Benefits of an Object-Oriented Multidimensional Data Model

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

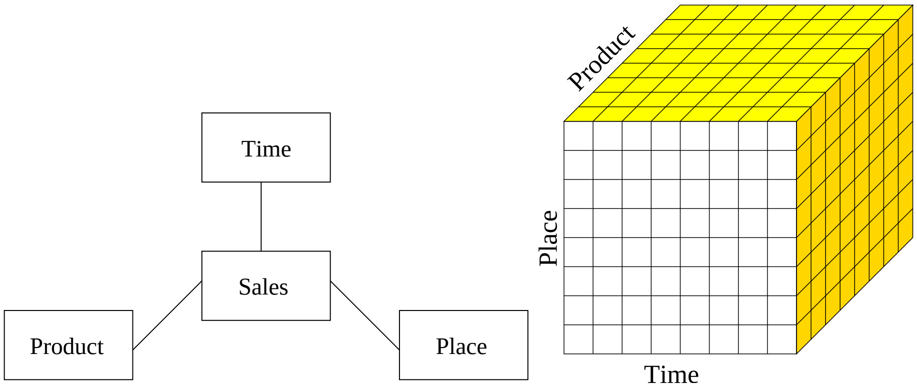
Abstract. In this paper, we try to outline the goodness of using an O-O model on designing multidimensional Data Marts. We argue that multidimensional modeling is lacking in semantics, which can be obtained by using the O-O paradigm. Some benefits that could be obtained by doing this are classified in six O-O-Dimensions (i.e. Classification/Instantiation, Generalization/Specialization, Aggregation/Decomposition, Behavioural, Derivability, and Dynamicity), and exemplified with specific cases.

Key Words: Object-Oriented Data Model, Semantics, Multidimensionality, Data Marts

1 Introduction

Data warehousing is a relatively new research area. W. Inmon is considered the father of the "Data Warehouse" (DW), whose definition can be read in [Inm96]: "A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions". Note that the definition does not refer to the size of the warehouse. However, if it is integrated and we assume that its scope is the whole corporation, it will contain a huge volume of data. Smaller storage systems, called "Data Marts" (DMs), with the same characteristics but devoted to satisfy the needs of a reduced set of users, are defined in order to improve response times. The DMs are customized to obtain good query performance (most of times by means of a query-driven design).

Closely related to the DMs are OLAP (On-Line Analytical Processing) applications, which, as said in [Pen99], are intended for "Fast Analysis of Shared Multidimensional Information" by using the data contained in DMs. Multidimensionality is just a design technique that separates the information into facts (what we want to analize) and dimensions (what we use to analize the facts). It gives rise to schemas with star shape (like that one depicted in figure 1(a)), having the abstraction representing the facts in the middle, and the analysis dimensions around it. The fact in a multidimensional schema is the entity containing the measures to be analyzed (mainly numeric attributes). The analysis



(a) Star schema (b) Cube metaphor

Fig. 1. Multidimensional modeling

dimensions are the points of view that will be used to analyze the fact, and mainly contain descriptive attributes that describe it. Frequently, the "Data Cube" metaphor (depicted in figure 1(b)) is used to explain multidimensionality. Each cell in the cube represents a unit of data (for instance, in the example above, *Sales* as the intersection of a *Product*, *Place*, and *Time*). A given position in every analysis dimension, by defining a position in the space, allows you to select exactly one of those cells. This kind of representation is close to the analysis concepts, and eases the usage of the data by final users.

Multidimensional modeling tries, on the one hand, to make the data schemas understadable to final users, and on the other hand, to improve query performance. The way to do it is by simplifying the data schemas, so that they only contain the essential things (i.e. a fact to be analyzed and its analysis dimensions). These schemas are close to the analysts conception of data, and suggest a specific kind of queries, so that the system can be easyly customized to solve them with good response times.

