YAM² (Yet Another Multidimensional Model): An extension of UML

Alberto Abelló
U. Politècnica de Catalunya (UPC)
Dept. de Llenguatges i Sistemes Informàtics
aabello@lsi.upc.es

José Samos
U. de Granada (UGR)
Dept. de Lenguajes y Sistemas Informáticos
jsamos@ugr.es

Félix Saltor
U. Politècnica de Catalunya (UPC)
Dept. de Llenguatges i Sistemes Informàtics
saltor@lsi.upc.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. YAM² allows the representation of several semantically related star schemas, 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.

There are several papers about multidimensional modeling (see [11] and [16] for two surveys of multidimensional models), but few of them place the discussion at conceptual level. Moreover, most of them focus on the representation of isolated star schemas, i.e. the representation of only one 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: [15] and [8]. These proposals use “Unified Modeling Language” (UML) standard (defined in [9]) in some way, but none of them proposes an extension of it to include multidimensionality. Just the “Common Warehouse Metamodel” (CWM) standard (defined in [10]) 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. Finally, section 6 shows the metaclasses of the model and their relationships with UML metaclasses.

2 YAM² is not JAM² (Just Another Multidimensional Model)

As stated in [3], 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 YAM² structures as an extension of a wide accepted modeling language, i.e. UML (each and every YAM² 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 YAM², we build our model on solid, well accepted foundations, and avoid the definition of basic concepts.

“Expressiveness” or “Semantic Power”, as it is defined in [12], 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. As outlined in [5], due to the presence of multidimensional aggregation, data warehouse - and specially OLAP - applications ask for the vital extension of the expressive power and functionality of traditional conceptual modeling formalisms. Therefore, this is crucial for conceptual multidimensional models like YAM². 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 [12] as the degree to which the model can accommodate not only one, but many different conceptions. This 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, YAM² also provides mechanisms (derivation relationships) 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 a YAM² schema, as well as the ways to obtain it. For instance, this information can be used at later design phases to decide materialization.

Therefore, main advantages of YAM² are its expressiveness and semantic relativism, besides the flexibility offered in the definition of summarization constraints (it generalizes the work in [6]). Moreover, from the separate study of characteristics of analysis dimensions and factual data in [1] and [2], we ensure that it is defined on solid foundations.

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. The “facts” represent measurements (in a general sense), while “dimensions” represent given information we already have before taking the measurements. 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 of target Level decomposes into a collection of instances of source Level (i.e. edges reflect part-whole relationships between instances of Levels). It is an specialization of Classifier UML metaclass.

Figure 1 shows an example of Dimension. It contains four Levels: Customer, AgeGroup, Bonanza, and All. Every instance of Customer Level represents a customer, which can be aggregated in two different ways to obtain either age or bonanza 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 graphs of Dimension 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 of target Cell decomposes into a collection of instances of source Cell (i.e. edges reflect part-whole relationships between instances of Cells). It is an specialization of Classifier UML metaclass.

Figure 2 shows an example of the structure of a Fact with two orthogonal Dimensions: Customer, already depicted in figure 1; and Clerk, composed by Clerk, Team, and All Levels. We can see that there is a Cell in the Fact for every combination of Levels in the Dimensions. Thus, a Fact contains all data regarding the same subject at any granularity. Having two independent Dimensions with 4 and 3 Levels respectively, means that the Fact will have 12
different Cells. These Cells and the part-whole relationships between them form a lattice. It is not necessary to represent all those Cells in the schema. Cells just containing derived data are optional, and should only be explicit to emphasize the importance of summarized data at a given aggregation level.

These six kinds of nodes are grouped in three pairs. At intermediate level, there are Cells and Levels. Looking at lower detail we see Measures and Descriptors. Moreover, at this level, we also define KindOfMeasure to show that several Measures in different Cells correspond to the same measured concept at different aggregation levels. Moreover, at upper detail level, we have Facts and Dimensions (one Fact and the Dimensions associated to it compose a Star).

**Definition 7** A Star is a modeling element composed by one Fact, and several Dimensions that can be used to analyze it. It is an specialization of Package UML metaclas.

