Differentiated Multiple Aggregations in 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), 118 Route de Narbonne, 31062 Toulouse cedex 9, France
{hassan, ravat, tournier, zurfluh}@irit.fr
[2] Université Toulouse 3 Paul Sabatier, IRIT (UMR 5505), 118 Route de Narbonne, 31062 Toulouse cedex 9, France
teste@irit.fr

Abstract. Many models have been proposed for modeling multidimensional data warehouses and most consider a same function to determine how measure values are aggregated according to different data detail levels. We provide a conceptual model that supports (1) multiple aggregations, associating to the same measure a different aggregation function according to analysis axes or hierarchies, and (2) differentiated aggregation, allowing specific aggregations at each detail level. Our model is based on a graphical formalism that allows controlling the validity of aggregation functions (distributive, algebraic or holistic). We also show how conceptual modeling can be used, in an R-OLAP environment, for building lattices of pre-computed aggregates.

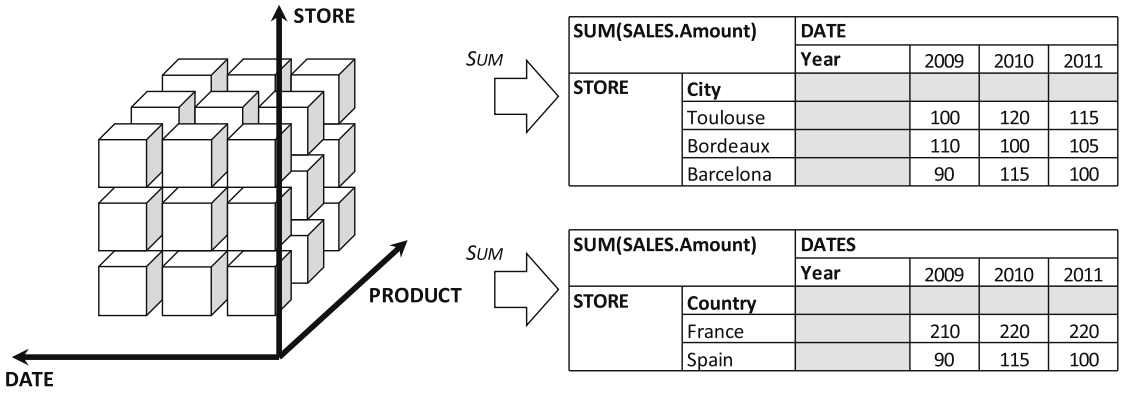
Keywords: Data warehouse · Conceptual modeling · Aggregate lattice · Multiple aggregations · Aggregation functions

1 Introduction

Decision support systems, such as data warehouses, have shown their ability to integrate large volumes of data by supporting effectively the analysis of stored data. These decision support systems are elaborated from data sources, usually the operational system of an organization; the data identified in the relevant sources are extracted, transformed and loaded [26] in a storage area called a data warehouse. To allow efficient querying and analysis of the data, specific data organization techniques have been developed using multidimensional databases (MDB) [3, 13]. This type of modeling considers the analyzed data from analysis indicators (i.e. measures grouped into facts) as points in a multidimensional space, forming a data cube [8]. Each dimension having various granularity/detail levels. Decision makers visualize extracts of data cubes, usually a two-dimensional slice of a cube. From this structure, called a multidimensional table (MT) [9], the decision maker can interact through manipulation operations [22]. The most emblematic operations are drilling operations which change the granularity level of the analyzed data and rotation operations which change the slice

of the cube. These operations are the most popular ones used for On-Line Analytic Processing (OLAP).

This environment provides a suitable analysis framework for decision makers; however, the imposed structure may be imperfect. In particular, a classical MDB supports only the calculation of a measure made by the same aggregation function while performing drilling or rotating operations (i.e. changing the analyzed slice of the cube). For example, if we consider sales amounts, these can be calculated as the sum of the products sold by cities and years (top part of Fig. 1). When drilling from cities to countries, the new amounts are calculated using the same aggregation function (SUM in the bottom part of Fig. 1). When the user wishes to change the aggregation function between two slices of the manipulated cube, the classical BDM no longer guarantees the validity of the calculated data, or even worse: does not support this type of manipulation.



SUM(SALES.Amount)		DATE			
		Year	2009	2010	2011
STORE	City				
	Toulouse		100	120	115
	Bordeaux		110	100	105
	Barcelona		90	115	100

SUM(SALES.Amount)		DATES			
		Year	2009	2010	2011
STORE	Country				
	France		210	220	220
	Spain		90	115	100

This paper aims at allowing non-uniform aggregations during user manipulations. To ensure the validity of such aggregations, we define **differentiated multiple aggregations**. Our proposal aims at developing a multidimensional model flexible enough for designing cubes with aggregation functions according to different levels.

1.1 Case Study

The case study concerns a diploma delivery jury. Here, decision makers (jury members) deliver diplomas by analyzing the marks (average, maximum, minimum) of students and their rate of absenteeism.

Students are split into groups and the academic year has two semesters. Each semester consists of Teaching Units (TU) and each TU is composed of several courses. Each course is associated with a coefficient that represents the importance of the course in the TU. We must take into account this coefficient to calculate the mark of the EU, which itself is linked to an ECTS (European Credit Transfer System) used to calculate the mark of semester. Each semester has the same amount of ECTS. In addition to the courses and students, analysts can analyze marks and absenteeism rates according to the dates (academic years).

Analysts may wish to observe absenteeism in two different ways:

- The first, called **simple**, is to calculate the percentage of absenteeism without distinction between different courses or TUs.
- The second, called **weighted**, uses the same coefficients (used for calculating the marks of TUs and semesters) to calculate absenteeism rates.

An MDB is implemented using extracting, transforming processes and loading data from the operational system, which we will not detail in this article. Figure 2 shows the conceptual star schema [7, 21] of the MDB of our case study. This MDB analyzes the measures (average marks 'Avg_Mark', maximum marks 'Max_Mark', minimum marks 'Min_Mark' and absenteeism rates 'Rate_Abs') by 'Courses', 'Students' and 'Dates' (dimensions).

The dimension 'Courses' has two hierarchies 'HCourse_Simple' and 'HCourse_weighted'. Each hierarchy corresponds to a way to analyze the absenteeism rate (simple and weighted). The other measures ('Avg_Mark', 'Max_Mark' and 'Min_Mark') are analyzed in the same way on the two hierarchies. A course is characterized by a course number (C_Id), a teaching unit number (TU_Id) and a semester. Each student has a student number (S_Id) and a group number (G_Id). Academic years 'Academic_year' of the dimension 'Dates' are aggregated by periods of five years 'period-5' and periods of ten years 'period-10'.

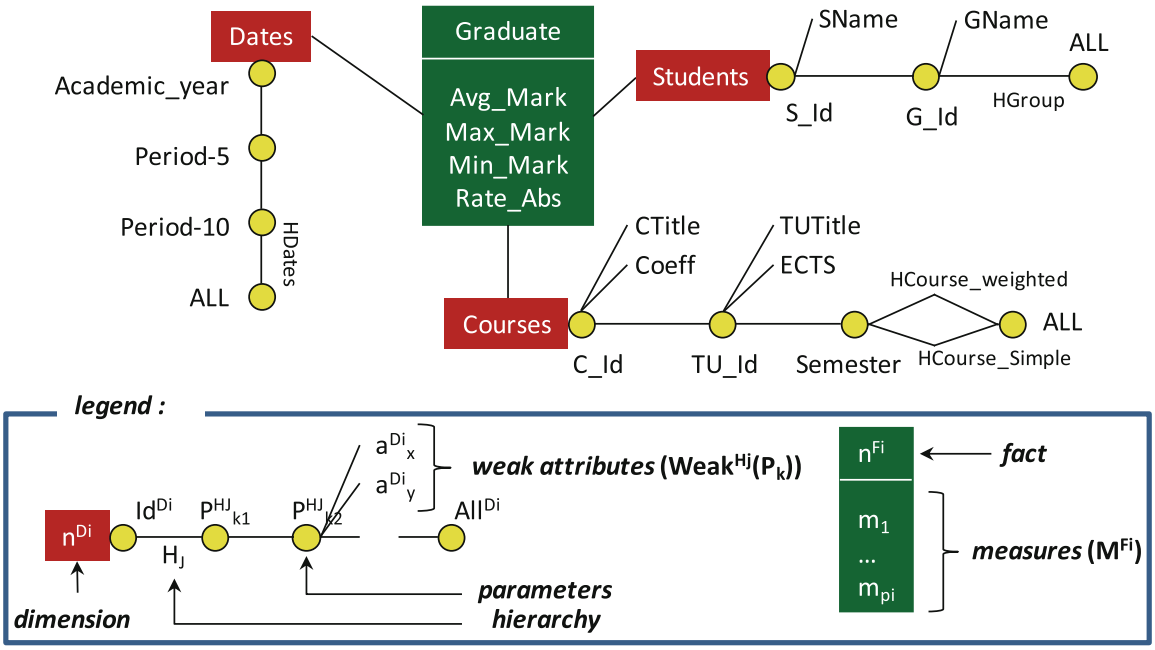


Fig. 2. The MDB of the diploma delivery case study

1.2 Illustration of the Problem

This schema allows getting average marks by courses and by students (Fig. 3). Obtaining the average mark by TU in this multidimensional environment requires aggregating the average marks by courses in accordance with the function associated

AVG(Graduate.Mark)		Courses			
		Semester	S1		
		TU_Id	U1		U2
		C_Id	M1	M2	M3
Students	S_Id (SName)				
	St1 (Tom)		14	10	12
	St2 (Sara)		8	10	9

Fig. 3. A MT visualizing the student's average marks by course

with the measure Mark (AVG). But this operation gives a result that does not correspond to examination modalities: an average mark by TU should be calculated from the course marks and taking into account the coefficient of each course (Eq. 1). Similarly, for average marks by semester, the ECTS of each TU (Eq. 2) has to be taken into account. However, to calculate the general average mark for each student, one must calculate the average of the TU marks (Eq. 3).

$$AVG\_TU = \frac{\sum Mark \times Coeff}{\sum Coeff} \tag{1}$$

$$AVG\_Semester = \frac{\sum AVG\_TU \times ECTS}{\sum ECTS} = \frac{\sum \left(\frac{\sum Mark \times Coeff}{\sum Coeff} \right) \times ECTS}{\sum ECTS} \tag{2}$$

$$AVG\_ALL = AVG(AVG\_Semester) \tag{3}$$

Therefore, classical approaches that consider a single aggregation function for all modeled aggregation levels in the star schema suffer from several limits:

- **Variability of the aggregation function.** Traditionally, models do not allow the use of aggregation functions that vary along dimensions or hierarchical levels. In our example, the aggregation function changes between the levels C_Id (courses), TU_Id (teaching units) and the semester level.
- **Shortcomings of basic functions.** When aggregating data across hierarchical levels, in our example, we use non-standard aggregation functions which use complementary data other than measure values (i.e. coefficients *Coeff*, weights *ECTS*).
- **Aggregation constraints.** The way to make the calculation of aggregation functions may be constrained. In our example, as shown in (Eq. 2), the average per semester cannot be obtained directly from the marks per courses. It is necessarily calculated from the averages per TUs. Similarly, the general average is calculated from the averages per semesters (Eq. 3).