One of the main focus of data warehousing and OLAP literature is on improving query response times. Always having performance on mind, the direct design of logical (usually Relational) schemas, skipping conceptual modeling, is well accepted. A survey of logical models is in [VS99]. Very few publications recognize the importance of conceptual design in this area. The existence of one fact table related by foreign keys to other flat, denormalized tables (one for each analysis dimension) is encouraged in the literature (see [Kim96]). By denormalizing the dimension tables, unnecessary joins are avoided, adding a little redundancy (with regard to the size of the fact table). However, other people argue for some kind of conceptual modeling (usually by means of E/R, like [GMR98], [SBHD99], [TBC99]). Others even propose the usage of semantically richer models (e.g. description logics in [HS97], or O-O in [Fir97], [TP98]).

Leaving aside the importance of performance, and paying special attention to conceptual design, in [AOSS00], we proposed a seven layers conceptual schemas architecture to integrate the DW in a federation of databases. It allowed us to see everything in a different context. Since [SCGS91] found O-O models as good "Canonical Data Models" for a federation (from a semantical point of view), we suggested the usage of a multidimensional O-O model to conceptually design the DMs. In next sections, we go further discussing the advantages of using the O-O paradigm in multidimensional design. In this paper, we are not proposing a new multidimensional model. The aim is just to outline how the O-O paradigm could be used to help multidimensional modeling by giving some examples.

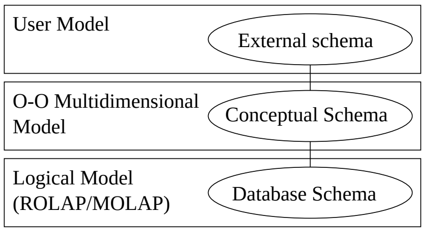


Fig. 2. Three-level schemas

In figure 2, it can be seen the level where the discussion is located. We are not interested neither in the best user model, nor the best kind of database to use (either ROLAP -Relational OLAP-, MOLAP -Multidimensional OLAP-, or even O3LAP -Object Oriented OLAP- presented in [BSH98] could be good). What we want is to present the benefits of an O-O data model to integrate the different multidimensional views and keep the semantics of the data at the conceptual level. If the user wishes to use a different one, it could always be translated to the desired model (maybe due to the tools being used). The same can be said about the internal level, the usage of an O-O multidimensional model does not imply we are storing the data in an O-O database.

Probably, the most important advantage of conceptualizing (multidimensionaly or not) the Universe of Discurs (UoD) by means of an O-O model is that the result is closer to the user conception. It reflects people's way of thinking. Every object or class modeled will have a correspondence with some UoD entity, making it quite easy to be understood. We can find other, not that abstract, benefits in the O-O paradigm, like the usage of Object Identifiers (OIDs), that solves the identification problem of being using keys; the allowance to use Non-First Normal Form (NF^2), which means the design of objects containing non-atomic values; semantics, an Object-Oriented model is semantically richer than others (for instance E/R or Relational); or even its proximity to Object-Oriented Software Engineering, which, for instance, eases some specific tasks like designing a Distributed Object System.

This paper is structured as follows: in section 2, six O-O-Dimensions (i.e. Classification/Instantiation, Generalization/Specialization, Aggregation/Decomposition, Behavioural, Derivability, and Dynamicity) are presented, and their usage in multidimensional modeling, one by one, is exemplified; the paper ends with some conclusions, acknowledgements, and references.

2 Benefits by O-O-Dimension

"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 structures 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.

In [Sal96], there were enumerated six O-O-Dimensions (i.e. Classification/Instantiation, Generalization/Specialization, Aggregation/Decomposition, Behavioural, Derivability, and Dynamicity). Each one of these O-O-Dimension adds a little of Semantic Power to a data model. We are going to see how each one of them helps multidimensional modeling, by allowing to represent different relationships among data.

Along these sections, we call nexus any relationship (tagged or not) between two objects. Usually, it is graphically represented by arrows. The nexus are specialized for every one of the O-O-Dimension to obtain the different meanings.

