Understanding Facts in a Multidimensional Object-Oriented Model

Alberto Abelló
U. Politècnica de Catalunya
C/ Manuel Girona 1-3
E-08034 Barcelona
aabello@lsi.upc.es

José Samos
U. de Granada
Avd. de Andalucia 38
E-18071 Granada
jsamos@ugr.es

Fèlix Saltor
U. Politècnica de Catalunya
C/ Manuel Girona 1-3
E-08034 Barcelona
saltor@lsi.upc.es

ABSTRACT

"On-Line Analytical Processing" tools are used to extract information from the "Data Warehouse" in order to help in the decision making process. These tools are based on multidimensional concepts, i.e. facts and dimensions. In this paper we study the meaning of facts, and the dependencies in multidimensional data. This study is used to find relationships between cubes (in an Object-Oriented framework) and explain navigation operations.

Keywords

Multidimensionality, Functional dependencies, Object-Oriented modeling, Cube operations

1. INTRODUCTION

SQL was created to be used in "On-Line Transactional Processing" (OLTP) systems. Along its years of existence, it proved to be really useful and wide accepted for this purpose. However, as time went by, due to the wide spread of computers, databases arrived to analysis systems. In this kind of environments, because of the huge amount of data, complexity of queries and unskillfulness of users, SQL has proved not to be the best solution.

For analysis purposes, data is stored in "Data Warehouses" (DW), which contain the historical, integrated data of all the company, that analysts use to make decisions. To bring these data near analysts, "On-Line Analytical Processing" (OLAP) tools appeared. By means of multidimensionality, this kind of tools allow non-expert users to formulate their own queries and obtain the results interactively (without the assistance of the IT department).

Multidimensionality is based on the duality fact-dimensions, i.e. facts are analyzed with regard to data in the dimensions. A fact represents a subject of analysis, while its dimensions show the different points of view we can use to study it. This is pretty intuitive and close to analysts' way of thinking. OLAP tools implement what users demand. However, there is no well accepted, strong mathematical foundation for Multidimensional databases as it is for Relational ones. In this paper we want to move a step towards it.

1.1 Related work

Lots of work have been devoted to multidimensional modeling. [1] contains a comparison of several multidimensional data models. Here we emphasize some that pay special attention to modeling facts at conceptual level. [7] presents a graphical model besides a methodology to obtain a multidimensional schema from the operational schemas (either E/R or Relational). It defines a "fact schema" as a set of "measures" related to "dimension attributes". As an specialization of E/R, we find [16] and [13]. Both extend E/R in order to capture multidimensional semantics. In the latter, "facts" are specialization of "relationships". [4] also contains a conceptual model. However, it provides a more formal approach where a "fact" is defined as a function over the cartesian product of domains of its analysis dimensions. In [15] and [11] we can see two Object-Oriented modeling approaches.

Regarding operations on multidimensional data, several algebras have also been proposed. [3] presents a set of algebraic operations claimed to be as powerful as Relational algebra. Multidimensional operations like *roll-up* can be built by composition of them. [8] uses "Description Logics" to describe semantics of multidimensional operators. Here, a cube is defined as an object which is associated to cells of similar form. [12] provides a formalism and an algebra that is closed and, at least, as strong as Relational algebra with aggregation functions. Finally, [18] also presents a complete and sound algebra.

With regard to relationships among data in multidimensional schemas, [9] states that the "fact table" has a composite primary key made up of the foreign keys to its "dimension tables". [6] agrees that, and emphasizes that records in the "fact table" represent points in the multidimensional space. [10] contains a proposal of normal forms for multidimensional modeling, based on "weak functional dependencies". It states that there is a functional dependency from analysis dimensions to "summary attributes" (i.e. measures). A schema in "multidimensional normal form" means dimensions are orthogonal to each other, and "summary attributes" are fully functionally determined by the set of "terminal category attributes" (i.e. atomic aggregation levels).

From our point of view, dependencies in the context of multidimensional databases need much more attention. A theoretical, wide study of dependencies is in [14]. For a more application-oriented explanation of dependencies, [5] contains two chapters devoted to dependencies and normal forms in Relational databases, and how they help on designing.

1.2 Paper contents and structure

In this paper, we try to clarify some concepts about facts, and how they should be modeled (this was already done for dimensions in [2]). "Functional Dependencies" (FDs) were successfully used on developing "Relational" theory. Thus, we are going to show how they could also be used to explain multidimensionality. It does not mean we plead for "Relational OLAP" (ROLAP) as opposed to "Multidimensional OLAP" (MOLAP) tools. We place our discussion at conceptual level, and it is independent of any kind of underlying system.

By better understanding multidimensionality and how it should be modeled, we can obtain several benefits. Firstly, it will help on designing multidimensional schemas, as normal forms do for relational ones. Secondly, users will also benefit from it, since querying will be easier and more understandable. Finally, storage and retrieve systems could also be improved, if knowledge about the real meaning of data is improved.

