2010

Representation of Aggregation Knowledge in OLAP Systems

Nicolas Prat
ESSEC, prat@essec.fr

Isabelle Wattiau
CNAM, wattiau@cnam.fr

Jacky Akoka
CNAM, akoka@cnam.fr

REPRESENTATION OF AGGREGATION KNOWLEDGE IN OLAP SYSTEMS

Journal:	*18th European Conference on Information Systems*
Manuscript ID:	ECIS2010-0300
Submission Type:	Research Paper
Keyword:	Database design/integration, Decision support systems (DSS), Object oriented modeling, Business intelligence

REPRESENTATION OF AGGREGATION KNOWLEDGE IN OLAP SYSTEMS

Nicolas Prat, ESSEC Business School, avenue Bernard Hirsch, BP 50105, 95021 CERGY Cedex, France, prat@essec.fr

Isabelle Comyn-Wattiau, CEDRIC-CNAM & ESSEC Business School, 292 rue St Martin, 75141 PARIS Cedex 03, France, wattiau@cnam.fr

Jacky Akoka, CEDRIC-CNAM & TEM, 292 rue St Martin, 75141 PARIS Cedex 03, France, akoka@cnam.fr

Abstract

Decision support systems are mainly based on multidimensional modeling. Using On-Line Analytical Processing (OLAP) tools, decision makers navigate through and analyze multidimensional data. Typically, users need to analyze data at different aggregation levels, using OLAP operators such as roll-up and drill-down. Roll-up operators decrease the details of the measure, aggregating it along the dimension hierarchy. Conversely, drill-down operators increase the details of the measure. As a consequence, dimensions hierarchies play a central role in knowledge representation. More precisely, since aggregation hierarchies are widely used to support data aggregation, aggregation knowledge should be adequately represented in conceptual multidimensional models, and mapped in subsequent logical and physical models. However, current conceptual multidimensional models poorly represent aggregation knowledge, which (1) has a complex structure and dynamics and (2) is highly contextual. In order to account for the characteristics of this knowledge, we propose to represent it with objects and rules. Static aggregation knowledge is represented using UML class diagrams, while rules, which represent the dynamics (i.e. how aggregation may be performed depending on context), are represented using the Production Rule Representation (PRR) language. The latter allows us to incorporate dynamic aggregation knowledge. We argue that this representation of aggregation knowledge allows an early modeling of user requirements in a decision support system project. In order to illustrate the applicability and benefits of our approach, we exemplify the production rules and present an application scenario.

Keywords: Decision Support System, Data warehouse, On-line Analytical Processing (OLAP), Aggregation, UML, Production rule.

1 INTRODUCTION

Data warehouses are the cornerstone of data-driven decision support systems (DSS) (March and Hevner, 2007). They rely on a multidimensional model, providing users with a business-oriented view of data. Using On-Line Analytical Processing (OLAP) tools, decision makers may then navigate through and analyze multidimensional data. Typically, users need to analyze data at different aggregation levels, which is achieved by means of the roll-up operator (and its converse operator, the drill-down operator). Therefore, in order to ensure correct and flexible aggregation, aggregation knowledge should be adequately represented in conceptual multidimensional models, and mapped in subsequent logical and physical models. This aggregation knowledge pertains to aggregation functions that can be applied, as well as to hierarchies along which they are applicable, etc. Incorporating this knowledge inside OLAP systems allows designers to provide final users with intuitive and user-friendly decision support systems. This knowledge also contributes to the quality of decision support systems, which itself affects the quality of management decision (Clark et al., 2007). When aggregating data, the effect of errors may be devastating, as errors may accumulate and propagate through the aggregation process (Parssian, 2006). In our approach, we don't control the quality of input data (input data are assumed as given). However, by representing aggregation knowledge explicitly, we control the aggregation process, thereby minimizing the risk of introducing new errors during aggregation.

A growing number of organizations refer to aggregation knowledge in applications of decision support systems. Typical applications include pricing models, planning and budgeting systems, sales tracking, etc. For example, in pricing models, users request the manipulation of the model's dimensions to evaluate pricing combinations. In planning and budgeting systems aggregations are used to evaluate financial requirements. Finally, sales tracking systems are based mainly on 'what if analysis' which can be performed using aggregations along several dimensions. In some cases it may be required to analyze sales at a brand level rather than at a category of brands. As it can be seen, aggregation is at the center of decision support systems.

The rest of the paper is organized as follows. A brief state of the art is proposed at Section 2. Section 3 focuses on the representation of static aggregation knowledge, using UML class diagrams. Section 4 proposes a typology of rules to represent dynamic aggregation knowledge, and illustrates how these rules may be represented and organized in the PRR formalism. Section 5 illustrates the approach with an example scenario. Section 6 concludes and points to further research.