2.1 Classification/Instantiation

This O-O-Dimension distinguishes between the occurrences and the schema. Every instance is related to, at least, a class in the schema by nexus in this O-O-Dimension. All instances sharing some attributes, and representing related concepts are grouped into a given class. In the same way, all elements in a schema (i.e. classes, nexus, ...) representing related concepts in a data model are grouped into a metaclass. To finish the recurrence, all metaclasses can be grouped into exactly one metametaclass, which is instance of itself. Of special interest in this dimension, present in all data models in one way or another, is the dynamic and multiple classification, explained in [MO96].

Dynamic classification refers to the ability of the instances to change the class they belong to. If we want to analize the sales depending on the goodness of our clients, and we have them classified into different classes, it is a matter of time we want to move a given client from a class to a different (hopefully better) one. We cannot delete the instance of the client in the database and create a new one in the new desired class because we would get a new identity (OID) for it. That is not what we want to represent, we did not lose a client and found a new one. It was just our consideration (classification) about a client what actually changed, and that is exactly what the data model should be able to represent.

On the other hand, multiple classification refers to the possibility of having an instance classified in more than one class (not related by Generalization/Specialization nexus) at the same time. For instance, it is absolutely possible to have a client as provider at the same time. Since there is not any relationship between the *Clients* and *Providers* classes, we need to have the same instance classified at both of them (multiply classified).

These characteristics are always desirable. Specifically in the field of data warehousing, the words "non-volatile" and "time variant", together with the OLAP need of analyzing relatively long periods of time, emphasizes their importance. Dynamic and Multiple Classification are really interesting due to the flexibility needed to represent the big amount of changes present along the long period of time that uses to be taken into account in analysis tasks.

2.2 Generalization/Specialization

Another O-O-Dimension is that of Generalization/Specialization relationships. The nexus in this O-O-Dimension relate two classes (or metaclasses). One of those classes has a more specific meaning than the other. The more general class is called "Superclass" with regard to the specific one, referred as "Subclass". As a consequence of this kind of nexus, we obtain inheritance. That is, the Subclass inherits the properties and methods of its Superclass (or Superclasses). If it is allowed to have more than one Superclass, we gain multiple inheritance (a class inherits from all its Superclasses at a time). Every class will have (besides its own attributes) the attributes and relationships of each one of its Superclasses. Note this is absolutely different from Multiple Classification where an instance is classified in multiple classes.

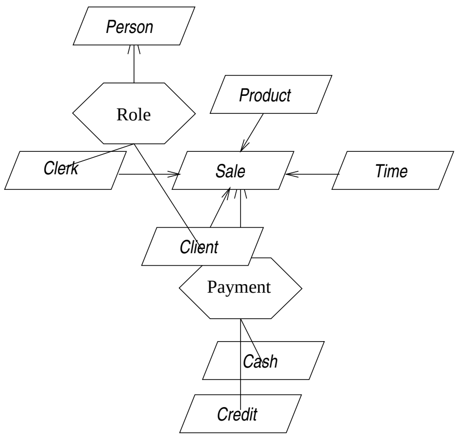


Fig. 3. Sales example

In figure 3, we can see an example of a multidimensional schema[1]. It has *Sales* as fact, and *Clerk*, *Time*, *Product*, and *Client* as analysis dimensions. Thus, the subject of analysis is *Sales*, and we want to analyze it depending on the clerk who sold, the moment it was done, the product sold and the client who bought. Besides that basic information, other details are also represented by means of nexus in this O-O-Dimension:

- the *Sales* fact is specialized in two different facts (i.e. *Cash*, and *Credit*) depending on the kind of payment, and
- two analysis dimensions (i.e. *Clerk*, and *Client*) are related by generalizing them in the same class (*Person*).

Specializing facts, you can generate new data cubes (if they contain any different data), or, at least, show a criteria to select the facts involved in the analysis. In our example, if *Sales* would have different attributes depending on the kind of payment, we would obtain three different cubes to be analyzed i.e. two containing the measures specific to each kind of payment, and another one with those measures shared by both of them. Conversely, if it would not have any other attribute but those common to both kinds of payment, we could analyze the *Sales* cube depending on whether the payment was done by cash, or by credit card. This could also be achieved by just adding an attribute to the facts, but it would give a slightly different tint.

