An Object Oriented Approach to Multidimensional Database Conceptual Modeling (OOMD)

Trujillo, J.*, Palomar, M.**
Grupo de Programación Lógica y Sistemas de Información
*Dpto. Economía Financiera
**Dpto. Lenguajes y Sistemas Informáticos
Email: * {juan.trujillo@ua.es}, ** {mpalomar@dlsi.ua.es}

Abstract. In the recent past, there has been an increasing interest in multidimensional databases (MDB) and On-line Analytical Processing (OLAP) scenarios. Several multidimensional models have been proposed in the last days. However, very few works have been focused on the area of multidimensional database conceptual modeling. Moreover, they are either conceptual extensions to the classical multidimensional model or translations from classical database conceptual models (such as the Entity-Relationship model). Nevertheless, we take the concepts and basic ideas of the classical multidimensional model (dimensions and facts) to propose a revolutionary approach based on the Object Oriented (OO) Paradigm to MDB conceptual modeling. Then, the basic elements of our Object Oriented Multidimensional Model (OOMD) such as dimension classes and fact classes are introduced. We then present cube classes as the basic structure to allow a subsequent analysis of the data stored in the system. We fairly believe that the utilization of the OO Paradigm will provide us a general conceptual model to MDB conceptual modeling in a more flexible, natural and simple way than the models proposed until now.

1 Introduction

Companies can adopt strategic decisions that may suppose a competitive advantage with respect to their competitors. In this context, the concept of Data Warehouses (DW) emerges in the decade of the ninety ([11], [12]) as an integrated data collection of the company oriented to decision making. The success of these DW is not only demonstrated by the amount of commercial products that have been emerging lately ([2], [18], [19]), but also in the proliferation of research projects and topics ([14], [21], [10] and [20]). A more detailed on-line bibliography focused on DW and OLAP technology can be found in [17].

However, the analysis of this historical data (DW) is carried out through user final tools that are based on OLAP technologies [6]. The storage structures (derived from the DW) used by these techniques are known with the name of hypercubes or multidimensional fact tables. These structures are suitable for this purpose since they represent in an intuitive way the factual data according to the characteristics (dimensions) that are considered relevant to the analysis. We will introduce a simple example to handle these concepts in the following section.

Traditional database systems are inappropriate for multidimensional analysis since they are optimized for On-line Transactional Processing (OLTP) in which an enormous number of concurrent transactions containing normally few records are involved. Nevertheless, OLAP techniques execute few complex queries involving a huge number of records. Current technology provides both OLAP servers and user final tools for the development of MDB. With reference to the OLAP servers we can find either relational systems (ROLAP) or multidimensional systems (MOLAP). A ROLAP system is an extended relational system that maps operations on the multidimensional data to standard relational operations (SQL). On the other hand, the MOLAP systems store and manipulate data directly in special structures called multidimensional arrays.

Modeling Multidimensional Databases (related work)

In both systems, MDB are modeled depending strictly on the corresponding implementation (ROLAP or MOLAP systems). Some problems emerge from this form of proceeding. Firstly, it does not exist a general conceptual model (independent of subsequent implementation details) to MDB conceptual modeling and valid for a subsequent implementation in any system. Secondly, the requirements posed for a subsequent data analysis need take into consideration tedious details of the data physical organization more than of its logical aspects (as also argued in [4]). We consider that these and other subsequent problems will be solved with the proposal of a general conceptual model independent of any subsequent implementation details. Furthermore, we fairly believe that the application of the OO Paradigm will provide us a general conceptual model with the total independence from physical aspects (as previously commented) to MDB conceptual modeling in a more natural and simple manner than the models proposed until now.

The traditional model used for MDB modeling is the known “star model” ([2], [11] and [12]) and its variants (“snowflake”, “fact constellation” and so on), mainly if a subsequent implementation is carried out in a ROLAP system. This model consists of two kinds of relational tables, dimensional tables and fact tables. The previous ones contain characteristics of the factual data and the latter ones contain the factual data itself (whose values are represented in some attributes called measures or fact attributes). Data contained
in dimensional tables present different levels of granularity in most cases, which is not taken into account in the model due to the fact that it only considers relational tables. We believe that this model is not suitable for MDB conceptual modeling in the sense that it refers to relational tables while the MDB modeling is being accomplished. However, our OOMD approach defines abstract objects without any reference to tables or their subsequent implementation. Furthermore, we define the cube as other collection of abstract objects on which a group of operations (defined/permitted on it) are carried out to allow a subsequent analysis of the data contained in it. Moreover, a special relation is applied to these objects to express the granularity of data in the conceptual model.