2 RELATED WORKS

Several authors have studied how data may be aggregated along hierarchies (a.k.a. summarizability), and several conceptual multidimensional models incorporate aggregation knowledge (Abello et al. 2006, Hüsemann et al., 2000, Lujan-Mora et al., 2006, Prat et al., 2006). (Abello et al., 2006) includes summarizability constraints. Horner et al. (2004) develop a taxonomy of the additive nature of measures (non-, semi-, fully-additive). Lujan-Mora et al. (2006) introduce non-additivity and semi-additivity as constraints on measures. Based on the typology of aggregation functions presented in (Pedersen and Jensen, 1999), Hüsemann et al. (2000) define aggregations using four restriction levels. Espil and Vaisman (2003) use intentional rules to define exceptions in aggregation hierarchies; this paper illustrates the applicability of rules to model aggregation knowledge, but does not consider aggregation functions. Lenz and Shoshani (1997) define three necessary conditions for summarizability: disjointness, completeness of category attributes, and type compatibility conditions. The disjointess condition states that the object attributes must form disjoint subsets over the objects. The completeness condition means that the grouping of objects into clusters is complete. Finally, the last condition verifies that summarization is performed correctly based on types of measures (stock,

flow, value-per-unit). Examples of summarization problems can be found in (Horner and Song, 2005). They can result from inaccuracies respectively at the schema, data, and computation level. For example, at the schema level, one hierarchy represents countries, states and cities. However, for some countries, there is no state level. At the data level, measurement instruments may lead to imprecise data. Finally, at the computation level, information may be stored using different units (dollars vs. euros). A recent survey synthesizes most summarizability issues in multidimensional modelling (Mazon et al., 2009).

One means to solve these summarizability problems is to introduce, at the schema level, some knowledge about aggregation. Lenz and Shoshani (1997) distinguish three types of measures: flow measures, stock measures and value-per-unit measures. This knowledge is needed to determine the appropriate aggregation operators. Despite the contributions of previous research, aggregation knowledge is still poorly or inadequately represented in current conceptual multidimensional models. Aggregation knowledge is difficult to represent in a simple way (Torlone, 2003). This knowledge (1) has a complex structure and dynamics and (2) is highly contextual in nature (e.g. the aggregation functions that may be applied at a given time may depend on the functions applied previously).

In order to account for the characteristics of aggregation knowledge, we propose to represent it with objects (UML class diagrams (OMG, 2009)) and rules (in Production Rule Representation language (OMG, 2009)). Static aggregation knowledge is represented in the UML class diagrams, combined with PRR rules which represent the dynamics (i.e. how aggregation may be performed depending on context). The next section describes the representation of static aggregation knowledge.

3 REPRESENTING STATIC AGGREGATION KNOWLEDGE

Users need to analyze data at different aggregation levels (using roll-up and drill-down functions). Therefore, aggregation knowledge should be adequately represented in conceptual multidimensional models, and mapped in subsequent logical and physical models. A conceptual multidimensional model should clearly distinguish between structure (schema) and content (instances) (Torlone, 2003). In our approach, this distinction is crucial, since aggregations (roll-ups) will be performed at the instance level. We therefore distinguish between the core conceptual multidimensional model (used for the conceptual representation of a data warehouse), and the data cube model. A data cube is a user view on multidimensional data. Like the core conceptual multidimensional model, the data cube model is represented at the conceptual level, i.e. independently of any OLAP implementation. Aggregations operate on and result in data cubes.

The meta-model described in Figure 1 represents the static view of the core conceptual multidimensional model. This meta-model draws on previous work described in (Akoka et al. 2001, Prat et al. 2006) and focuses more specifically on concepts related to aggregation. A conceptual multidimensional schema is composed of facts and dimensions. For example, in a university, the fact Teaching may be dimensioned by the dimensions Professor, Course, and Time. Facts are composed of measures and dimensions are composed of hierarchies. For example, the fact Teaching may be characterized by the measures hours_taught and hours_billed; the dimensions Professor, Course and Time may be composed (respectively) of the hierarchies Professor→Department→All, Course→Programme→All and Term→Year→All. Hierarchies are built upon rollup relationships between dimension levels (e.g. the hierarchy Professor→Department→All has 3 dimension levels, linked by rollup relationships). The roles of rollup relationships are characterized by their multiplicity. For example, when the lower multiplicity of the source role is 0, we have an asymmetric hierarchy (Malinowski and Zimanyi, 2006). In the hierarchy Agency→Branch→Bank→All, if some branches are not subdivided into agencies, we have an asymmetric hierarchy. When the upper multiplicity of the target role is * (i.e. the target is plural), we have a non-strict hierarchy (Malinowski and Zimanyi, 2006), which may require the definition of a coefficient (Akoka et al., 2001). In our university example, if some courses are common to several programmes, Course→Programme→All is a non-

