.

The following example specifies the definition of a multidimensional table for analyzing the average temperature by month and department in the Rhône-Alpes region:

TM = (Temperature,
<(Dates, Hmonth, <Year, MonthN>),
(Geography, Hgeo_Scien, <Region, Department>)>,

[3] The customized aggregation function 'AVG_W' calculates a weighted average. It receives a parameter (TYPE DATA_WEIGHTED AS OBJECT (value NUMBER, weight NUMBER)) that consists in the data and its associated weight.

<{(2, Avg(Tem_Avg), {}, {}, {}, 0),
(1, Avg(Tem_Avg), {Geography}, {Hgeo_Scien}, {City}, 0)}>,
GEOGRAPHY.Region = 'Rhône-Alpes')

The graphical representation of this specification is shown in Fig. 3 (b):

Temperature AVG(Tem_Avg)			**Geography|Hgeo_Scien**		
			Region	Rhône-Alpes	
			Department	Rhône	Isère
Dates| Hmonth	**Year**	**MonthN**			
	2012	2012-1		1	2
		2012-2		5	4
Geography.Region = 'Rhône-Alpes'					

(a) classic

Temperature <2> AVG(Tem_Avg)			**Geography|Hgeo_Scien**		
			Region	Rhône-Alpes	
			Department <1> AVG(Tem_Avg)	Rhône	Isère
Dates| Hmonth	**Year**	**MonthN**			
	2012	2012-1		1	2
		2012-2		5	4
Geography.Region = 'Rhône-Alpes'					

(b) extended

Fig. 3. Graphical representation of a multidimensional table

In order to adapt the visualization of the MT to display several aggregation functions that can be used by a unique measure Fig. 3(b), the MT allows displaying the aggregation functions (along with their inputs, their execution orders and their constraints) used to obtain the displayed elements.

- The general function is displayed instead of the corresponding measure,
- The multiple dimensional function is displayed besides the name of the corresponding dimension,
- The multiple hierarchical function is displayed besides the name of the corresponding hierarchy,
- The differentiated function is displayed besides the name of the corresponding parameter,
- The function inputs are displayed between parentheses '()' after the function name,
- The function execution order is displayed between '<>' before the function name,
- The function constraints are displayed after the inputs, at the end of the function.

This visualization is based on the simplification of the functions as much as possible:

- Simplifying aggregation constraints: if a function is not constrained (the constraint value is 0), the MT does not display this constraint Fig. 3
- Simplifying execution orders: if all displayed aggregation functions have the same execution order, the MT hides all these execution orders. For example, if we analyze the average temperatures by department (using the 'Hgeo_simp' hierarchy) and by month, the functions used (the general 'AVG(Tem_Avg)' and the multiple hierarchical 'Select_center(Adm_lev, Tem_Avg)') have the same execution order (2). The resulting MT is shown in Fig. 4,

– Reduce the number of displayed functions:
  - If all the displayed parameters on a dimension have a differentiated function, the MT displays neither the multiple hierarchical function nor the multiple dimensional function on the considered dimension,
  - If a displayed hierarchy has an aggregation function, the MT does not display the multiple dimensional function of the considered dimension,
  - If the two displayed dimensions have a multiple dimensional function or a multiple hierarchy function for the displayed hierarchy or even a differentiated function for each displayed parameter, the MT does not display the general function.

Temperature AVG(Tem_Avg)			**Geography|Hgeo_Simp Select_center(Adm_lev, Tem_Avg)**		
			Region	Rhône-Alpes	
			Department	Rhône	Isère
Dates| Hmonth	**Year**	**MonthN**			
	2012	2012-1		2	2
		2012-2		4	4
Geography.Region = 'Rhône-Alpes'					

Fig. 4. MT with simplified execution orders

4 Experiments

Our proposal is implemented in the prototype 'OLAP-Multi-Functions'. We use Java 7 on top of Oracle 11g DBMS. It allows the definition of a constellation with multiple and differentiated aggregations, as well as visualizing and querying multidimensional data. Aggregation functions are described in a meta-schema. This meta-schema also describes the structures of the multidimensional schema (facts, dimensions and hierarchies). To oversee the analysis, the prototype has a generator of SQL queries. The analyst selects the desired measure and aggregation levels. The prototype translates interactions by generating an executable SQL script in the context of a R-OLAP implementation.

In this section, we study the additional execution time required by the extended operators compared to classical operators.

Collection: We use the example shown in Fig. 1, where temperatures are recorded eight times a day (every three hours). Size grouping for the geography dimension is set to 5. This means that each instance of a higher level corresponds to five instances of lower level (for example, each department has five cities).

Protocol: We observe the execution time of the five queries above in accordance with the number of tuples of the fact (from two to eight millions). The fifth query includes a customized function 'Avg_w' that affects the execution time because the function is not optimized contrary to the standard functions (Sum, Avg, Count, Max, Min); therefore, we also study a sixth query (R6) identical to the fifth query but using a standard function: 'Avg' instead of 'Avg_w'.