We argue that the meaning of multidimensionality has not been well stated. We are not going to re-discover multidimensionality, but just clarify and justify some points. Section 2 explains multidimensional concepts (placing them at different detail levels), with regard to n-dimensional spaces and *functional dependencies* (FD) between them. Section 3 explains and exemplifies relationships between *cubes* and *Facts*, while section 4 presents a complete and sound algebra for *cubes*. Conclusions, acknowledgements and bibliography close the paper.

2. MULTIDIMENSIONAL ELEMENTS UNLEASHED

The DW contains lots of *measures* analysts want to understand and compare. Studying all together would be almost impossible. In this section, we are going to show how these data can be successively grouped at different detail levels to ease its management. We will have *measures* grouped into *cells*, of different *classes* (represented as n-dimensional *cubes*), which will be grouped based on the *kind of fact* they represent.

2.1 Measures and cells

Usually, for the same *kind of fact* subject of analysis, at lower detail level, we have several *kinds of measures*. For instance, for a `Sale` we could keep `cost`, `revenue`, `amount of product`, etc. Thus, when a cloud of *measures* must be faced, those corresponding to the same *fact* are always grouped in the mind of analysts.

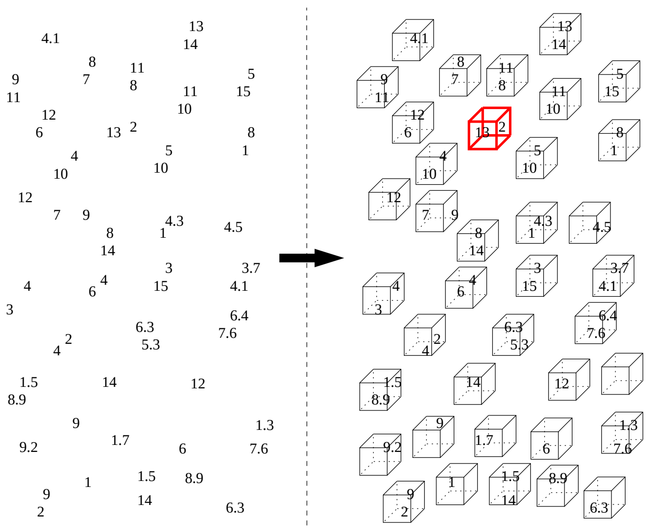

Figure 1: Measures grouped into *cells* corresponding to *facts*

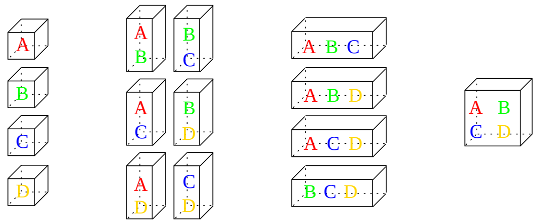

Figure 2: $\mathcal{P}(C_i)$ being $C_i = \{A, B, C, D\}$

Definition 1. A *cell* is a set of *measures* (possibly empty), representing a *fact*.

Figure 1 sketches this by drawing several *measures*. Those that correspond to the same *fact* are inside a *cell*, which represents the *fact*. One of these *cells* (i.e. an instance of a *kind of fact*) contains all *measures* we have about what was sold to John Doe last Monday in Barcelona (i.e. we sold him 2 items and charged 13$). Nevertheless, grouping *measures* of the same *fact* is not enough to be able to make decisions. Several *facts* can be grouped, and it gives rise to more complex *facts*. Algebraically, the set of *cells* (C) representing all possible *facts* in the DW forms a commutative semigroup with union ($x \cup y$ means *cells* x and y are grouped into a new, complex *cell*). $< C, \cup >$ fulfills the following properties:

Closed: $\forall x, y \in C, x \cup y \in C$

Commutative: $\forall x, y \in C, x \cup y = y \cup x$

Associative: $\forall x, y, z \in C, (x \cup y) \cup z = x \cup (y \cup z)$

Neutral element: $\forall x \in C, x \cup \emptyset = x$

If we call C_A the set of all *cells* representing atomic *facts* (i.e. those that cannot be decomposed) and we allow the

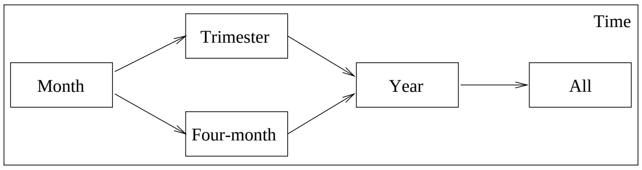

Figure 3: Example of analysis dimension

union of any kind of *cells*, $\mathcal{P}(C_A)$ should be considered, which contains $2^{Card(C_A)} - 1$ *cells* (figure 2 shows a set with four atomic *cells*). Fortunately, what analysts really want to study is only a subset of $\mathcal{P}(C_A)$. This subset is defined by the different *kinds of facts*. We do not need to consider $\mathcal{P}(C_A)$ but, at most, $\bigcup \mathcal{P}(C_i)$, being every C_i the set of all atomic *cells* of a given *kind of fact* so that $\bigcup C_i = C_A$. For example, if `A` and `D` in figure 2 are of the `Sales` *kind of fact* (S), while `C` and `B` are of the `Productions` *kind of fact* (P), $\bigcup_{i \in \{S,P\}} \mathcal{P}(C_i) = \{A, B, C, D, AD, BC\}$.