strict hierarchy; for the measure hours_billed, we need to define a coefficient to determine how the hours_billed will be divided between the different programmes (e.g. the programme with most students in the course will be in charge of paying the hours, or the relative share of each programme will be computed based on the number of students from the programme in the course). Our model enables the data warehouse designer to specify some applicable aggregation functions (an applicable aggregation function may be specified for a measure i.e. independently of dimensions and hierarchies, for all the hierarchies of a dimension, for a specific hierarchy, or for a sub-hierachy) ; however, specifying aggregation functions at this stage is not compulsory since PRR rules (described in Section 3) are specifically aimed at representing dynamic aggregation knowledge, including applicable aggregation functions. In line with the aggregation functions generally studied in extant literature, our research considers the aggregation functions SUM, AVG, MIN, MAX and COUNT. Following Lenz and Shoshani (1997), the core conceptual multidimensional model distinguishes stocks (a measure is either a stock or a flow), and values-per-unit. These characteristics of measures will have a direct impact on applicable aggregation functions. Finally, we distinguish between elementary measures and pre-aggregated measures. This distinction accounts for the fact that data warehouses often don't store data at the detail level of transactions, but pre-compute some aggregates.

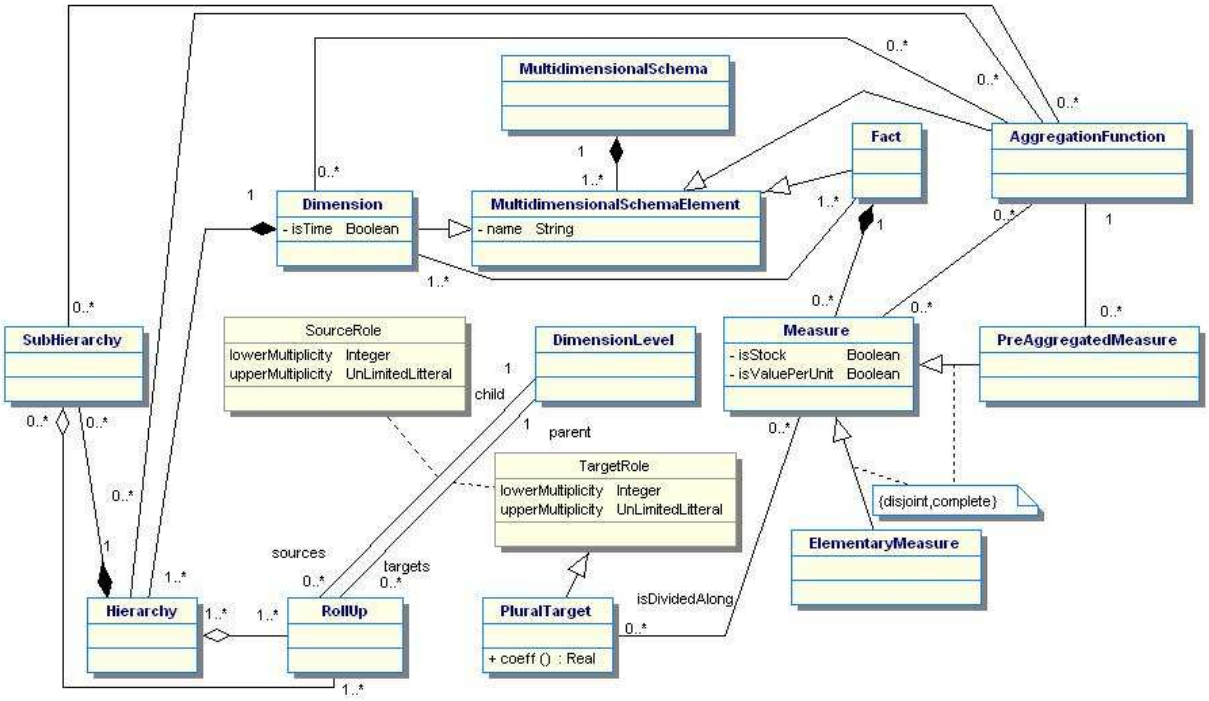

Figure 1. Core Conceptual Multidimensional Model (MM): static view.

Figure 2 represents the static view of a data cube ("MM" refers to classes of the core conceptual multidimensional model). A data cube is composed of cells and axes. An axis corresponds to one (and only one) dimension in the core conceptual multidimensional model. Each axis has a default hierarchy (the default hierarchy used for rollup, as proposed in the Common Warehouse Metamodel (OMG, 2009)). A cell is composed of cell values (one value for each measure). Cell values may be base cell values or aggregated cell values (i.e. cell values resulting from previous rollups).

In our university example, Teaching is an example of data cube. This cube is tri-dimensional (one axis for each of the dimensions Professor, Course and Time). As discussed above, each of these 3 dimensions has one hierarchy. A cell in the cube has coordinates along each axis (e.g. coordinates ("Dalton", "Math I" and "T1 2008-2009"), which are instances of the dimension levels Professor, Course and Term respectively). Each cell contains the value of each measure for this cell (e.g. Dalton has taught, and also billed, 15 hours of Math I in T1 2008-2009). Based on the cube Teaching, rollups

will be performed (e.g. the user may first consider summing the hours taught by year, along the dimension Time, resulting in a new cube).

