Multidimensional database modelling with differentiated multiple aggregations

Ali Hassan, Franck Ravat, Olivier Teste, Ronan Tournier, Gilles Zurfluh

Multidimensional database modelling with differentiated multiple aggregations

Ali Hassan[a]*, Franck Ravat[a], Olivier Teste[b], Ronan Tournier[a] and Gilles Zurfluh[a]

[a]*University Toulouse 1 Capitole – IRIT (UMR 5505) 118, Route de Narbonne – 31062, Toulouse cedex 9, France;* [b]*University Toulouse 2 Le Mirail – IRIT (UMR 5505) 118, Route de Narbonne – 31062, Toulouse cedex 9, France*

Many solutions have been defined for multidimensional database modelling. These propositions consider the same aggregation function to determine the values of an indicator according to different levels of granularity into the multidimensional space. We provide a more flexible conceptual model that supports multiple differentiated aggregations. Multiple aggregations allow associating different aggregation functions to the same measure for each dimension and for each hierarchy. Differentiated aggregation allows specific aggregations at each level (parameter). Our model is based on a double graphical formalism, expressive enough to control the validity of aggregation functions. We also study the consequences of this conceptual modelling for building lattices of pre-computed aggregates in a relational online analytical processing (R-OLAP) environment.

Keywords: multiple aggregations; multidimensional database; data warehouse conceptual modelling; multidimensional lattices

1. Introduction

Online analytical processing (OLAP) analyses consist in monitoring indicators represented as points in a multidimensional space using different analysis axes. According to hierarchies associated with analysis axes, aggregation functions are used for obtaining a synthetic view of indicator values. Data are grouped according to the selected detail level and aggregated by the functions. Drilling operations (*roll-up* and *drill-down*), often used in OLAP analyses, make intensive use of these aggregation functions.

Multidimensional databases (MDB) provide an appropriate framework for decision-maker analyses. However, in some cases, the default aggregation functions provided may be maladjusted for decision-making. For example, the analysis of temperatures can be performed uniformly with a same aggregation function. The annual maximum temperature is obtained from the monthly maximum temperatures. The latter are obtained from daily maximum temperatures. However, the analysis of temperatures by departments is usually based on the average temperatures by cities, while the analysis of temperatures by region can be performed from an aggregation of the department temperatures, which takes into consideration the department's area (a weighted average). Thus, this analysis involves non-uniform different aggregation functions (such as average or weighted average) according to a considered analysis level (department,

*Corresponding author. Email: hassan@irit.fr

region, etc.). Moreover, in classical analyses, aggregation at a granularity level can be generally obtained from the aggregation of any lower level (the annual maximum temperature is the maximum temperature either monthly or daily). But in case of temperatures based on a spatial distribution, calculating temperatures by regions cannot be obtained directly from the city temperatures (the minimum granularity available). To get the temperature of a region, it is necessary to calculate the average temperature by department and then to do a weighted average of these temperatures.

This article presents a new multidimensional model expressive enough to support several cases of aggregation. We study the consequences of this conceptual model on the lattice of pre-aggregates (Gray, Bosworth, Layman, & Pirahesh, 1996) at the logical level.

In previous work (EDA, 2012; Hassan, Ravat, Teste, Tournier, & Zurfluh, 2012), we detailed our conceptual and logical models. Here, we:

- Extend the conceptual model with a new type of aggregation (hierarchical);
- Revisit the execution order mechanism in order to be more expressive;
- Implement our prototype to study the consequences on lattice reductions.

This paper is organised as follows: Section 2 presents related work. Section 3 recalls the classical conceptual multidimensional model then defines our extensions for differentiated multiple aggregations, followed by the graphical formalism of these extensions. Section 4 details the logical R-OLAP model along with its optimisation relations and the impact of our extensions on this model. We validate our proposal in Section 5 by presenting our prototype and detailing our experiments.

2. Related Work

There are typically two approaches for modelling multidimensional databases: the first approach is based on the data cube metaphor according to which the MDB is represented by cubes, and the second one is known as multidimensional modelling, where the MDB is described by a star schema or constellation (Kimball, 1996). The cube metaphor is based on an equivocal separation between structure elements and values (Torlone, 2003). Modelling analysis axes is not very expressive, due to the difficulty of representing multiple hierarchical organisations of dimensions. It also collides with the hypercube representation when the multidimensional space consists of more than three analysis axes. Finally, it is limited when it comes to representing constellations of facts and shared dimensions. Due to these limitations, our approach falls in the second category.

In the context of multidimensional modelling, several synthetic reviews (Chaudhuri & Dayal, 1997; Lechtenbrger & Trujillo, 2009; Vassiliadis & Sellis, 1999) and comparative surveys (Abelló, Samos, & Saltor, 2006; Boulil, Bimonte, & Pinet, 2011; Gyssens & Lakshmanan, 1997; Luján-Mora, Trujillo, & Song, 2005; Oliveira, Rodrigues, Martins, & Moura, 2011; Pedersen, Jensen, & Dyreson, 2011; Prat, Wattiau, & Akoka, 2010; Ravat, Teste, Tournier, & Zurfluh, 2008; Vassiliadis & Skiadopoulos, 2000) are available in the scientific literature. Some, such as Lechtenbrger and Trujillo (2009), deal with problems related to complex structures such as non-strict, roll-up incomplete and drill-down incomplete hierarchies. We don't address this kind of problem. We focus on the problem of using several aggregation functions in an analysis.

Most of the existing contributions consider that a measure is associated with only one aggregation function, used with all modelled aggregation levels. This function calculates the same aggregation for all combinations of all modelled parameters.

Initial contributions (Gyssens & Lakshmanan, 1997; Vassiliadis & Skiadopoulos, 2000) allow manipulating several aggregation functions for each measure during OLAP analyses, but do not include these aggregation functions in the model. In Pedersen et al. (2011), the authors, in their conceptual model, can link several aggregation functions to a single measure. However, in these three papers, the same function will be used with all the dimensions and all aggregation levels. The model YAM2 (Yet Another Multidimensional Model; Abelló et al., 2006) supports a different aggregation function for each dimension. However, the model does not support function change either between hierarchies or within the hierarchical levels. The aggregation model of Prat et al. (2010) allows associating an aggregation function to each dimension or each hierarchy or sub-hierarchy, but the model considers only standard functions (SUM, AVG, MIN, MAX and COUNT). In Boulil et al. (2011), the authors overcome this limit. However, these last two papers (Boulil et al., 2011; Prat et al., 2010) suffer from a limitation: the authors do not consider the case where aggregation functions are non-commutative (for example, average and weighted average).

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 in several ways (Harinath, Zare, Meenakshisundaram, Carroll, & Guang-Yeu Lee, 2009):

- By using unary operators for solving the aggregation problem over a particular type of hierarchy (parent-child attributes hierarchy). The parent-child hierarchies are built from a single attribute with a reflexive join relationship on the attribute itself (technically, a join on the dimension table itself);
- By using MDX (multidimensional expressions) scripts, either directly or by using the attribute property 'CustomRollupColumn' which indicates a column where MDX scripts are stored.

Here, aggregation functions are related neither to a specific dimension nor to 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 (Harinath et al., 2009). Moreover, binding 'custom rollup' with a specific instance can be problematic with data updates.

The MDX language allows the possibility for building data sets (that will be aggregated by aggregation functions) using functions: PeriodsToDate, YTD (year-to-date), QTD (quarter-to-date), MTD (month-to-date), Crossjoin, Cousin, Descendants, Children, Hierarchize and Members. However, this possibility is not related to our problem: changing the aggregation function according to a considered analysis dimension or hierarchy or level.

Our aim is to remove these limits by designing a conceptual multidimensional model integrating differentiated multiple aggregates. By *multiple* we mean that the same measure can be aggregated by several aggregation functions according to hierarchies or analysis axes, and by *differentiated* we mean that these aggregations may vary, depending on the chosen aggregation level.

In addition, aggregation functions are classified:

- From an aggregation mechanism point of view, aggregation functions belong to three different categories (Gray et al., 1996). The first corresponds to **distributive**

functions that calculate, at a level of granularity, aggregated values from the values already aggregated at the level of granularity just below (e.g. the sum of an amount per year can be calculated from the summed values of each semester). The second corresponds to **algebraic** functions that calculate aggregated values from stored intermediate results (e.g. the average of an amount per year can be calculated from the sum of the amounts and the count of occurrences). Finally, the third corresponds to **holistic** functions that cannot be calculated from intermediate results. In this case, aggregated values must be calculated from the elementary values of the lowest level of granularity, the base level (e.g. the RANK function).

- From a summarisability point of view, aggregation functions are classified into two groups (Abelló et al., 2006): (1) 'transitive' which guarantees summarisability, (2) 'non-transitive' which implies that aggregations must always be calculated from the base level.
- From a measure (data) point of view, aggregation functions are of three types (Pedersen et al., 2011): (1) applicable to additive data; (2) applicable to snapshot data that can be used for average calculations; (3) applicable to constant data or non-numerical data, i.e. data that can only be counted.

All these proposals as well as aggregation function classifications assume that the measure aggregation can be calculated from the base level. Our goal is to consider cases when the measure cannot be aggregated from the base level, using a concept named *aggregation constraints*.