The objective of this paper is to propose a multidimensional model sufficiently expressive to support these various aggregations. Then, we study the impacts of this conceptual model on the lattice of pre-aggregates [8] at the logical level.

In previous work [11], we detailed our conceptual model and presented simply the logical model. Here, we:

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

The rest of this paper is organized as follows: Sect. 2 reviews related work. Section 3 defines our conceptual multidimensional model: classical concepts, extensions for differentiated multiple aggregations and associated graphical formalisms. Section 4 shows the logical R-OLAP model of our star schema and its optimization relations. We detail our prototype and experiments in Sect. 5 and the last section concludes this work and states some future work.

2 Related Work

There are typically two approaches for modeling multidimensional databases. The first is based on the data cube (or hypercube) metaphor according to which the MDB is represented by cubes. The second is known as multidimensional modeling, where the MDB is described by a star schema or a constellation [13]. Our work falls in the second category. A cube is based on an equivocal separation between the structure elements and the values [24]: modeling analysis axes is not very expressive especially due to the difficulty for representing the hierarchical organization of the data. It is also limited for representing constellations of facts with shared dimensions.

Several surveys of the domain [3, 17, 25] and comparative studies [1, 2, 7, 9, 14, 16, 18–23, 27] are available in the scientific literature. One, [17], deals 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 during an analysis.

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

The treatment of aggregation of measures in the multidimensional space has evolved (Table 1). Two contributions [9, 27] do not specify aggregation functions at the measure level; however, they leave the possibility to use several aggregation functions for each measure during OLAP analyses. This provides great flexibility, but allows the user to do errors by using inappropriate aggregation functions. In addition, one advantage of specifying the aggregation functions in the conceptual model is to use them for the cube computation, i.e. for the pre-computation of the aggregates. In [19, 21, 23], the authors, in theirs conceptual models, can link to a single measure a set of functions which includes only valid functions. However, in these three papers, the same function will be used with all the dimensions and all aggregation levels.

In [4, 6], the authors assume that the aggregation function is determined for a measure in the analysis queries. This function can change from one query to another one while concerning the same measure. But in each query, the aggregation function

will be used uniformly over all the dimensions involved in the analysis. In [5], although the authors store multiple aggregations data in a hierarchical organization according to the time granularities, they use the same function for all granularities.

The YAM^2 model [1] and the work presented in [7] support a different aggregation function with each dimension. However, these models do not support function change neither between hierarchies nor within the hierarchical levels. This limit has been lifted by the aggregation model of [20] which 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 [2] the authors overcome this limit. However, these last two papers [2, 20] 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 that a "custom rollup" can be applied in a hierarchy in several ways [10]:

- By using unary operators to solve the aggregation problem over a particular type of hierarchy (parent-child attributes hierarchy). These hierarchies are built from a single attribute with a reflexive join relationship on the attribute itself (i.e. technically a join on the dimension table itself).
- By using MDX scripts, either directly or by using the attribute property "CustomRollupColumn" which indicates a column where MDX scripts are stored.

These two ways concern aggregation functions but it is not related to a specific dimension or an aggregation level. It is 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 poses a storage problem and reduces performance [10]. Moreover, binding a "custom rollup" with a specific instance can cause difficulties when updating data.

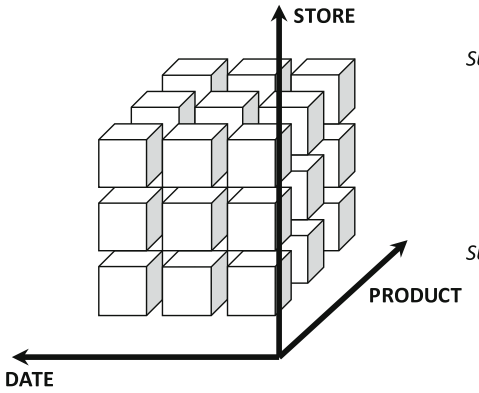
The MDX language allows the possibility for building data sets (that will be aggregated by aggregation functions) using functions: PeriodsToDate, YTD, QTD, MTD, Crossjoin, Cousin, Descendants, Children, Hierarchize, and Members. However, this possibility is not related to our problem: changing the aggregation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

The above was about how the integration of aggregation functions within the multidimensional model. But, there is another point that should be taken into consideration; it is the aggregation functions itself. Aggregation functions are classified:

- From an aggregation mechanism point of view, aggregation functions belong to three different categories [8]: The first corresponds to **distributive** functions that calculate aggregated values of the selected granularity level from the values already aggregated at the lower level (e.g. yearly amounts can be calculated by summing monthly values). The second corresponds to **algebraic** functions that calculate aggregated values from stored intermediate results (for example, the average of an amount per year can be calculated from the sum of the amounts and counting the occurrences from a month level). Finally, the third corresponds to **holistic** functions

of the cube. These operations are the most popular ones used for On-Line Analytic Processing (OLAP).

This environment provides a suitable analysis framework for decision makers; however, the imposed structure may be imperfect. In particular, a classical MDB supports only the calculation of a measure made by the same aggregation function while performing drilling or rotating operations (i.e. changing the analyzed slice of the cube). For example, if we consider sales amounts, these can be calculated as the sum of the products sold by cities and years (top part of Fig. 1). When drilling from cities to countries, the new amounts are calculated using the same aggregation function (SUM in the bottom part of Fig. 1). When the user wishes to change the aggregation function between two slices of the manipulated cube, the classical BDM no longer guarantees the validity of the calculated data, or even worse: does not support this type of manipulation.



SUM

SUM(SALES.Amount)		DATE			
		Year	2009	2010	2011
STORE	City				
	Toulouse		100	120	115
	Bordeaux		110	100	105
	Barcelona		90	115	100

SUM

SUM(SALES.Amount)		DATES			
		Year	2009	2010	2011
STORE	Country				
	France		210	220	220
	Spain		90	115	100

Fig. 1. Uniform aggregation in slices of a cube

This paper aims at allowing non-uniform aggregations during user manipulations. To ensure the validity of such aggregations, we define **differentiated multiple aggregations**. Our proposal aims at developing a multidimensional model flexible enough for designing cubes with aggregation functions according to different levels.

1.1 Case Study

The case study concerns a diploma delivery jury. Here, decision makers (jury members) deliver diplomas by analyzing the marks (average, maximum, minimum) of students and their rate of absenteeism.

Students are split into groups and the academic year has two semesters. Each semester consists of Teaching Units (TU) and each TU is composed of several courses. Each course is associated with a coefficient that represents the importance of the course in the TU. We must take into account this coefficient to calculate the mark of the EU, which itself is linked to an ECTS (European Credit Transfer System) used to calculate the mark of semester. Each semester has the same amount of ECTS. In addition to the courses and students, analysts can analyze marks and absenteeism rates according to the dates (academic years).

Differentiated Multiple Aggregations in 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), 118 Route de Narbonne, 31062 Toulouse cedex 9, France
{hassan, ravat, tournier, zurfluh}@irit.fr
[2] Université Toulouse 3 Paul Sabatier, IRIT (UMR 5505), 118 Route de Narbonne, 31062 Toulouse cedex 9, France
teste@irit.fr

Abstract. Many models have been proposed for modeling multidimensional data warehouses and most consider a same function to determine how measure values are aggregated according to different data detail levels. We provide a conceptual model that supports (1) multiple aggregations, associating to the same measure a different aggregation function according to analysis axes or hierarchies, and (2) differentiated aggregation, allowing specific aggregations at each detail level. Our model is based on a graphical formalism that allows controlling the validity of aggregation functions (distributive, algebraic or holistic). We also show how conceptual modeling can be used, in an R-OLAP environment, for building lattices of pre-computed aggregates.

Keywords: Data warehouse · Conceptual modeling · Aggregate lattice · Multiple aggregations · Aggregation functions

1 Introduction

Decision support systems, such as data warehouses, have shown their ability to integrate large volumes of data by supporting effectively the analysis of stored data. These decision support systems are elaborated from data sources, usually the operational system of an organization; the data identified in the relevant sources are extracted, transformed and loaded [26] in a storage area called a data warehouse. To allow efficient querying and analysis of the data, specific data organization techniques have been developed using multidimensional databases (MDB) [3, 13]. This type of modeling considers the analyzed data from analysis indicators (i.e. measures grouped into facts) as points in a multidimensional space, forming a data cube [8]. Each dimension having various granularity/detail levels. Decision makers visualize extracts of data cubes, usually a two-dimensional slice of a cube. From this structure, called a multidimensional table (MT) [9], the decision maker can interact through manipulation operations [22]. The most emblematic operations are drilling operations which change the granularity level of the analyzed data and rotation operations which change the slice

We define the attribute set and the hierarchy set respectively as

$$A = \cup_{i=1}^{m} A^{D_i} \text{ and } H = \cup_{i=1}^{m} H^{D_i}$$

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

- $\mathrm{n^{Hj}} \in \mathrm{N}$ is the name that identifies the hierarchy,
- $\mathrm{P^{Hj}} = \left\{p_1^{H_j}, \ldots, p_{q_j}^{H_j}\right\}$ is a set of attributes called *parameters*, $\mathrm{P^{Hj}} \subseteq \mathrm{A^{Di}}$,
- $\prec^{\mathrm{Hj}} = \left\{\left(p_x^{H_j}, p_y^{H_j}\right) | p_x^{H_j} \in \mathrm{P^{H_j}} \wedge p_y^{H_j} \in \mathrm{P^{H_j}}\right\}$ is an antisymmetric and transitive binary relation between parameters. Remember that the antisymmetry means that $\left(p_{k_1}^{H_j} \prec^{H_j} p_{k2}^{H_j}\right) \wedge \left(p_{k_2}^{H_j} \prec^{H_j} p_{k1}^{H_j}\right) \Rightarrow p_{k1}^{H_j} p = p_{k2}^{H_j}$ while the transitivity means that $\left(p_{k_1}^{H_j} \prec^{H_j} p_{k2}^{H_j}\right) \wedge \left(p_{k_2}^{H_j} \prec^{H_j} p_{k3}^{H_j}\right) \Rightarrow p_{k1}^{H_j} \prec^{H_j} p_{k3}^{H_j}$.
- $\mathrm{Weak^{Hj}} : \mathrm{P^{Hj}} \rightarrow 2^{A^{D_i} \setminus P^{H_j}}$ is an application that associates to each parameter a set of dimension attributes, called *weak attributes* (2^{N} represents the power set of N).