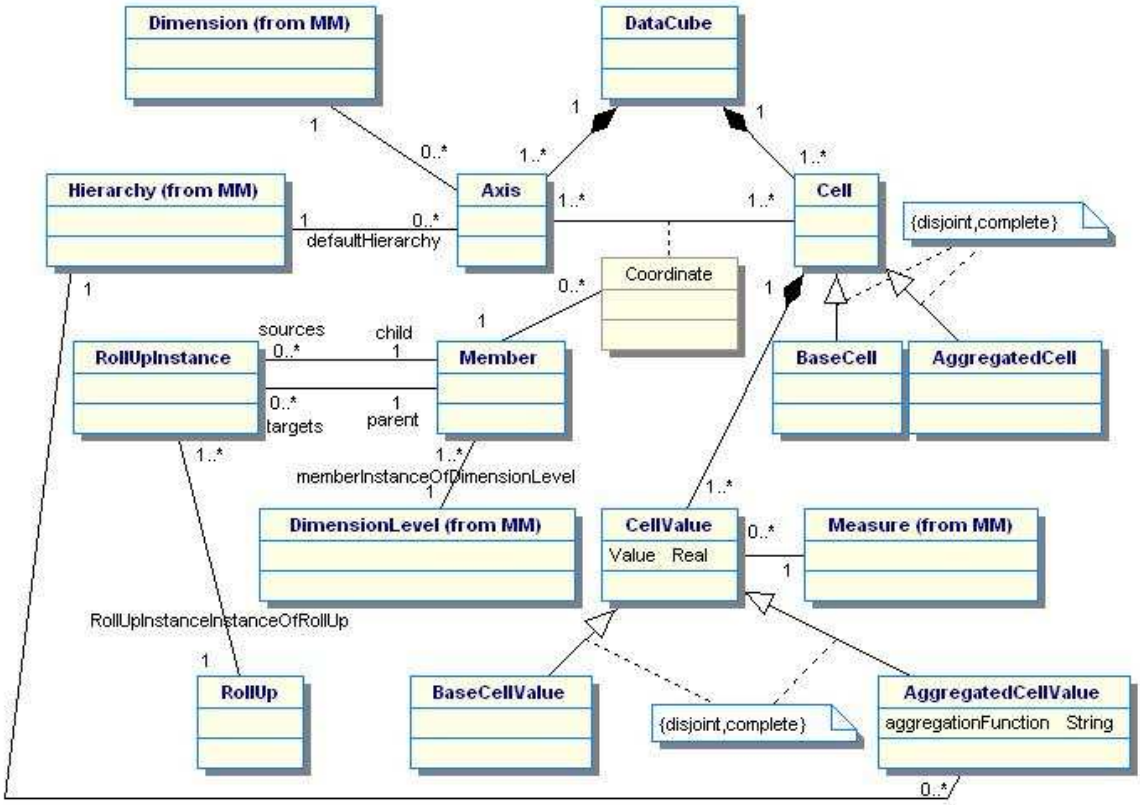

Figure 2. Data cube model: static view.

The UML class diagrams of the two meta-models are limited to the representation of static knowledge related to aggregation. In order to integrate more knowledge, we need to take into account the dynamic aspects. To achieve this aim, we propose to use a rule language described in the next section.

4 REPRESENTING DYNAMIC AGGREGATION KNOWLEDGE: PRR RULES

In order to completely represent aggregation knowledge, we must specify how, in a given context, aggregation may be performed on a given data cube (i.e. how to choose the aggregation function, and how to perform the aggregation once the aggregation function has been chosen). Since this knowledge is complex and highly contextual (depending on the data cube, the user preferences…), it is appropriately represented with rules. Rules also enable us to trace and explain why a particular aggregation function has been chosen in a given context and how it has been applied.

To represent aggregation rules, we have chosen the Production Rule Representation language, complying with our choice of UML for representing the conceptual multidimensional model. The Production Rule Representation language enables the representation of rules related to the UML class diagrams presented in Section 3, independently of subsequent implementations.

4.1 The Production Rule Representation language (PRR)

The Production Rule Representation language (PRR) has been proposed by OMG for high-level (tool-independent) representation of rules in conjunction with UML.

PPR rules are grouped into rulesets. A ruleset is a collection of rules with a particular mode of execution (sequential or inferencing). When inferencing is chosen as a mode of execution, priorities may be defined to constrain the order in which rules will be executed. PRR currently supports only forward chaining.

A production rule is typically represented as *if [condition] then [action-list]*. For example, an action may be the invocation of an operation associated with a UML class, or the assertion (creation) of a new object.

Variables may be defined at the ruleset level or at the rule level. Rules variables are used for binding.

PRR rules can be represented formally, based on an extension of the Object Constraint Language (OCL) (OMG, 2009).