### 3.2 Arcs

![Figure 3. UML Relationships](image)

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 different subclasses and stereotypes of Relationship. As depicted in figure 3, Generalization relationships relate two GeneralizableElements, one with a more specific meaning than the other. Classifiers and Associations 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 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. It shows part-whole relationships. Finally, UML allows to represent different kinds of Dependency relationships between ModelElements like Binding, Usage, Permission, or Abstraction. We are not going to consider the three first, because they are rather used on application modeling, and YAMF 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.

In this section, we systematically see how these relationships can be used to relate multidimensional constructs at every detail level. For every pair of constructs at each detail level we will show whether they can be related by a given kind of Relationship or not. Moreover, if two constructs can be related, we will also show whether they must belong to the same construct at the level above, or not (i.e., inter or intra relationships, respectively).

#### 3.2.1 Upper detail level

<table>
<thead>
<tr>
<th>Generalization</th>
<th>Inter</th>
<th>Intermediate</th>
<th>Intra</th>
<th>Inter/Intra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>Inter</td>
<td>Intermediate</td>
<td>Inter</td>
<td>Inter/Intra</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Inter</td>
<td>Intermediate</td>
<td>Inter</td>
<td>Inter/Intra</td>
</tr>
<tr>
<td>Derivation</td>
<td>Inter</td>
<td>Intermediate</td>
<td>Intra</td>
<td>Inter/Intra</td>
</tr>
</tbody>
</table>

### Table 1. Relationships at 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 Facts, they must belong to different Stars. Therefore, relationships between Facts will always be inter-stellar, but for reflexive Associations and Aggregations. 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.

![Figure 4. YAMF schema at upper detail level](image)

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 on, 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 are customers, 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 Club by means of an Aggregation relationship. Every instance of Club 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. Derivations 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 Dimensions 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 Facts 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 Facts 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 obtaining of one from the other should be restricted to derivation mechanisms. For instance, a Fact cannot eventually become a Dimension.

3.2.2 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 Cells (i.e. AtomicSale) depending 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 Cells can be related by Generalization, but they must belong to different Facts (i.e. it is an
inter-factual relationship). Cells in different Facts 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 Cells 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 Levels, and are not necessary in the schema. However, we could also find that different Cells are aggregated to obtain a Cell about a different kind of fact (when both Facts are also related like ProductSale and Deal). In this case, we do not group Cells along any analysis dimension, i.e. it does not generate coarser Cells in the same Fact, but Cells 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 an inter-factual relationship). Derivation relationships can be used to hide information, change names, or Measures in the Cells, giving rise to new Facts.

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, and vice versa (showing the granularity of the Cells).

### 3.2.3 Lower detail level

Since elements at this level are neither Classifiers nor GeneralizableElements, but just Attributes, 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, evolution cannot convert a Measure into a Descriptor, nor vice versa.

![Figure 5. YAM² schema at intermediate detail level](image)

**Table 3. Relationships a lower detail level**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Derivation</td>
<td>OtherStore</td>
<td>time</td>
<td>State</td>
<td>time</td>
</tr>
<tr>
<td>Association</td>
<td>Product</td>
<td>time</td>
<td>Store</td>
<td>item</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Region</td>
<td>time</td>
<td>Region</td>
<td>time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Atom/SaleByStore</th>
<th>Income</th>
<th>Revenue</th>
<th>IncomeAverage</th>
<th>IncomeAverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Income</td>
<td>Revenue</td>
<td>IncomeAverage</td>
<td>IncomeAverage</td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>Income</td>
<td>Revenue</td>
<td>IncomeAverage</td>
<td>IncomeAverage</td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td>Income</td>
<td>Revenue</td>
<td>IncomeAverage</td>
<td>IncomeAverage</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 6. YAM² schema at lower detail level](image)

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 (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 Cells and Levels, 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 aspects in multidimensional modeling, namely placement 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. This improves the understanding of those data and allows the implementation of specific storage techniques. It is important that the n dimensions of the space (i.e. Cube) are orthogonal. If not, i.e. if a Dimension determines others, the visualization of data will be unnecessarily complicated (we are showing more information than 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, maybe resulting in a waste of space. This does not mean that all Dimensions in a Star must be orthogonal. Nevertheless, those defining Cubes (which are used for visualization as well as storage purposes) should be, or at least the user should know whether they are.

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 placement 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 Bases (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 Levels Clerk, Minute, and Product, so that AtomicSale is fully functionally determined by those three Levels (a Base of the space).

