YAM^2 (Yet Another Multidimensional Model): An extension of UML

Alberto Abelló [(a)], José Samos [(b)], and Fèlix Saltor [(a)]

[(a)] U. Politècnica de Catalunya (UPC), Dept. de Llenguatges i Sistemes Informàtics
{aabello,saltor}@lsi.upc.es
[(b)] U. de Granada (UGR), Dept. de Lenguajes y Sistemas Informáticos
jsamos@ugr.es

Abstract. This paper presents a multidimensional conceptual Object-Oriented model, its structures, integrity constraints and query operations. It has been developed as an extension of UML core metaclasses to facilitate its usage, as well as to avoid the introduction of completely new concepts. YAM^2 allows the representation of several semantically related stars, as well as summarizability and identification constraints.

1 Introduction

A "Data Warehouse" (DW) is roughly a huge repository of data used on the decision making process. To help on the management and study of that enormous quantity of data appeared "On-Line Analytical Processing" (OLAP) tools. The main characteristic of this kind of tools is multidimensionality. They represent data as if these were placed in an n-dimensional space, allowing a study in terms of facts subject of analysis, and dimensions showing the different points of view according to which data can be analyzed.

Several papers appeared in the last years regarding multidimensional modeling. However, few of them place the discussion at a conceptual level. Moreover, most of them focus on the representation of isolated star schemas, i.e. the representation of only a kind of facts surrounded by its analysis dimensions. In spite of the dominant trend in data modeling is the "Object-Oriented" (O-O) paradigm, there exist only a couple of proposals on O-O multidimensional modeling: [TP98] and [NTW00]. These proposals use "Unified Modeling Language" (UML) standard (defined in [OMG99]) in some way, but none of them proposes an extension of it to include multidimensionality. Just the "Common Warehouse Metamodel" (CWM) standard (defined in [OMG01]) extends UML metaclasses to represent some multidimensional concepts. However, it is too general, and not conceived as a conceptual model.

Next section explains the main contributions of our multidimensional model. Then, sections 3, 4, and 5 present its structures, inherent integrity constraints, and operations, respectively. Section 6 shows the metaclasses of the model and their relationships with UML metaclasses. Finally, section 7 compares YAM^2 with other multidimensional models against several items (most of them already introduced by other authors). Conclusions, and references close the paper.

2 Not just another multidimensional data model

As stated in [AHV95], a "database model" provides the means for specifying particular data structures, for constraining the data sets associated with these structures, and for manipulating the data. It is also explained there that, as relations are the data structures of the Relational model, so graphs are the structures of O-O models. We provide a precise, easily understandable semantics for graphs in our O-O model, by defining YAM^2 structures as an extension of a wide accepted modeling language, i.e. UML (each and every YAM^2 metaclass is a subclass of a UML metaclass). There are some multidimensional models that use UML notation, but no one extends its concepts for multidimensional purposes. By using UML as a base for the definition of structures of YAM^2, we build our model on solid, well accepted foundations, and avoid the definition and exemplification of basic concepts. It makes unnecessary to explain what classes, attributes, etc. are.

The main goal of multidimensionality is to help non-expert users to query data. Therefore, the data structures of a multidimensional model should show how data can be accessed, driving users in their understanding. They should keep as much information as possible, but the resulting schema must be easily understandable by final users. Thus, the different modeling elements in YAM^2 have been defined at three levels (i.e. upper, intermediate, and lower), so that they are successively decomposed to give the desired detail.

"Expressiveness" or "Semantic Power", as it is defined in [SCGS91], is the degree to which a model can express or represent a conception of the real world. It measures the power of the elements of the model to represent conceptual structures, and to be interpreted as such conceptual structures. The most expressive a model is, the better it represents the real world, and the more information about the data gives to the user. This is crucial for conceptual models like YAM^2, since they are used to represent user ideas. Therefore, we will define different kinds of nodes and arcs in the graphs to improve the "Expressiveness" of our model. The applicability of the different kinds of relationships supported by UML has been systematically studied.

Another important point for a data model is its "Semantic Relativism". It is defined in [SCGS91] as the degree to which the model can accommodate not only one, but many different conceptions. It is really important because since different persons perceive and conceive the world in different ways, the data model should be able to capture all of them. The information kept in the DW should be shown to users in the form they expect to see it, independently of how it was previously conceived or is actually stored. Therefore, YAM^2 also provides mechanisms (derivation relationships at different detail levels) to model the same data from different points of view.

Our model also pays special attention to show how data can be classified and grouped in a manner appropriate for subsequent summarization. Summarized data can be reflected in the schema, as well as the ways to obtain it. For instance, this information can be used at later design phases to decide materialization.

In section 7, we compare YAM^2 with other models to show its advantages and disadvantages. There, contributions of our model can be clearly seen, regarding specific items.

3 Structures

In this section, we define the structures in our O-O model (i.e. nodes and arcs).

3.1 Nodes

Multidimensional models are based on the duality Fact-Dimensions. Intuitively, a "Fact" represents data subject of analysis, and "Dimensions" show different points of view we can use in analysis tasks. "Facts" represent measurements (in a general sense), while "Dimensions" represent given information we already have before taking the measurements (on the understanding that they can always be modified). As previous work for the definition of this model, we separately studied "Dimensions" and "Facts" in [ASS01b] and [ASS01c], respectively. The reader is referred to them for an specific, deeper explanation of each of both kinds of data. Now we are going to give the definition of the different nodes we find in a multidimensional O-O schema.

Definition 1. *A* Level *represents the set of instances of the same granularity in an analysis dimension. It is an specialization of* Class *UML metaclass.*

Definition 2. *A* Descriptor *is an attribute of a* Level*, used to select its instances. It is an specialization of* Attribute *UML metaclass.*

Definition 3. *A* Dimension *is a connected, directed graph representing a point of view on analyzing data. Every vertex in the graph corresponds to a* Level*, and an edge reflects that every instance at target* Level *decomposes into a collection of instances of source* Level *(i.e. edges reflect part-whole relationships between instances of* Level*s). It is an specialization of* Classifier *UML metaclass.*

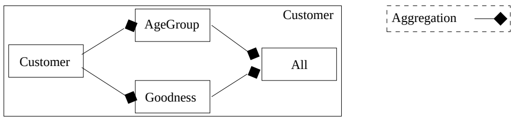

Fig. 1. Example of analysis dimension

Figure 1 shows an example of *Dimension*. It contains four *Level*s: `Customer`, `AgeGroup`, `Goodness`, and `All`. Every instance of `Customer` *Level* represents a customer, which can be aggregated in two different ways to obtain either age or goodness groups of customers. At top we have `All` level with exactly one instance representing the group of all customers in the *Dimension*. The structure of *Dimension*'s graphs and their properties were carefully explained in [ASS01b].

Just to note here that it forms a lattice, and due to the transitive property of part-whole relationships, some arcs are redundant, so that they do not need to be explicited (for instance, `Customer` being aggregated into `All`).