4.2 Typology and examples of rules

We distinguish the following types of aggregation rules:

- *Semantic aggregation rules*, which are based on the semantics of elements of the conceptual multidimensional model (e.g. semantics of dimensions, measures, aggregation functions).
- *Syntactic aggregation rules*, which express the mathematical properties of aggregation functions (e.g. commutativity, related to the concept of distributivity (Lenz and Thalheim, 2001)).
- *User preferences* (e.g. "For a given measure, the same aggregation function should be applied to all dimensions along all hierarchies.").
- *Aggregation execution rules*, indicating how to perform aggregation along a specific hierarchy.

Semantic and syntactic aggregation rules indicate which aggregations are correct; user preferences indicate which aggregations are preferable (in case several candidate aggregation functions are applicable). Finally, aggregation execution rules indicate how a particular aggregation function should be executed once it has been chosen (e.g. how to perform aggregation along a non-strict (Malinowski and Zimanyi, 2006) hierarchy).

We give illustrations for each category of rules.

4.2.1 Semantic aggregation rules

These rules are complex and may depend, among other things, on the semantics of measures, dimensions and hierarchies. By representing aggregation knowledge as rules, we enable easy update of semantic aggregation knowledge (new rules are added as new semantic aggregation knowledge is acquired).

Example R1: Measures of type stock are not additive along temporal dimensions (Lenz and Shoshani, 1997).

Example R2: Ratios (value-per-units) are not additive along any dimension (Lenz and Shoshani, 1997).

Example R3: For a non-elementary (i.e. pre-aggregated) measure of type COUNT, along certain dimensions, aggregation may be incorrect from a certain dimension level, or a change in aggregation function may be necessary.

As an illustration of rule R3, consider the number of credits of a module (i.e. the number of credits that the student will get in the module if he passes). Suppose we have the hierarchy

Module→Diploma→Institution. When rolling up from Module to Diploma, it makes sense to use the aggregation function SUM (total number of credits for the diploma). However, totaling the number of credits for the different diplomas of an institution does not make sense. Other aggregation functions may be used instead, e.g. MIN.

4.2.2 Syntactic aggregation rules

These rules indicate the correct sequencing of aggregation functions, within a given dimension, or between different dimensions.

Example R4: For a given measure, making the sum of averages, minima or maxima, does not make sense. (Horner et al., 2004)

As another example of syntactic aggregation rules, these rules may be used to express distributivity. Citing Hurtado et al. (2005, p.861), "A distributive aggregate function af can be computed on a set of measures by partitioning the set into disjoint subsets, aggregating each separately, and then computing the aggregation of these partial results with another aggregate function we will denote as af^c. Among the SQL aggregate functions, COUNT, SUM, MIN, and MAX are distributive. We have that $COUNT^c$ =SUM, and for SUM, MIN, and MAX, af^c =af."

Example R5: SUM, MIN, MAX and COUNT are distributive.

Example R6: AVG is not distributive.

4.2.3 User preferences

These rules enable the representation of preferences for a given user, a profile of users, or for all users.

Example R7: If applicable aggregation functions are specified in the conceptual multidimensional schema, these functions should be applied in priority.

Example R8: Distributive aggregation functions should be applied preferably to non-distributive aggregation functions.

Example R9: For a given measure, the same aggregation function should be applied to all dimensions along all hierarchies.

Example R10: For a given measure and a given hierarchy, the same aggregation function should be applied along all levels of the hierarchy.

Example R11: Aggregation should be stopped when null values are obtained.

(Null values may be obtained when aggregating measures along asymmetric hierarchies (Malinowski and Zymanyi, 2006) for example).

4.2.4 Aggregation execution rules

These rules are needed, in particular, to deal with non-standard (e.g. non-strict or asymmetric) hierarchies. They may also explicit how null values are taken into account in computing aggregation, which is crucial in OLAP applications (Lenz and Thalheim, 2001).

Example R12: Sums along strict hierarchies are computed by considering null values as 0.

Example R13: Minima and maxima along strict hierarchies are computed by excluding null values.

The following two rules concern the computation of sums along non-strict hierarchies.

Rule R14 deals with the case when a measure is divided. As an illustration, in our university example, we have the non-strict hierarchy Course→Programme→All (a course may be shared by several programmes; in other words, the upper multiplicity from Course to Programme is "*", meaning

"several"). In this case, if the measure is divided, the coefficient is applied before performing the sum (here, if we consider the measure hours_billed, the coefficient is applied to this measure before performing the sum by Programme, along the dimension Course).

Example R14: In case of a non-strict hierarchy, when a measure is divided, the coefficient should be applied first, before performing the SUM.

Rule R15 deals with the case when a measure is undivided. In our university example, hours_taught are undivided; all the hours_taught in a course count as hours taught in the programme; however, at the upper level (when considering all programmes of the university), hours taught in courses common to several programmes should only be counted once.

Example R15: In case of a non-strict hierarchy, when a measure is undivided, no coefficient should be applied before performing the SUM. However, at the upper level of the hierarchy, the measure should not be counted multiple times.