With regard to relating two analysis dimensions, it shows a common domain between them, so that it is allowed to compare the classes, or restrict both at the same time. In the example, the analyst could formulate queries comparing instances of *Client* and *Clerk*, because the data schema shows both as Subclasses of the same class (i.e. *Person*). Moreover, we can consider the possibility of class *Person* being used in a different multidimensional schema, which would become directly related to that of *Sales* by means of the generalization nexus between the analysis dimensions. This would point out the relationship between facts, easing the navigation through the data.

2.3 Aggregation/Decomposition

By means of this O-O-Dimension, it is possible to build new objects as result of the aggregation of others, which in turn can be aggregations, as well. We distinguish two different nexus belonging to it. Attending to their strength, a nexus in the Aggregation/Decomposition O-O-Dimension can denote:

Composition, if the new object is conceived as composed by others, which are its parts. This is called "Part-Whole Relation" by some authors, and implies an existence dependency between both sides of the nexus (i.e. the whole cannot exists without its parts).

[1] The syntax used in the example schemas along this paper is that of BLOOM99, and can be found in [AORS99]

Simple Aggregation, if the aggregating objects are just characteristics of the new one. They could have an existence dependency too, but it is not an implication of the existence of the nexus itself.

The usage of this O-O-Dimension in multidimensional design is mandatory. It helps to represent some of the most common situations, and other maybe not so common:

- Firstly, it helps to define the analysis dimension hierarchies by means of Composition links. A dimension hierarchy must be a lattice with the class corresponding to the maximum level of detail in the facts at the bottom, and a class representing the whole set of points in the dimension at the top. In between, we have other levels corresponding to different data granularities. For instance, if we collect data hourly, the time analysis dimension would have *Hours* class at the bottom, which would compose *Days* above it, which would give raise to *Weeks* and *Months*, and so forth. The lattice would be closed at the top by an *Eternity* class containing exactly one instance representing all time points in the database. These hierarchies are used to roll-up the data in the database, augmenting its granularity. Moving a query up (e.g. rolling-up from days to months) or down (e.g. drilling-down from months to weeks) along a hierarchy we obtain more or less detail in the data.
- On the other hand, using any kind of aggregation, we can relate either analysis dimensions classes or facts classes to their attributes. These attributes will be used to ease the selection of facts to be considered in a given analysis by allowing to group them depending on the values.
- Nexus between the facts class and the bottom classes in every dimension hierarchy are aggregation nexus, as well. They can be Composition or Simple Aggregation nexus, but whether denoting composition or not, a fact will be identified by exactly one object at each linked analysis dimension (or more than one if the dimension has more than one nexus with the facts class). Thus, the nexus with the analysis dimensions will form the class-key of the facts, and that is what really distinguishes them from other attributes. In figure 3, we associate a sale with a point in the 4-dimensional space defined by dimensions *Clerk*, *Time*, *Product*, and *Client*. Therefore, *Sales* functionally depends on those four dimensions.
- Finally, Composition relationships can be found between facts classes. By reflecting them in the schema, we will allow the navigation between them.

The example in figure 4 depicts two facts classes, sharing some analysis dimensions, and related by a Composition nexus. The first facts class is that of *Flight*. We are interested in analyzing each flight depending on the time it takes place, the airline company that owns the plane, and its origin and destination airports (it is related to the corresponding analysis dimensions by Simple Aggregation nexus). At the same time, we want to analyze the sequences of flights that give rise to whole trips sold by travel agencies. *Trip* and *Flight* being connected by a composition nexus represents the fact that an instance of *Trip* is composed of a set of instances of *Flight*. *Trip* is also connected to the corresponding analysis dimensions by Simple Aggregation nexus. Moreover, it is important to notice