Results: Figure 5 shows the curves corresponding to the six queries. The time required to execute query R3 (the basic analysis of the average temperatures of departments by month for the extended operator) is greater than the time required to execute query R1 (the basic analysis using the classical operator) because of the complexity of the query (using several functions) that uses the extended operator.

The time required to execute the queries R2 and R4 (that benefit from the intermediate values of the results Result1 and Result2 of the previous queries R1 and R3 respectively) is remarkably low (about 0.1 seconds) because the data has previously been highly aggregated. Here, we do not notice any difference between the classical operator and the extended operator.

The time required to execute the query R5 (that uses the extended operator in the absence of the possibility of using intermediate values) is greater than the time required to execute R3 (the basic analysis for the extended operator). However, a large part of that time is due to the fact that the customized function 'Avg_W' is not optimized, this is what we see clearly in the difference between the time required to execute R5 and R6 These results show that the additional time required to execute extended operators is relatively large, thus it is necessary to optimize queries and the used functions and benefit as much as possible of previously calculated values.

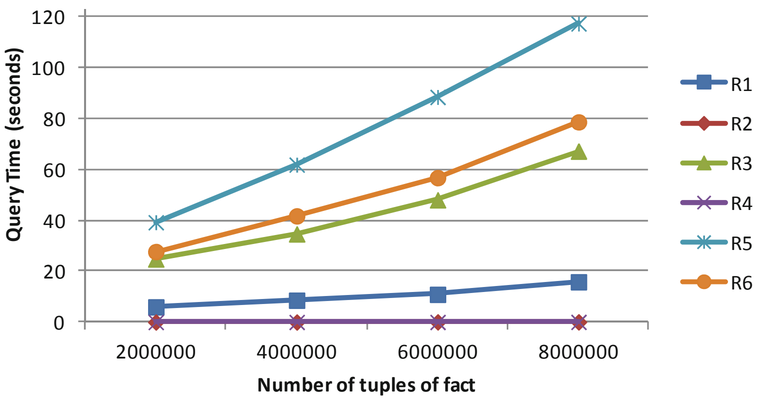


Fig. 5. Execution time of 6 queries according to the number of aggregated tuples

5 Conclusion

In this paper, we have studied the impact of our conceptual data representation model on the multidimensional table and the associated multidimensional algebra. This model allows associating to each measure several aggregation functions according to the dimensions, hierarchies and parameters. The non-commutative aggregation functions necessitate the use of execution orders as these later may influence the internal mechanism of OLAP operators (especially Rotate and RollUp). In order to adapt the visualization of the multidimensional table to our multifunction model, the multidimensional table allows displaying the aggregation functions with their inputs, execution orders and constraints; however, the multidimensional table simplifies the presentation of the functions in terms of aggregation constraints, execution order and number of displayed functions.

We are developing our prototype to benefit from intermediate values and temporary results of OLAP operators to improve the performance. We are also considering optimizing the extended operators algorithms and performing experiments based on our prototype.

References

[1] Abelló, A., Samos, J., Saltor, F.: YAM2: A multidimensional conceptual model extending UML. Information Systems 31, 541–567 (2006)

[2] Boulil, K., Bimonte, S., Pinet, F.: Un modèle UML et des contraintes OCL pour les entrepôts de données spatiales. De la représentation conceptuelle à l'implémentation. Ingénierie des Systèmes d'Information (ISI) 16(6), 11–39 (2011) (in French)

[3] Codd, E.F.: Providing OLAP (on-line analytical processing) to user analysts: an IT mandate. Technical Report, E.F. Codd and Associates (1993)

[4] Golfarelli, M., Maio, D., Rizzi, S.: Conceptual Design of Data Warehouses from E/R Schemes. In: Intl. Conf. HICSS 1998, vol. 7, pp. 334–343 (1998)

[5] Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: Intl. Conf. ICDE, vol. 96, pp. 152–159 (1996)

[6] Gyssens, M., Lakshmanan, L.V.S.: A Foundation for Multi-Dimensional Databases. In: Intl. Conf. VLDB., vol. 97, pp. 106–115 (1997)

[7] Harinath, S., Zare, R., Meenakshisundaram, S., Carroll, M., Guang-Yeu Lee, D.: Professional Microsoft SQL Server Analysis Services 2008 with MDX. Wiley Publishing, Indianapolis (2009)

[8] Hassan, A., Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Differentiated Multiple Aggregations in Multidimensional Databases. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 93–104. Springer, Heidelberg (2012)

[9] Pedersen, T., Jensen, C., Dyreson, C.: A foundation for capturing and querying complex multidimensional data. Information Systems 26(5), 383–423 (2001)

[10] Prat, N., Wattiau, I., Akoka, J.: Representation of aggregation knowledge in OLAP systems. In: The 18th European Conference on Information Systems, ECIS (2010)

[11] Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. International Journal of Data Warehousing and Mining 4(1), 17–46 (2008)

[12] Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Graphical Querying of Multidimensional Databases. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 298–313. Springer, Heidelberg (2007)

[13] Vassiliadis, P., Skiadopoulos, S.: Modelling and Optimisation Issues for Multidimensional Databases. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 482–497. Springer, Heidelberg (2000)