4.3 PRR rules organization and examples

The context of aggregation consists in a triple <*data cube*, *measure*, *axis*>. *Data cube* is the active data cube, *measure* is the measure that the user wants to aggregate, and *axis* is the axis along which the user wants to perform the aggregation. We define a first ruleset for choosing the aggregation function. Inside the ruleset, the candidate aggregation functions are determined by means of the semantic and syntactic aggregation rules, and user preferences. These rules add or delete aggregation functions in the list of candidate aggregation functions. Ultimately, a unique aggregation function is chosen. (We illustrate a few examples of the rules described in Section 4.2.)

The ruleset chooseAggregationFunction takes as input the aggregation context (triple <*data cube*, *measure*, *axis*>). It returns the chosen aggregation function. Initially, the set of candidate aggregation functions is the set of all possible aggregation functions.

```
RuleSet chooseAggregationFunction
   (in currentDataCube: DataCube, currentMeasure: Measure,
   currentAxis: Axis,
   out chosenAggregation: String)
Variable:
   candidateAggregationFunctions: Set =
       Set{'SUM', 'AVG', 'MIN', 'MAX', 'COUNT'}
```

Rule R1 (measures of type stock are not additive along temporal dimensions) is expressed as follows in PRR. The rule expresses that if SUM is among the candidate aggregation functions and the current measure is a stock and the dimension of the current axis is temporal, SUM should be suppressed from the list of candidate aggregation functions. It should be noted that this rule (as the following rules) is expressed with concepts of the data cube model presented in Section 3 (which itself refers to concepts of the core multidimensional model).

```
Rule R1_stockNotAdditiveTime
Condition:
   candidateAggregationFunctions->includes('SUM') and
   currentMeasure.isStock=true and
   currentAxis.dimension.isTime=true
Action:
   candidateAggregationFunctions =
   candidateAggregationFunctions->excluding('SUM')
```

Similarly, rule R4 is expressed as follows:

```
Rule R4_NotSumAfterAvgMinMax
RuleVariable:
   ?cell: Cell = currentDataCube.cells->any
```

```
        (c:Cell|currentDataCube.cells->first()=c)
    ?cellValues: Set = ?cell.cellValues->select
        (cv:CellValue|
        cv.oclIsTypeOf('AggregatedCellValue') and
        cv.measure=currentMeasure and
        cv.hierarchy.dimension=currentAxis.dimension and
        (cv.aggregationFunction='AVG' or
        cv.aggregationFunction='MIN' or
        cv.aggregationFunction='MAX'))
Condition:
   candidateAggregationFunctions->includes('SUM') and
    ?cellValues->notEmpty()
Action:
   candidateAggregationFunctions =
   candidateAggregationFunctions->excluding('SUM')
```

Rule R8 expresses that distributive aggregation functions are preferred: if the list of candidate aggregation functions contains AVG and at least one other aggregation function, AVG is removed from the list of candidate aggregation functions.

```
Rule R8_distributiveAggregationPreferable
Condition:
   candidateAggregationFunctions->includes('AVG') and
   candidateAggregationFunctions->size()>1
Action:
   candidateAggregationFunctions =
   candidateAggregationFunctions->excluding('AVG')
```

Once a unique aggregation function has been chosen, aggregation execution rules are used to perform the aggregation. For this purpose, we define a second ruleset. The ruleset takes as input the current data cube, the current measure, the current axis, as well as the aggregation function chosen previously. The ruleset declares the new data cube, which will be constructed as a result of performing the aggregation.

```
RuleSet executeAggregation
   (in currentDataCube: DataCube, currentMeasure: Measure,
   currentAxis: Axis,
   chosenAggregation: String)
Variable:
   newDataCube: DataCube
   theAggregatedCell: AggregatedCell
   theMembers: Set = Set{}
```

The aggregation execution rules are defined in this second ruleset. For space reasons, these rules are not shown.

5 EXAMPLE SCENARIO

In order to illustrate the applicability and benefit of our approach, we consider the example of teaching in a university. This example, which we have started to introduce in preceding sections, is illustrated in Figure 3. The bottom of the Figure shows the data cube, while the corresponding multidimensional schema is shown in the top. We assume here that hours_taught is the only measure.

There are several difficulties in this case. One difficulty is the non-strict hierarchy mentioned above: a course may be taught for several programmes. Another difficulty may be encountered when aggregating hours by department: some professors are not attached to departments but directly to the university. Finally, as multidimensional cubes often contain many null values, it is necessary to decide how these values will be dealt with.