3. Conceptual data model

3.1. Classical concepts

Let us define N, F and D such as:

$N = \{n_1, n_2, \ldots\}$ is a finite set of non-redundant names,
$F = \{F_1,\ldots,F_n\}$ is a finite set of facts, $n \geq 1$,
$\{D_1,\ldots,D_m\}$ is a finite set of dimensions, $m \geq 2$.

Definition 1. A *fact*, denoted F_i, $\forall$ i $\in[1...n]$, is defined by (n^{Fi}, M^{Fi}), where

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

We define a global measure set as $M = \bigcup_{i=1}^{n} M^{F_i}$

Definition 2. A *dimension*, denoted D_i, $\forall$ i $\in[1...m]$, is defined by (n^{Di}, A^{Di}, H^{Di}), where

$n^{Di} \in N$ is the name that identifies the dimension,
$A^{Di} = \{a_1^{D_i},\ldots,a_{r_i}^{D_i}\}$ is the set of the *attributes of the dimension*,
$H^{Di} = \{H_1^{D_i},\ldots,H_{s_i}^{D_i}\}$ is a set of *hierarchies*

Hierarchies organise dimension attributes (parameters and weak attributes) from the finest graduation level to the most general graduation level. Thus, a hierarchy defines valid navigation paths on an analysis axis.

We define the attribute set $A = \bigcup_{i=1}^{m} A^{D_i}$ and the hierarchy set $H = \bigcup_{i=1}^{m} H^{D_i}$

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

- $n^{Hj}\in N$ is the name that identifies the hierarchy,
- $P^{Hj} = \{p_1^{H_j},..., p_{q_j}^{H_j}\}$ is a set of attributes called *parameters*, $P^{Hj} \subseteq A^{Di}$,
- $\prec^{Hj} = \{(p^{Hj}_x, p^{Hj}_y) \mid p^{Hj}_x \in P^{Hj} \wedge p^{Hj}_y \in P^{Hj}\}$ is an antisymmetric and transitive binary relation between parameters. Recall that the antisymmetry means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k1}) \Rightarrow p^{Hj}_{k1} = p^{Hj}_{k2}$ while the transitivity means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k3}) \Rightarrow p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k3}$.
- $Weak^{Hj} : P^{Hj} \rightarrow 2^{A^{D_i}\setminus P^{H_j}}$ is an application that associates to each parameter a set of attributes, called *weak attributes* (2^N represents the power set of N).

We define parameter sets $P^{D_i} = \bigcup_{j=1}^{Si} P^{H_j}$ and $P = \bigcup_{i=1}^{m} P^{D_i} = \bigcup_{i=1}^{m} \bigcup_{j=1}^{Si} P^{H_j}$

Lemma 1. For each dimension D_j, a *root parameter*, denoted $Id^{Di} \in P^{Di}$, exists and defined as follows: $\forall j\in[1...s_i]$, $\forall p_k^{H_j}\in P^{Di}$, $Id^{Di} \neq p_k^{H_j} \mid Id^{Di} \prec^{Hj} p_k^{H_j}$.

Lemma 2. For each dimension D_i, an *extremity parameter*, denoted $All^{Di} \in P^{Di}$, exists and defined as follows: $\forall j\in[1...s_i]$, $\forall\ p_k^{H_j}\in P^{Di}$, $All^{Di} \neq p_k^{H_j} \mid p_k^{H_j} \prec^{Hj} All^{Di}$.

We also define weak attribute sets $W^{D_i} = \bigcup_{\forall j\in[1...s_i],\forall k\in[1...q_j]} Weak^{H_j}(p_k^{H_j})$ and $W = \bigcup_{i=1}^{m} W^{D_i} = \bigcup_{i=1}^{m} \bigcup_{\forall j\in[1...s_i],\forall k\in[1...q_j]} Weak^{H_j}(p_k^{H_j})$

Lemma 3. For each dimension D_i, all its attributes are exclusively either parameters or weak attributes, $P^{Di} \cap W^{Di} = \varnothing$ and $P^{Di} \cup W^{Di} = A^{Di}$.

Example 1. We illustrate our approach with an example of weather analysis. In this example, analysts study maximum, minimum and average temperatures, and wind speeds. To support these analyses, we establish an MDB with two facts defined as follows:

- $F_{Temperature}$ = ("Temperature", {Tem_Avg, Tem_Max, Tem_Min}).
- F_{Wind} = ("Wind", {Speed}).

These analyses are performed according to geographic, temporal and meteorological (wind direction) information. We consider that each country is composed of several regions. Each region includes departments, which contain cities. We also consider that every city has an administrative level, which can be a country capital, a department prefecture or a regional capital. Near cities, there are weather stations that measure temperatures and wind speeds several times a day.

The *Geography* dimension has two hierarchies that allow taking into account two ways for observing temperatures: simple or scientific. The **simple** aggregation adopts the same method used in weather forecasting: when presenting a country's maximum or minimum temperatures, it refers to the capital city's temperatures. Similarly, when temperatures in a region or a department are presented, these are the temperatures of a significant city (regional capital or departmental prefecture). The **scientific** aggregation takes into account the temperatures of all regions to calculate those of a country. Similarly, when calculating temperatures in a region or in a department, we take into account respectively all its departments or all its cities.

The *Time* dimension has only one hierarchy that organises hourly granulations at which temperatures have been measured during the day. The *Direction* dimension has a hierarchy that has only the root and the ALL levels. The *Date* dimension has several hierarchies that organise the different granularities of the day.

In the following paragraph, we present the formal definitions of the dimensions:

- D_{Time} = (*"Time"*, {$a_{Every\_3\_hours}$, $a_{Quarter\text{-}day}$, $a_{Half\text{-}day}$, ALL^{Time}}, {H_{HTime}}) with
  - H_{HTime} = (*"HTime"*, {$a_{Every\_3\_hours}$, $a_{Quarter\text{-}day}$, $a_{Half\text{-}day}$, ALL^{Time}}, {($a_{Every\_3\_hours}$, $a_{Quarter\text{-}day}$), ($a_{Quarter\text{-}day}$, $a_{Half\text{-}day}$), ($a_{Half\text{-}day}$, ALL^{Time})}).
- $D_{Direction}$ = (*"Direction"*, {a_{from}, $ALL^{Direction}$}, {H_{Hdir}}) with
  - H_{Hdir} = (*"Hdir"*, {a_{from}, $ALL^{Direction}$}, {(a_{from}, $ALL^{Direction}$)}).
- D_{Date} = (*"Date"*, {a_{Day}, a_{LibD}, a_{Month}, a_{LibM}, a_{Week}, a_{Season}, a_{Year}, ALL^{Date}}, {H_{Hmonth}, H_{Hweek}, $H_{Hseason}$}) with
  - H_{Hmonth} = (*"Hmonth"*, {a_{Day}, a_{Week}, a_{Year}, ALL^{Date}}, {(a_{Day}, a_{Week}), (a_{Week}, a_{Year}), (a_{Year}, ALL^{Date})}, {(a_{Day}, {a_{LibD}})}),
  - H_{Hweek} = (*"Hweek"*, {a_{Day}, a_{Month}, a_{Year}, ALL^{Date}}, {(a_{Day}, a_{Month}), (a_{Month}, a_{Year}), (a_{Year}, ALL^{Date})}, {(a_{Day}, {a_{LibD}}), (a_{Month}, {a_{LibM}})}), and
  - $H_{Hseason}$ = (*"Hseason"*, {a_{Day}, a_{Season}, a_{Year}, ALL^{Date}}, {(a_{Day}, a_{Season}), (a_{Season}, a_{Year}), (a_{Year}, ALL^{Date})}, {(a_{Day}, {a_{LibD}})}).
- $D_{Geography}$ = (*"Geography"*, {a_{City}, $a_{Administrative\_Level}$, $a_{Department}$, $a_{D\_Surface}$, a_{Region}, $a_{R\_Surface}$, $a_{Country}$, $a_{C\_Surface}$, $ALL^{Geography}$}, {H_{Hgeo_Scien}, H_{Hgeo_simp}}) with
  - H_{Hgeo_Scien} = (*"Hgeo_Scien"*, {a_{City}, $a_{Department}$, a_{Region}, $a_{Country}$, $ALL^{Geography}$}, {(a_{City}, $a_{Department}$), ($a_{Department}$, a_{Region}), (a_{Region}, $a_{Country}$), ($a_{Country}$, $ALL^{Geography}$)}, {(a_{City}, {$a_{Administrative\_Level}$}), ($a_{Department}$, {$a_{D\_Surface}$}), (a_{Region}, {$a_{R\_Surface}$}), ($a_{Country}$, {$a_{C\_Surface}$})}), and
  - H_{Hgeo_simp} = (*"Hgeo_simp"*, {a_{City}, $a_{Department}$, a_{Region}, $a_{Country}$, $ALL^{Geography}$}, {(a_{City}, $a_{Department}$), ($a_{Department}$, a_{Region}), (a_{Region}, $a_{Country}$), ($a_{Country}$, $ALL^{Geography}$)}, {(a_{City}, {$a_{Administrative\_Level}$})}).