Definition 4. *A* Cell *represents the set of instances of a given kind of fact measured at the same granularity for each of its analysis dimensions. It is an specialization of* Class *UML metaclass.*

Definition 5. *A* Measure *is an attribute of a* Cell *representing measured data to be analyzed. Thus, each instance of* Cell *contains a (possibly empty) set of measurements. It is an specialization of* Attribute *UML metaclass.*

Definition 6. *A* Fact *is a connected, directed graph representing a subject of analysis. Every vertex in the graph corresponds to a* Cell*, and an edge reflects that every instance at target* Cell *decomposes into a collection of instances of source* Cell *(i.e. edges reflect part-whole relationships between instances of* Cell*s). It is an specialization of* Classifier *UML metaclass.*

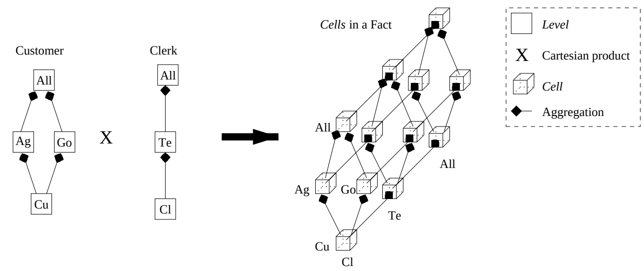

Fig. 2. Graph of *Cell*s in a *Fact* with two *Dimensions*

Figure 2 shows an example of the structure of a *Fact* with two *Dimension*s: `Customer`, already depicted in figure 1; and `Clerk`, composed by `Clerk`, `Team`, and `All` *Level*s. We can see that there is a *Cell* in the *Fact* for every combination of *Level*s in the *Dimension*s. Thus, a *Fact* contains all data regarding the same subject at any granularity. Having two *Dimension*s with 4 and 3 *Level*s respectively, means that the *Fact* will have 12 different *Cell*s. These *Cell*s and the part-whole relationships between them form a lattice, as was already explained in [ASS01c].

These six kinds of nodes are grouped in three pairs. At upper detail level, we have *Fact*s and *Dimension*s (one *Fact* and the *Dimension*s associated to it compose a *Star*). At intermediate level, there are *Cell*s and *Level*s. Finally, looking at lower detail we see *Measure*s and *Descriptor*s. Moreover, at this level, we also define *KindOfMeasure* to show that several *Measure*s in different *Cell*s correspond to the same measured concept at different aggregation levels.

3.2 Arcs

Once the nodes have been defined, in this section we are going to see the different kinds of arcs we could find between them. UML provides four different kinds of

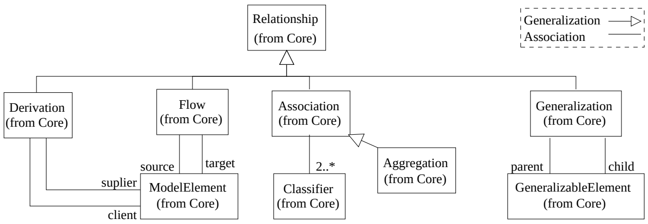

Fig. 3. UML Relationships between model elements

relationships: *Generalization*, *Flow*, *Association*, and *Dependency*. As depicted in figure 3, *Generalization* relationships relate two *GeneralizableElement*s, one with a more specific meaning than the other. *Classifier*s and *Association*s are *GeneralizableElements*. *Flow* relationships relate two elements in the model, so that both represent different versions of the same thing. *Association*, as defined in UML specification, defines a semantic relationship between two *Classifiers*. By means of a stereotype of *AssociationEnd*, UML allows to use a stronger type of *Association* (i.e. *Aggregation*), where one classifier represents parts of the other. If parts cannot be shared by different wholes, we have a stronger form of *Aggregation* known as *Composition*. Both kinds of *Aggregation* show part-whole relationships, so we will not distinguish them in the study, but only in some diagrams. Finally, UML allows to represent different kinds of *Dependency* relationships between *ModelElement*s like *Binding*, *Usage*, *Permission*, or *Abstraction*. We are not going to consider the three first, because they are rather used on application modeling, and YAM^2 is just a data model. Moreover, due to the same reason, out of the different stereotypes of *Abstraction* we are only going to use *Derivation*. Derivability, also known as "Point of View", helps to represent the relationships between model elements in different conceptions of the UoD.

The usability of these relationships between concepts was briefly explained and exemplified in [ASS01a]. Here we are systematically going to see how they can be used to relate multidimensional constructs at every detail level. For every pair of constructs at each detail level we will show if they can be related by a given kind of *Relationship* or not. Moreover, if two constructs can be related, we will also show if they must belong to the same construct at the level above, or not (i.e. inter or intra relationships, respectively).

	Fact-Fact	*Fact-Dimension*	*Dimension-Fact*	*Dimension-Dimension*
Generalization	Inter	-	-	Intra/Inter
Association	Inter	Intra/Inter	Intra/Inter	Intra/Inter
Aggregation	Inter	-	-	Intra/Inter
Flow	Inter	-	-	Intra/Inter
Derivation	Inter	Intra/Inter	-	Intra/Inter

Table 1. Relationships between elements at upper detail level

Upper detail level Table 1 shows the different relationships we can find at this detail level. Since a *Star* only contains one *Fact*, in order to have two related *Fact*s, they must belong to different *Star*s. Therefore, relationships between

Facts will always be inter-stellar. However, we can have inter-stellar as well as intra-stellar relationships between two *Dimensions*, because a *Star* contains several *Dimensions*, which can be related.

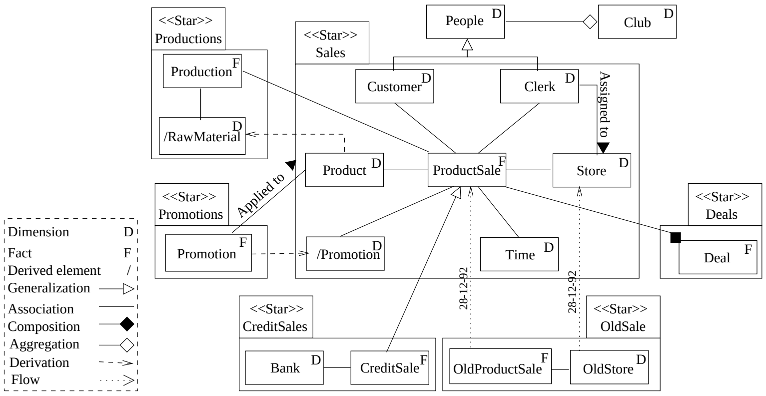

Fig. 4. Example of YAM^2 schema at upper detail level

Figure 4 shows examples of most relationships at this level. Firstly, corresponding to the upper-left corner of the table, we see that two *Facts* can be related by *Generalization* (i.e., `ProductSale` and `CreditSale`). We will have different information for the more specific *Fact* (for example, number of credit card). Thus, analysis dimensions are inherited from the more general *Fact*, but others could be added, like `Bank`. `ProductSale` and `Production` are related by *Association* to show the correspondences between produced and sold items. We can also find *Aggregation* relationships between *Facts*. A *Fact* in a *Star* can be composed by *Facts* in another *Star*. For instance, a `Deal` is composed by several individual `ProductSale`. Notice that it is not always possible to calculate all measurements of `Deal` from those of `ProductSale` (for instance, discount in the deal). Data sources, measure instruments, or calculation algorithms are probably going to change, and these changes should be reflected in our model by means of *Flow* relationships between *Facts*. All these changes are not reflected by just relating our *Facts* to `Time` *Dimension*, since we actually have different *Cells*. On `December 28th of 1992`, we started recording discount checks in `ProductSale`, so that we kept both incomes (i.e. cash, and discount checks). From that day, we have different *Facts* containing the same kind of data before and after the acceptance of the checks (i.e. `OldProductSale`, and `ProductSale`). Finally, two *Facts* could also be related by *Derivation* relationships to show that they are the same concept from different points of view.