We define parameter sets as

$$P^{D_i} = \cup_{j=1}^{s_i} P^{H_j} \text{ and } P = \cup_{i=1}^{m} P^{D_i} = \cup_{i=1}^{m} \cup_{j=1}^{s_i} P^{H_j}$$

Lemma 1. For each dimension $\mathrm{D_i}$, a *root parameter*, denoted $\mathrm{Id^{Di}} \in \mathrm{P^{Di}}$, exists. It is defined as follows: $\forall \mathrm{j} \in [1..\mathrm{s_i}], \forall p_k^{H_j} \in \mathrm{p^{D_i}}, \mathrm{Id^{D_i}} \neq p_k^{H_j} | \mathrm{Id^{D_i}} \prec^{H_j} p_k^{H_j}$.

Lemma 2. For each dimension $\mathrm{D_i}$, a *extremity parameter*, denoted $\mathrm{All^{Di}} \in \mathrm{P^{Di}}$, exists. It is defined as follows: $\forall \mathrm{j} \in [1..\mathrm{s_i}], \forall p_k^{H_j} \in \mathrm{p^{D_i}}, \mathrm{A11^{D_i}} \neq p_k^{H_j} | p_k^{H_j} \prec^{H_j} \mathrm{A11^{D_i}}$.

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

3.2 Extensions for Differentiated Multiple Aggregations

We enrich the multidimensional model for specifying how the aggregations calculations are performed during OLAP analysis. This corresponds to three extensions:

- The first extension concerns the aggregation process which allows using several aggregation functions for the same measure:
  - **Differentiated aggregation**. It consists in aggregating measure values between two parameters (aggregation levels) of a hierarchy. The aggregation function is associated with one measure and one parameter. This kind of aggregation allows a specific aggregation over each level of granularity.
  - **Multiple hierarchical aggregation**. It is used to aggregate the measure values between all the parameters over a hierarchy. This is a simplified representation

instead of a repeated use of the same differentiated function over several levels of granularity. It is important to note that several aggregation functions can be associated to a same measure; one for each hierarchy.
    - **Multiple dimensional aggregation**. It consists in aggregating measure values using different aggregation functions depending on the used dimension. Similarly to multiple hierarchical aggregation, multiple dimensional aggregation is a simplified representation instead of a repeated use of the same multiple hierarchical aggregation over several hierarchies. The same aggregation is performed over each level of granularity of a dimension. The function is associated with one measure and a dimension.
    - **General aggregation**. This function is associated only with a measure without taking into account neither parameter nor hierarchy nor dimension. This is a simplified representation instead of a repeated use of the same multiple dimensional function over several dimensions. This is equivalent to aggregation functions in classical models.
- The second extension concerns the **execution order** for handling the case of non commutative aggregation functions. It is possible to have different aggregation functions during an analysis. These functions are generally not commutative. Therefore, it is necessary to plan in the MDB an execution order when using the functions between the different dimensions.
- The third extension concerns **aggregation constraints** which aim at handling the case where the measure cannot be calculated from the base level. All aggregations are not carried out uniformly using systematically all lower hierarchical levels (contrarily to the aggregation process designed in classical multidimensional models). Therefore, we introduce a constraint mechanism on the aggregation process to indicate the valid aggregation level that allows obtaining the upper level.

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

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

- $F = \{F_1, \ldots, F_n\}$ 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*,
- $D = \{D_1, \ldots, D_m\}$ 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.
- 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. Aggregate defines the different types of aggregation functions supported by our model:

    - If 2^D, 2^H and 2^P are not used ($2^D = \varnothing$, $2^H = \varnothing$ and $2^P = \varnothing$) then the function is a *general aggregation* function.
    - If 2^H and 2^P are not used ($2^D \neq \varnothing$, $2^H = \varnothing$ and $2^P = \varnothing$) then the function is a *multiple dimensional aggregation* function. Here, the function aggregates the measure over the entire considered dimension.

- If 2^{P} only is not used ($2^{D} \neq \varnothing$, $2^{H} \neq \varnothing$ and $2^{P} = \varnothing$) then the function is a *multiple hierarchical aggregation* function. Here, the function aggregates the measure over the entire considered hierarchy.
- If $2^{D} \neq \varnothing$, $2^{H} \neq \varnothing$ and $2^{P} \neq \varnothing$ then the function is a *differentiated aggregation* function. Here, the function aggregates the measure between a considered parameter and the parameter directly above it in the same hierarchy.

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

$\mathbb{N}^-$ is to constraint 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.

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.

$$\begin{cases} \forall i \in [1..n], \forall m_k \in \mathrm{M}^{\mathrm{Fi}}, \exists f \in \mathcal{F}, \exists x_1 \in \mathbb{N}^*, \exists x_2 \in \mathbb{N}^-, \\ \quad \left| \left(x_1 f, \{\}, \{\}, \{\}, x_2\right) \in Aggregate(m_k) \right. \\ \forall D_j \in Star(F_i) \left| \left(x_1 f, \{D_j\}, \{\}, \{\}, x_2\right) \in Aggregate(m_k) \right. \\ \forall H_s \in H^j \left| \left(x_1 f, \{D_j\}, \{H_s\}, \{\}, x_2\right) \in Aggregate(m_k) \right. \\ \forall P_q \in P^s \backslash \{All^j\} \left| \left(x_1 f, \{D_j\}, \{H_s\}, \{P_q\}, x_2\right) \in Aggregate(m_k) \right. \end{cases}$$

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 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 Formalisms

Textual Formalisms. The example of the diploma delivery illustrated in case study, is defined formally by (F, D, Star, Order, Aggregate) where:

- F = {$F_{Graduate}$}, where the fact is defined by $F_{Graduate}$ = (‘*Graduate*’, {Avg_Mark, Max_Mark, Min_Mark, Rate_Abs }).
- D = {$D_{Courses}$, $D_{Students}$, D_{Dates}}, where the dimensions are defined by:
  - $D_{Courses}$ = (‘*Courses*’, {a_{C_Id}, a_{Coeff}, a_{CTitle}, a_{TU_Id}, a_{ECTS}, $a_{TUTitle}$, $a_{Semester}$, $ALL^{DCourses}$}, {$H_{HCourse\_Simple}$, $H_{HCourse\_weighted}$}) with
    - $H_{HCourse\_Simple}$ = (‘*HCourse_Simple*’, {a_{C_Id}, a_{TU_Id}, $a_{Semester}$, $ALL^{DCourses}$}, {(a_{C_Id}, a_{TU_Id}), (a_{TU_Id}, $a_{Semester}$), ($a_{Semester}$, $ALL^{DCourses}$)}, {(a_{C_Id}, {a_{Coeff}, a_{CTitle}}), (a_{TU_Id}, {a_{ECTS}, $a_{TUTitle}$})}),
    - $H_{HCourse\_Simple}$ = (‘*HCourse_Simple*’, {a_{C_Id}, a_{TU_Id}, $a_{Semester}$, $ALL^{DCourses}$}, {(a_{C_Id}, a_{TU_Id}), (a_{TU_Id}, $a_{Semester}$), ($a_{Semester}$, $ALL^{DCourses}$)}, {(a_{C_Id}, {a_{Coeff}, a_{CTitle}}), (a_{TU_Id}, {a_{ECTS}, $a_{TUTitle}$})}).
  - $D_{Students}$ = (‘*Students*’, {a_{S_Id}, a_{SName}, a_{G_Id}, a_{GName}, $ALL^{DStudents}$}, {H_{HGroup}}) with
    - H_{HGroup} = (‘*HGroup*’, {a_{S_Id}, a_{G_Id}, $ALL^{DStudents}$}, {(a_{S_Id}, a_{G_Id}), (a_{G_Id}, $ALL^{DStudents}$)}, {(a_{S_Id}, {a_{SName}}), (a_{G_Id}, {a_{GName} })}).
  - D_{Dates} = (‘*Dates*’, {$a_{Academic\_year}$, $a_{Period\text{-}5}$, $a_{Period\text{-}10}$, ALL^{DDates}}, {H_{HDates}}) with
    - H_{HDates} = (‘*HDates*’, {$a_{Academic\_year}$, $a_{Period\text{-}5}$, $a_{Period\text{-}10}$, ALL^{DDates}}, {($a_{Academic\_year}$, $a_{Period\text{-}5}$), ($a_{Period\text{-}5}$, $a_{Period\text{-}10}$), ($a_{Period\text{-}10}$, ALL^{DDates})}).
- Star : $F \rightarrow 2^{D}$ |
  Star($F_{Graduate}$) = {$D_{Courses}$, $D_{Students}$, D_{Dates} }
- Aggregate : $M \rightarrow 2^{N* \times \mathcal{F} \times 2^{D} \times 2^{H} \times 2^{P} \times N^{-}}$ |

Aggregate (Avg_Mark) = {(2, AVG(Avg_Mark), {}, {}, {}, 0),[1]
(1, AVG_W(Avg_Mark, Coeff), {Courses}, {HCourse_weighted}, {C_Id}, 0),
(1, AVG_W(Avg_Mark, ECTS), {Courses}, {HCourse_weighted}, {TU_Id},-1),[2]
(1, AVG(Avg_Mark), {Courses}, {HCourse_weighted}, {Semester}, -1),
(1, AVG_W(Avg_Mark, Coeff), {Courses}, {HCourse_Simple}, {C_Id}, 0),
(1, AVG_W(Avg_Mark, ECTS), {Courses}, {HCourse_Simple}, {TU_Id}, -1),
(1, AVG(Avg_Mark), {Courses}, {HCourse_Simple}, {Semester}, -1)}
Aggregate (Max_Mark) = {(2, MAX(Avg_Mark), {}, {}, {}, 0),
(1, AVG_W(Avg_Mark, Coeff), {Courses}, {HCourse_weighted}, {C_Id}, 0),

[1] Note that there is no constraint on the aggregation.