3.2. Extensions for differentiated multiple aggregations

In order to specify the differentiated multiple aggregations, the previous definitions are extended as follows:

- The first extension concerns the aggregation process.
  - **General aggregation** consists in aggregating the values of a measure using any hierarchical level and always the same function.
  - **Multiple dimensional aggregation** consists in aggregating measure values using different aggregation functions depending on the used dimension.
  - **Multiple hierarchical aggregation** consists in aggregating measure values using different aggregation functions that depend on the used hierarchy.
  - **Differentiated aggregation** consists in aggregating measure values by changing the aggregation function between two consecutive aggregation levels.
- The second extension concerns the **execution order** of the aggregation functions between dimensions. It is possible to combine several different aggregation functions over different dimensions during an analysis. These functions are often non-commutative. Thus, an execution order has to be considered.
- The third extension concerns **aggregation constraints**. All aggregations are not carried out uniformly using systematically all lower hierarchical levels

(contrarily to the aggregation process designed in classical multidimensional models). As a consequence, we introduce a constraint mechanism on the aggregation process to indicate the valid aggregation level that allows obtaining the upper level.

Let $\mathcal{F} = \{f_1, f_2, ...\}$ be a finite set of aggregation functions.

Definition 4. A *multidimensional schema*, denoted S, is defined by (F, D, Star, Aggregate), where:

- $F = \{F_1,...,F_n\}$ is the set of facts, if $|F|=1$ then the multidimensional schema is called a *star schema* while if $|F|>1$ it is a *constellation schema*,
- $D = \{D_1,...,D_m\}$ is the 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 (note that 2^D represents the powerset of D).
- Aggregate: $M \rightarrow 2^{\mathbb{N}^* \times F \times 2^D \times 2^H \times 2^P \times \mathbb{N}^-}$ associates each measure to a set of aggregation functions. Aggregate defines different types of aggregation functions supported by our model (general, multiple dimensional, multiple hierarchical, differentiated):
  - *General aggregation*: 2^D, 2^H and 2^P are not used ($2^D = \varnothing$, $2^H = \varnothing$ and $2^P = \varnothing$).
  - *Multiple dimensional aggregation*: 2^H and 2^P are not used ($2^H = \varnothing$ and $2^P = \varnothing$). Here, the function aggregates the measure over the entire considered dimension.
  - *Multiple hierarchical aggregation*: 2^P is not used ($2^P = \varnothing$). Here, the function aggregates the measure over the entire considered hierarchy.
  - *Differentiated aggregation*: the function aggregates the measure between a considered parameter and the parameter directly above it in the same hierarchy.

$\mathbb{N}^*$ binds to each aggregation function an execution order. The aggregation function with the smallest order has the highest priority. If two aggregation functions are commutative, then both functions will have the same order.

$\mathbb{N}^-$ is used to constrain aggregations by indicating a specific level from which the considered aggregation must be calculated. An unconstrained aggregation will be associated with 0 while a constrained aggregation will be associated with a negative value to force the calculation from a chosen level lower than the considered level.

Example 2. Following the example introduced in Example 1, decision makers analyse wind speeds according to their *Direction*, the *Geography* and the *Date*, whereas they analyse temperatures according to *Time, Date* and *Geography.* We formally define this MDB as (F, D, Star, Aggregate) where

- $F = \{F_{Temperature}, F_{Wind}\}$
- $D = \{D_{Geography}, D_{Date}, D_{Time}, D_{Direction}\}$,
- Star: $F \rightarrow 2^D$ |
  $Star(F_{Temperature}) = \{D_{Geography}, D_{Date}, D_{Time}\}$
  $Star(F_{Wind}) = \{D_{Geography}, D_{Date}, D_{Direction}\}$
- Aggregate: $M \rightarrow 2^{\mathbb{N}^* \times F \times 2^D \times 2^H \times 2^P \times \mathbb{N}^-}$
  Aggregate(Speed) = {(1, AVG(Speed), {}, {}, {}, 0),

(1, SELECT_CENTRE(Administrative_Level, Speed), {Geography}, {}, {}, 0),
(1, AVG(Speed), {Geography}, {H_geo_simp}, {Country}, −4)[1],
(1, AVG(Speed), {Geography}, {H_geo_scien}, {Country}, −4)}

- Aggregate(Tem_Avg) = {(2, AVG(Tem_Avg), {}, {}, {}, 0),
(2, SELECT_CENTRE(Administrative_Level, Tem_Avg), {Geography}, {H_geo_simp}, {}, 0),
(2, AVG(Tem_Avg), {Geography}, {H_geo_simp}, {Country}, −1),[2]
(1, AVG(Tem_Avg), {Geography}, {H_geo_scien}, {City}, 0),
(1, AVG_W(Tem_Avg, D_Surface), {Geography}, {H_geo_scien}, {Department}, −1),
(1, AVG_W(Tem_Avg, R_Surface), {Geography}, {H_geo_scien}, {Region}, −1),
(1, AVG_W(Tem_Avg, C_Surface), {Geography}, {H_geo_scien}, {Country}, −1)}
- Aggregate(Tem_Min) = {(1, MIN(Tem_Min), {}, {}, {}, 0),
(1, SELECT_Centre(Administrative_Level, Tem_Min), {Geography}, {H_geo_simp}, {}, 0),
(1, MIN(Tem_Min), {Geography}, {H_geo_simp}, {Country}, −4)}
Aggregate(Tem_Max) = {(1, MAX(Tem_Max), {}, {}, {}, 0),
(1, SELECT_CENTRE (Administrative_Level, Tem_Max), {Geography}, {H_geo_simp}, {}, 0),
(1, MAX(Tem_Max), {Geography}, {H_geo_simp}, {Country}, −4)}

The function Avg_W(X, Y) takes as input two numerical parameters. It returns the average of values of X weighted by Y. In other words, the weighted average Avg_W (X, Y) = $\frac{\sum(X \times Y)}{\sum Y}$. The function Select_Centre(I,M) takes as input two numerical parameters. It returns the value M which corresponds to the value of Max(I). If there are several Max(I), then the function selects the average of the corresponding M values. For example, if Select_Centre (Administrative_Level, Tem_Avg) is done at the Country level, it returns the temperature of the capital city (the city with the highest administrative level). Moreover, if applied at the department or region level, it returns respectively the temperature of the department prefecture or that of the region's capital city.

If we analyse for example the average temperatures using the *Date* and the *Time* dimensions, the decisional system must use the general function (AVG) to aggregate the measure values because there is no other specific function for these dimensions. If we analyse using the *simple* hierarchy of the *Geography* dimension, the system uses the multiple hierarchical function (SELECT_CENTRE). Except for the *ALL* level, the system must calculate the aggregated values from the *country* level using the differentiated function (AVG). However, using the *scientific* hierarchy, the system uses on each aggregation level a different differentiated aggregation function. Aggregation is done using the level directly below (AVG to aggregate the *department* level using the *city* level and AVG_W for the other levels). Furthermore, if we analyse data using two or more dimensions, then functions over the *scientific* hierarchy are a priority; that means that we must apply it before the other functions.

Lemma 4. Aggregation functions ensure the full *coverage* of multidimensional schemas. Thus, there does not exist any parameter (i.e. aggregation levels) for which the aggregation function to be applied is unknown.

$$\forall\, i \in [1..n],\ \forall\, m_k \in \mathrm{M}^{\mathrm{Fi}},\ \exists\, f \in \mathcal{F},\ \exists\, x_1 \in \mathbb{N}^*,\ \exists\, x_2 \in \mathbb{N}^-,$$

$$\begin{cases} |(x_1, f, \{\}, \{\}, \{\}, x_2) \in Aggregate(m_k) \\ \forall D_j \in Star(F_i) | (x_1, f, \{D_j\}, \{\}, \{\}, x_2) \in Aggregate(m_k) \\ \forall H_s \in H^{D_j} | (x_1, f, \{D_j\}, \{H_s\}, \{\}, x_2) \in Aggregate(m_k) \\ \forall P_q \in P^{H_s} \backslash \{All^{D_j}\} | (x_1, f, \{D_j\}, \{H_s\}, \{P_q\}, x_2) \in Aggregate(m_k) \end{cases}$$

Less formally, the *coverage* of the schema is carried out in several ways:

- By using a general aggregation function,
- By using a multiple dimensional aggregation function for each dimension,
- By using a multiple hierarchical aggregation function for each hierarchy,
- By using a differentiated aggregation function for each aggregation level,
- By combining multiple dimensional and hierarchical aggregation functions with differentiated ones. Each dimension or hierarchy having no multiple function must have a differentiated function for each aggregation level (i.e. parameter).

3.3. *Graphical formalisms*

We introduce a graphical formalism to ease the understanding of the schema of the MDB. We distinguish two levels of graphical formalism.

Structural schema. The structural schema allows globally visualising the multidimensional schema of the MDB by hiding the aggregation mechanisms. This global view is obtained from the *Star* function.