In the upper-right corner of table 1, we can see that there exist *Generalization* relationships between *Dimensions*. For instance, `People` *Dimension* generalizes `Clerk` and `Customer` ones. Notice that if we suppose that all people is a customer, both related *Dimensions* would belong to the same *Star*. It is also possible to have analysis dimensions related by *Association*. Thus, `Clerk` is associated with `Store`

Dimension to show that clerks are assigned to stores. We can also find stronger associations between analysis dimensions, if we join more than one to give rise to another. For example, `People` *Dimension* is used to define `Clubs` by means of an *Aggregation* relationship. Every instance of `Clubs` is composed by a set of people. Several years ago, when our local business grew, `Store` *Dimension* was changed to reflect the new *Level* `Region`. At conceptual level, those changes are represented by a *Flow* relationship between `OldStore` and `Store`. *Derivation*s allow to state that there are different views of the same *Dimension*. We could find that the same concept has different names depending on the subject we are. Thus, a *Dimension* could be used in different *Stars*. For example, `Product` is considered `RawMaterial` in a different context. Therefore, the same *Dimension*, with exactly the same instances, needs a different name depending on the context. These *Dimension*s could even have different aggregation hierarchies or attributes of interest to the users. For example, studying the raw material grouped by profit margin can be meaningless.

The middle columns in table 1 show how a *Fact* can be related to a *Dimension* and vice versa. Firstly, we see that a *Fact* is related to its analysis dimensions by means of *Association* relationships. Moreover, they can also be associated to *Fact*s in another *Star* as shown in the example, where `Promotion` *Fact* is associated to `Product` *Dimension* in the `Sales` *Star*. A *Dimension* can be obtained by deriving it from a *Fact*. The name can be changed, some aggregation levels added or removed, others modified, some instances selected, etc. in order to adapt it to its new usage. In our example, some people is interested in the analysis of promotions. Thus, the promotions selected by studying `Promotion` *Fact*, can be used as *Dimension* to study `ProductSale`. Notice the difference between deriving a *Dimension* and associating it to a *Fact* in another *Star*. The former allows to study the sales performed during a promotion, while the latter shows all promotions that have been applied to a kind of product. That *Derivation* between a *Fact* and a *Dimension* uses to be an inter-stellar relationship (i.e. from a *Fact*, we derive a *Dimension* to analyze another *Fact*). However, we could also use information derived from a *Fact* to analyze the same *Fact*. It is also important to say that a *Fact* cannot be derived from a *Dimension*, because *Fact*s represent measurements, so that they cannot be found a priori in the form of *Dimension*. The rest of relationships (i.e. *Generalization*, *Aggregation*, and *Flow*) cannot be found between a *Fact* and a *Dimension*, nor vice versa. All three imply obtaining a new element based on a preexisting one, and the difference between *Fact* and *Dimension* is so important that the obtaining of one from the other should be restricted to derivation mechanisms. For instance, a *Fact* cannot eventually become a *Dimension*.

Intermediate detail level Table 2 shows the relationships we can find at this level. Most of them are exemplified in figure 5. Our company (resulting from the fusion of preexisting smaller companies) is organized in autonomous regions. Thus, the information systems in one of these regions collect data that those in other regions do not, so we specialize our *Cell*s (i.e. `AtomicSale`) depending

	Cell-Cell	*Cell-Level*	*Level-Cell*	*Level-Level*
Generalization	Inter	-	-	Inter
Association	Intra/Inter	Inter	Inter	Intra/Inter
Aggregation	Intra/Inter	-	-	Intra/Inter
Flow	Inter	-	-	Inter
Derivation	Inter	Inter	-	Inter

Table 2. Relationships between elements at intermediate detail level

on the region. This specialization is due to the specialization of the kind of fact they are representing. Therefore, we can see in the upper-left corner of the table that two *Cell*s can be related by *Generalization*, but they must belong to different *Fact*s (i.e. it is an inter-factual relationship). *Cell*s in different *Fact*s can be associated (for instance, each *Cell* representing a sale with its corresponding *Cell* representing the production of what was sold). Moreover, we can also have *Association* relationships between *Cell*s in the same *Fact* (for instance, computers are associated to those other products that are plugged to them). In general, we only have intra-factual *Aggregation* relationships, which correspond to those relationships between *Level*s, and are not necessary in the schema. However, we could also find that different *Cell*s are aggregated to obtain a *Cell* about a different kind of fact (when both *Fact*s are also related like `ProductSale` and `Deal`). In this case, we do not group *Cell*s along any analysis dimension, i.e. it does not generate coarser *Cell*s in the same *Fact*, but *Cell*s in another *Fact* (i.e. `AtomicDeal`). If a new *Measure* would appear for a kind of fact, we would obtain a new *Cell* related to the old one by means of a *Flow*. Both would represent the same concept. However, they would belong to different versions of the same *Fact* (it is a inter-factual relationship). *Derivation* relationships can be used to hide information, change names, or *Measure*s in the *Cell*s, giving rise to new *Fact*s.

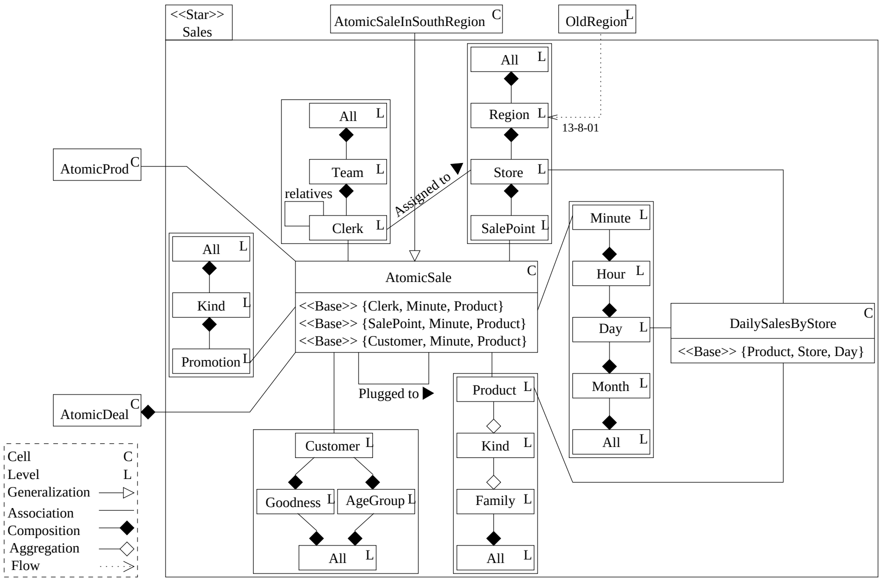

Fig. 5. Example of YAM^2 schema at intermediate detail level

The rightmost column shows that we could also find *Generalization* relationships between two *Levels*. As in the case of *Cells*, it must be an inter-dimensional relationship, because both *Levels* cannot be related, at the same time, by *Generalization* and part-whole relationships. *Associations* between *Levels* can be intra- as well as inter-dimensional. The *Level* representing clerks is associated with other clerks (his/her relatives) in the same *Dimension*, and with stores in another *Dimension*. Intra-dimensional *Aggregations* define the graph of the *Dimension*. However, we could also find inter-dimensional *Aggregations* between *Levels*, if two *Dimensions* are so related. When the company was restructured and the regional division changed, the aggregation level showing it also changed. Both, new and old *Levels* are related by means of a *Flow* (although they represent the same concept, they belong to different versions of the same *Dimension*). Finally, as for any other concept, a *Level* could be derived from another one to show it from a different point of view.