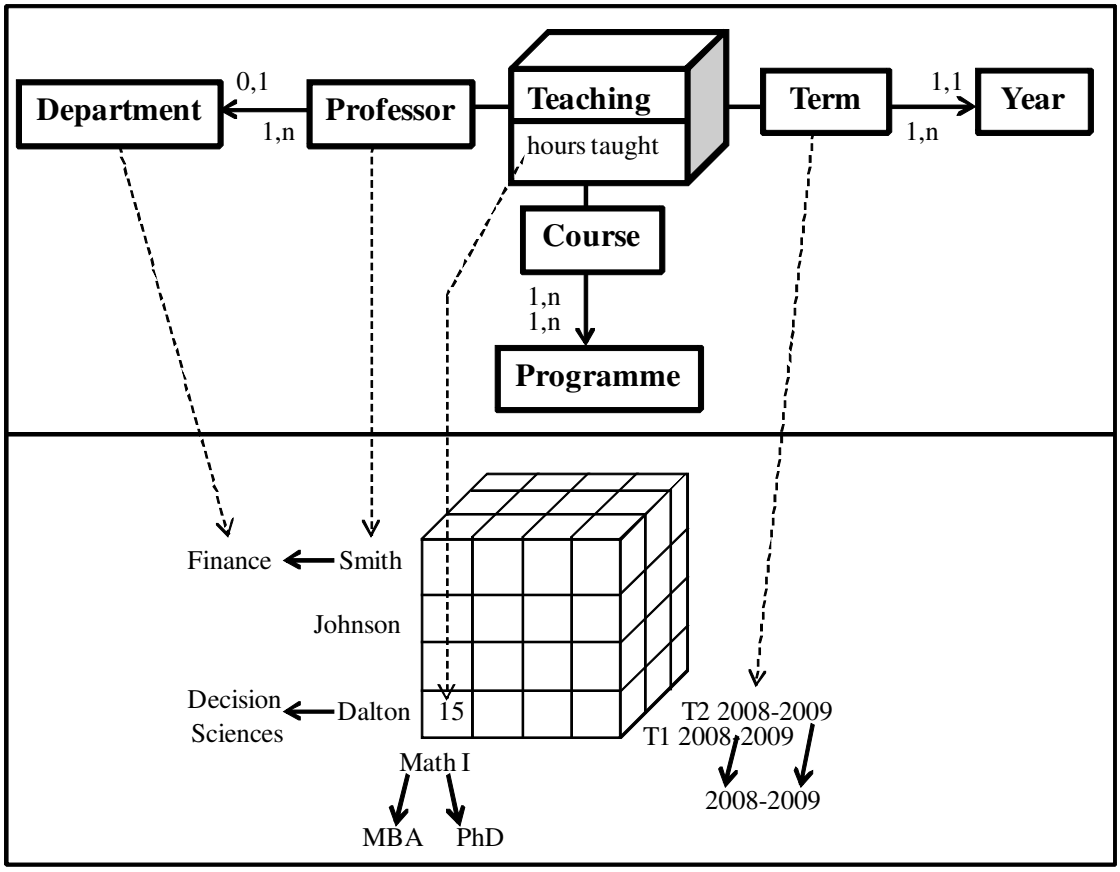

Figure 3. Example scenario: data cube (bottom) and corresponding multidimensional schema (top).

As mentioned previously, the rules pertaining to user preferences may be defined for a specific user or a category of users. In the present case, we will assume that rules R9 and R10 do not apply to the user of the data cube (we will assume that the user is used to handling data cubes and performing rollups). We will also assume that no applicable aggregation functions have been specified in the multidimensional schema (rule R7 does not apply).

The user starts by choosing the axis for aggregation. He chooses the axis of terms (time dimension). Together with the initial data cube and the measure hours_taught, this constitutes the context of aggregation.

By default, the candidate aggregation functions are initially SUM, AVG, MIN, MAX and COUNT. By application of rule R8, AVG is suppressed from the list of candidate aggregation functions. Aggregation is performed along a temporal dimension, but hours_taught is a flow, therefore rule R1 does not apply. Similarly, rule R2 does not apply. Finally, the list of candidate aggregation functions is SUM, MIN, MAX and COUNT. The user chooses SUM. The aggregation (sum of hours taught by year, for each professor and each course) is then executed, using rule R12. The resulting cube is the input (context) for the second aggregation.

For the second aggregation, the user wants to perform aggregation along the axis Course. The candidate aggregations are SUM, MIN, MAX and COUNT (rule R8 is applied again. In our case we have excluded rule R9. However, if it had applied, this would have reduced the list of possible aggregations to the function SUM, applied in the first aggregation). The user chooses to perform a SUM. This sum is implemented by rule R15 (the sum is performed along a non-strict hierarchy, and the measure hours_taught is not divided between the programmes). We now have a data cube with the number of hours taught by professor, programme and year.

For the third aggregation, the user wants to perform aggregation along the axis Professor. The candidate aggregation functions are again SUM, MIN, MAX and COUNT. This time, the user chooses MAX. The aggregation is performed by rule R13. The resulting cube shows, by department, programme and year, the number of hours taught for that programme that year by the professor of the department who has taught most for the programme. The user decides to stop aggregation here. (If he decides to continue, SUM is excluded from the list of possible aggregation functions, by rule R4).