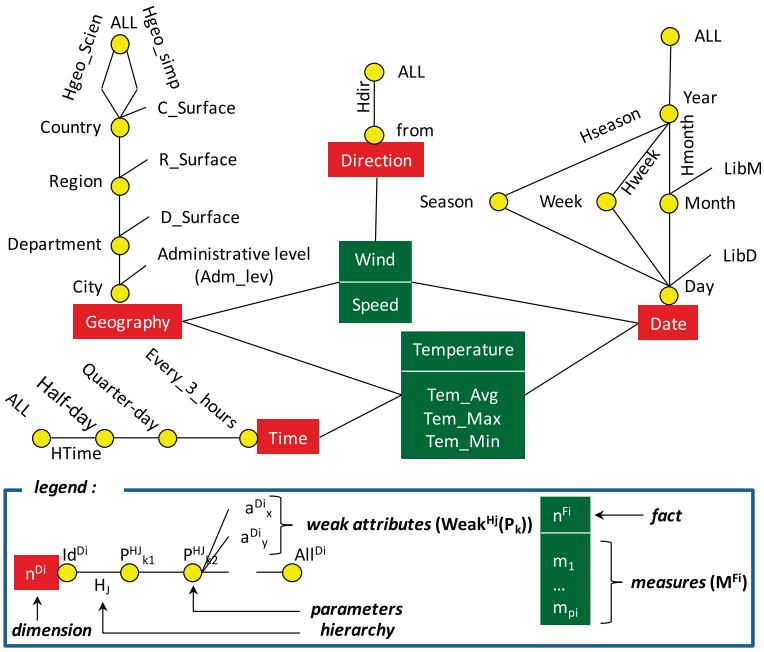

Figure 1. Structural schema of the multidimensional database (MDB).

Example 3. The MDB formally described in the previous examples is displayed using our graphical formalism as in Figure 1. The proposed graphical formalism is based on the works of Golfarelli et al. (1998) and Ravat, Teste, Tournier, and Zurfluh (2007; Ravat et al., 2008). This MDB allows analysis of weather indicators (measures): average, minimal and maximal temperatures as well as wind speeds. These measures are organised according to two facts: *Wind* and *Temperature*. Each fact is associated to three dimensions among four available dimensions: *Geography*, *Date*, *Time* and *Direction*.

Aggregation schema. An aggregation schema is defined for each measure $m_i \in F_i$ using the function *Aggregate*. Each schema details the aggregation mechanisms during an analysis for a selected measure by displaying only structural elements directly linked to

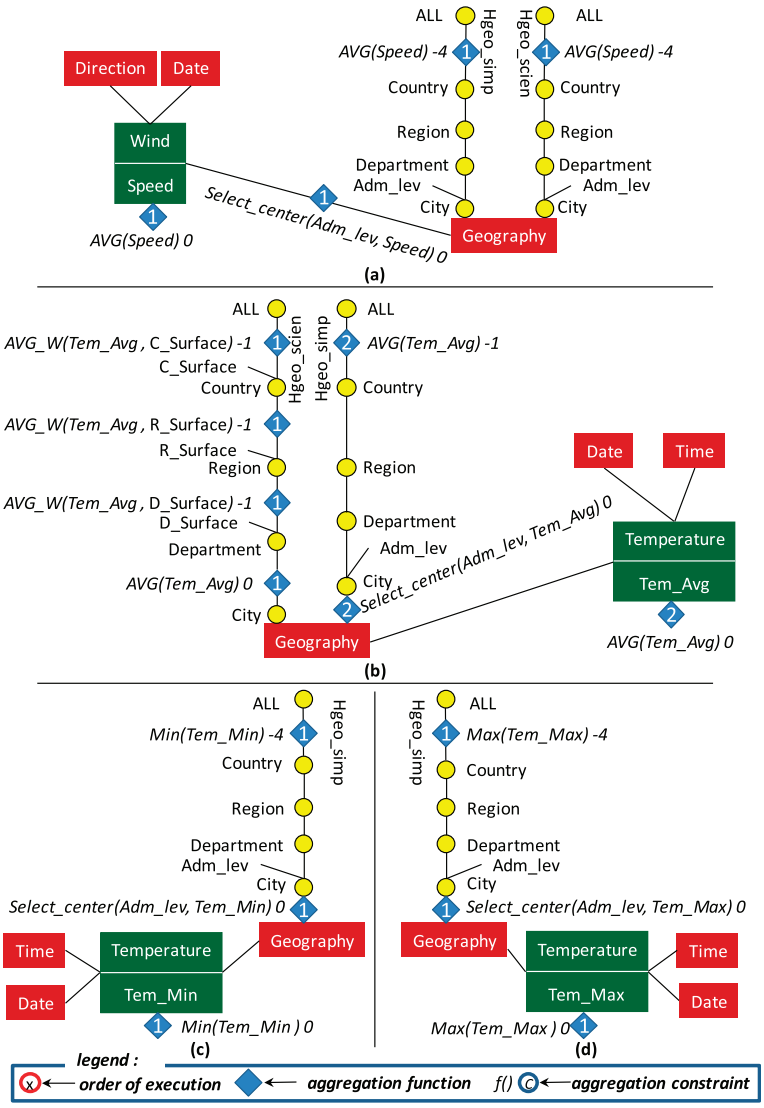

Figure 2. Aggregation schemas.
Note: *Adm_lev* = *Administrative_Level*.

the measure: F_i and $Star(F_i)$. This schema is an extension of our previous work (Hassan et al., 2012).

Example 4. From the previously presented structural schema, it is possible to obtain the aggregation schema of the different measures. Figure 2 details four aggregation schemas that correspond to the four measures *Speed*, *Tem_Avg, Tem_Min* and *Tem_Max* of our case study.

As shown in Figure 2, the hierarchies are presented in split version, unlike the structural schema which is presented in compact version. The aggregation functions are represented with lozenges. Each lozenge shows also the execution order and the possible aggregation constraint. The positions of the lozenges depend on the function type:

- A general function is represented with a lozenge on the side of the fact,
- A multiple dimensional function is represented on the edge linking the fact to the dimension,
- A multiple hierarchical function is represented in the bottom of the hierarchy,
- A differentiated function is represented on the edge between two parameters.

The aggregation with a constraint assigned to –1 is calculated from the level directly below; e.g. the average temperature for each country in the *scientific* hierarchy is calculated from temperatures per regions, whereas the general average wind speed (level ALL) is calculated from the wind speeds per city because the constraint is assigned to –4.

In summary, we suggest to graphically represent an MDB from two levels: on the one side, the structural schema provides a global view of the multidimensional structural elements (facts, dimensions and hierarchies) whereas the complexity due to aggregations is hidden. On the other side, aggregation schemas provide a detailed view of the aggregation mechanisms of one measure (general, multiple dimensional, multiple hierarchical and differentiated aggregations, aggregation constraints and aggregation order) while the complexity due to the constellation schema is minimised.

4. Logical model

4.1. Classical approach

Our current implementation uses an R-OLAP approach, consisting in using relational databases for implementing multidimensional schemas (Kimball, 1996). This approach has several advantages, such as reusing well-proven data management properties, and the capacity of systems to handle large data volumes.

The conceptual multidimensional structures (facts and dimensions) are translated at the logical level into relations (Kimball, 1996). The hierarchical structure of the dimension attributes (hierarchies) at the conceptual level is used to optimise the MDB. This optimisation consists in pre-calculating aggregations required by decision makers that perform OLAP analyses within the multidimensional space (Gray et al., 1996) using queries. These pre-aggregations can be modelled using a pre-aggregation lattice (Chaudhuri & Dayal, 1997) where each node represents a pre-aggregate and each edge represents a path for calculating the aggregates. When the aggregation function used is distributive or algebraic, an aggregate can be calculated from the level just below, while in the case of a holistic aggregation function, the aggregate has to be calculated using the all-the-way-down base relations (the lowest level).

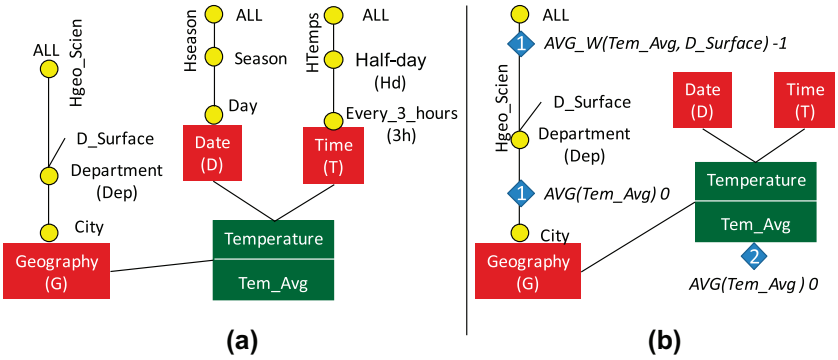

Figure 3. Structural and aggregation schemas of simplified example.

Example 5. To illustrate the classical R-OLAP implementation, we use a simplified MDB example (see Figure 1); our structural schema is detailed in Figure 3a.

Here, facts and dimensions are translated into relational tables. Hierarchies are used to complement the schema with a set of relations that pre-compute aggregates, these being possibly required by user queries. The lattice of pre-computed aggregates, displayed in Figure 4, represents all these additional relations.