All relationships in the central columns must be inter-structure, because *Cells* and *Levels* always belong to different structures (i.e. *Facts* and *Dimensions*, respectively). As for relationships at upper detail level, a *Cell* cannot be converted into a *Level* nor vice versa by means of *Generalization*, *Aggregation*, or *Flow*. It must always be done using derivation mechanisms. Moreover, because of the same reason that a *Fact* cannot be derived from a *Dimension*, a *Cell* cannot be derived from a *Level*. Nevertheless, if a *Dimension* is derived from a *Fact*, its *Levels* are also derived from the *Cells* of the *Fact*. *Associations* exist between *Cells* and *Levels*, or vice versa (showing the granularity of the *Cells*).

	Measure-Measure	*Measure-Descriptor*	*Descriptor-Measure*	*Descriptor-Descriptor*
Flow	Inter	-	-	Inter
Derivation	Intra/Inter	Inter	Inter	Intra/Inter

Table 3. Relationships between elements at lower detail level

Lower detail level Elements at this level are neither *Classifiers* nor *GeneralizableElements*, but just *Attributes*. Therefore, as it is shown in table 3, they can only be related by those relationships between *ModelElement* (i.e. *Derivation*, and *Flow*).

If a change affects a *Measure* or *Descriptor*, they will belong to new versions of their *Cell* and *Level*, respectively. Thus, *Flow* relationships are in both cases inter-structure. Moreover, time cannot convert a *Measure* into a *Descriptor*, nor vice versa.

It is always possible to define derived *Measures* from other *Measures* in the same *Cell*, as well as *Descriptors* from other *Descriptors* in the same *Level*. Moreover, in both cases, supplier *Attributes* could also be in other *Classes*. *Measures* in a *Cell* could be obtained by applying some operation to *Measures* in other *Cells*. For instance, looking to lower detail level elements in figure 6, we see that measurements of `revenue` in `AtomicSale` are obtained from subtracting `cost` in `Production`. What is more, a *Descriptor* can be obtained from some *Measures*

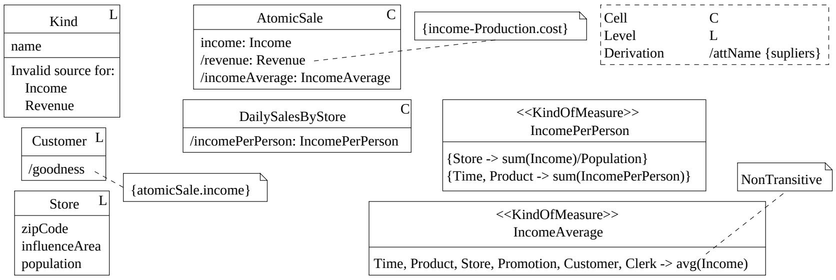

Fig. 6. Example of YAM^2 schema at lower detail level

(for example, the goodness of a customer from the income of his/her purchases), or vice versa (for example, impact of sales obtained by dividing incomes by the population of the influence area of the store). This figure does not show arcs between *Cell*s and *Level*s, because they are at intermediate detail level.

4 Inherent integrity constraints

The metaclasses of the model define constraints on multidimensional schemas, but constraints should also be defined on their instances. In this section, we are going to address that kind of constraints, paying special attention to two important points in multidimensional modeling, namely visualization of data in an n-dimensional space, and summarizability of data.

The main contribution of multidimensionality is the placement of data in an n-dimensional space. It improves the understanding of those data and allows the implementation of specific storage techniques. From our point of view, it is important that the n dimensions of the space are orthogonal. If not, i.e. if a dimension determines others, the visualization of data will be unnecessarily complicated (we are showing more information that it is needed and it will be more difficult for users to understand it); moreover, storage mechanisms are affected, as well, because they are not considering that several combinations of dimension values are impossible, resulting in a waste of space.

A *Cell* instance is related to one object or set of objects (if it is an *Association* with upper-bound multiplicity greater than one) at each associated analysis dimension, and those objects or sets of objects completely identify it. Thus, regarding visualization of data in n-dimensional spaces, we could say that the set of *Levels* a *Cell* is associated with form a "superkey" (in Relational terms) of that *Cell*. We call *Base* to every minimal set of *Levels* being "superkey" (i.e. "key" in the Relational model) of a *Cell*. When one of these *Base*s (that define spaces of orthogonal dimensions) is associated to a *Cell*, we obtain a *Cube*. For instance, `AtomicSale` (in figure 5) can be associated with points in the 3-dimensional space defined by *Level*s `Clerk`, `Minute`, and `Product`, so that `AtomicSale` is functionally determined by those three *Level*s (a *Base* of the space).

Definition 7. *A* Cube *is an injective function from an n-dimensional finite space (defined by the cartesian product of n functionally independent* Level*s*

$\{L_1, .., L_n\}$), *to the set of instances of a* Cell (C_c).

$$c : L_1 \times .. \times L_n \rightarrow C_c, \ injective$$

If the *Levels* were not functionally independent (i.e. they did not form a *Base*), we would use more *Dimensions* that strictly needed to represent the data, and would generate empty meaningless zones in the space.

Another interesting group of constraints to deal with is that related to summarization anomalies and how to solve (or prevent) them. In multidimensional modeling, it is essential to know how a given kind of measure must be aggregated to obtain it at a coarser granularity. [LS97] identifies three necessary (intuitively also sufficient) conditions for summarizability:

1. Disjointness: subsets of objects must be disjoint.
2. Completeness: the union of subsets must constitute the entire set.
3. Compatibility: category attribute (*Level*), summary attribute (*KindOfMeasure*), and statistical function (*Summarization*) must be compatible.

The first two conditions are absolutely dependent on constraints over cardinalities in the part-whole relationships of the *Dimensions*, because these define the grouping categories. Therefore, let us briefly talk also about this third group of integrity constraints of our model.

To avoid those anomalies on summarizing data, some models forbid "to-many" relationships in the aggregation hierarchies. This means that instances of a *Part Level* can only belong to one *Whole*. Nevertheless, there is no mereological axiom forbidding the sharing of parts among several wholes. A given product `Kinder Surprise` (at *Level* `Product`) belongs to two different kinds of products at the same *Level* `Kind` (i.e. `Candies`, and `Toys`). We argue that this case should not be ignored by a multidimensional model. Therefore, non-strict hierarchies are allowed in the *Dimensions*, and they need to be taken into account to decide summarizability of *Measures*.