6 CONCLUSION AND FURTHER RESEARCH

The aggregation concept is of central concern in data warehouse design and multidimensional modelling. However, it is generally poorly represented, due to the fact that multidimensional models mainly focus on static knowledge representation. To overcome this limitation, we proposed in this paper to use Production Rule Representation to enrich UML class diagrams with dynamic aggregation knowledge. The literature mentions approaches which mainly represent information on aggregation hierarchies. We go beyond by proposing PRR as a means to incorporate dynamic aggregation knowledge in the multidimensional model. The main contributions of this paper are in i) collecting knowledge on aggregation in the literature, ii) classifying it into semantic rules, syntactic rules, user-preference rules, and execution rules, iii) homogenizing the description of these rules in a unique formalism using the PRR language, and iv) demonstrating the expressive power of the PRR language.

This work presents models representing aggregation knowledge. It offers a typology of aggregation rules, and examples for each type of rule (some of which are illustrated in the present paper). Through the use of UML, all aggregation knowledge (static and dynamic) is nicely integrated. Moreover, beyond the examples given in section 4, aggregation rules can themselves be described by means of a meta-model (similarly to the meta-models presented in section 3 for static aggregation knowledge). We have shown a scenario illustrating the benefit and applicability of our approach. The problems raised by this example are difficult to address if aggregation knowledge is represented only in static models and constraints, as is generally the case in extant approaches. Rules are needed to account for the complex dynamics of aggregation and the fact that this aggregation is context-dependant.

As a further research and beyond the scenario and examples of PRR rules as illustrated in this paper, we plan to perform larger experiments based on a prototype. The next step of our research will consist of the definition of mapping rules to transform PRR aggregation rules into commands that could be executed by an OLAP tool. The typology of aggregation rules can also be refined and used to define rule execution strategies. Finally, sequence diagrams could be used to combine rulesets, as proposed in (Abdullah et al., 2007).

References

Abdullah, M., Benest, I., Paige and R., Kimble, C. (2007). Using Unified Modeling Language for conceptual modeling of Knowledge-Based Systems, Proc. ER 2007, Auckland, New Zealand.

Abello, A., Samos, J., and Saltor, F. (2006) YAM^2: a multidimensional conceptual model extending UML, Information Systems, 31 (6).

Akoka, J., Comyn-Wattiau, I. and Prat, N. (2001). Dimension hierarchies design from UML generalizations and aggregations, Proc. ER 2001, Yokohama, Japan.

Clark, T., Jones, M., and Armstrong, C. (2007). The dynamic structure of management support systems: theory development, research focus, and direction, MIS Quarterly, 31 (3), September.

Espil, M.M. and Vaisman, A. (2003). Revising aggregation hierarchies in OLAP: a rule-based approach, Data & Knowledge Engineering, 45 (2).

Horner, J., Song, I.-Y. and Chen, P. (2004). An analysis of additivity in OLAP systems, Proc. DOLAP'04, Washington, DC, USA.

Horner, J. and Song, I.Y. (2005). A taxonomy of inaccurate summaries and their management in OLAP systems, L. Delcambre et al. (Eds.): ER2005, LNCS 3716, Springer-Verlag, pp. 433-448.

Hurtado, C., Gutierrez, C. and Mendelzon, A. (2005). Capturing summarizability with integrity constraints in OLAP, ACM Transactions on Database Systems, 30 (3), September.

Hüsemann, B., Lechtenbörger, J. and Vossen, G. (2000). Conceptual data warehouse design, Proc. DMDW 2000, Stockholm, Sweden.

Lenz, H.-J. and Shoshani, A. (1997). Summarizability in OLAP and statistical data bases, Proc. SSDBM'97, Olympia, Washington, USA.

Lenz, H.-J. and Thalheim, B. (2001). OLAP databases and aggregation functions, Proc. SSDBM 2001, Fairfax, Virginia, USA.

Lujan-Mora, S., Trujillo, J. and Song, I.-Y. (2006). A UML profile for multidimensional modeling in data warehouses, Data & Knowledge Engineering, 59 (3).

Malinowski, E. and Zimanyi, E. (2006). Hierarchies in a multidimensional model: from conceptual modeling to logical representation, Data & Knowledge Engineering, 59 (2).

March, S. and Hevner, A. (2007). Integrated decision support systems: a data warehousing perspective, Decision Support Systems, 43 (3).

Mazon, J.N., Lechtenbörger, J. and Trujillo, J. (2009). A survey on summarizability issues in multidimensional modeling, Data & Knowledge Engineering, 68 (12).

OMG, 2009. Object Management Group specifications, available at http://www.omg.org/technology/documents/spec_summary.htm. (visited January 10, 2010).

Parssian, A. (2006). Managerial decision support with knowledge of accuracy and completeness of the relational aggregate functions, Decision Support Systems, 42 (3), December.

Pedersen, T. and Jensen, C. (1999). Multidimensional data modeling for complex data, Proc. ICDE'99, Sydney, Australia.

Prat, N., Akoka, J. and Comyn-Wattiau, I. (2006). A UML-based data warehouse design method, Decision Support Systems, 42 (3), December.

Torlone, R. (2003). Conceptual multidimensional models. In M. Rafanelli (ed), Multidimensional databases: problems and solutions, Idea Group.