Each node represents a relation; e.g. the nodes *Dep_3h_Day* and *ALL^{G}_3h_Day* correspond to the following respective relations:

Dep_3h_Day(**Department, Day, Each_3_Hours,** Tem_Avg)

ALL^{G}_3h_Day(**Day, Each_3_Hours,** Tem_Avg)

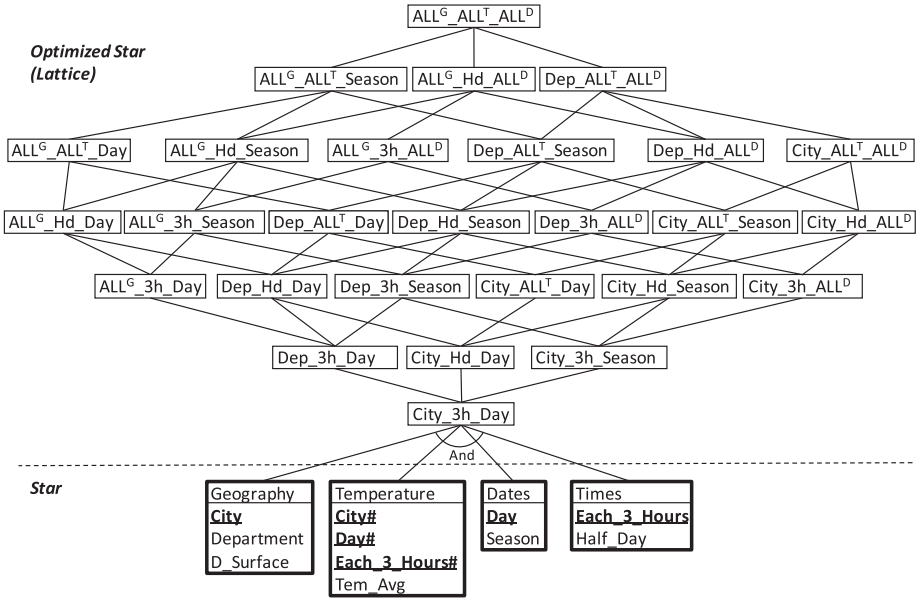

Figure 4. Classical lattice.
Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; ALL^{G} = ALL^{Geography}; ALL^{T} = ALL^{Time}; ALL^{D} = ALL^{Date}.

Within these relations, the attribute *Tem_Avg* represents the average temperatures calculated by the aggregation function AVG. Note that this function is algebraic. In the classical approach, contrarily to our proposition, a unique aggregation function is used in the whole lattice for the measure *Tem_Avg*.

4.2. Extending the approach with multiple and differentiated aggregations

Extensions previously defined impact the "classical' lattice.

Edge types. Unlike the classical approach, where only one aggregation function is considered, multiple and differentiated aggregation functions require the use of different aggregations on each edge of the lattice.

Using different aggregation functions for the same measure according to parameters requires differentiating lattice edges. This allows defining for each pair of nodes the corresponding aggregation function.

Example 6. Figure 3b presents the aggregation schema of our simplified example. Three aggregation functions are used to process average temperatures. For each department, we use the classical average (AVG) function. However, the average temperature over all departments takes into account the size of each department and uses a weighted function (AVG_W). Thus, in the lattice, it is necessary to distinguish the edge between *City* and *Department* parameters (that use AVG) from the edge between *Department* and *ALL* (that use AVG_W). In Figure 5, simple lines correspond to the AVG function and double lines are for AVG_W.

Pruning the lattice. In the conceptual model, there is an order between the functions. This represents the order in which the aggregation functions have to be used among the dimensions. This order generates invalid edges in the lattice; hence, it is possible to prune them.

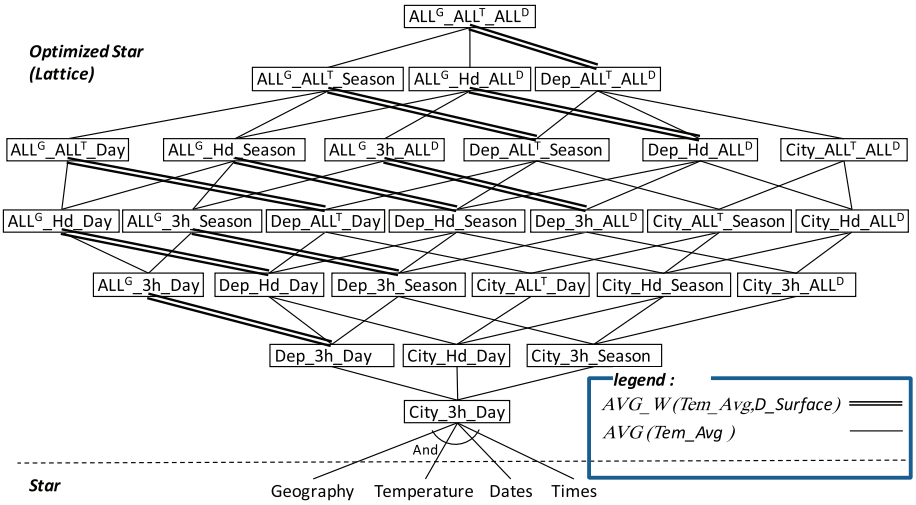

Figure 5. Lattice with typed edges.
Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; $ALL^G = ALL^{Geography}$; $ALL^T = ALL^{Time}$; $ALL^D = ALL^{Date}$.

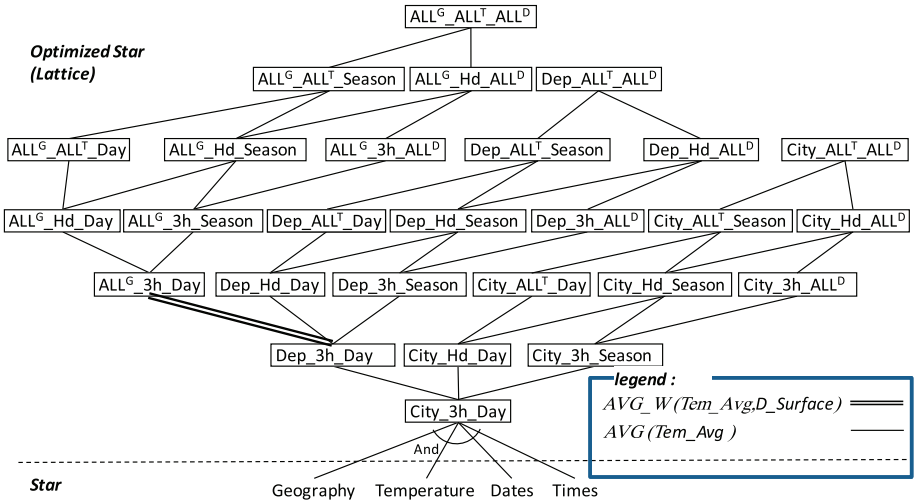

Figure 6. Lattice with pruned edges.
Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; $ALL^G = ALL^{Geography}$; $ALL^T = ALL^{Time}$; $ALL^D = ALL^{Date}$.

Example 7. In our example (see Figure 3 b), the weighted average function, *AVG_W* (*Tem_Avg*, *D_Surface*) cannot be used on the *Geography* dimension after the use of *AVG*(*Tem_Avg*) on the *Date* dimension. In other words, it is impossible to process the season average temperature (node $ALL^G\_ALL^T\_Season$) from department average temperatures for each season (node $Dep\_ALL^T\_Season$). Thus, the edge between these nodes is deleted. Figure 6 shows the lattice after deleting the invalid edges.

Modifying edges. We also represent the possibility of calculating aggregations from another level than the one directly below the selected one. We use a constraint on the

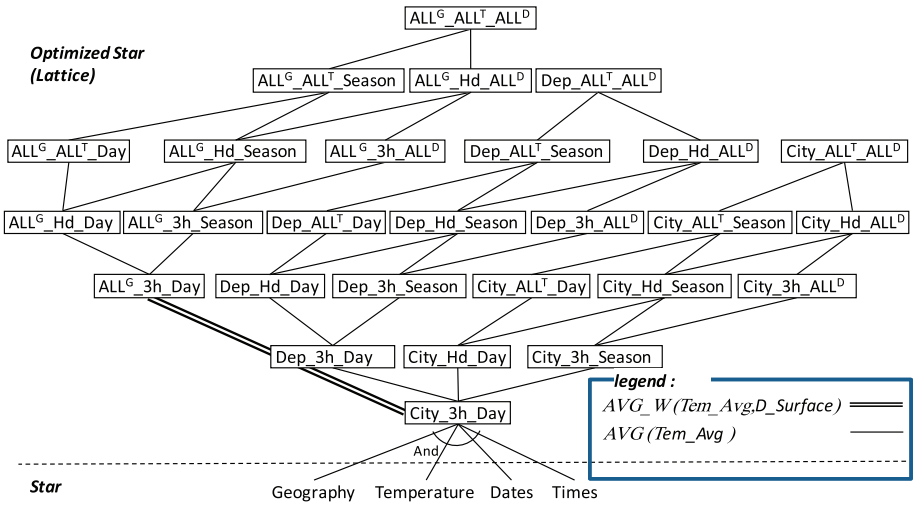

Figure 7. Lattice with constraint = –2.
Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; $ALL^G = ALL^{Geography}$; $ALL^T = ALL^{Time}$; $ALL^D = ALL^{Date}$.

aggregation with a value of 0 (the aggregation can be calculated from any lower level) or –1 (the aggregation can only be calculated from the level directly below the selected one). Such constraints imply possible path changes in the lattice.