2.2 Analysis dimensions and aggregation levels

The *facts* themselves are almost meaningless. They only have meaning when analysis dimensions identify them. Talking about `Sales` is senseless, if you do not know who sold what, when, whom, etc. *Cells* are usually grouped to give rise to more complex *cells*, which contain summarized *measures*. However, most combinations of *cells* do not give rise to meaningful more complex *cells*. It must be done based on analysis dimensions (for example, we should not group data regarding months with those regarding years). In [2] we already studied semantics and structure of *Dimensions*, which show the different points of view analysts use to study *facts*. Each *Dimension* contains a graph indicating how the *facts* can be aggregated along the analysis dimension.

Definition 2. A *Dimension* is a connected, directed graph. Every vertex in the graph corresponds to an *aggregation level* containing instances, and an edge reflects that every instance at target level can be decomposed as a collection of instances of source level (i.e. edges reflect Part-Whole relationships between instances of *aggregation levels* in the *Dimension*).

Figure 3 shows an example of *Dimension*. It contains five aggregation levels, i.e. `Month`, `Trimester`, `Four-month`, `Year`, and `All`. Every instance of `Month` level represents a month, which can be aggregated in two different ways to obtain either trimesters or four-month periods. Both kinds of instances (i.e. `Trimester` or `Four-month`) can be grouped to obtain years. Finally, at top we have `All` level with exactly one instance representing the group of all months in the *Dimension*.

2.3 Classes of cells (n-dimensional cubes)

We can associate every atomic *cell* to an instance of an *aggregation level* in each of its analysis dimensions, showing the meaning of its *measures*. If all *cells* in a complex *cell* are associated with instances in an *aggregation level* l_1, and there exists an instance of l_2 exactly composed by those instances in l_1, we can associate the complex *cell* with the instance of l_2. For example, if all *cells* composing another one are at level `Month` and correspond to exactly those months in a trimester, the complex *cell* is associated to an instance of `Trimester` level.

Definition 3. A *class of cells* (*class* for short) contains those *cells* representing the same *kind of fact* and being associated with instances of the same *aggregation level* for each of the dimensions we use to analyze it.

For example, all *cells* representing `Sales` during a given month in a given store by a given customer form a *class*. Instances of this *class* differ in one or more of the instances of the *Dimensions* they are associated to (i.e. the month it was sold, the store where it was sold, or the customer who bought it). Two *measures* regarding the same *kind of fact*, and the same instance in every *Dimension* will be in the same *cell*. Thus, *Dimension* instances identify *cells* in a *class*.

If we allowed to compare any set of *cells*, we would find that there exists $2^{2^{Card(C)}-1} - 1$ possible sets of *cells* in $\mathcal{P}(\mathcal{P}(C))$. Thus, *classes of cells* are defined to ease the study of these huge amount of sets of cells. Only *measures* in *cells* of the same *class* can be compared or treated together, because they represent exactly the same kind of information (i.e. *kind of fact*) at the same granularity (i.e. *aggregation level*). What's more, analysts are not interested in all *cells* in $\mathcal{P}(C_i)$, but only in those corresponding to *facts* at an *aggregation level* in each of the *Dimensions*. They are only interested in subsets of every $\mathcal{P}(C_i)$ determined by aggregation hierarchies in analysis dimensions. Aggregation hierarchies in the *Dimensions* restrict the union of *cells* to those of the same *class*. For example, a *cell* associated to an instance of `Month` cannot be grouped with another *cell* at `Year` level to give rise to a more complex *cell*.

[10] explains that analysis dimensions of a "summary attribute" should be orthogonal. This means that there are no dependencies between them. In our case, it can be translated as having no dependencies between the *aggregation levels* a *class of cells* is associated with. All possible combinations of instances in the *Dimensions* must be possible, which may be stated as multivalued dependencies with the empty set in the left hand side (*degenerated dependency*) for every pair of *aggregation levels*. Being L the set of *aggregation levels* with which a *class of cells* is associated, $\forall l_i, l_j \in L$ *and* $i \neq j$, $\emptyset \rightarrow\rightarrow l_i \mid l_j$.

Degenerated dependencies for every pair of *aggregation levels* mean we are talking about the cartesian product of all them. Since we also have that *aggregation levels* identify the *cells* in a *class*, that cartesian product fully functionally determines the *cells*, i.e. $l_1 \times .. \times l_n \rightarrow C_{class}$. Thus, a *class of cells* (either atomic or complex) determined by analysis dimensions could be drawn, as those in figure 4, forming an n-dimensional cube at intermediate detail level.