The other problem on cardinalities is that of "non-onto" and "non-covering" hierarchies (as presented is [Ped00]). That is, having different part-whole structures for instances at the same *Level* is allowed. For example, if we would have a state-city (like `Monaco` in a `Geographic` linear *Dimension* with *Levels* `City`, `State`, and `All`), we could generate both situations. If we consider that `Monaco` is a city, we have a "non-covering" hierarchy (we are skipping `State` level). On the other hand, if it is considered a state, we obtain a "non-onto" hierarchy (we have different path lengths from the root to the leaves depending on the instances). In this case, we propose the usage of what some authors call "Dummy Values" to guarantee the existence of at least one part for every whole in the hierarchy. These values are not dummy at all. `Monaco` being a state-city does not mean it is either a state or a city, but a state and a city at the same time. Thus, both instances will represent city and state facets of the same entity.

Therefore, in *YAM*^{2}, cardinalities in aggregation hierarchies are "1..*" parts for every whole, and "*" wholes for every part, on the understanding that *Dimension* instances can always be defined so that there are "1..*" wholes for every part. Please, refer to [ASS01b] for a deeper explanation of these cardinalities.

Going back to the group of constraints regarding summarizability, in our model, there are three different elements to deal with that problem (all exemplified in figure 6). These elements allow to represent summarizability conditions in a more flexible way than just distinguishing "additive", "semi-additive", and "non-additive" measures. Firstly, we have that some *Level*s are an *InvalidSource* for the calculation of a given *KindOfMeasure* (for example, `Kind` is an invalid source for `Income` and `Revenue`). It means that measurements at an aggregation level cannot be used to obtain data at higher aggregation levels. This can be due to the instances of that *Level* are not disjoint or not complete (i.e. summarizability conditions 1 and 2 mentioned above). A *Level* being invalid or not cannot be deduced just from the cardinalities of its associations, but also depends on the *KindOfMeasure*. For instance, if a *Measure* is obtained as the minimum of a set of measurements, it does not matter whether the source sets of instances are disjoint or not. In some cases, double counting could even be desirable.

Moreover, *Induce Association* shows the summarization that must be performed on aggregating a given *KindOfMeasure* along a *Dimension*. This constraint regards the third condition mentioned above. Along a given analysis dimension we can use a summarization operation, while along a different analysis dimension we use a different function. For instance, we aggregate `IncomePerPerson` along `Time` and `Product` by means of sum, while along `Store` it needs to be recalculated from `Incomes`. Incompatibilities are not always associated to `Time` *Dimension*. Furthermore, inductions could be partially ordered, if necessary, to show that operations are not commutative, and must be performed in a given order, as pointed out in [Tho97]. For example, sums along a *Dimension* must be performed before averages along another one, so that, we aggregate up to the desired *Level* in a *Dimension*, and then we aggregate along the other.

Finally, another point to take into account, usually forgotten in other models, is that of transitivity. If a summarization operation is not transitive, we cannot use precalculated aggregates at a given *Level* to obtain those at higher levels. Going to the atomic source is mandatory (for instance, we should not perform the average of averages, if we want to obtain the average of raw data).

5 Operations

The multidimensional model is just a query model, i.e. it does not need operations for update, since this is not directly performed by final users. YAM^2 operations focus on identifying and uniformly manipulating sets of data, namely *Cube*s. In a *Cube*, data are identified by their properties. Thus, these operations are separated from the physical storage of the data.

Detail level	Subject of analysis	Point of view
Upper	Drill-across	ChangeBase
Intermediate	-	Roll-up
Lower	Projection	Selection

Table 4. YAM^2 operations

As everything in a multidimensional model, operations are also marked by the duality Fact-Dimensions. Table 4 shows the operations in two columns. The first one contains those operations having effect on the subject of analysis (i.e. *Fact*, *Cell*, and *Measure*). They select the part of the schema we want to see. In the other column, there are those operations affecting the point of view we will use in the analysis (i.e. *Dimension*, *Level*, and *Descriptor*). They allow to reorganize the data, modify their granularity, and focus on a specific subset, by selecting the instances we want to see.

$$\begin{tikzcd}
L^i_1 \, x \, .. \, x \, L^i_k \, x \, .. \, x \, L^i_m \arrow[r, "c_i"] \arrow[d, "h"] & C^i_c \arrow[d, bend left, "g"] \\
L^o_1 \, x \, .. \, x \, L^o_k \, x \, .. \, x \, L^o_n \arrow[r, "c_o"] \arrow[u, bend left, "f"] & C^o_c
\end{tikzcd}$$

Fig. 7. Operations as composition of functions

In the sense of [AHV95], these operations are conceptually a "procedural language", because queries are specified by a sequence of operations that construct the answer. We generally say that a query is from (or over) its input schema to its output schema. Thus, there exists an input m-dimensional *Cube* (c_i), and we want to obtain an output n-dimensional *Cube* (c_o). Since, we defined a *Cube* (see definition 7) as a function, operations must transform a function into another function. Operations in the first column work on the image of the function, while operations in the second column change its domain. Therefore, as depicted in figure 7, we have three families of functions (i.e. f, g, and h), that can be used to transform a *Cube*.

Drill-across: This operation changes the image set of the *Cube* by means of a bijective function ψ of the family g (relationships in section 3.2 can be used for this purpose). This function relates instances of a *Fact* to instances of another one. $c_o(x) = \delta_\psi(c_i) = \psi(c_i(x))$

Projection: This just selects a subset of *Measure*s from those available in the selected *Cell*. $c_o(x) = \pi_{m_1,..,m_k}(c_i) = c_i(x)[m_1, .., m_k]$

ChangeBase: This operations changes the domain set of the *Cube* by means of a bijective function ϕ of the family f (i.e. ϕ relates points in an n-dimensional finite space to points in an m-dimensional finite space). Thus, it actually modifies the analysis dimensions used. $c_o(x) = \gamma_\phi(c_i) = c_i(\phi(x))$

Roll-up: It modifies the granularity of data, by means of an exhaustive function φ of the family h (i.e. φ relates instances of two levels in the same *Dimension*, corresponding to a part-whole relationship). $c_o(x) = \rho_\varphi(c_i) = \bigcup\limits_{\varphi(y)=x} c_i(y)$

Dice: By means of a predicate P over *Descriptor*s, this operation allows to choose the subset of points of interest out of the whole n-dimensional space. $c_o(x) = \sigma_P(c_i) = \begin{cases} c_i(x) & if\ P(x) \\ undef & if\ \neg P(x) \end{cases}$

It is clear that there is one operation missing, which would allow to select the *Cell* we want to query in the same way we choose *Measure*s or *Fact*s. However, the specific *Cell* we analyze cannot be selected by itself, but it is absolutely determined by the selected aggregation levels in every *Dimension*.