Example 8. In our example, the general average temperature (level ALL) is calculated from department temperatures (constraint value = –1). Had we chosen the hypothesis of calculating this temperature using city temperatures, the constraint would have been assigned to –2 and the lattice would have been as in Figure 7.

Blocking transitivity. Constraints associated to aggregations also have an impact on the corresponding constrained edges. This implies that a node can only be calculated from another precise node. It is thus forbidden to calculate an upper node using transitivity on the lower nodes, as would normally be possible. Thus, aggregation paths are blocked upwards as soon as there is a constrained edge.

Example 9. The node *City_Hd_Season* can be calculated from the node directly below *City_3h_Season*; by transitivity it could also be calculated from *City_3h_Day*. However, the edge generated from the constraint on the function *AVG_W*(*Tem_Avg*, *D_Surface*) associating the nodes *Dep_3h_Day* and ALL^G*_3h_Day* blocks transitivity. Thus, ALL^G*_3h_Day* can be calculated from *Dep_3h_Day* (node directly below) but not from another lower node (such as *City_3h_Day*).

Similarly, different execution orders between functions also block transitivity. This implies non-transitive edges within the lattice (aggregations will only be calculated from the node directly below).

Example 10. The node *Dep_Hd_Season* can be calculated by transitivity from *Dep_3h_Day* but not from *City_3h_Day*. Indeed, the schema forces the calculus to start with temperatures according the Avg() function over the *Geography* dimension (node

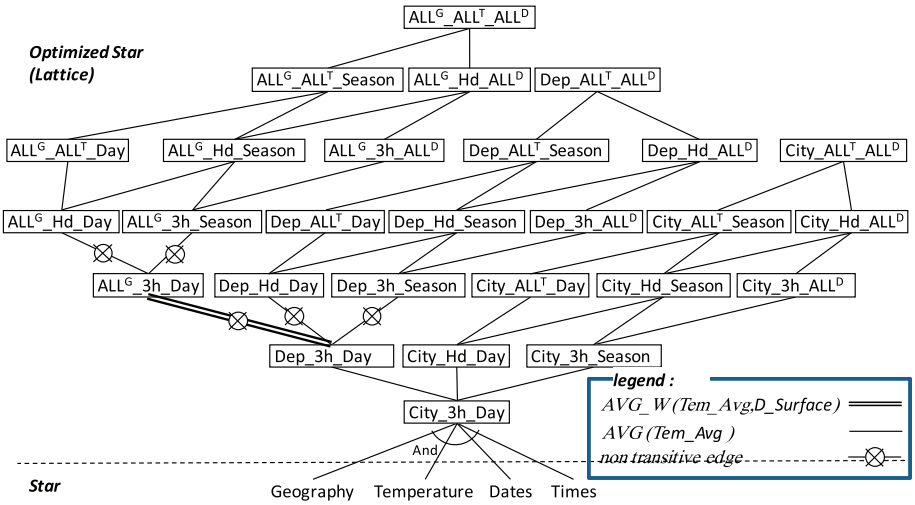

Figure 8. Controlled pre-aggregate lattice.
Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; $ALL^G = ALL^{Geography}$; $ALL^T = ALL^{Time}$; $ALL^D = ALL^{Date}$.

Dep_3h_Day) in order to be able to process temperatures according to the Avg() function over *Date* and *Time* dimensions (node *Dep_Dh_Season*).

Figure 8 shows the final controlled pre-aggregate lattice. Edges with crossed circles are obtained either from aggregation constraints or from an execution order.

5. Validations

We implemented a prototype, described hereafter. We then detail the meta-schema on which the prototype is based. Finally, validation experiments are detailed.

5.1. OLAP-Multi-Function Prototype

Our prototype *OLAP-Multi-Function* was implemented using Java 6 on top of the Oracle 11 *g* Database Management System (DBMS). *OLAP-Multi-Function* is an extension of a previous prototype: *Graphic OLAP* (Ravat et al., 2007, 2008). *Graphic OLAP* allows defining and manipulating an R-OLAP constellation as well as visualising and querying the multidimensional data with the use of a graphical representation and a multidimensional table (MT). Displaying the constellation is done using a meta-schema that describes the multidimensional model (see Figure 9). The prototype consists of three levels: interfaces, processing and storage.

Interfaces. The main functionality of *OLAP-Multi-Function* is visualising and facilitating the integration and the use of aggregation functions in the multidimensional model. By using the meta-schema and a hyperbolic space, we present the aggregation schemas (rear window in Figure 10). We use a graphic interface (top window in Figure 10) allowing the definition of our four types of aggregation functions:

General: we determine only the function.
Multiple dimensional: we define the function and dimension.
Multiple hierarchical: we specify the function, dimension and hierarchy.
Differentiated: we determine the function, dimension, hierarchy and parameter.

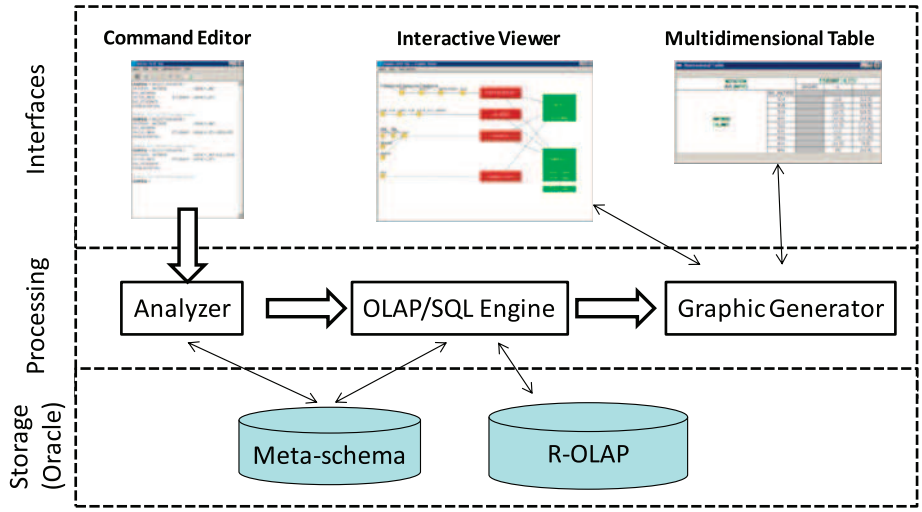

Figure 9. Prototype architecture.

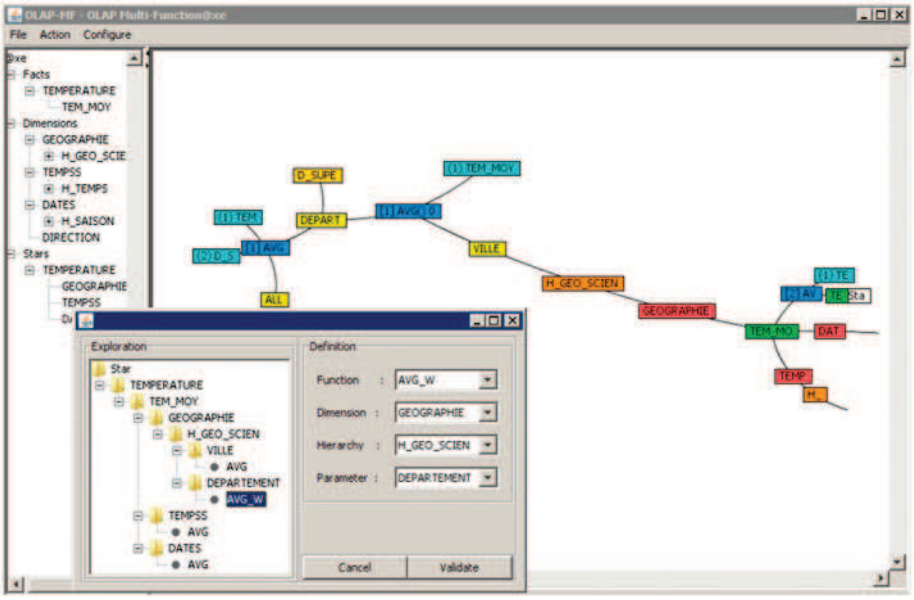

Figure 10. Hyperbolic view and aggregation functions definition. (French implementation of our example).

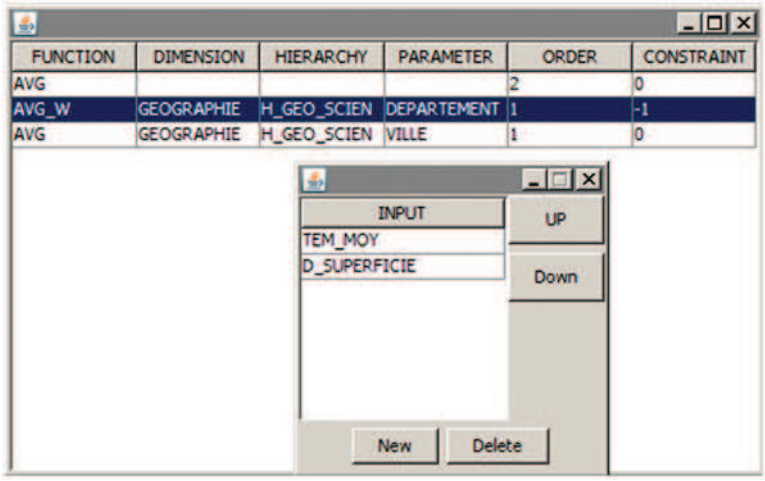

Figure 11. Defining function inputs, execution order and aggregation constraints.