[2] The aggregated values are computed from the values at the level directly below the one considered.

(1, AVG_W(Avg_Mark, ECTS), {Courses}, {HCourse_weighted}, {TU_Id},-1),
(1, AVG(Avg_Mark), {Courses}, {HCourse_weighted}, {Semester}, -1),
(1, AVG_W(Avg_Mark, Coeff), {Courses}, {HCourse_Simple}, {C_Id}, 0),
(1, AVG_W(Avg_Mark, ECTS), {Courses}, {HCourse_Simple}, {TU_Id}, -1),
(1, AVG(Avg_Mark), {Courses}, {HCourse_Simple}, {Semester}, -1)}

Aggregate (Rate_Abs) = {(2, AVG(Rate_Abs), {}, {}, {}, 0),
(1, AVG_W(Rate_Abs, Coeff), {Courses}, {HCourse_weighted}, {C_Id}, 0),
(1, AVG_W(Rate_Abs, ECTS), {Courses}, {HCourse_weighted}, {TU_Id},-1),
(1, AVG(Rate_Abs), {Courses}, {HCourse_weighted}, {Semester}, -1)}

Aggregate(Min_Mark) is identical to Aggregate(Max_Note) except that it uses the MIN function instead of MAX.

The function Avg_W(X, Y) takes as input tow numerical parameters. It returns the average of values of X weighted by Y. In other words, the weighted average:

$$\text{Avg\_}W(\text{X}, \text{Y}) = \frac{\Sigma(\text{X} \times \text{Y})}{\Sigma \text{Y}}$$

Regarding the measures 'Avg_Mark', 'Max_Mark' and 'Min_Mark', it is aggregated in an identical way on the two hierarchies of the dimension 'Courses'. Moreover, the aggregation of the measures 'Max_Mark' and 'Min_Mark' is based on the aggregation of 'Avg_Mark'. This clearly appears through the use of the measure 'Avg_Mark' in aggregation functions of 'Max_Mark' and 'Min_Mark'. For the maximum mark 'Max_Mark' of a course or a teaching unit for a group of students, we must first calculate the mark 'Avg_Mark' of this course or TU for each student, and then we determine from the obtained marks, the maximum mark.

If we analyze for example the average marks 'Avg_Mark' using the dimensions 'Dates' and 'Students', the decisional system must use the general function 'AVG (Avg_Mark)' to aggregate the measure values because there is no other specific function for these dimensions. If we analyze using the dimension 'Courses', the system uses on each aggregation level a different differentiated aggregation function. Aggregation is done using the level directly below (AVG_W to aggregate the 'TU_Id' and 'Semester' levels using the 'C_Id' and 'TU_Id' levels respectively and AVG for 'ALL' level using the 'Semester' level). Furthermore, if we analyze data using two or more dimensions then functions over the dimension 'Courses' are a priority; that means that we must apply it before the other functions.

Graphical Formalisms. Associated with the formal definitions, we introduce a two-level graphical formalism for easing the understanding of the MDB schema:

- **Structural Schema.** The structural schema is used to display globally the multidimensional elements (facts, dimensions and hierarchies) hiding aggregation mechanisms. This global view (see Fig. 2) is defined by the function *Star*. The graphical formalism is based on [7, 22].
- **Aggregation schema.** For each measure $m_k \in M^{Fi}$, an aggregation schema is obtained using the function *Aggregate*. This schema details the aggregation

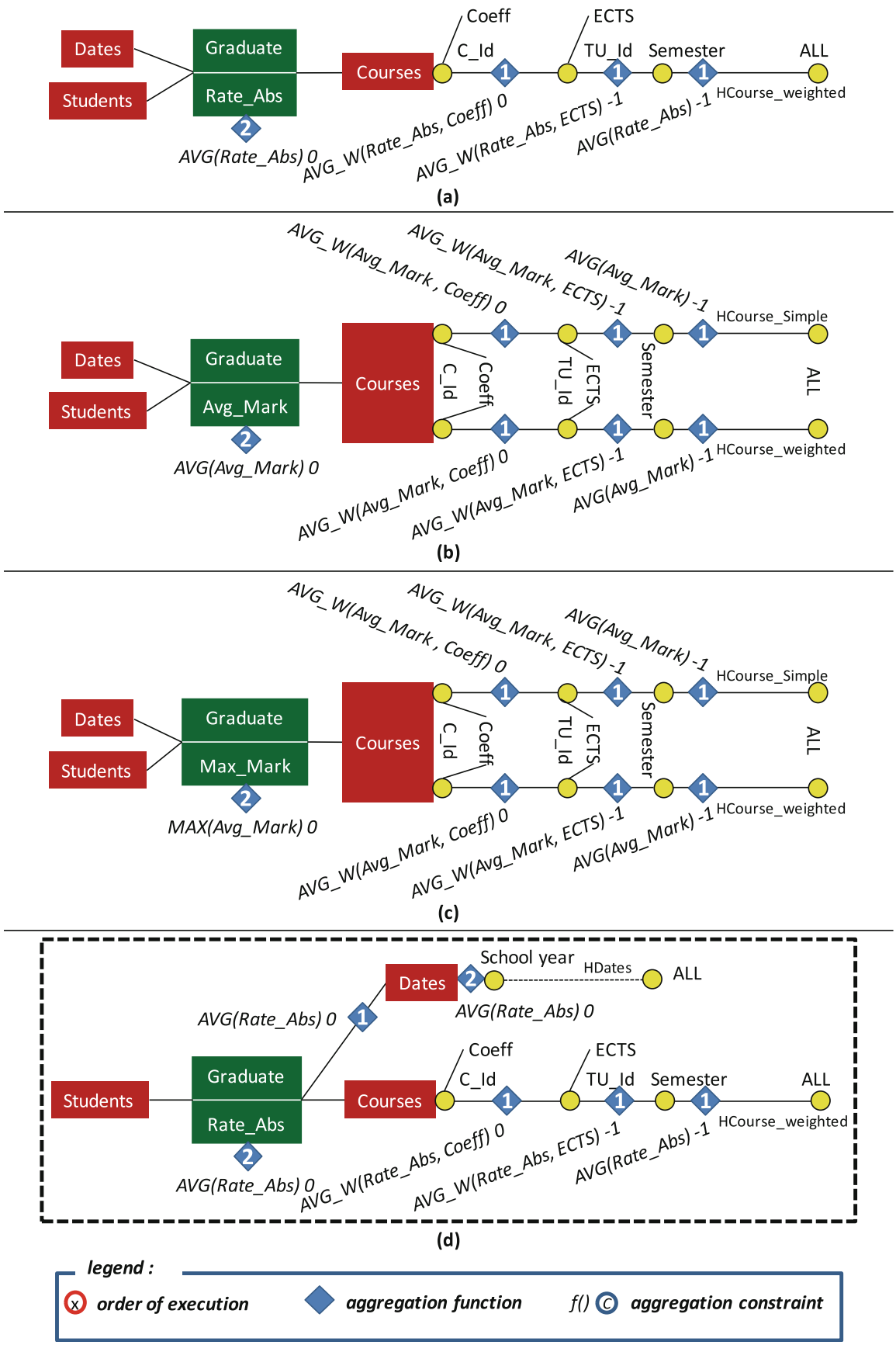


Fig. 4. Graphical notation extensions (Aggregation schemas)

mechanisms involved in the selected measure analysis (multiple, differentiated and general aggregations, constraints of aggregation and execution order) but shows simply the structural elements directly related to the measure. This schema is an extension of our previous work [11].

Figure 4 illustrates three aggregation schemes (a, b, c) corresponding to the measures 'Rate_Abs', 'Avg_Mark' and 'Max_Mark' (we do not present the measure 'Min_Mark'). As shown in Fig. 4, the hierarchies are presented in split version, unlike the structural schema (Fig. 2) where it is presented in compact version, e.g. hierarchies 'HCourse_weighted', 'HCourse_Simple' in Fig. 4 (b and c).

The aggregation functions are modeled by diamonds. Each diamond also indicates the execution order and the possible aggregation constraint. The positions of the diamonds depend on the type of function:

- A general function is represented by a diamond on the fact,
- A multiple dimensional function is on the edge connecting facts to dimensions,
- A multiple hierarchical function is represented on the bottom of the hierarchy,
- A differentiated aggregation function is a label on the edge linking two parameters.

Figure 4 (d) presents multiple aggregation (dimensional and hierarchical) functions and commutativity in the execution order. We assume that there is a multiple dimensional function AVG(Rate_Abs) on the dimension 'Dates'. This function is commutative with the functions of the dimension 'Courses'. We assume also that there is a multiple hierarchical function on the hierarchy 'HDates'. This function is commutative with the general function.

Aggregation with a constraint assigned to -1 is calculated from the directly lower level. E.g. the average mark 'Avg_Mark' by semester is calculated from average marks by UEs. In case, we would have chosen to calculate this average by semester from the marks by courses, the constraint would be assigned to -2.

4 Relational-OLAP (R-OLAP) Logical Model

Current multidimensional schema implementations use mainly the relational approach R-OLAP [13]. This approach has many advantages such as reusing proven data management mechanisms and the ability to manage very large volumes of data.

4.1 R-OLAP Star

In this relational context, the conceptual multidimensional structures (facts and dimensions) are translated into relations [13]. Applied to our example, the R-OLAP schema is the following:

COURSES (**C_Id**, Coeff,CTitle,TU_Id,ECTS,TUTitle,Semester)
STUDENTS (**S_Id**, SName, G_Id, GName)
DATES (**Academic_year**, Period-5, Period-10)
GRADUATE (**C_Id#,S_Id#,Academic_year#**, Avg_Mark, Rate_Abs)

According to aggregation functions for the maximum and minimum marks 'Max_Mark', 'Min_Mark' (cf. Fig. 4); these measures are calculated from the average mark measure 'Avg_Mark'. So their values can be obtained directly from those of the measure 'Avg_Mark' without storing them in the relational table corresponding to the fact of model.