To our best knowledge, only three works focused on the conceptual design of MDB have been presented until now. In [7], the conceptual design is outlined from the schemes provided by the ER model of the company OLTP systems. Nevertheless, the nature of DW makes necessary, in most cases, to include data that is not in the original OLTP systems and therefore, it does not exist in the ER schemes. Moreover, in this approach, facts and dimensions are defined from the entities and relations of these schemes; i.e. only data is taken into account. However, our approach provides a higher level of abstraction since not only are the data static properties (data itself) considered, but also the dynamic ones (operations to be applied on data). Furthermore, we can find several references such as [13] that consider the ER model inappropriate for DW conceptual modeling.

Another proposal ([14]) extends a multidimensional model (MD) proposed in [3] in which a declaratory query language and the research of its expressiveness were mainly introduced. This extension of the MD defines a schema of MDB as a schema of fact tables to provide a general design methodology for MDB. This MD considers all the necessary concepts for the conceptual design of MDB. However, we consider that not only does it use the concept of tables in the conceptual design phase, but it is also closed to the classical relational model (with the normal extensions to allow the definition and subsequent multidimensional data analysis). We fairly believe in a more revolutionary proposal and a higher level of abstraction in the design phase. According to this, our OOMD considers abstract objects and we do not take any assumption neither about the logical model to be used nor fact tables. Moreover, this higher level of abstraction will allow us to define a cube class in which both static and dynamic properties will be taken into account. As a consequence, we will achieve a more restrictive way on a subsequent data analysis phase.

In [15] we find the first OO approximation for the design of MDB in the proposed Nested Multidimensional Data Model, which is a conceptual extension to the classical multidimensional model to allow us to model complex OLAP scenarios. It introduces the concept of multidimensional object to define the multidimensional cube in which a group of operations are defined on it to permit a subsequent data analysis. This cube consists of dimensional and classification attributes (for a previous classification of the dimensional attributes) to express data features. Our approach only considers dimension attributes on which an attribute roll-up relation (ARR) is defined. We consider that this relation can provide us a higher flexibility in the design phase in the sense that new ARR’s may be defined at any time without being necessary to change the existing ARR’s. Nevertheless, the main contribution of [15] is to demonstrate that cube structures are nested and therefore, their analysis can be simplified. However, our OOMD introduces for first time in this field the concept of classes to encapsulate data and operations to apply on it, which provide us a higher level of abstraction.

Multidimensional cubes can be seen as different classic views of databases that users can have. According to [15], we consider that the definition of an abstract entity (abstract objects) to encapsulate data that the cube contains as well as the operations permitted on it in the design phase will achieve a clearer design of MDB, higher design flexibility and a better restriction during the data analysis phase.

On the other hand, several multidimensional models (formal logical models) have been proposed. However, they are mainly guided to the study of OLAP query languages. A common feature to all of them is that they are guided to a specific implementation and therefore, they are less suitable to the conceptual design of MDB. In the rest of this section, we will make reference to three multidimensional models that we consider the most relevant ones presented until now.

In [1], a model based on the notion of the multidimensional cube (whose first definition was introduced in [8]) and an algebraic query language to allow analysis operations on this cube are proposed. However, there are aspects that we consider relevant in the design of MDB as the dimension attribute classification hierarchy that are considered using a special operator in the query language. However, in our proposal, this relevant element is considered from the first step in the conceptual design providing the ARR on the dimension attributes. Furthermore, the model proposed in [1] is based on the idea of a subsequent mapping to the traditional model adopting the presumption of a subsequent implementation in a relational system.

In [9], a logical model for MDB in which the contents are clearly separate of the structural aspects is proposed. As above-commented, we find basic elements in the design of MDB as the aggregation levels of dimensions that are not explicitly considered. Furthermore, this model is focused on the development of a query language based on the structures previously defined. Finally, we should say that the best success of this model is its complex mapping to the relational model. On the other hand, in [16] a multidimensional model (MDD) for OLAP techniques is proposed. In this model, a query language called "grouping algebra" based on a basic component called multidimensional cube is developed.

In conclusion, no general conceptual model with a high level of abstraction and independent of any subsequent implementation, and consequently, suitable for the conceptual design of MDB has been proposed until now. However, the current proposals are conceptual extensions to the classical multidimensional model, translations from classical database conceptual models such as the E-R model or mappings to the relational model.
**Paper layout.** In next section, we introduce a simple example, which will be used throughout the paper, to handle all the concepts and basic ideas of the classical multidimensional model. In the third section we present the basic definitions of our proposal, i.e. the notion of fact classes and dimension classes with an adequate domain definition. In the fourth section we introduce the concept of cube class as an abstract entity on which data (objects) and operations permitted on it (them) are encapsulate to allow a subsequent data analysis. Finally, in the fifth section we present the conclusions and future works that emerge from this first approach.