If we want to know the production cost of every product sold under a given promotion, by month and plant, we should perform the following operations over our `AtomicSale` schema: 1) *Dice* to select promotion "A", 2) *Drill-across* to "Production" *Fact*, 3) *Projection* to see just the desired *Measure* "cost", 4) *Roll-up* to obtain data at "Month" *Level* (notice that summarization operation is not explicited, because a *YAM*² schema shows how a given *KindOfMeasure* must be summarized along each *Dimension*), and finally 5) *ChangeBase* to choose the appropriate n-dimensional space to place data.

$$\gamma_{Month \times Plant \times Product}(\rho_{Month}(\pi_{cost}(\delta_{Production}(\sigma_{Promotion="A"}(AtomicSale)))))$$

6 Metaclasses

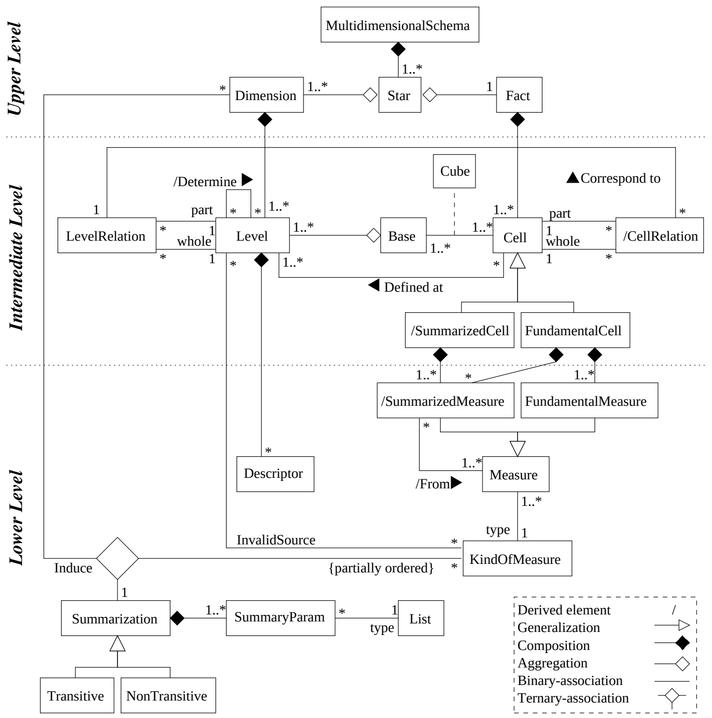

Fig. 8. *YAM*² metaclasses in UML notation (as in [OMG99])

As it was defined in [Inm96], a DW is a subject-oriented set of data. When analysts want to study a given subject, they want to see together all data regarding it. Thus, we propose a subject-oriented model, where all classes related to a subject are shown together in the multidimensional schema. For this purpose we use the upper detail level which, as depicted in figure 8, shows that a *Star* is composed by one *Fact* and several *Dimensions*. Subject-oriented does not imply subject-isolated. Therefore, relationships between different *Star*s will exist, as it was shown in section 3.2.

At the intermediate detail level, we can see that *Dimension*s are composed by *Level*s related by *LevelRelation*s, representing part-whole relationships. Hence, a *Dimension* is a lattice stating how measured data can be aggregated. On the other hand, we see that a *Fact* is composed by a set of *Cell*s. Each of those *Cell*s is defined at an aggregation level for each of the analysis dimensions of its *Fact*. If there is a *Level* (l_2) whose elements are obtained by grouping those of another *Level* (l_1) at which a *Cell* (c_1) is defined, then we have another *Cell* (c_2) related to l_2 whose instances are composed by those of c_1. *Cell*s c_1 and c_2 are related by a *CellRelation*, which corresponds to the *LevelRelation* between l_1 and l_2. A set of functionally independent *Level*s form a *Base*, and the pair *Base-Cell* (where the *Base* fully determines instances of the *Cell*) is a *Cube*.

Some data must be physically stored while other will or could be derived. In the same way, some model elements must be explicited in the schema, while other (for instance, *CellRelation*) can be derived. In this sense, we distinguish those *Cell*s that need to be explicited (i.e. *FundamentalCell*s), from those that do not (i.e. *SummarizedCell*s), because all data they contain can be derived.

At lower detail level, we can see information regarding the attributes of the concepts we are representing. The *Level*s contain *Descriptor*s, and the *Cell*s contain *Measure*s. *SummarizedCell*s only contain data that can be derived (i.e. *SummarizedMeasure*s), and *FundamentalCell*s can contain derived or not derived data. *SummarizedMeasure*s are obtained from other *Measure*s, while *FundamentalMeasure*s are not. Notice that it is possible to obtain one *Measure* from more than one supplier (for instance, to be able to weigh an average).

Every *Dimension* induces a *Summarization* over a given *KindOfMeasure*. In general, *SummarizedMeasure*s are obtained by sum of other. However, this is not always the case, product, minimum, maximum, average, or any other operation could be used. It depends on the *KindOfMeasure* and the *Dimension* along which we are summarizing ([LS97] studies the influence of the `Time` dimension on three different kinds of attributes). Thus, when we want to obtain a *SummarizedMeasure* in a *Cell* (c_1), from a *Measure* in another *Cell* (c_2), the *Summarization* performed is that induced by the *Dimension* that contains the *LevelRelation* to which the *CellRelation* between c_1 and c_2 corresponds.

*Summarization*s over a *KindOfMeasure* are partially ordered to state that some must be performed before others. Moreover, some data at an aggregation level could be an invalid source to summarize some *KindOfMeasure*s, which is also captured in the schema. A summarization operation being non-transitive, implies that any summarization that uses it must be done from the atomic data.

Figure 9 shows how all these multidimensional concepts perfectly fit into UML. A *Star* is a *Package* that contains a subject of analysis. *Fact*s and *Dimensions* are *Classifier*s containing *Class*es (i.e. *Cell*s, and *Level*s respectively). Finally, *Measure* and *Descriptor* are just *Attribute*s of the *Class*es. All other

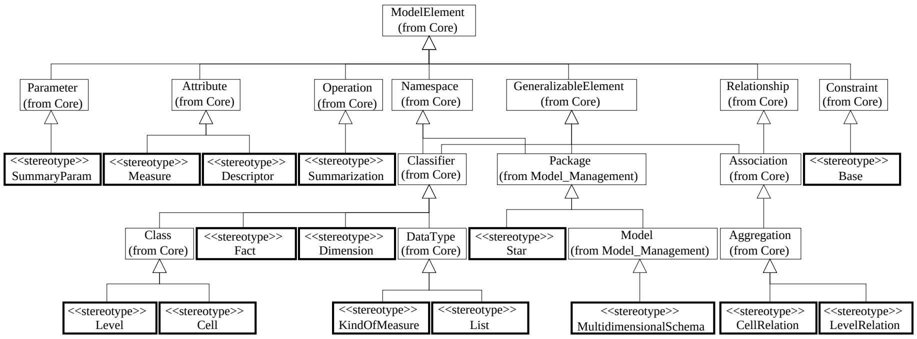

Fig. 9. Extension of UML with *YAM*^{2} stereotypes

elements in *YAM*^{2} have also been placed as specialization of a UML concept. Maybe, the most relevant ones are *CellRelation* and *LevelRelation* that are *Aggregation*s. Moreover, a *Base* is just a *Constraint* stating that a set of functionally independent *Level*s fully determine instances of a *Cell*.