The aggregation functions are stored in the database. We use a meta-schema (not detailed here, for more information see [12]) to describe the multidimensional schema (facts, dimensions and hierarchies) corresponding to the R-OLAP relations that store the analysis data. It also describes the different aggregation functions and the possible aggregation constraints.

4.2 Optimized Star

Conceptual modeling allows structuring hierarchically the analysis axis (dimension) graduations (parameters). These hierarchies are exploited for pre-computing the aggregations required by decision makers to navigate and to perform analyses in the multidimensional space (using OLAP). Traditionally, these pre-aggregations are modeled by a *lattice* of pre-computed aggregates [3, 8] where:

- each node represents a pre-computed aggregate and
- each edge represents a path for computing aggregates. If the aggregation function used is *distributive* or *algebraic*, the aggregate can be calculated from the directly lower aggregate, while if it is *holistic*, the calculus is from the base relation [8].

To avoid that the lattice is too complex, we simplify the example of the diploma delivery jury. We take into account only two dimensions:

- 'Courses' with two hierarchies 'HCourse_Simple' and 'HCourse_weighted'
- 'Students' with one hierarchy 'HGroup'.

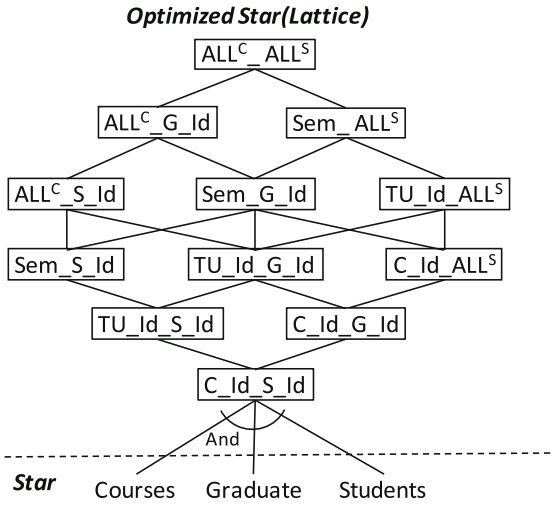


Fig. 5. Classical optimization lattice (We use abbreviations ('Sem' for 'Semester', 'ALL^{C}' for '$ALL^{DCourses}$', 'ALL^{S}' for '$ALL^{DStudents}$'))

Figure 5 shows the lattice of pre-aggregates of the measure 'Rate_Abs'. Each node represents a relation. E.g. the nodes 'TU_Id_S_Id' and 'C_Id_ALL^S' correspond to the following respective relations:

```
TU_Id_S_Id (TU_Id, S_Id, Rate_Abs, Abs_sum, Abs_count)
C_Id_ALL^S (C_Id, Rate_Abs, Abs_sum, Abs_count)
```

In these relations, the attribute 'Rate_Abs' represents the absenteeism rate calculated by the aggregation function AVG. Here, it is a case of algebraic function, so we store intermediate values (the sum 'Abs_sum' and the count 'Abs_count' of occurrences of the absenteeism rate) that will be used to calculate the upper nodes. In the classical approach, contrarily to our proposition, a unique aggregation function is used in the whole lattice for the measure 'Rate_Abs'.

4.3 Extending the Approach with Multiple and Differentiated Aggregations

The flexibility introduced in the conceptual model impacts the lattice.

Increasing the Number of Nodes. In our model, by using the multiple hierarchical and/or differentiated aggregation functions, we can associate the same parameter in different hierarchies with different aggregation functions. This gives different results for the same analysis depending on the used hierarchy. Thus, new nodes compatible with results of all these possible aggregations will be produced in the lattice (Fig. 6). For example, the absenteeism rate 'Rate_Abs' of a TU by groups of students can be calculated by the average function 'AVG(Rate_Abs)' over the hierarchy 'HCourse_Simple' or by the weighted average 'AVG_W(Rate_Abs, Coeff)' on the hierarchy 'HCourse_weighted' (see Fig. 4 (a)); of course each function gives different results.

The number of nodes in the classical lattice (Fig. 5) is calculated by multiplying the number of parameters in each dimension:

$$\text{number of nodes } = \prod_{i=1}^{m} \left| P^{D_i} \right|$$

In our model, assuming that each parameter has its own aggregation function, the number of nodes in the lattice (Fig. 6) is calculated by multiplying the sum in each dimension of number of parameters in each hierarchy; here, we must be careful for not count the root parameter of a dimension several times with the different hierarchies:

$$\text{number of nodes} = \prod_{i=1}^{m} \left(\sum_{j=1}^{S_j} \left(\left| P^{H_j} \right| - 1 \right) + 1 \right)$$

Edge Types. The differentiated and multiple aggregation functions involve using different aggregation computations for each edge of the lattice (Fig. 6), contrary to the traditional approach which usually considers only a single aggregation function.

When multiple paths are possible, the less costly path is preferred. The cost function (not detailed here) favors the most effective computation time [15]. However, the use of different aggregation functions on each edge of the lattice makes the cost estimate more complex than in usual lattices.

The possibility of use different aggregation functions for a same measure requires differentiating lattice edges. This typing to indicate the corresponding aggregation function between two nodes. For example, Fig. 4 (a) presents the aggregation schema of the absenteeism rate. Three aggregation functions are used to calculate the absenteeism rate 'Rate_Abs'. For each teaching unit and semester over the hierarchy 'HCourse_weighted', the absenteeism rate takes into account the courses coefficients and teaching units ECTS and uses a weighted function (AVG_W). Thus, in the lattice, it is necessary to distinguish the edges between ('C_Id' and 'TU_Id') and between ('TU_Id' and 'Semester') parameters over the hierarchy 'HCourse_weighted' (that use AVG_W) from the other edges (that use AVG).

In Fig. 6, simple lines correspond to the AVG function and double or triple lines are for AVG_W functions.

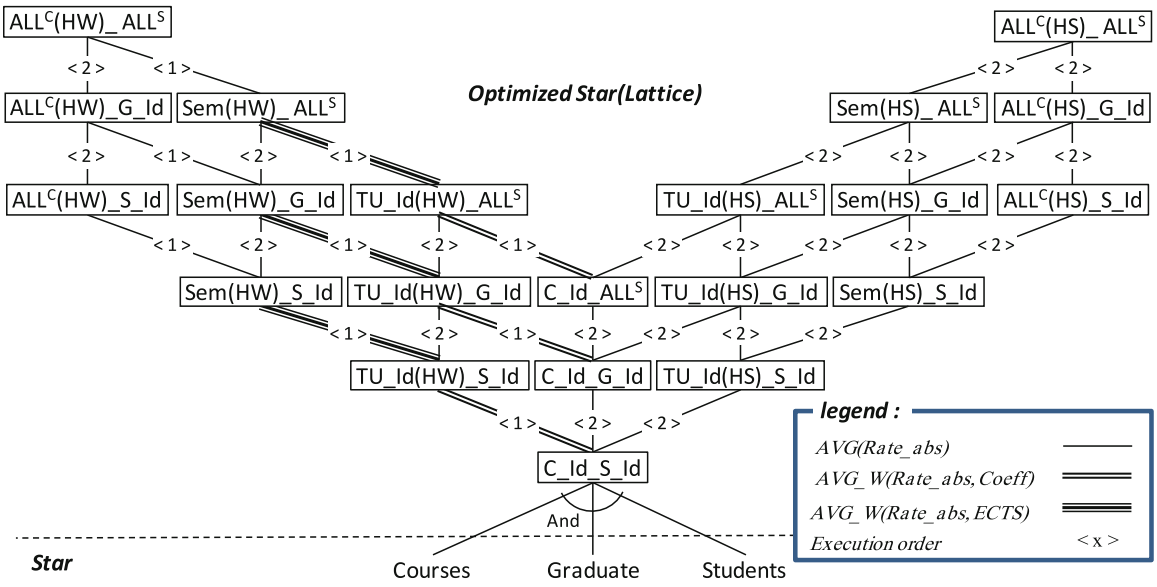


Fig. 6. Lattice with typed edges (Execution orders (< x >) were added to facilitate the understanding of the next impacts (blocking transitivity and pruning the lattice). And we use abbreviations ('HW' for the hierarchy 'HCourse_weighted', 'HS' for the hierarchy 'HCourse_Simple'))

Blocking Transitivity. Constraints (the specific level from which the considered aggregation must be calculated) associated with the aggregation functions have repercussions on the lattice. Edges with a symbol (crosses in a circle in Fig. 7) come from these constraints which require calculating the node from another specific node. It is then forbidden to calculate an upper node using transitivity from lower nodes as it would be in a classical schema. Thus the computing paths are blocked as soon as such an edge is encountered; e.g. the node 'Sem(HS)_S_Id' is calculable from the direct lower node 'TU_Id(HS)_S_Id'; using transitivity, it is also calculable from the lower

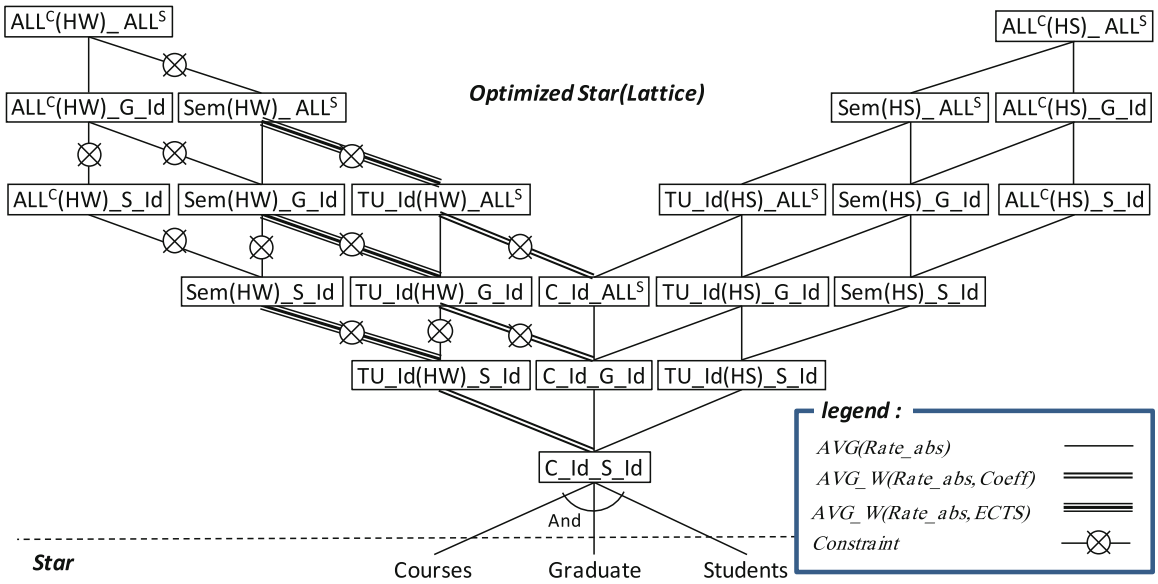


Fig. 7. Lattice with non transitive edges

node 'C_Id_S_Id'. However, the blocked edge resulting from the constraint of the function 'AVG_W(Rate_Abs, ECTS)' which operates on the edge ('TU_Id(HW)_S_Id', 'Sem(HW)_S_Id') blocks the calculation transitivity. Therefore, the node 'Sem(HW)_S_Id' is calculable from the direct lower node 'TU_Id(HW)_S_Id' but not from another lower node (such as 'C_Id_S_Id').