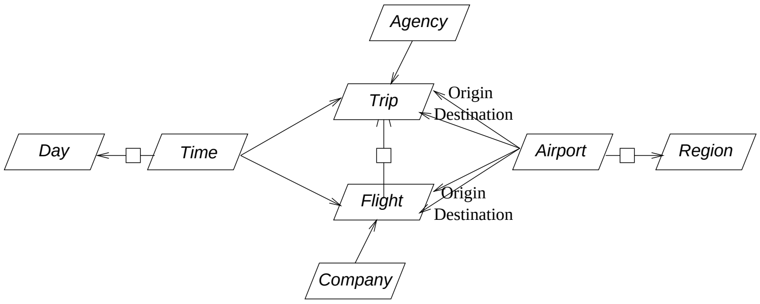


Fig. 4. Traveling example

that two of those dimensions contain more than one class, connected by Composition nexus. A *Region* is composed of a set of *Airports*, in the same way that a *Day* is a set of *Times*.

In order to keep it simple and understandable, the example does not contain the nexus representing the attributes of the facts and dimension classes, which would belong to the Aggregation/Decomposition O-O-Dimension, as well. Because of the same reason, the four classes at the top of each one of the analysis dimension hierarchies are not depicted (i.e. in the time dimension, the *Eternity* class mentioned above). These classes always exist and contain exactly one instance corresponding to the whole set of instances in the lowest granularity level of the dimension. Therefore, one can infer them.

2.4 Behavioural (Caller/Called)

In O-O, the objects interchange messages. A class accepts certain kinds of messages from instances of other classes, which trigger the execution of methods (i.e. queries, updates, calculations, etc.). The nexus in this Behavioural O-O-Dimension, also known as "Caller/Called", show when a class is allowed to invoke a given method in another class.

As pointed out in [Fir97], Relational entities represent tables, purely passive containers for data, and since they are not real objects, are independent of behaviours. The inclusion of methods in the data model helps to model the behaviour together with the data. It looks like a bad idea to have two different, separated models for statics and dynamics. Specifically, in multidimensional modeling, by associating operations to a domain, we will be able to know which aggregation functions can be used on a given fact measure. For instance, as explained in [GMR98], we can find semi-additive attributes (those that are not additive along one or more dimensions), or non-additive attributes (which are additive along no dimension). *Temperature* should be marked as non-additive (nobody could call an additive method on it), and *Inventory level* as semi-additive, since it cannot always be added (e.g. along *Time* dimension). It does not imply that other aggregation operations could be applied on those measures. Therefore, we need to show the applicability of every different operation.

Moreover, methods facilitate the implementation of complex aggregate functions. In an analysis environment, it is important to keep track of the way the measures are obtained. It is not advisable to allow the users to implement their own ad hoc functions. It is error prone, and drive to misunderstandings. O-O concepts such as inheritance, polymorphism, or encapsulation look really nice at this point. For instance, suppose we would like to obtain the delay of a flight, defined as the difference between the expected and real durations (actually, not a complex function). The problem could arise if the expected duration of the flight were kept as a time interval. If this is the case, the difference could be done by subtracting the minimum, maximum, or even midpoint expected duration, which result in completely different values. Probably, it does not matter how it is obtained, but we must ensure it is always calculated in the same, easy to change way to be able to compare the obtained values among different users or even sessions.

Leaving those considerations aside, this O-O-Dimension is also important because of security reasons, but that is completely out of the scope of this paper.

2.5 Derivability

"Semantic Relativism" of a data model 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 Derivability O-O-Dimension, also known as "Point of View", helps to represent the relationships between abstractions in different conceptions of the UoD. The database does not need to physically keep all those conceptions, but only their definitions and different relationships among them. In general, it is not good to store derived data (unless because of performance reasons, not considered in this paper). What we do really need to store is that derived data exists and how it is obtained. Here is the importance of this O-O-Dimension. Derivation mechanisms can be used to easily restructure the schemas to show them in the way the user wants, in order to be closer to his/her thoughts. Summing up, it is used to define derived data.