Definition 4. A *cube* is an injective function from an n-dimensional finite space defined by the cartesian product of

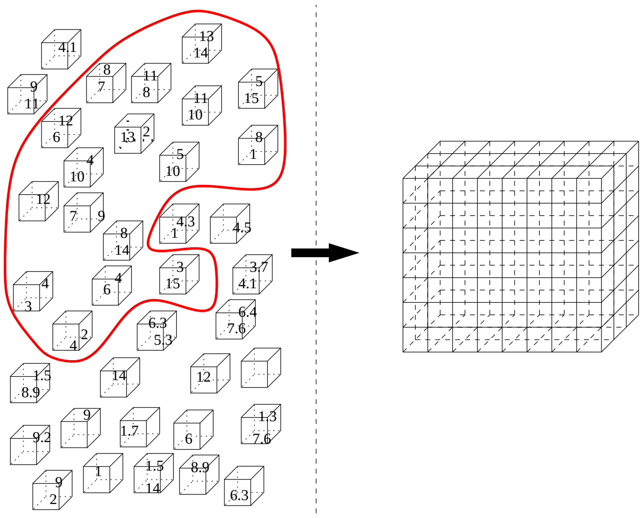

Figure 4: Diagram of a *class of cells* with three analysis dimensions

n functionally independent *aggregation levels*, to the set of *cells* in a *class* (C_c).

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

Being a function means it is not allowed to have "holes" in the *cube*. Any combination of *Dimension* instances must be valid (i.e. related to a *cell*). However, missing *cells* should be allowed it they mean that the *fact* is "unknown" or that it could have happened, but it did not. To avoid these "holes", this can be represented as a boolean *measure* per *cell* meaning whether the corresponding *fact* happened or not (a "null" value in this boolean *measure* means we do not know if it happened). What must be forbidden is an sparse *cube* because of "inapplicable" combinations in the cartesian product, since it means we have dependencies between *Dimensions*, which is a bad conceptual design.

In general, different *classes of cells* are determined by cartesian products of different *aggregation levels*. However, it could also be that the same set of *aggregation levels* determine two different *cubes* for different *kinds of facts* (for example, `Sales` and `Purchases` in our business, both being analyzed by `Month`, `Region`, and `Product`). That is, *Dimensions* can be freely reused for different *cubes*.

2.4 Facts

Only *cells* of the same *class* can be grouped to obtain a coarser *cell*. Thus, *cells* in a *cube* are obtained by union of *cells* in another *cube*. This is always done following aggregation paths in the analysis dimensions. The *cubes* generated by grouping *cells* in another *cube* are always in the same *kind of fact*. Thus, we group *cubes* into *Facts* at upper detail level.

Definition 5. A *kind of fact* (*Fact* with capital "F" for short) is a connected, directed graph. Every vertex in the graph corresponds to a *cube* containing *cells*, and an edge reflects that every *cell* at target *cube* can be decomposed as a collection of *cells* of source *cube* (i.e. edges reflect Part-Whole relationships between *cells* in *cubes* in the *Fact*).

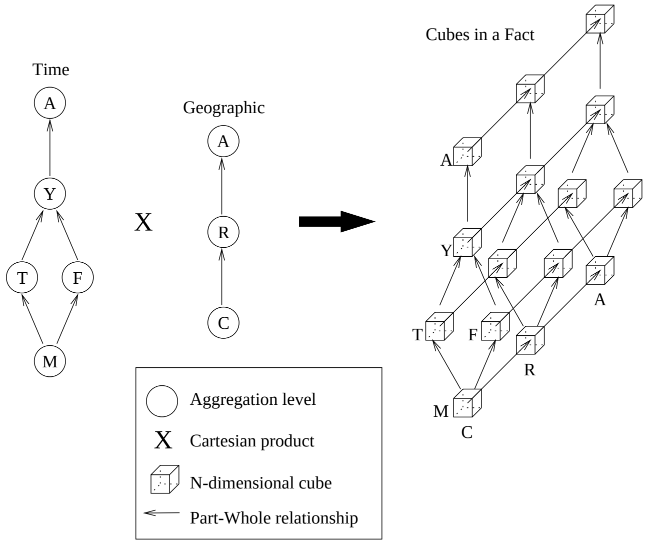

Figure 5: Graph of *cubes* in a *Fact* with two *Dimensions*

Figure 5 shows an example of the structure of a *Fact* with two *Dimensions*: `Time` already depicted in figure 3, and `Geographic` composed by `City`, `Region`, and `All` *aggregation levels*. We can see that there is a *cube* in the *Fact* for every combination of *aggregation levels* in the *Dimensions*. Having two *Dimensions* with 5 and 3 *aggregation levels* respectively, means that the *Fact* will have 15 *cubes*. These *cubes* and the Part-Whole relationships between them form a lattice. All atomic *cells* are in the *cube* at the bottom, while the *cube* at top contains only one cell which is the union of all atomic *cells*.

A *cell* may contain any kind of data. It uses to be numerical, because we always know how to summarize numerical data (i.e. *sum*, *avg*, etc.). However, we just need a set of aggregation operations for a non-numerical data type to be able to keep it in *cells*. For instance, we could aggregate character strings by set-union. Therefore, we can also have descriptive attributes in *cells*. Aggregation operations, for boolean *measures*, would be *count*, *and*, *or*, etc. Anyway, if the data types of the *measures* have an order, we can always aggregate calculating the median. Thus, *measures* in *cells* can always be aggregated to obtain the *measures* in *cubes* with more complex *cells*. Different aggregation functions (i.e sum, average, minimum, etc.) could be used to obtain different *measures* in a complex *cell*.