Similarly, the change of execution orders or functions between edges blocks also transitivity. In other words, if all previous edges for a specific edge correspond to different functions or different execution orders, then this edge is non transitive; e.g. the edge ('ALL^C(HW)_S_Id', 'ALL^C(HW)_G_Id') corresponds to the function 'AVG(Rate_abs)' with an execution order of value 2 (see Fig. 6). This edge has a single previous edge ('Sem(HW)_S_Id', 'ALL^C(HW)_S_Id') which corresponds to the same function 'AVG(Rate_abs)' but with an execution order of value 1 (see Fig. 6). Because of the difference between the execution orders, the edge ('ALL^C(HW)_S_Id', 'ALL^C(HW)_G_Id') is not transitive. Therefore, the node 'ALL^C(HW)_ ALL^S' is calculable by transitivity from the node 'ALL^C(HW)_S_Id' but it is not calculable by transitivity from the node 'Sem(HW)_S_Id', because the aggregation schema (Fig. 4 (a)) requires to calculate firstly the absenteeism rates according to the dimension 'Courses' (node 'ALL^C(HW)_S_Id') in order then to calculate absenteeism rate based on the dimension 'Students' ('ALL^C(HW)_ ALL^S').

Figure 7 shows the resulting pre-aggregate lattice. Edges with crossed circle are obtained either from aggregation constraints or from the change of execution orders or aggregation functions between the edges.

Pruning the Lattice. Some paths or edges are invalid; therefore, these can be eliminated to reduce the lattice size (Fig. 8). This pruning is possible using the execution order. An edge can be deleted if it is preceded by an edge with a larger execution order (see Fig. 6).

In our example (see Fig. 4 (a)), we cannot apply the weighted average function 'AVG_W(Rate_Abs, Coeff)' on the 'Courses' dimension (with an execution order of

value 1) after the function 'AVG(Rate_Abs)' on the dimension 'Students' (with an execution order of value 2) as this would give erroneous results. Thus, to obtain the node 'TU_Id(HW)_G_Id' (absenteeism rate by group and TU on the hierarchy 'HCourse_weighted'), it is impossible to calculate it from the node 'C_Id_G_Id' (absenteeism rate by group and course on the hierarchy 'HCourse_weighted'). Therefore, the edge between 'C_Id_G_Id' and 'TU_Id(HW)_G_Id' can be deleted.

Figure 8 shows the final controlled pre-aggregate lattice after deleting the invalid edges.

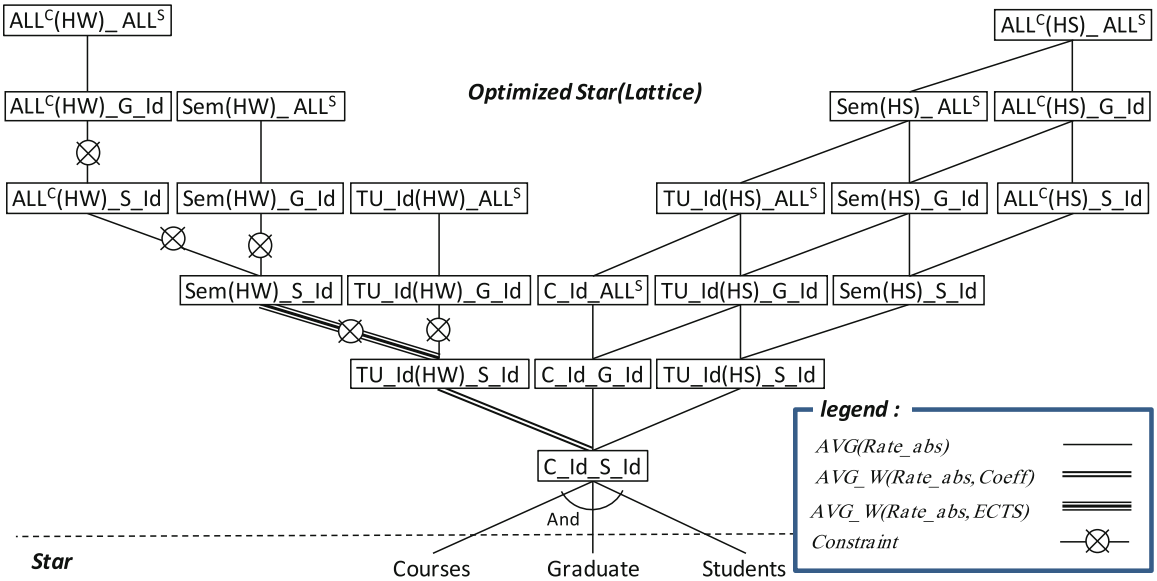


Fig. 8. Controlled pre-aggregate lattice (with pruned edges)

Modifying Edges. In our model, we have proposed a mechanism of aggregation constraint to fix the valid aggregation level from which a higher level is calculated. This valid level is not necessarily the one directly lower level. We express this case when we use a constraint value other than 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.

In our example, the absenteeism rate by semester on the hierarchy 'HCourse_ weighted' is calculated from the absenteeism rates by TU (constraint value = -1) (see Fig. 4 (a)). In case we had chosen to calculate this rate by semester from the rates by courses, the constraint would have been assigned to -2 and the lattice would have been as Fig. 9.

5 Validations

To demonstrate the feasibility of our approach, we have produced a prototype: *OLAP-Multi-Function*, described hereafter. We validate our proposal by overcoming the limits suffered by the software "Business Objects" with our prototype. Finally, experiments based on our prototype are detailed.

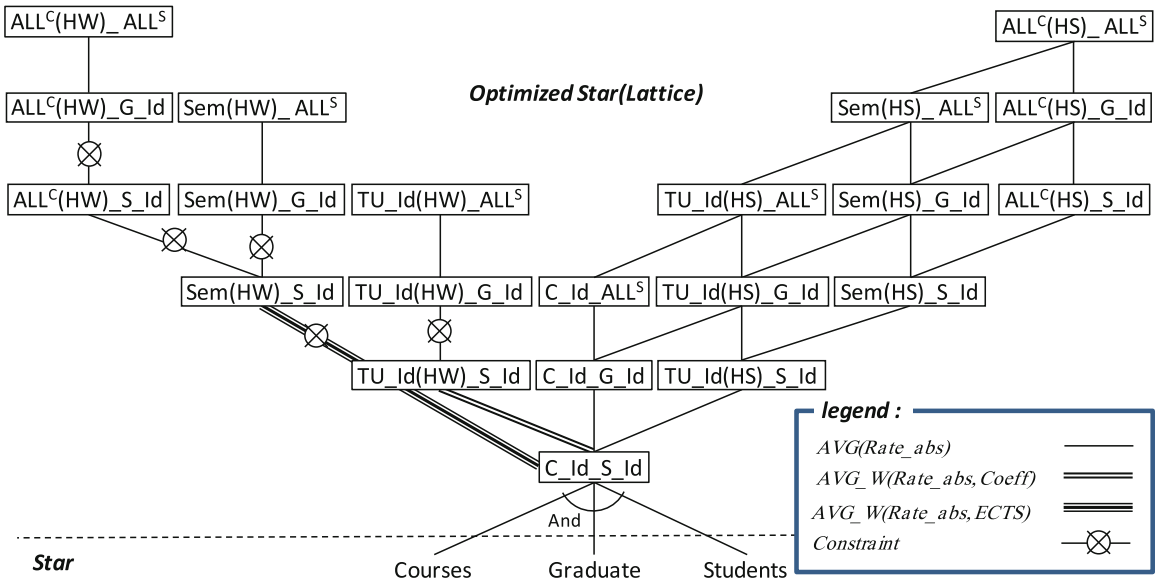


Fig. 9. Lattice with constraint = -2

5.1 OLAP-Multi-Function Prototype

Our prototype was implemented using Java 7 on top of the Oracle 12g DBMS. It allows designing a MDB with differentiated and multiple aggregation functions as well as supervising the OLAP manipulations carried out by analysts using a graphical representation.

Prototype Architecture. The main functionality of *OLAP-Multi-Function* (Fig. 10) is visualizing and facilitating the integration of aggregation functions in the multidimensional model. It is based on a set of graphic interfaces (Constructor) for defining the four types of aggregation functions (general, multiple dimensional, multiple hierarchical, differentiated), their execution orders and aggregation constraints.