Some analysts do not mind whether a data is atomically stored in the database or not. In this sense, it is desirable that either derived or atomic measures are treated equally. However, others would like to know how measures are obtained. Therefore, the definition of the derived measures should be in the schema of the database, as Relational views are. It allows either to hide the complexity, or to know where something comes from, depending on the user needs. At the same time, as in the Behavioural O-O-Dimension, this also makes possible that groups of users have available the same definitions.

In multidimensional modeling, it is specially important to have the powerful possibilities offered by this O-O-Dimension. When a fact is being analyzed, what really matters is to be able to see it from as many points of view as possible. Therefore, it is crucial to have the mechanisms to define those different views of the data. For instance, all summarized data is related to its detail data by a nexus in this O-O-Dimension. If we did not have it, we would not have any kind of summarized data.

Going back to our example in figure 4, we can see that the *origin* of a *Trip* would be derived from the *origins* of the *Flights* that compose it, by taking the first one; *destination* would be defined in the same way; the duration of a *Trip* would be function of the duration and taking off times of the different *Flights*; and so on.

2.6 Dynamicity

This O-O-Dimension refers to changes along time. We consider these changes at three different levels:

Object Objects are created, deleted, and also updated. Keeping the history of those updates is often referred as "Versioning".

Class As well as the objects, the data schema can be updated, too. New classes are created, old ones are deleted, and others just modified in what is called "Schema Evolution".

Metaclass In the same way we can modify classes, we can add new metaclasses (notice we can neither modify nor delete them). This means having an "Extensible Data Model".

If we just wanted to represent the current reality, we would not need to consider Dynamicity O-O-Dimension. However, it is common to need past states. Therefore, changes need to be kept, and often stamped with some kind of time tag to know when they happened.

In multidimensional analysis tasks, time is an omnipresent dimension. Moreover, to worsen it, analysts frequently consider a scale of years. If we add how fast things change nowadays, we can see the importance of this O-O-Dimension for multidimensional modeling. It is almost impossible to find a business that has not changed at all in the last three or five years, and those changes must be reflected in the corresponding information system. Leaving aside changes in metaclasses, we want to see the need of considering the other two kinds of changes (i.e. those in objects and classes).

The importance of user requirements makes Schema Evolution an important issue. When the user requirements or conceptions change, it is advisable to change the data schema in accordance with them. A change in a class or nexus should be shown in the schema by connecting the old an the new version with a Dynamicity nexus. By doing it, the analist can easily see the available data, and the meaning of the results he is obtaining. For instance, when the definition of a derived measure changes, the analyst is able to compare the results using the new and the old definitions. Moreover, if some attribute is not kept any more, or a new one is added, the analist can know whether it can be queried or not at a given point in time.

The problem of changes in the data is referred in [Kim96] as "Slowly Changing Dimensions". It arises when attributes in analysis dimension classes are modified. The old values must be kept, because the facts previous to the change are probably still related to them, while the new ones will be referred by the facts occurring from now on. However, both instances represent the same entity in reality, and it has to be outlined by a nexus between them. Clearly, if an airport increases its number of tracks, it would be incorrect to analyze the air traffic previous to the enlargement with regard to the new number of tracks. Therefore, we need to have two instances of the same airport related by a Dynamicity nexus showing that they represent the same object.

3 Conclusions

Along the paper we have introduced the problematic of multidimensional modeling. Probably because of the interest of the industry in the subject, it is being mainly developed in a specially commercial way. This means stressing performance, and passing over semantics and conceptual modeling. Multidimensional semantics are really important because of their proximity to the inherent structure of the problem domain, but they are not the only ones to be represented. Other semantics should not be forgotten. It is not enough having an isolated multidimensional schema reflecting how the user will access the information, leaving aside the representation of other data relationships.

We have shown the applicability of six O-O-Dimensions (i.e. Classification/Instantiation, Generalization/Specialization, Aggregation/Decomposition, Behavioural, Derivability, and Dynamicity) to semantically enrich multidimensional schemas, by exemplifying how they could be used to interelate and integrate some of those schemas. This is a really important point since most people consider isolated star schemas composed by a central fact table, and different flat, denormalized dimension tables arranged around it, each one related to the central table by a foreign key. That is not a bad idea at all, but we are sure there is much more information about the data subject of analysis that the schema could contain, which would be really useful to the analysts, users of the system.