Some *cubes* could contain *measures* that are not obtained by aggregation of those from other *cubes*. For example, some data could be collected yearly, so that *cubes* at `Month` level cannot contain it. [6] distinguishes between analytical and non-analytical data. Sometimes, we are interested in analyzing data at a given *aggregation level*, and ignore atomic data. However, in spite of we might not collect *measures* at the lower level of granularity due to either availability, performance or legal reasons (i.e. personal data uses to be private), we could be interested in keeping some information about instances at that level (for example, names of people in the census). Thus, we have cases where *measures* in a *cube* are not present for coarser or more detailed levels.

If we know the *Dimensions* that define the *cubes* in a *Fact*, we know which are those *cubes* and how they are related. Thus, it could be inferred that it is not necessary to show them in multidimensional modeling. However, we argue that this is not true. As stated before, some cubes could have specific *measures*, or other could be specially important to be shown to users. As some derived attributes are show in a conceptual schema for the sake of completeness and clearness, so some *cubes* with complex *cells* should also be shown in a multidimensional schema. Most of those *cubes* will be calculated on the fly, but other could be physically stored to improve performance, or just keep specific *measures*.

The structure of *cubes* in a *Fact* (a lattice) exactly coincides with that of *aggregation levels* in a *Dimension*. Not only structure, but meaning coincides as well. In both cases, there is a *class* of atomic instances at bottom, that are successively aggregated in instances of other *classes*, until we obtain an instance of the top *class* which contains all atomic instances. Both, *Facts* and *Dimensions*, contain a graph of Part-Whole relationships between classes. The difference is that the aggregation graph of a *Dimension* depends on its proper semantics, while the aggregation graph of a *Fact* depends on the aggregation hierarchies of its analysis dimensions. Thus, we could consider a *Dimension* as a unidimensional, self-qualified *Fact*. All we need to obtain a *Dimension* from a *Fact* is to change its base appropriately (as explained in section 3.1).

[6] defines a "degenerate fact" as a measure recorded in the intersection table of a many-to-many relationship between facts. It could be seen as data in a *cube* being related to two different *cubes* (by one-to-many relationships). Thus, we could also see it as two *Facts* acting as *Dimensions* of another *Fact*. Therefore, the duality *Fact-Dimensions* only exists if we look to an isolated multidimensional schema. Looking to all multidimensional schemas together means that what is considered a *Fact* by an analyst, could be considered a *Dimension* by another one, or vice versa.

3. OTHER RELATIONSHIPS BETWEEN CUBES

In this section, we are going to see how different *cubes* can be conceptually related. Firstly, we present the same *cube* being redistributed to be analyzed from different points of view (i.e. base changes in the space). Then, we will also show how some Object-Oriented relationships (i.e. Generalization/Specialization, Aggregation, and Derivation) between *cells* are represented as relationships between *cubes* and *Facts*.

3.1 Base changes

Steinitz's theorem regarding vectorial spaces states that if $\{e_1, .., e_n\}$ are a base for a space, and $\{v_1, .., v_m\}$ are linearly independent, we can change m elements in the base by v_i,

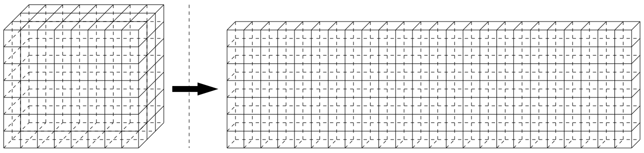

Figure 6: Reduction of a tri-dimensional *cube* to a bidimensional *cube*

and it still be a base. Since *cubes* are nothing else that finite spaces, we can also find that two *cubes* are related by a base change (*dimensions* change in our case). Both *cubes* contain the same *cells*, but just place them in a space defined by different analysis dimensions. Thus, *Dimensions* in one of them must functionally determine the ones in the other.

If *aggregation levels* $\{L_1, .., L_n\}$ determine a *cube*, and there exists a set of functionally independent *aggregation levels* $\{L'_1, .., L'_m\}$ so that $\forall i \leq m,\ L'_i \to L_i$, we can change m *aggregation levels* in the *cube*. If $L'_i \to L_j \times L_k$, dimensionality can be reduced by replacing L_j and L_k by L'_i, as sketched in figure 6. Some authors propose to join two correlated analysis dimensions in order to avoid meaningless combinations. This is not the case. The number of *cells* is exactly the same. They are just placed in another way. As dimensionality of the *cube* can be decreased, it can also be increased, if $L'_i \times L'_j \to L_k$. All these base changes between *cubes* can be seen as an application of the transitive property of FDs between *aggregation levels*.

As a special case, a surrogate generated by a sequence is always a base for the (unidimensional) space. However, it can be considered a degenerate case, since it is meaningless for analysts and implies the loss of all benefits in multidimensionality. Nevertheless, as mentioned in previous section, it is important to convert a *Fact* in a *Dimension*. In [6], the *Dimension* of a unidimensional space is called a "shadow dimension", which has a one-to-one relationship with the "fact table".