7 Related work

Some O-O multidimensional models have already been defined, and some of them used UML syntax to do it. However, to the best of our knowledge, this is the first extension of UML for multidimensional modeling. As previously said, CWM does extend UML. Nevertheless, it is not a multidimensional data model, but a metadata standard for data warehousing.

	Metamodel		Structures													Constraints						Operations				
Reference	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
[Kim96]	NL	Rel.	✓	-	✓	✓	✓	-	-	p	-	-	p	-	-	-	-	-	-	p	-	-	Headers	✓	-	p
[LW96]	Maths	Rel.	✓	p	✓	✓	-	-	-	p	-	-	-	-	-	-	-	-	-	-	-	A	Relations	✓	✓	-
[AGS97]	Maths	-	-	p	✓	-	p	-	✓	-	-	-	-	-	-	-	-	p	-	-	-	A	Cubes	✓	✓	-
[HS97]	Maths	DL	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	p	p	p	-	-	-	A	Cubes	-	✓	-
[GL97]	Maths	Rel.	✓	-	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	A & C	Cubes	✓	✓	-
[DT97]	Maths	-	✓	-	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	p	-	-	-	A	Cubes	✓	✓	-
[CT98]	Maths	-	✓	✓	✓	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	A & C	Cubes	✓	✓	-
[Leh98]	NL	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	A	Cubes	-	-	-
[GMR98]	NL	-	✓	✓	✓	✓	✓	-	-	✓	-	-	-	-	-	-	-	-	-	✓	-	QL	Cubes	-	-	p
[TP98]	NL	O-O	✓	✓	✓	✓	✓	-	-	✓	p	p	-	p	-	-	-	✓	✓	✓	-	-	Cubes	-	-	-
[SBHD99]	E/R	E/R	✓	✓	✓	✓	✓	-	-	✓	p	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[TBC99]	NL	E/R	✓	✓	✓	✓	✓	-	p	-	p	p	-	-	-	-	-	✓	✓	p	-	-	-	-	-	-
[NTW00]	UML	-	-	✓	✓	✓	✓	-	-	✓	-	-	-	-	-	-	-	✓	-	-	-	-	Cubes	-	-	-
[Vas00]	Maths	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	Cubes	-	-	-
[Ped00]	Maths	-	✓	✓	✓	-	✓	p	✓	-	-	-	✓	-	✓	✓	✓	✓	✓	✓	-	A	Cubes	-	-	-
[TKS01]	NL	-	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	✓	-	-
YAM^2	UML	UML	✓	✓	✓	✓	✓	✓	p	✓	✓	✓	✓	✓	-	-	-	✓	✓	✓	✓	A	Cubes	p	✓	✓

Tick: Supported in the model.
p: Partially supported.
Rel.: Relational.
E/R: Entity-Relationship.
O-O: Object-Oriented paradigm.
A: Algebra.

Hyphen: Not supported or not explained how to support it.
NL: Natural Language.
DL: Description Logics.
UML: Unified Modeling Language.
QL: Query Language.
C: Calculus.

Table 5. Comparison between *YAM*^{2} and other models

In [BSHD98], a list of requirements for a multidimensional model in order to be suitable for OLAP were derived from general design principles, and from characteristics of OLAP applications. [Ped00] also presents eleven requirements (found in clinical data warehousing) for multidimensional data models. [Vas00] gives yet another classification of logical cube models, which we are not going to consider, because our model is at conceptual level. Let us briefly explain the items we use in the comparison of models (most of them taken from those papers), summarized in table 5.

1. **Language used to define the model.** This column shows the language mainly used by every multidimensional model to express its metaschema.
2. **Extended framework.** Some models redefine or extend concepts in other, more general models or design frameworks, which is reflected in this column. In spite of [TP98] uses UML notation, we consider that it is not extending UML, because neither stereotypes, properties nor constraints (i.e. the extension mechanisms of UML) are used on defining the multidimensional model.
3. **Explicit separation of structure and contents** (from [BSHD98]). The data structure should be represented in the schema, while the contents should correspond to instances.
4. **Explicit aggregation hierarchies** (from [BSHD98] and [Ped00]). The model should show how data can be successively aggregated along analysis dimensions.
5. **Multiple hierarchies in each *Dimension*** (from [Ped00]). Although, aggregation hierarchies can be linear, most dimensions show multiple aggregation paths along the same analysis dimension, so this should also be allowed.
6. ***Dimension* attributes** (from [BSHD98]). Showing other characteristics of the analysis dimensions that do not define hierarchies should also be possible.
7. ***Measure*s sets** (from [BSHD98]). This refers to the possibility of defining complex *Cell* structures (grouping more that one *Measure*) related to the same *Fact*. Support provided by [AGS97] is considered partial, because in spite of it allows to manage tuples of measurements, they do not have any extra meaning as a whole.
8. ***Measure*s at different levels of granularity**. Measurements could be taken at different aggregation levels. If so, *Measure*s belonging to the same *Fact*, or even showing the same kind of measure should be related in some way. [Ped00] proposes a comparison item slightly similar to this. However, it is stated as having exactly the same kind of measure being measured at different aggregation levels, so that sometimes it should be stored in a *Cell*, and others in a different one. It would be solved in YAM^2 by specializing the *Cell*s depending on whether the *Measure* is derived or not.
9. **Treat descriptions and measurements symmetrically** (from [BSHD98] and [Ped00]). The data model should allow *Fact*s to be treated as *Dimension*s and vice versa. YAM^2 allows the usage of measurements as descriptors for another measurements by means of derivation mechanisms.
10. **Multi-star schemas**. Users should not be restricted to an isolated subject. They need to see several *Fact*s in one schema. It is not enough sharing *Dimension*s, as in [Kim96], since richer semantic relationships can be used.
11. ***Generalization* relationships**. *Generalization*s should be shown.

12. ***Association* relationships**. Representing *Associations* should be allowed.
13. **Change and time** (from [Ped00]). Although the business being reflected in the schema change, it should be possible to compare data over time.
14. **Derived elements** (from [BSHD98]). The definition of concepts by means of other concepts should be part of the schema.
15. **Imprecision** (from [Ped00]). We just decided not to tackle the problem of representing and querying imprecise data in our model.
16. **Non-onto hierarchies** (from [Ped00]). That is, hierarchies with paths of different lengths from the root to the leaves should be represented. YAM^2 does not fulfill this point because, from our point of view, every object in an aggregation level must have the same structure, i.e. the class structure. Thus, it is not possible that some instances of a class can be divided into parts, while others can not (if so, it should be specialized in some way).
17. **Non-covering hierarchies** (from [Ped00]). That is, hierarchies where there exist relationships between elements of levels that are not directly related. It is not necessary to be supported in our model, because we consider that if those relationships really exist, they should be explicitly represented in the schema by a part-whole relationship between the corresponding Levels.
18. **Many-to-many relationships between two aggregation levels** (from [Ped00]). Some models just mention the possibility of having this kind of relationships (i.e. [AGS97], [HS97], and [DT97]).
19. **Many-to-many relationships between facts and dimensions** (from [Ped00]). There is no constraint forbidding this in YAM^2. Like these relationships are allowed in UML, so they are in YAM^2. However, we can always see it as the fact being related to one set of elements in the *Dimension* so that we obtain a "to-one" relationship with a new *Dimension* of sets of elements.
20. **Additivity semantics** (from [BSHD98] and [Ped00]). Multidimensional models should show how a concept is obtained (if it can) at coarser granularities, and which aggregation functions can be applied to a given *Measure*.
21. **Identification of facts**. The model should show how the different data subject of analysis can be identified by means of other data. Most models just show the aggregation levels at which data are taken, but they do not show the functional dependency that fully determine the measurements. [Vas00] mentions that the data set in a cube is a set of tuples such that contains a primary key. However, it is not reflected by his model in any way.
22. **Mathematical construct used for the operations** (from [BSHD98]). This column shows the mathematical formalism used in the models to define the operations over data.
23. **Elements over which operations are defined**.
24. **Queries using ad-hoc hierarchies not included in the schema** (from [BSHD98]). In order to roll data up, it is necessary a function showing the correspondence between levels. If that function is not in the schema, where is it? YAM^2 allows to define specific star schemas for every user profile. Thus, ad-hoc hierarchies for ad-hoc queries can be defined there.
25. **User defined aggregation functions** (from [BSHD98]). As any operation can be defined in a UML schema, so YAM^2 supports it.