For the sake of simplicity and understandability, the stars use to be represented in an isolated manner. We have tried to stress the necessity of providing an overall view of the data. Multidimensional analysis is used in decision making processes. Therefore, the most global view is provided, the more the schema helps the users. It is really important to offer an integrated vision of the business or subject of analysis, in order to give the analist a unified set of data instead of lots of puzzle pieces. We have proposed to relate the puzzle pieces by means of nexus in the different O-O-Dimensions.

As a future work, we plan to add multidimensional semantics to a semantically rich data model (i.e. BLOOM, which syntax is described in [AORS99], and has been used along this paper). Previously, the role of the O-O-Dimensions in multidimensional design has to be studied in more depth. Moreover, it would be interesting to find patterns in a semantically rich data schema that suggest the possibility of a multidimensional analysis.

Acknowledgements

This work has been partially supported by the Spanish Research Program PRONTIC under projects TIC99-1078-C02-01 and TIC99-1078-C02-02, as well as the grant 1998FI-00228 from the Generalitat de Catalunya.

References

[AORS99] A. Abelló, M. Oliva, E. Rodríguez, and F. Saltor. The syntax of BLOOM99 schemas. Technical Report LSI-99-34-R, Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1999.

[AOSS00] A. Abelló, M. Oliva, J. Samos, and F. Saltor. Information System Architecture for Secure Data Warehousing. In *Proceedings of the Third Int. Workshop on Engineering Federated Information Systems (EFIS'2000)*, Dublin (Ireland), June 2000.

[BSH98] J. W. Buzydlowski, I. Song, and L. Hassell. Framework for Object-Oriented On-line Analytical Processing. In *Int. Workshop on Data Warehousing and OLAP (DOLAP)*, 1998.

[Fir97] J. M. Firestone. Object–Oriented Data Warehousing. Technical report, Executive Information Systems, Inc., 1997. White Paper No. Five.

[GMR98] M. Golfarelli, D. Maio, and S. Rizzi. Conceptual Design of Data Warehouses from E/R Schemes. In *Proceedings of the Hawaii Int. Conference On System Sciences*, 1998.

[HS97] Mohand-Saïd Hacid and Ulrike Sattler. An object-centered multidimensional data model with hierarchically structured dimensions. In *Proceedings of the IEEE Knowledge and Data Engineering Workshop, Newport Beach, CA, USA*, pages 65–72. IEEE Computer Society, November 1997.

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

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

[MO96] J. Martin and J. Odell. *Object-Oriented Methods: Pragmatic Considerations.* Prentice-Hall, 1996.

[Pen99] N. Pendse. The OLAP report. Bussiness Intelligence Ltd., 1999. http://www.olapreport.com/fasmi.htm.

[Sal96] F. Saltor. Semántica de datos. In *Panorama Informático*, pages 39–64. Federación Española de Sociedades de Informática (FESI), 1996.

[SBHD99] C. Sapia, M. Blaschka, G. Hoefling, and B. Dinter. Extending the E/R Model for the Multidimensional Paradigm. *Lecture Notes in Computer Science*, 1552:105–116, 1999. ER Workshops 1998.

[SCGS91] F. Saltor, M. Castellanos, and M. García-Solaco. Suitability of Data Models as Canonical Models for Federated DBs. *ACM SIGMOD Record*, 20(4):44–48, 1991.

[TBC99] N. Tryfona, F. Busborg, and J. Christiansen. starER: A Conceptual Model for Data Warehouse Design. In *Int. Workshop on Data Warehousing and OLAP (DOLAP)*, pages 3–8, 1999.

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

[VS99] P. Vassiliadis and T. Sellis. A Survey of Logical Models for OLAP Databases. *ACM SIGMOD Record*, 28(4):64–69, December 1999.