Another problem, already discused by some authors, is how *measures* can be transformed into analysis dimensions for its own *Fact*. For example, [10] stated that using a *measure* as *Dimension* means a change in the schema. In our framework, we have just a base change in the space. Whether numerical or descriptive, if a set of attributes fully functionally determines *cells* in a *cube*, they can be used as analysis dimensions. Thus, *measures* could also be used as analysis dimensions, if they allow to identify *facts*.

3.2 Generalization/Specialization

As it was previously said, *cells* in a given *cube* could have *measures* that *cells* in other *cubes* do not have. For instance, if our company (may be the result of a fusion of preexisting smaller companies) is organized by autonomous regions, it could be that the information systems in one of these regions collect data that those in other regions do not. Thus, we will specialize our data *cells* depending on the region.

Specialization of *cells* is due to the specialization of the *kind of fact* they are representing. Specializing means dividing

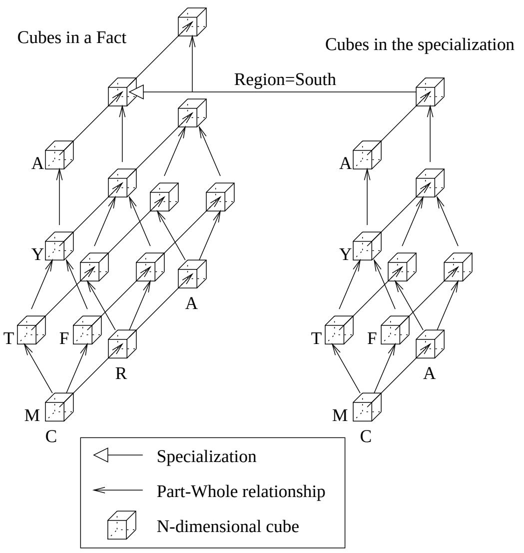

Figure 7: Specialization of a *Fact* by region

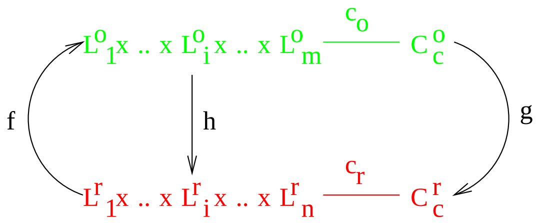

Figure 8: Operations as composition of functions

the instances of a "superclass" into different "subclasses". Notice that the sets of atomic *cells* in each of these "subclasses" (i.e. sets of **north**, **east**, **west**, and **south** *cells*) are in $\mathcal{P}(C_i)$ (being C_i the *cells* in the superclass). Therefore, if they are meaningful for analysts, there will be a *cube* in the *Fact* (maybe after a base change to work with the appropriate analysis dimensions) so that each "subclass" in the specialization corresponds to a *cell* in the *cube* (i.e. the *cube* will have four *cells*, one per region). Thus, we should rather specialize a *Fact* based on a *cube*.

To specialize a *Fact*, we have to choose the appropriate base. Then, the *cube* that contains the *cells* corresponding to the desired "subclasses" is specialized into *cubes*, with exactly one instance, that will be the *cube* at top of the lattice of *cubes* in the new more specific *Facts*. The example in figure 7 shows the same *Fact* in figure 5, that we want to specialize now by region. Thus, we take the *cube* containing data by regions and specialize it in one *cube* with one *cell*. It would give rise to a new *Fact* having a *cubes* sub-graph of that of the "superclass", which will be the lattice having the "subclass" at top. Notice that `Geographic` *Dimension* in the *Fact* specialization is an specialization of the `Geographic` *Dimension* in the original *Fact* (i.e. `All` *aggregation levels* do not coincide).

3.3 Aggregation

We can also find that different *cells* are aggregated to obtain a *cell* about another subject (a different *kind of fact*). For instance, a deal is composed by several individual sales. Notice that *measures* of `Deals` are not necessarily obtained from those of `Sales` (for example, discount in the deal). In this case, we do not group *cells* along any analysis dimension. It does not generate coarser *cells* in the same *Fact*, but *cells* in another *Fact*. There could be, or not, coincidences in analysis dimensions. Depending on it, the *cubes* lattice will have a more or less similar form.

Usefulness of this kind of relationships between *cubes* is twofold. On one hand, it allows to define complex *Facts* from simpler ones, which will improve understandability of data. On the other hand, two *Facts* can be related, so that navigation between them is possible. If we are studying a set of sales, it can be interesting to see data corresponding to deals in which they were done. Coincidences or differences in *Dimensions* do not matter. We should be able to travel from a *cube* to another one just because the aggregation relationship between the *Facts*.

3.4 Derivation

Another possibility is that *measures* in a *cell* are obtained by operating *measures* in *cells* about a different *kind of fact*. If this is the case, we say that there is a derivation relationship between both *cubes* (extensively, between both *Facts*). For example, on analyzing efficiency of employees, some *measures* could be obtained by operating the benefits of some products sold (the best sales, sales involving relevant products, etc.).