26. **Drill-across.** Some models allow to *drill-across* if the *Star*s share analysis dimensions. However, we can find semantic relationships that also allow it.

8 Conclusions

In the last years, lots of work have been devoted to OLAP technology in general, and multidimensional modeling in particular. However, there is no well accepted model, yet. Moreover, in spite of the acceptance of the O-O paradigm, only a couple of efforts take it into account for conceptual modeling.

In this work, we have presented YAM^2, a multidimensional conceptual model, which allows the usage of semantic O-O relationships between different *Star*s. The model has been defined as an extension of UML to make it much more understandable, and avoid its definition from scratch.

Structures in the model have been defined by means of metaclasses, which are specialization of UML metaclasses. Thus, possible relationships among multidimensional elements have been systematically studied in terms of UML relationships among its elements, so that they allow to show semantically rich multi-star schemas. The inherent integrity constraints of the model pay special attention to identification of data, and summarizability (providing much more flexibility than those of previous multidimensional models). Finally, a set of intuitive, algebraic operations on *Cube*s have been defined in terms of operations over mathematical functions.

References

[AGS97] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling Multidimensional Databases. In *Proc. of 13th. Int. Conf. on Data Engineering (ICDE)*. IEEE Press, 1997.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[ASS01a] A. Abelló, J. Samos, and F. Saltor. Multi-star Conceptual Schemas for OLAP Systems, 2001. Submitted for publication.

[ASS01b] A. Abelló, J. Samos, and F. Saltor. Understanding Analysis Dimensions in a Multidimensional Object-Oriented Model. In *3rd International Workshop on Design and Management of Data Warehouses (DMDW)*. SwissLife, 2001.

[ASS01c] A. Abelló, J. Samos, and F. Saltor. Understanding Facts in a Multidimensional Object-Oriented Model. In *4th Int. Workshop on Data Warehousing and OLAP (DOLAP)*. ACM, 2001. To appear.

[BSHD98] M. Blaschka, C. Sapia, G. Höfling, and B. Dinter. Finding your way through multidimensional data models. In *Proc. of 9th Int. Conf. on Database and Expert Systems Applications (DEXA)*, volume 1460 of *LNCS*. Springer, 1998.

[CT98] L. Cabibbo and R. Torlone. A Logical Approach to Multidimensional Databases. In *Advances in Database Technology - EDBT'98*, volume 1377 of *LNCS*. Springer, 1998.

[DT97] A. Datta and H. Thomas. A Conceptual Model and an algebra for On-Line Analytical Processing in Data Warehouses. In *Workshop on Information Technologies and Systems (WITS)*, 1997.

[GL97] M. Gyssens and L. V. S. Lakshmanan. A Foundation for Multi-dimensional Databases. In *Proc. of 23rd Int. Conf. on Very Large Data Bases (VLDB)*. Morgan Kaufmann Publishers, 1997.

[GMR98] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *Int. Journal of Cooperative Information Systems*, 7(2&3), 1998.

[HS97] M.-S. Hacid and U. Sattler. An Object-Centered Multi-dimensional Data Model with Hierarchically Structured Dimensions. In *Proc. of the IEEE Knowledge and Data Engineering Workshop*. IEEE Computer Society, 1997.

[Inm96] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, 1996.

[Kim96] R. Kimball. *The Data Warehouse toolkit*. John Wiley & Sons, 1996.

[Leh98] W. Lehner. Modeling Large Scale OLAP Scenarios. In *Advances in Database Technology - EDBT'98*, volume 1377 of *LNCS*. Springer, 1998.

[LS97] H.-J. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *Proc. of the 9th Int. Conf. on Scientific and Statistical Database Management (SSDBM)*. IEEE Computer Society, 1997.

[LW96] C. Li and X. Wang. A data model for supporting on-line analytical processing. In *Int. Conf. on Information and Knowledge Management (CIKM)*, 1996.

[NTW00] T. B. Nguyen, A M. Tjoa, and R. R. Wagner. An Object Oriented Multidimensional Data Model for OLAP. In *Int. Conf. on Web-Age Information Management (WAIM)*, volume 1846 of *LNCS*. Springer, 2000.

[OMG99] OMG. *Unified Modeling Language Specification*, June 1999. Version 1.3.

[OMG01] OMG. *Common Warehouse Metamodel*, February 2001. Version 1.0.

[Ped00] T. B. Pedersen. *Aspects of Data Modeling and Query Processing for Complex Multidimensional Data*. PhD thesis, Faculty of Engineering & Science (Aalborg University), 2000.

[SBHD99] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the E/R Model for the Multidimensional Paradigm. In *Int. Workshop on Data Warehouse and Data Mining (DWDM)*, volume 1552 of *LNCS*. Springer, 1999.

[SCGS91] F. Saltor, M. Castellanos, and M. García-Solaco. Suitability of Data Models as Canonical Models for Federated DBs. *ACM SIGMOD Record*, 20(4), 1991.

[TBC99] N. Tryfona, F. Busborg, and J. G. B. Christiansen. starER: A conceptual model for data warehouse design. In *2nd Int. Workshop on Data Warehousing and OLAP (DOLAP)*. ACM, 1999.

[Tho97] E. Thomsen. *OLAP Solutions*. John Wiley & Sons, 1997.

[TKS01] A. Tsois, N. Karayannidis, and T. Sellis. MAC: Conceptual Data Modeling for OLAP. In *3rd International Workshop on Design and Management of Data Warehouses (DMDW)*. SwissLife, 2001.

[TP98] J. C. Trujillo and M. Palomar. An Object-Oriented Approach to Multidimensional Database Conceptual Modeling. In *1st Int. Workshop on Data Warehousing and OLAP (DOLAP)*. ACM, 1998.

[Vas00] P. Vassiliadis. *Data Warehouse Modeling and Quality Issues*. PhD thesis, Department of Electrical and Computer Engineering (National Technical University of Athens), 2000.