To define aggregation constraints and execution orders of the functions, we use another graphic interface (Figure 11).

A command line editor allows entering textual orders in an OLAP-SQL language (Ravat, Teste, & Zurfluh, 2002) The result of a query is a multidimensional table containing the requested data.

Example 11. For analysing the average temperatures by season and by department (see Figure 3 a), the OLAP-SQL command is as follows:

```
SELECT Tem_Avg
ON ROWS Geography USING Hgeo_Scien (Department)
ON COLUMNS Date USING Hseason (Season)
FROM Temperature;
```

Processing. Each OLAP-SQL command is lexically and syntactically analysed to be validated. Validated commands are translated by the OLAP/SQL engine and send to the RDBMS. Queries are calculated and results are shown by the graphic generator in a multidimensional table.

Example 12. To perform the command of Example 11, the OLAP/SQL engine sends to the database the following query:

```
SELECT Department, Season, AVG(Tem_Avg) AS Tem_Avg
FROM (SELECT Date.Season, Geography.Department, Time.Each_3_Hours,
             Date.Day, AVG(Temperature.Tem_Avg) AS Tem_Avg
      FROM Date, Geography, Temperature, Time
      WHERE Temperature.Each_3_Hours = Time.Each_3_Hours
        AND Temperature.City = Geography.City
        AND Temperature.Day = Date.Day
      GROUP BY Date.Season, Geography.Department,
             Time.Each_3_Hours, Date.Day)
GROUP BY Season, Department
```

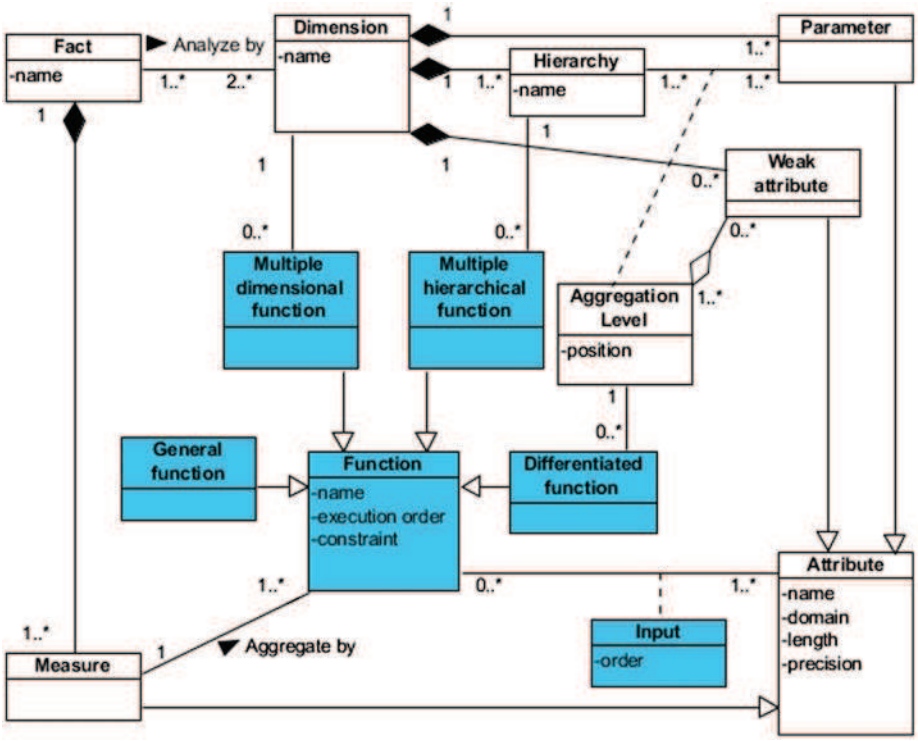

Figure 12. Meta-schema.

Storage. The storage level includes two databases. The first one contains facts and dimension data implemented with the R-OLAP model. The second one contains the meta-schema describing the structures of the multidimensional schema (facts, dimensions and hierarchies) as well as the aggregation functions to build valid and coherent SQL queries. Figure 12 shows this meta-schema using a Unified Modeling Language (UML) class diagram. White classes describe the classical model (*F*, *D* and *Star* of Definition 4) while shaded classes describe our extensions (*Aggregate* of Definition 4).

We can note that, according to this meta-schema, a fact is composed of measures and can be analysed by several dimensions. Each dimension consists of parameters, hierarchies and weak attributes. Parameters are ordered to represent aggregation levels of hierarchies. A parameter can belong to several hierarchies. For each aggregation level, there may exist a set of weak attributes.

In this meta-schema, note that a measure can be aggregated by several aggregation functions (class *Function*) but the aggregation function is linked to only one measure. The class *Function* has an attribute (*constraint*) that is used to force the aggregation, indicating a lower aggregation level from which the considered aggregation must be calculated. It has also an attribute (*execution order*) used to order the execution of non-commutative aggregation functions. The function takes at least one input which is either a measure, a parameter or a weak attribute.

We have four types of aggregation function that inherit from class *Function*:

- A *General function* is linked only with the measure. The measure has at most one general aggregation function. This function is used to aggregate the measure values in the multidimensional space where there is no other specific aggregation function.
- A *Multiple dimensional function* is associated with only one dimension over which this function is applied. The dimension can have several aggregation functions, one for each different measure.
- A *Multiple hierarchical function* is linked with only one hierarchy over which we apply this function. As for the previous function, the hierarchy can have several aggregation functions, one for each different measure.
- A *Differentiated function* is associated with only one aggregation level for aggregating the measure values between this level and the level directly above it in the same hierarchy. Similarly, the aggregation level can have several aggregation functions, one for each different measure.

5.2. Experiments

During our research, we carried out a series of experiments.

Experiment 1.

We studied relations between the functions' priorities (execution order) and the complexity of a lattice of a measure.

Collection. We use three multidimensional schemas with four, five and six dimensions. We consider that each dimension has only one hierarchy with four granularity levels (parameters) and functions used on a same dimension have the same execution order.

Protocol. Firstly, all dimensions have the same execution order. Then we change the execution order of dimensions one by one until each dimension has its own execution order. We monitor the evolution of number of edges of the lattice.

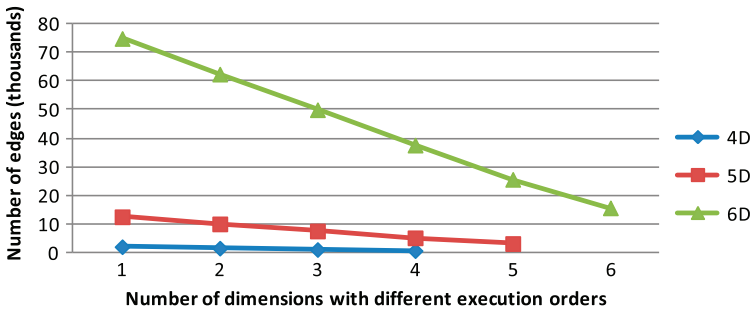

Figure 13. Number of edges according to the number of execution orders.

Results. Figure 13 shows that the number of edges of the lattice decreases as the number of different execution orders increase. Note that for all three cases this decrease is linear. These results show that the execution order allows us to significantly reduce the number of edges of the lattice.

Experiment 2.

We studied relations between the number of dimensions and the complexity of a lattice of a measure.

Collection. We use two multidimensional schemas with dimensions of four parameters. In the first, all dimensions have the same priority, i.e. a single value of execution order for all functions. The second has different execution orders for each dimension.

Protocol. We observe the number of nodes, edges and non-transitive edges depending on the number of dimensions (from two to six)

Results. Figure 14 shows that if all dimensions have the same priority, the number of edges (curve *edges*) increases much faster than the number of nodes (curve *nodes*). If each dimension has a different value of execution order, the increase in the number of non-transitive edges (curve *non-transitive edges*) is clearly less than the increase in the total number of edges (curve *optimised edges*). The curve *optimised edges* is identical to the curve *nodes* (indicating that for each node there is only one edge). Thus, if each dimension has a different execution order, the optimised lattice is no longer a graph, but a tree (Figure 15); hence, to calculate each node, there is only one path from the base level. This reduces the number of edges and makes some nodes critical as many paths require the calculation of these nodes (for example, the nodes *Dep_3h_Day* and ALL^G*_3h_Day* in Figure 15).

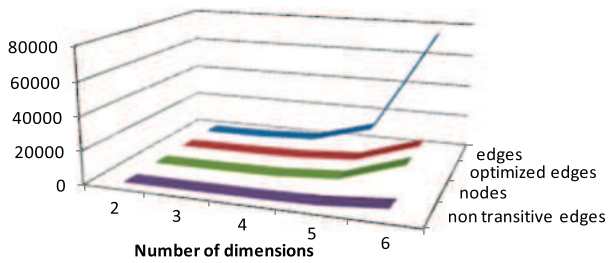

Figure 14. Number of edges and nodes according to the number of dimensions.