Derivation relationships can also be used to hide information, change names, or units of *measures*. Most *Dimensions* will be likely shared by both *cubes*. However, they are related because of relationships between *cells*, not because of the *Dimensions*. This does not correspond to Part-Whole relationships in the lattice of a *Fact*, because both cubes belong to different subjects, and grouping of *cells* is not performed by means of aggregation hierarchies, but by conditions over the *measures* themselves.

4. MULTIDIMENSIONAL OPERATIONS

In this section, we are going to present a definition of the well-known multidimensional operations (i.e. *Slice*, *Dice*, *Roll-up*, and *Drill-across*), plus *Change-base* (corresponding to the relationship explained in section 3.1), based on functions between n-dimensional spaces of *aggregation levels*, and *classes of cells*. Each one of these operations transforms a original *cube* (c_o) and a function (ζ) into a new result *cube* (c_r). Depending on the multidimensional operation, ζ belongs to a different family of functions, and the new *cube* is defined in a different way.

Notice that a *cube* has been defined in definition 4 as an injective function. Thus, as depicted in figure 8, obtaining c_r from c_o, can be seen as mathematical composition of functions ($c_r = \psi \circ c_o \circ \phi$, belonging ψ and ϕ to the families of functions g and f, respectively). Firstly, we can see how *Change-base*, *Dice*, and *Slice*, given c_o and ζ (a function belonging to a family of functions f between the finite spaces defined by cartesian product of *aggregation levels* of each *cube*), we obtain a new *cube* ($c_r = c_o \circ \zeta$). Nevertheless, *Drill-across* does change the *class of cells*. Thus, it works in the opposite way, in the sense that it needs a *cube* c_o and the function ζ (belonging to a family of functions g from a *class of cells* to a *class of cells*) to obtain the new *cube* ($c_r = \zeta \circ c_o$).

Change-base reallocates exactly the same *class of cells* in a new space. ζ belongs to the family of functions:

$$f : L_1^r \times .. \times L_n^r \rightarrow L_1^o \times .. \times L_m^o,\ bijective$$

$$f(x_1^r, .., x_n^r) = (x_1^o, .., x_m^o)$$

Dice selects a subset of the *class of cells*, by choosing subsets of values in one or more *aggregation levels*. ζ belongs to the family of functions:

$$f : L_1^r \times .. \times L_n^r \rightarrow L_1^o \times .. \times L_n^o,$$

so that $\forall i \leq n, L_i^r \subseteq L_i^o$

$$f(x_1, .., x_n) = (x_1, .., x_n)$$

Slice reduces the dimensionality of the original *cube* by fixing a point in a *Dimension*. It can be derived by means of *Dice* and *Change-base* operations. ζ belongs to the family of functions:

$$f : L_1^r \times .. \times L_{i-1}^r \times L_{i+1}^r \times .. \times L_m^r \rightarrow L_1^o \times .. \times L_m^o,$$

so that $\forall j \leq m\ j \neq i,\ L_j^r = L_j^o$

$$f(x_1, .., x_{i-1}, x_{i+1}, .., x_m) = (x_1, .., x_{i-1}, k, x_{i+1}, x_m),$$

where $k \in L_i^o$

Drill-across changes the *class of cells* of the *cube*, but the space remains exactly the same. ζ belongs to the family of functions:

$$g : C_c^o \rightarrow C_c^r,\ bijective$$

Unfortunately, it is not possible to define all operations like these. *Roll-up* changes the space as well as the *class of cells*. Thus, obtaining it as a composition of functions is not possible, because a coordinate in the space of c_r corresponds to several points in c_o. Therefore, there is no ζ, so that c_r is a composition of ζ and c_o.

Roll-up groups *cells* in the *cube* along an analysis dimension, based on a function of the family h that states Part-Whole relationships between *aggregation levels*. It reduces the number of *cells*, but not the number of *Dimensions*. ζ belong to the family of functions h, and the cube c_r is not defined by composition, but:

$$\forall j < n\ j \neq i,\ L_j^r = L_j^o\ and\ h : L_i^o \rightarrow L_i^r\ is\ exhaustive$$

$$c_r(x_1, .., x_i^r, .., x_n) = \bigcup_{h(x_i^o) = x_i^r} c_o(x_1, .., x_i^o, .., x_n)$$

Drill-down is the inverse of *Roll-up*. It can neither be expressed as composition of function, since g would relate a cell in C_c^o with several cells in C_c^r. Thus, we can only apply it, if we previously performed a *Roll-up* and did not lose the correspondences between *cells*. This can be expressed as an "undo" of *Roll-up*, or by means of views over the atomic *class* as in [17].

The *cube* algebra composed by these operations is sound (i.e. the result of all operations is always a *cube*), and complete (i.e. any valid *cube* can be computed as the combination of a finite set of operations). Being sound seams clear, for the first four operations, since composition of functions is always a function, and all functions in these operations are injective. Therefore, all operations result in an injective function from a cartesian product of *aggregation levels* to a *class of cells*. In the case of *Roll-up*, ζ being exhaustive implies that multidimensional operation defines a function over a *class of cells*. Moreover, ζ being a function means the result is injective.