Definition 8 A Cube is an injective function from an n-dimensional finite space (defined by the cartesian product of n functionally independent Levels), to the set of instances of a Cell \((C, c : \prod_{i=1}^n L_i \rightarrow C)\).

If the Levels were not functionally independent (i.e. they did not form a Base), we would use more Dimensions than 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. [6] identifies three necessary (intuitively also sufficient) conditions for summarizability:

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

The first two conditions are absolutely dependent on constraints over cardinalities in the part-whole relationships in 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 in [11]). 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
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 \(\text{YAM}^2\), 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.

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 Levels are an InvalidSource for the calculation of a given KindOfMeasure (for example, Kind is an invalid source for Income and Revenue). This means that measurements at an aggregation level cannot be used to obtain data at higher aggregation levels. We must go to Levels below to obtain the source of data for the calculation.

This can be due to the fact that 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 can 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.

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 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, instances of Induction 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 [14]. 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 overlooked 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. \(\text{YAM}^2\) operations focus on identifying and uniformly manipulating sets of data, namely Cubes. In a Cube, data are identified by their properties. Thus, these operations are separated from the physical storage of the data. Moreover, they are not presentation oriented like those in [13].

![Table 4. \(\text{YAM}^2\) operations](image)

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.

![Figure 7. Multidimensional operations as composition of functions](image)

In the sense of [3], 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 as a function (see definition 8), operations must transform a function into another function. Operations in the first column work on the image of the function (i.e. Cell), while operations in the second column change its domain (i.e. Base). 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.
Obtaining $c_o$ from $c_i$, can be seen as mathematical composition of functions ($c_o = \psi \circ c_i \circ \phi$, with $\psi$ and $\phi$ belonging to the families of functions $g$ and $f$, respectively). Firstly, we can see how ChangeBase, given $c_i$ and $\phi$ (a function belonging to a family of functions $f$ between the finite spaces defined by cartesian product of Levels of each Cube), we obtain a new Cube ($c_o = c_i \circ \phi$). Nevertheless, Drill-across does change the Cell. Thus, it works in the opposite way, in the sense that it needs a Cube $c_i$ and the function $\psi$ (belonging to a family of functions $g$ from a Cell to another Cell) to obtain the new Cube ($c_o = \psi \circ c_i$).

Unfortunately, it is not possible to define all operations in such a way. Roll-up changes the space as well as the Cell. Thus, obtaining it as a composition of functions is not possible, because a coordinate in the space of $c_o$ corresponds to several points in $c_i$. Therefore, there is no $\zeta$, so that $c_o$ is a composition of $\zeta$ and $c_i$. It can neither be defined as an homomorphism (like those in [4]), because the problem is not the conversion of a set of instances into one instance (which is always performed by union), but deciding which is the set of instances to be converted (defined by a function of family $h$).

### Drill-across

This operation changes the image set of the Cube by means of an injective function $\psi$ of the family $g$ (relationships in section 3.2 can be used for this purpose). The space remains exactly the same, only the cells placed in it change. This function relates instances of a Fact to instances of another one. $\psi : C^i_o \rightarrow C^o$, injective so that $c_o(x) = \delta_o(c_i) = \psi(c_i(x))$

### Projection

This just selects a subset of Measures from those available in the selected Cell. Since it works at the attribute level, it is absolutely equivalent to the homonym operation in Relational algebra. $c_o(x) = \pi_{m_1, \ldots, m_k}(c_i) = c_i(x)[m_1, \ldots, m_k]$

### ChangeBase

This operation reallocates exactly the same Cell in a new space. It changes the domain set of the Cube by means of an injective function $\phi$ of the family $f$ (i.e. $\phi$ relates points in an n-dimensional finite space to points in an m-dimensional finite space). Thus, it actually modifies the analysis dimensions used. $\phi : L^o_1 \times \ldots \times L^o_n \rightarrow L^i_1 \times \ldots \times L^i_m$, injective so that $c_o(x) = \gamma_o(c_i) = c_i(\phi(x))$

### Roll-up

It groups cells in the Cube based on an aggregation hierarchy. This operation modifies the granularity of data, by means of an exhaustive function $\varphi$ of the family $h$ (i.e. $\varphi$ relates instances of two Levels in the same Dimension, corresponding to a part-whole relationship). It reduces the number of cells, but not the number of Dimensions. $c_o(x) = \rho_h(c_i) = \bigcup_{\varphi(y)=x} c_i(y)$

### Dice

By means of a predicate $P$ over Descriptors, this operation allows to choose the subset of points of interest out of the whole n-dimensional space. Like Projection, it is absolutely equivalent to an operation of Relational algebra. In this case, the operation is “Selection”. $c_o(x) = \sigma_{P(c_i)} = \begin{cases} c_i(x) & \text{if } P(x) \\ \text{undefined} & \text{if } \neg P(x) \end{cases}$

Looking to the empty cell in table 4, it is clear that there is another operation missing, which would allow to select the Cell we want to query in the same way we choose Measures or Facts. 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 YAM² 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.