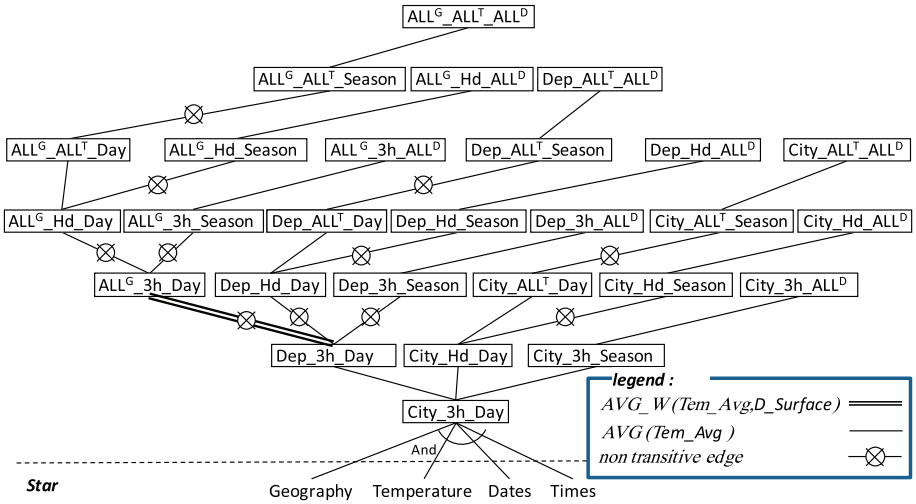

Figure 15. Controlled lattice of pre-computed aggregates (each dimension has a different order). Note: 3h = Every_3_hours; Dep = Department; Hd = Half-day; $ALL^G = ALL^{Geography}$; $ALL^T = ALL^{Time}$; $ALL^D = ALL^{Date}$.

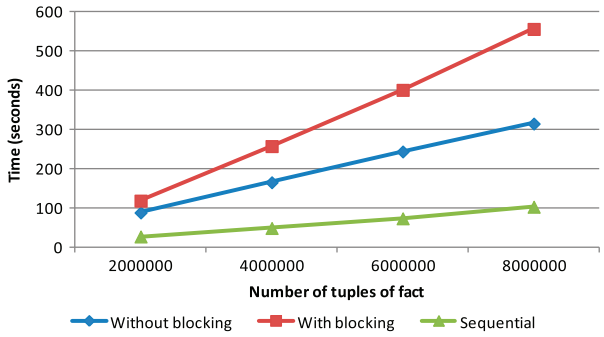

Figure 16. Time of creating the lattices according to the number of tuples of fact.

Experiment 3.

We studied the impacts of non-transitive edges on the time of creating the lattice.

Collection. We work on two different versions of the lattice of Figure 15. The first is identical to Figure 15 where 42% of the edges are non-transitive. The second differs from the first as it does not contain any non-transitive edges. To avoid ambiguities resulting from the difference between the functions, we use only one function (AVG) on the entire lattice.

Protocol. We observe the time for creating the entire lattice in accordance with the number of tuples of fact (from 2 to 8 million). We create the lattice in two ways: (1) we create each node from the base node (fact); (2) we create each node from the node directly below.

Results. Non-transitive edges do not change the time required to build the node from the node directly below. Accordingly, time required to build the two versions of the lattice is identical (curve *Sequential* in Figure 16). However, non-transitive edges significantly increase the time required for building a node from the base node. This is

shown clearly in Figure 16, where the additional time to build the lattice with 42% of non-transitive edges (curve *With blocking*) is up to 75% of the time required to build the lattice without non-transitive edges (curve *without blocking*). Here, in the case of a large number of non-transitive edges, the need to use pre-aggregates is clearly shown.

6. Conclusion

In this paper, we propose a conceptual model for multidimensional data that allow associating several aggregation functions to a measure taking into account the analysis axes (multiple functions) and the granularity levels (differentiated functions). Our model is expressive enough to control the validity of these functions. Aggregation constraints allow us to define the level where aggregation must be calculated, and execution order defines the order between functions required to calculate the aggregations. We also provide a two-level graphical formalism: the structural schema describes the multidimensional structures while hiding the aggregation complexity, and aggregation schemas describe the aggregation mechanisms for each measure. Moreover, at the logical level, we show how to use our conceptual model to provide controlled pre-aggregation lattices.

As future work, we are considering revisiting algorithms that compute pre-aggregates by using our model and then by defining an adapted cost function in order to evaluate the building and maintenance cost of an optimised MDB. We are also studying the impact of OLAP algebraic operators on our model.

Notes

1. Values are aggregated directly from the base level.
2. Values are aggregated from the already aggregated values of the level directly below the considered one.

References

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

Boulil, K., Bimonte, S., & Pinet, F. (2011). Un modèle UML et des contraintes OCL pour les entrepôts de données spatiales. De la représentation conceptuelle à l'implémentation [A UML model and OCL constraints for spatial data warehouses. From conceptual design to implementation]. *Ingénierie des Systèmes d'Information (ISI)*, *16*, 11–39.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD Record, 26*, 65–74.

Golfarelli, M., Maio, D., & Rizzi, S. (1998). Conceptual design of data warehouses from E/R schemes. In *Proceedings of the 31st annual Hawaii international conference on system sciences: HICSS '98* (Vol. 7, pp. 334–343). Washington, DC: IEEE Computer Society.

Gray, J., Bosworth, A., Layman, A., & Pirahesh, H. (1996). Data Cube: A relational aggregation operator generalizing Group-By, Cross-Tab, and Sub-Total. In S. Y. W. Su (Ed.), *Proceedings of the 12th international conference on data engineering (ICDE'96)* (pp. 152–159). Washington, DC: IEEE Computer Society.

Gyssens, M., & Lakshmanan, L. V. S. (1997). A foundation for multi-dimensional databases. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos & M. A. Jeusfeld (Eds.), *VLDB'97, Proceedings of the 23rd international conference on very large data bases* (pp. 106–115). San Francisco, CA: Morgan Kaufman.

Harinath, S., Zare, R., Meenakshisundaram, S., Carroll, M., & Guang-Yeu Lee, D. (2009). *Professional Microsoft SQL server analysis services 2008 with MDX*. Indianapolis, IN: Wiley.

Hassan, A., Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2012). Differentiated multiple aggregations in multidimensional databases. In A. Cuzzocrea & U. Dayal (Eds.), *Proceedings of the 14th international conference on data warehousing and knowledge discovery (DAWAK'12), LNCS Vol. 7448* (pp. 93–104). Berlin: Springer.

Kimball, R. (1996). *The data warehouse toolkit: Practical techniques for building dimensional data warehouses*. New York, NY: John Wiley & Sons.

Mazón, J. N., Lechtenbörger, J., & Trujillo, J. (2009). A survey on summarizability issues in multidimensional modelling. *Data & Knowledge Engineering, 68*, 1452–1469.

Lujàn-Mora, S., Trujillo, J., & Song, I. Y. (2005). A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering, 59*, 725–769.

Oliveira, R., Rodrigues, F., Martins, P., & Moura, J. P. (2011). Extending the dimensional templates approach to integrate complex multidimensional design concepts. In A. Cuzzocrea & U. Dayal (Eds.), *Proceedings of the 13th international conference on data warehousing and knowledge discovery (DaWaK 2011)* (pp. 26–38). Berlin: Springer.

Pedersen, T., Jensen, C., & Dyreson, C. (2011). A foundation for capturing and querying complex multidimensional data. *Information Systems, 26*, 383–423.

Prat, N., Wattiau, I., & Akoka, J. (2010). Representation of aggregation knowledge in OLAP systems. In P. M. Alexander, M. Turpin, & J. P. van Deventer (Eds.), *Proceedings of the 18th European conference on infomation systems (ECIS 2010)*. Pretoria, South Africa.

Ravat, F., Teste, O., & Zurfluh, G. (2002). Langage pour bases multidimensionnelles: OLAP-SQL [Language for multidimensional databases]. *Ingénierie des Systèmes d'Information (ISI)*, *7*, 11–38.

Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2007). Graphical querying of multidimensional databases. In Y. E. Ioannidis, B. Novikov, & B. Rachev (Eds.), *Proceedings of the 11th East-European conference on advances in databases and information systems (ADBIS'07), LNCS Vol. 4690* (pp. 298–313). Berlin: Springer.

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

Torlone, R. (2003). Conceptual multidimensional models. In M. Rafanelli (Ed.), *Multidimensional databases: Problems and solutions* (pp. 69–90). Hershey, PA: IGI Global.

Vassiliadis, P., & Sellis, T. K. (1999). A survey of logical models for OLAP databases. *SIGMOD Record, 28*, 64–69.

Vassiliadis, P., & Skiadopoulos, S. (2000). Modelling and optimisation issues for multidimensional databases. In B. Wangler & L. Bergman (Eds.), *Proceedings of the 12th international conference on advanced information systems engineering (CAiSE 2000), LNCS Vol. 1789* (pp. 482–497). Berlin: Springer.