Being complete is also true since if there exists an FD between two *cubes* in the closure of FDs, there is a sequence of operations that allows to obtain one from the other. We can change the left hand side of the function defining a *cube* (i.e. the cartesian product of *aggregation levels*) in two ways: the domain (by means of *Change-base*) and its elements (by means of *Dice*). As can be seen in definition 3, the right hand side of that function, a *class of cells*, is defined by two characteristics: a subject (that can be changed by *Drill-across*), and an *aggregation level* (that can be changed by *Roll-up*).

5. CONCLUSIONS

The aim of this paper is to help on clarifying what multidimensionality means. We have used n-dimensional spaces, and functional dependencies to explain what "measures", "cells", "cubes", and "facts" exactly are, which will help on designing as well as querying multidimensional data.

As summarized in table 1, we have distinguished three different detail levels. At lower detail level, we have *measures* that are grouped into *cells*, if they refer to the same *fact*. Then, we can group *cells* into different *classes* that can be drawn as n-dimensional *cubes* (at intermediate detail level), thanks to that the different analysis dimensions defining a *cube* are functionally independent. Finally, at upper level, several *cubes* representing the same *kind of fact* at different *aggregation levels* are grouped into a *Fact*. Parallelism with the structure of analysis dimensions has been outlined (see [2] for an in depth explanation of the corresponding column).

Moreover, a set of well-known multidimensional operations has been explained in this framework by means of functions. It has been shown as a complete and sound algebra for *cubes*. Specifically, an operation to change the *Dimensions* of a *cube* (i.e. *Change-base*) has been defined. Thus, by having candidate bases and this operation, the most appropriate representation of data can be selected in every situation. It has also been explained and exemplified how different *cubes* and *Facts* can be related by Generalization/Specialization, Aggregation, and Derivation.

Facts have been deeply studied in this work. However, summarizability of measures has been left as future work.

Table 1: Summary table of the different elements found in a multidimensional model

Detail level	Subject of analysis	Analysis dimensions
Lower	*Measures* (grouped into *cells* that correspond to *facts*)	Descriptors[1]
Intermediate	*Cubes* (representing a *class of cells*)	*Aggregation levels*
Upper	*Facts* (representing a *kinds of facts*)	*Dimensions*

[1] These elements at lower detail level allow to select instances of the *aggregation levels*.

Acknowledgements

This 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 a grant from the Generalitat de Catalunya (code 1998FI-00228).

6. REFERENCES

[1] A. Abelló, J. Samos, and F. Saltor. A framework for the classification and description of multidimensional data models. In *12th International Conference on Database and Expert Systems Applications (DEXA)*, volume 2113 of *LNCS*, pages 668–677. Springer, 2001.

[2] 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.

[3] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling Multidimensional Databases. In *Proc. of 13th. Int. Conf. on Data Engineering (ICDE)*, pages 232–243. IEEE Press, 1997.

[4] L. Cabibbo and R. Torlone. A Logical Approach to Multidimensional Databases. In *Advances in Database Technology - EDBT'98*, volume 1377 of *LNCS*, pages 183–197. Springer, 1998.

[5] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Benjamin Cummings, third edition, 2000.

[6] W. A. Giovinazzo. *Object-Oriented Data Warehouse Design*. Prentice Hall, 2000.

[7] 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.

[8] 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.

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

[10] W. Lehner, J. Albrecht, and H. Wedekind. Normal Forms for Multidimensional Databases. In *Proc. of 8th Int. Conf. on Statistical and Scientific Database Management (SSDBM)*. IEEE Computer Society, 1998.

[11] T. B. Nguyen, A. M. Tjoa, and R. R. Wagner. An Object Oriented Multidimensional Data Model for OLAP. In *Proc. of 1st Int. Conf. on Web-Age Information Management (WAIM)*, volume 1846 of *LNCS*, pages 83–94. Springer, 2000.

[12] T. B. Pedersen and C. S. Jensen. Multidimensional data modeling for complex data. In *Proc. of 15th Int. Conf. on Data Engineering (ICDE)*, pages 336–345. IEEE Computer Society, 1999.

[13] 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)*, number 1552 in LNCS. Springer, 1999.

[14] B. Thalheim. *Dependencies in Relational Databases*. B.G. Teubner, 1991.

[15] J. C. Trujillo, M. Palomar, and J. Gómez. Applying Object-Oriented Conceptual Modeling Techniques to the Design of Multidimensional Databases and OLAP applications. In *Proc. of 1st Int. Conf. on Web-Age Information Management (WAIM)*, volume 1846 of *LNCS*, pages 83–94. Springer, 2000.

[16] N. Tryfona, F. Busborg, and J. G. B. Christiansen. starER: A conceptual model for data warehouse design. In *Proc. of ACM 2nd Int. Workshop on Data Warehousing and OLAP (DOLAP)*, pages 3–8, 1999.

[17] P. Vassiliadis. Modeling Multidimensional Databases, Cubes and Cube operations. In *Int. Conf. on Scientific and Statistical Database Management (SSDBM)*. IEEE Computer Society, 1998.

[18] P. Vassiliadis. *Data Warehouse Modeling and Quality Issues*. PhD thesis, Department of Electrical and Computer Engineering, 2000.