2 The classical multidimensional model throughout an example

We wish to design a MDB for a company whose commercial activity is devoted to the vehicle sales to different stores. We wish to know details on the vehicle sales, concretely we wish to analyze the sold units and which is the sales value. Furthermore, we wish to know features of the store, vehicle and date of the sales. Concretely, we wish to know of the store its name, country, area, city and street where it is located, of the vehicle its group, family and brand and of the date its year, semester, month, date and day of the week.

In figure 1 a multidimensional cube is presented to show the general idea of the multidimensional data model. In each cell of the cube, we will be able to store the concrete data of the sales that are studied, i.e. the sold units and their values. This particular data receives the name of fact attributes or characteristics (or measures). On the other hand, it can be observed that the cube has three sides (dimensions), one for each feature that we wish to analyze, i.e. vehicle, store and date. Finally, each dimension consists of a number of attributes (features) called dimension attributes that describe each dimension in more detail (described in the previous paragraph).

A last relevant feature of these cubes is the classification hierarchy that is defined on the attributes along each dimension, which permits the values of these attributes to be assembled (classified or aggregated). The oriented arrows in figure 1 show the attribute classification hierarchy that has been defined along each dimension. This will allow us to aggregate attribute values (roll-up operation) or to analyze them in a larger detail level (drill-down). In our particular example, we suppose that all the stores in our database are located in Spain (Spain is divided into four areas, North, South, East and West). Furthermore, we have currently located sales in the cities of Alicante, Valencia and Sevilla. By analyzing the sales with respect to the cities, we can accomplish a roll-up operation along the store dimension and from the city attribute to the area one. Thus, we will obtain the result that we have obtained sales in the East area (Alicante and Valencia belong to the East area) and in the South one (Sevilla belongs to the South area). This classification hierarchy could also be written as city ➔ area ➔ country. The reverse operation will be to crumble those areas to obtain the concrete cities where we have sold vehicles. Then, we will accomplish a drill-down operation along the store dimension and from the area attribute to the city one. Thus, we will obtain the result that the cities in the East area where we have sold vehicles are Alicante and Valencia.

In addition to the operations roll-up and drill-down, according to [5] we can slice/dice (selection and projection along one or more dimensions) and pivoting (re-orienting the multidimensional view of data). Other authors such as [1] increase the number of operations to apply on the cube (for example, they propose operations to add and delete a dimension on a cube).

3 Dimension class and fact class

In the real life there are objects that have common characteristics. Following an OO Paradigm we will group objects in classes. These classes will encapsulate both static and dynamic properties of these objects. For example, with reference to stores, their static properties are that all of them are located in a city and that a city belongs to a concrete area and subsequently the store is placed in a specific country (attributes and their attribute classification hierarchy). Dynamic properties are the operations that can be accomplished on objects to change their characteristics (for example, for change the store location). However, we know that in a context of MDB objects (data) are static in the sense that once they exist in our system they will not modify their characteristics (static properties) until they are carried to an auxiliary store. Therefore, the two first actions to apply on these objects will be to create and destroy them.

Following the nomenclature of the multidimensional model and applying the OO paradigm, we will firstly distinguish among dimension classes (DC) and fact classes (FC). DC will contain dimension objects (DO) that provide characteristics of the factual data, while FC will contain fact objects (FO). The latter classes will be built from DC. We will firstly introduce the necessary definitions to allow us to define DC and FC (basic elements in our OOMD model).

**Definition 1** Let attributes (A) be an n-tuple (a1, a2,..., an) where each element (ai) is a feature that have the objects of a specific class, i.e. this tuple characterizes the objects of a class.

**Definition 2** Let V_i be the set of values or instances that can be taken by an attribute a_i ∈ A following the definition of Data Abstract Type (DAT) of ai.

**Note** that we will firstly take into account basic DAT’s such as string, real, float, integer and so on with their possible operations.

![Figure 1](image-url)
Definition 3 Let \( a_i \subseteq A \) an attribute and \( vi \) be the set of possible values (instances) to be taken by \( a_i \), we define the domain function as \( \text{dom}: a_i \rightarrow V_i \), where for a given attribute \( a_i \), it will return a subset of values \((v_i \subseteq v_i)\) for \( a_i \).

Definition 4 Let \( A \) be a set of attributes being hold by the objects (elements) of a particular class, we say that the key attribute \((KA)\) is an attribute \( a_i \subseteq A \) that defines univocally every object of that particular class.