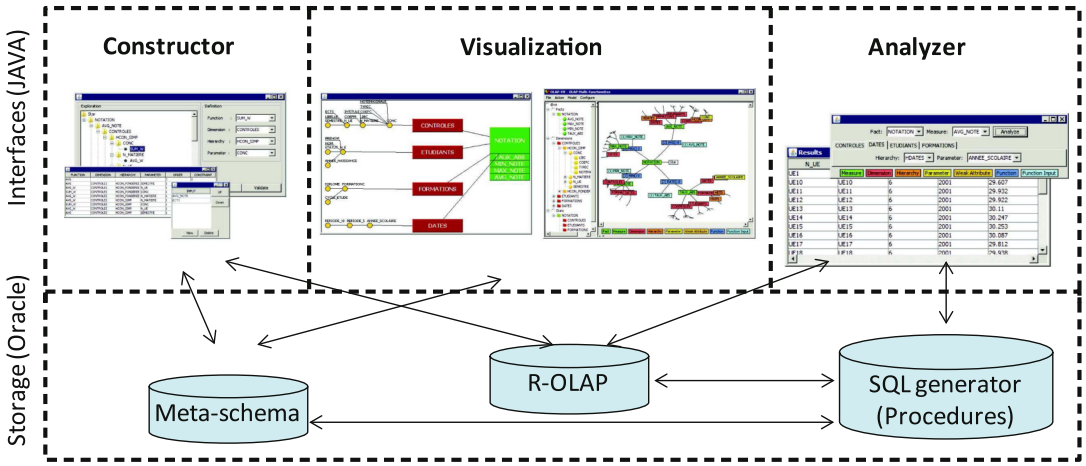


Fig. 10. Prototype architecture

The structural schema is displayed as a constellation graph based on graphic formalisms of facts, dimensions, and hierarchies introduced in [21, 22]. Different aggregation schemas are visualized in the form of a hyperbolic graph. For querying, the analyst selects the measure and the desired aggregation levels. After validation, *OLAP-Multi-Function* automatically calculates the result and presents it in the form of an R-OLAP table.

The storage level includes two databases. The first one contains the meta-schema that describes the structural elements of the multidimensional schema (facts, dimensions and hierarchies) as well as the aggregation functions, execution orders and aggregation constraints to build valid and coherent SQL queries (for more information about the meta-schema, see [12]). The second one contains facts and dimensions data implemented with the R-OLAP model.

SQL Queries Generator. To supervise the analysis, the prototype has a SQL query generator. The analyst configures the calculations to be done: the user must specify the measure and the desired aggregation levels. The generator translates interactions into SQL scripts executable in the context of an R-OLAP implementation. The generation process consists of the four following steps, described using a BPMN diagram (Business Process Modeling Notation) in Fig. 11:

1. Detecting tables of the logical R-OLAP model: this step identifies the tables used to store analysis data.
2. Determining aggregation functions: using the meta-schema and the required aggregation levels, this step identifies aggregation functions to be applied to perform the analysis.
3. Simplifying aggregation functions: this step is for detecting possible redundant calculations, i.e. a needless repetition of an aggregation function
4. Generating the SQL script: from the meta-schema and the previous steps, this step generates the final SQL query. It sends it to the DBMS that calculates the query and returns the results to the prototype.

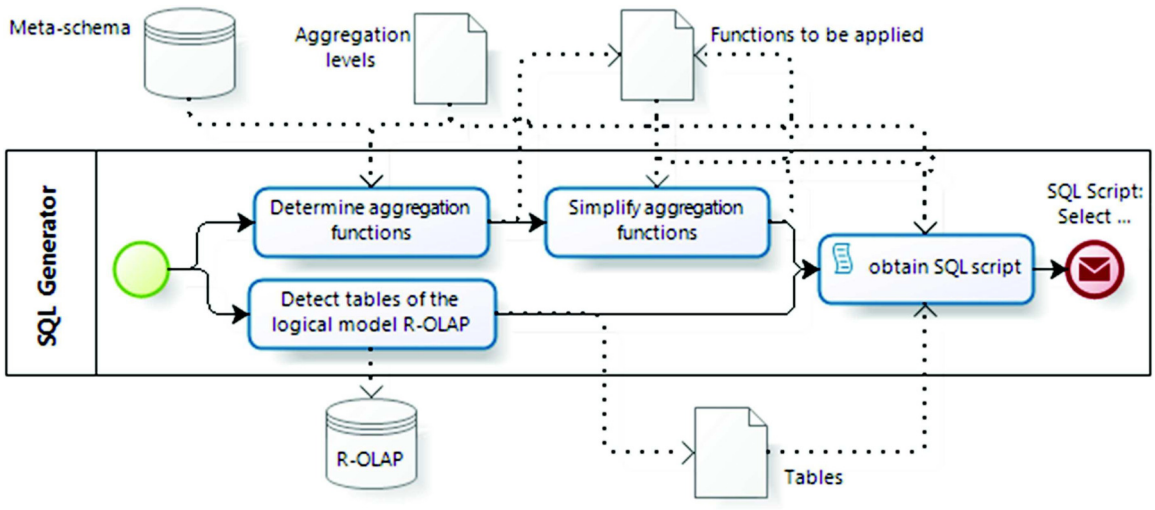


Fig. 11. SQL queries generator (shown in BPMN)

5.2 Discussing

We present in this section the advantages of our prototype *OLAP-Multi-Function* over one of the most used commercial tool: "Business Objects" (BO).

Business Objects. According to our knowledge, the major limit of BO is to use only one aggregation function for each measure. To know how far we can overcome this problem, we have applied our example (Fig. 2) in BO. We associated the measure 'Avg_Mark' with the aggregation function AVG. Thus, we can perform all possible analyses on dimensions 'Students' and 'Dates'. For example, we can analyze the average marks of courses ('C_Id' level) by periods of five years ('period-5' level and 'ALL' level on the 'Students' dimension). This analysis can be performed by the following SQL query:

```
SELECT C.C_Id, D.Period-5, AVG(G.Avg_Mark) AS Avg_Mark
FROM Courses C, Dates D, Graduate G
WHERE G.C_Id = C.C_Id AND G.Academic_year = D.Academic_year
GROUP BY C.C_Id, D.Period-5
```

But for analyzing the data along the dimension 'Courses', we use a non-standard aggregation function: 'AVG_W' (weighted average). To solve this problem, there are two proposals:

- The use of a calculated measure: this proposal means defining a new measure (AVG_Mark_TU) calculated by Eq. 1, defined in Sect. 1.2 (Fig. 12). The problem with this proposal is that this equation ("Select:" in Fig. 12) will not be used to calculate the measure at the TU level but at the base level ('C_ID'), then to calculate the measure at the TU level, its own aggregation function will be used to aggregate the values.
- The use of a variable: the advantage of this proposal is that the variable can use values of an aggregated measure contrary to the calculated measure that use only the base values. The problem is that if the variable uses values other than the measure, these values must be used in the analysis, otherwise there will be errors; e.g. the variable 'AVG_Var' (Fig. 13) is calculated by Eq. 1 where the values of 'Coeff' are not used in the analysis, hence the errors. To overcome this problem and get the requested results, we can define two new measures: the first M1 = SUM (Coeff * AVG_Mark); the second M2 = SUM(Coeff) and then the variable becomes AVG_Var = M1 /M2.

Thus, by using variables, we can calculate:

1. A non-standard function,
2. A second aggregation function from the results of the main function associated with the measure. This is similar to associating two aggregation functions with one measure.

The limits of the use of a variable are when that variable is used for a specific level (as the variable 'AVG_Var' in our example); then there is no constraint that forbids the user to use it for a different level and that would give a wrong result. Another limit is that we cannot use a variable to calculate another variable otherwise there will be errors

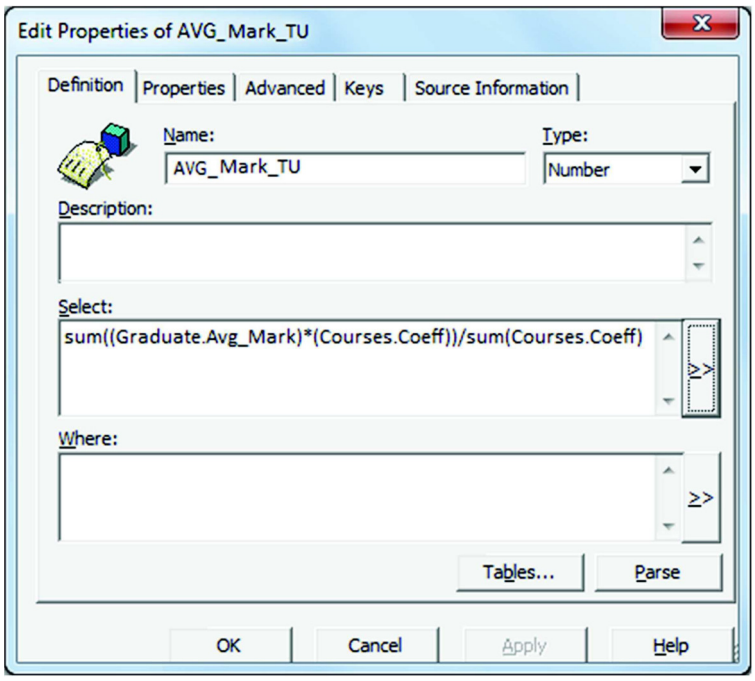


Fig. 12. Use of a calculated measure in a BO query

	Nom	ST1		ST2	
Semestre	TUTitle	AVG_Mark	AVG_Var	AVG_Mark	AVG_Var
5	TU1	12,00	#IERR	12,67	#IERR
5	TU2	11,00	#IERR	13,00	#IERR
6	TU3	15,00	#IERR	12,00	#IERR
6	TU4	10,50	#IERR	10,00	#IERR

Fig. 13. Use of a variable in a BO report

(Fig. 13). Thus we cannot use the variable 'AVG_Var' to calculate the average marks at the Semester level.

OLAP-Multi-Function. Our prototype integrates several aggregation functions for the same measure in the multidimensional model. It overcomes the two principal limits of BO: the use of non-standard functions and the use of several aggregation functions. To use the non-standard weighted average function 'AVG_W', a generic aggregation function was implemented:

- An Oracle object type (class) was used to implement the four routines of the Data cartridge interface ODCIAggregate: ODCIAggregateInitialize, ODCIAggregateIterate, ODCIAggregateMerge and ODCIAggregateTerminate. These methods correspond to internal operations that each aggregation function performs (respectively Initialize, Iterate, Merge and Terminate).
- Then, our aggregation function 'AVG_W' was created to compute a weighted average based on our previous object type. This function takes one parameter composed of the data to aggregate and the weight (TYPE data_weighted AS OBJECT (value NUMBER, weight NUMBER)).