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

These operations allow to build Cubes on solid mathematical foundations. Semantic relationships in the multi-dimensional schema define functions between Classes. By composing those functions appropriately, we can obtain the desired vision of data. If we want to analyze instances of a given Class in the space defined by the cartesian product of a set of Classes, all we have to do is find the appropriate composition of functions. If that “chain” of functions exists, we can analyze data in the desired way. Thus, properties of mathematical functions can be applied.

It is also important to outline that this set of operations is minimal (i.e. none can be expressed in terms of others, nor can any be dropped without affecting their functionality) and the operations are atomic (i.e. each operation performs exactly one task). This can be easily inferred from the explanation above, and table 4. Each operation works inside only one detail level. Moreover, they work either on factual or dimensional data. Roll-up could be thought as working on both sides, however, it really operates only on factual data based on dimensional data.

### 6 Metaclasses

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 Stars will exist, as it was shown in section 3.2.

At the intermediate detail level, we can see that Dimensions are composed by Levels related by LevelRelations, 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 Cells. Each of those Cells is defined at an aggregation level for each of the analysis dimensions of its Fact. If there is a Level (l1) whose elements are obtained by grouping those of another Level (l2) at which a Cell (c1) is defined, then we have another Cell (c2) related to l2 whose instances are composed by those of c1. Cells c1 and c2 are related by a CellRelation, which corresponds to the LevelRelation between l1 and l2. A set of functionally independent Levels 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 explicated in the schema, while other (for instance, CellRelation) can be derived. In this sense, we distinguish those Cells that need to be explicated (i.e. FundamentalCells), from those that do not (i.e. SummarizedCells), 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 Levels contain Descriptors, and the Cells contain Measures. SummarizedCells only contain data that can be derived (i.e. SummarizedMeasures), and FundamentalCells can contain derived or not derived data. SummarizedMeasures are obtained from other Measures, while FundamentalMeasures 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, SummarizedMeasures 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 ([6] studies the influence of the temporal dimension on three different kinds of attributes). Thus, when we want to obtain a SummarizedMeasure in a Cell (c1), from a Measure in another Cell (c2), the Summarization performed is that induced by the Dimension that contains the LevelRelation to which the CellRelation between c1 and c2 corresponds.

Summarizations 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 KindOfMeasures, which is also captured in a YAM² 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. Facts and Dimensions are Classifiers containing Classes (i.e. Cells, and Levels respectively). Finally, Measure and Descriptor are just Attributes of the Classes. All other elements in YAM² have also been placed as specialization of a UML concept. Maybe, the most relevant ones are CellRelation and LevelRelation that are Aggregations. Moreover, a Base is just a Constraint stating that a set of functionally independent Levels fully determine instances of a Cell.

This proves that multidimensional modeling is just an specialization of general data modeling. We could roughly say that all we are doing is splitting elements in the model based on whether they refer to factual or dimensional data. It can be seen that some specific concepts are defined, besides properties and constraints of the new structures. [7] claims that E/R provides the complete functionality and support necessary for OLAP applications. Here, we can see that UML also provides such support. However, it is well known that the more specific the Classes in a schema are, the better they represent reality. In the same way, the more specific our data model is, the better it will represent reality.
7 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 Stars. The model has been defined as an extension of UML to make it much more understandable, and avoid its definition from scratch. As a side effect, this shows that multidimensional modeling is just an special case of data modeling.

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 Cubes have been defined in terms of operations over mathematical functions.

Acknowledgements

Our work has been partially supported by the Spanish Research Program PRONTIC under projects TIC2000-1723-C02-01 and TIC2000-1723-C02-02, as well as the grant 1998FI-00228 from the Generalitat de Catalunya.

References