In order to use several aggregation functions in the same analysis, our SQL generator can generate nested queries. Note that the SQL queries are generated using an interface where the user manipulates only multidimensional concepts. Thus, the complexity of both the aggregation and the logical structure of the MDB are hidden. E.g. the SQL query generated by our prototype for analyzing the average marks by semester and by group of students is as follows:

```
SELECT G_Id, Semester, AVG(Avg_Mark) AS Avg_Mark
FROM ( SELECT G_Id, Semester, S_Id, Academic_year,
       AVG_W(DATA_WEIGHTED(Avg_Mark, ECTS)) AS Avg_Mark
       FROM ( SELECT S.G_Id, C.Semester, S.S_Id,
              D.Academic_year,C.TU_Id, C.ECTS,
              AVG_W(DATA_WEIGHTED(G.Avg_Mark, C.Coeff))
                                                AS Avg_Mark
              FROM STUDENTS S, COURSES C, DATES D, GRADUATE G
              WHERE G.S_Id = S.S_Id AND G.C_Id = C.C_Id
              AND G.Academic_year = D.Academic_year
              GROUP BY S.G_Id, C.Semester, S.S_Id,
                       D.Academic_year, C.TU_Id, C.ECTS)
       GROUP BY G_Id, Semester, S_Id, Academic_year)
GROUP BY G_Id, Semester
```

5.3 Experiments

The SQL query generator serves as an experimental platform for which we show a series of experiments.

Experiment 1. The first experiment is intended to study the impact of our proposal on the execution time of OLAP analysis queries.

Collection: to our knowledge, there are no benchmarks that use for a same measure, different aggregation functions according to analysis axes, hierarchies and aggregation levels. Therefore, we use data related to the diploma delivery jury; the data grouping size on the dimension 'Courses' is five, i.e. each instance of a higher level corresponds to five instances of lower level (for example, each semester has five TUs).

Protocol: we observe the execution time (in seconds) in accordance with the number of tuples of the fact (from two to ten millions) of three queries:

- The first query aggregates average marks at the TU level. It uses (as in the classical model) a single aggregation function ('AVG_W'),
- The second query aggregates average marks at the semester level. It is based on two aggregation functions ('AVG_W' twice),
- The third query aggregates average marks at the 'ALL' level. It requires three aggregation functions ('AVG_W' twice and 'AVG' once).

We chose these three queries to present the impact of using several functions (second and third queries) compared with the classical model that uses a single function (first query).

Results: Fig. 14 (right) shows the execution time of the three queries. Queries execution times increase regularly with the number of tuples. The distance between the curves of the first query (aggregation in the classical model) and the second query (aggregation in our proposed model) represents the overhead time of our model required to apply the second aggregation function. This time is approximately 5 % of the total query execution time. The additional time to apply the third function seems to be non-remarkable (the curve of third query is nearly on top of the curve of second query). In fact, this phenomenon is related to the data volume that decreases with the functions previously applied. Thus, when calculating the third function, the data volume is significantly reduced compared with the initial volume.

Experiment 2. The second experiment aims at studying the relationship between the execution time and the data grouping size. By grouping size, we mean the number of values of a lower parameter that are grouped into one value of a higher parameter.

Collection: we work on two different versions of our example of the diploma delivery jury; the first one with data grouping size 2 on the dimension 'Courses' and the second one with data grouping size 5.

Protocol: we observe the execution time in accordance with the number of tuples and the size of the data grouping of the four queries:

- Two queries at the TU level (one with a grouping size of 2 and the other with5) that use an aggregation function.
- Two queries at 'ALL' level (one with a grouping size of 2 and the other with 5) that use three aggregation functions.

Results: Fig. 14 (left) shows the execution time of the four queries. The execution time of queries with grouping size 5 is less than that of the queries with a grouping size of 2. We note that the query execution time seems mainly influenced by the grouping size. Thus, the query with grouping size 2 and a single aggregation function (TU (2)) is more

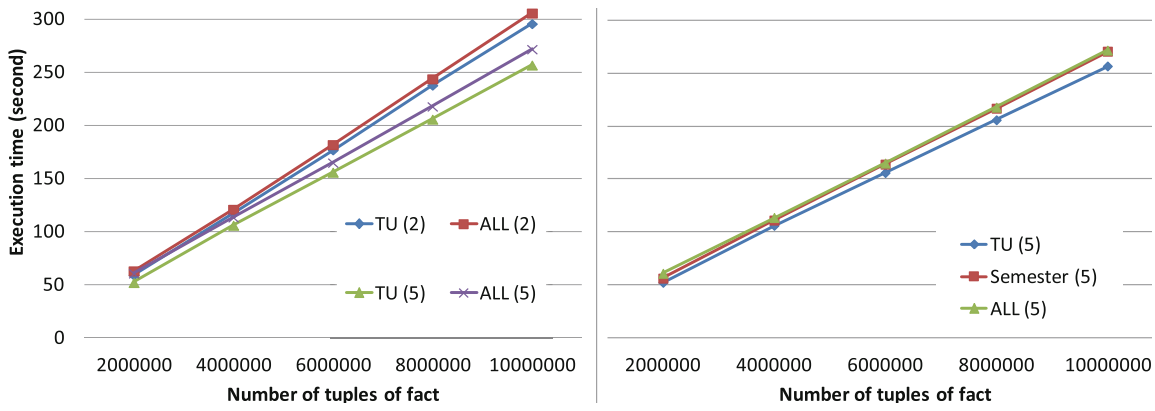


Fig. 14. Experiments

expensive in terms of computing time than the query with a grouping size of 5 despite three aggregation functions (ALL (5)). The grouping size appears to have a crucial impact on the execution time.

6 Conclusion and Future Work

This paper defines a conceptual multidimensional data model flexible enough to allow the designer to specify differentiated and multiple aggregations. *Multiple*, as the same measure can be aggregated by several aggregation functions according to analysis axes or hierarchies and *differentiated* as these aggregations may vary, depending on the aggregation level. Furthermore, the model is expressive enough to check the function calculations validity. Aggregation constraints define the level from which the aggregation should be calculated. The execution order defines the necessary order between non-commutative aggregation functions.

This model is based on a two-level graphical formalism: the structural schema describes the multidimensional structures while hiding the aggregation complexity and aggregation schemas detail the aggregation mechanisms for each measure.

At the logical level, the implementation can be optimized by a controlled lattice of pre-computed aggregates, where invalid edges can be pruned.

We plan to continue our work by revisiting algorithms that compute pre-aggregates, adapting them to our model and studying the effects of changes in the lattice when selecting nodes for improving performance. We also plan to study OLAP manipulation operators on our model.

References

1. Abelló, A., Samos, J., Saltor, F.: YAM2: a multidimensional conceptual model extending UML. Inf. Syst. **31**(6), 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. In: Ingénierie des Systèmes d'Information (ISI), vol. 16(6), pp. 11–39 (2011). (In French)
3. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
4. Cuzzocrea, A.: Providing probabilistically-bounded approximate answers to non-holistic aggregate range queries in OLAP. In: 8th International Workshop on Data Warehousing and OLAP (DOLAP 2005), pp. 97–106 (2005)
5. Cuzzocrea, A., Furfaro, F., Masciari, E., Saccà, D., Sirangelo, C.: Approximate query answering on sensor network data streams. In: Stefanidis, A., Nittel, S. (eds.) GeoSensor Networks, pp. 53–72. CRC Press, Boca Raton (2004)
6. Cuzzocrea, A., Saccà, D.: Balancing accuracy and privacy of OLAP aggregations on data cubes. In: 13th International Workshop on Data Warehousing and OLAP (DOLAP 2010), pp. 93–98 (2010)
7. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual design of data warehouses from E/R schemes. In: International Conference on HICSS 1998, vol. 7, pp. 334–343 (1998)

8. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: International Conference on ICDE 1996, pp. 152–159 (1996)
9. Gyssens, M., Lakshmanan, L. V. S.: A foundation for multi-dimensional databases. In: International Conference on VLDB 1997, pp. 106–115 (1997)
10. 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)
11. 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)
12. Hassan, A., Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Agrégations multiples différentiées dans les bases de données multidimensionnelles. In: Ingénierie des Systèmes d'Information (ISI), vol. 18(2), pp. 75–102 (2013). (In French)
13. Kimball, R., Ross, M.: The Data Warehouse Toolkit: the Definitive Guide to Dimensional Modeling, 3rd edn. John Wiley & Sons, NY (2013). ISBN 978-1-118-53080-1
14. Jaecksch, B., Lehner, W.: The planning OLAP model - a multidimensional model with planning support. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 14–25. Springer, Heidelberg (2011)
15. Kotidis, Y., Roussopoulos, N.: DynaMat: a dynamic view management system for data warehouses. In: International Conference on SIGMOD 1999, pp. 371–382 (1999)
16. Lujàn-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. Data Knowl. Eng. **59**, 725–769 (2006)
17. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modelling. Data Knowl. Eng. **68**, 1452–1469 (2009)
18. Oliveira, R., Rodrigues, F., Martins, P., Moura, J.P.: Extending the dimensional templates approach to integrate complex multidimensional design concepts. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 26–38. Springer, Heidelberg (2011)
19. Pedersen, T.B., Jensen, C., Dyreson, C.: A foundation for capturing and querying complex multidimensional data. Inf. Syst. **26**, 383–423 (2001)
20. Prat, N., Wattiau, I., Akoka, J.: Representation of aggregation knowledge in OLAP systems. In: the 18th European Conference on Information Systems ECIS. (2010)
21. 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)
22. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. Int. J. Data Warehous. Min. **4**(1), 17–46 (2008)
23. Silva, J., Times, V.C., Salgado, A.C.: A set of aggregation functions for spatial measures. In: 11th International Workshop on Data Warehousing and OLAP (DOLAP 2008), ACM. ISBN: 978-1-60558-250-4, pp. 25–32 (2008)
24. Torlone, R.: Conceptual multidimensional models. In: Rafanelli, M. (ed.) Multidimensional Databases: Problems and Solutions, pp. 69–90. IGI Publishing Group, PA (2003)
25. Vassiliadis, P., Sellis, T.K.: A survey of logical models for OLAP databases. SIGMOD Rec. **28**(4), 64–69 (1999)
26. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Modeling ETL activities as graphs. In: International Conference on DMDW 2002, pp. 52–61 (2002)
27. 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)