Definition 5 Let \( A \) be a set of attributes being hold by a set of dimension objects \((DO)\), we define dimension attributes \((DA)\) as a n-tuple \((a_1, a_2, ..., a_n)\) that characterizes these DO, where the KA of these DO is not included in this tuple.

Definition 6 Let \( A' \subseteq A \) be a subset of attributes, we define an attribute roll-up relation \((ARR)\) as an n-tuple \((a_1, a_2, ..., a_n)\) where a partial order relation is defined, such that \( a_1 \leq a_2 \leq ... \leq a_n \) and that given two attributes \( a_i, a_j \) such that \( a_i \leq a_j \) there is not any \( a_k \) such that \( a_k \leq a_i \leq a_j \).

Note that this relation can be applied to any subset of \( A \) \((A')\), even though the KA is within \( A' \).

We can observe that this relation to allow us to define the attribute classification hierarchy, i.e. \( a_1 \supseteq a_2 \supseteq ... \supseteq a_n \) \((a_i \text{ rolls up to } a_2, a_2 \text{ rolls up to } a_3 \text{ and so on, as commented in the second section})\).

Definition 7 Let \( ARR \) be an attribute roll-up relation, we define the roll-up domain function as \( \text{domroll-up}: v_i, a_i \rightarrow v_j \) to obtain the attribute values according to the classification hierarchy.

Conversely, we define the drill-down domain function as \( \text{domdrill-down}: v_j, a_j \rightarrow v_i \).

Definition 8 Let any kind of object be, \( E \) is the set of events (operations) to apply on these objects. These operations are “new” and “delete” to create and destroy any object respectively.

Definition 9 We define a dimension class \((DC)\) as a tuple \((A, ARR, E)\), where
- \( A= KA \cup DA \)
- \( ARR \) is a possible attribute roll-up relation defined on \( A' \), where \( A' \subseteq A \)
- \( E \) is the set of events allowed on the class objects, i.e. “new” and “delete”

Example We apply these definitions to the example presented in section 2. According to this, we will have three dimension classes, the vehicle dimension class, store dimension class and date dimension class. We now apply the previous definitions to only one dimension class (the store class). The reader can easily apply the definitions to the rest of the dimension classes.

We will have a store dimension class as the tuple \((A, ARR, E)\), where
- \( A= KA \cup DA \), where,
  - \( KA \) is the store_code,
  - \( DA=\text{store_name, country, area, city, street} \),
  - \( ARR=\text{store_name, city, area, country} \)
  - \( E=\text{new, delete} \)

We will then show some examples of how the domroll-up and domdrill-down functions will operate to obtain aggregated data, taking into account the different values that currently exist in the system (see the example in section 2).

\[
\begin{align*}
\text{domroll-up: alicante, area} & \rightarrow \text{East} \\
\text{domroll-up: sevilla, area} & \rightarrow \text{South} \\
\text{domroll-up: East, country} & \rightarrow \text{Spain} \\
\text{domroll-down: Spain, area} & \rightarrow \text{East, South} \\
\text{domdrill-down: East, city} & \rightarrow \text{alicante, valencia} \\
\text{domdrill-down: South, city} & \rightarrow \text{sevilla}
\end{align*}
\]

Note that these functions together with the definition of \( ARR \)'s are necessary to express the granularity of data. The following step is to build the fact class \((FC)\), which will contain fact objects \((FO)\), from dimension classes. Before the fact class definition we define the concept of fact attributes or measures as follows.

Definition 10 Let \( A \) be a set of attributes and \( FO \) be a set of fact objects, we define measure or fact attributes \((FA)\) as a n-tuple \((a_1, a_2, ..., a_n)\) that characterize these \( FO \), where the KA of these \( FO \) is not included in this tuple. These are the emerging characteristics that provide specific information (factual information).

Definition 11 Let \( DC_1, DC_2, ..., DC_n \) be \( n \) dimension classes, a fact class \((FC)\) from these \( n \) dimension classes is a tuple \((A, ARR, E)\), where
- \( A= KA \cup FA \), where,
  - \( KA \) is the key attribute for \( FO \)
  - \( FA \) is the set of fact attributes or measures
- \( ARR \) is a possible attribute roll-up relation defined on a subset of \( FA \) (\( FA' \subseteq FA \))
- \( E \) is the set of events allowed on \( FO \), i.e. “new” and “delete”

Example Following the same example, we will obtain the sales fact class from the vehicle, store and date dimension classes as the tuple \((A, ARR, E)\) where:
- \( A= KA \cup FA \), where,
  - \( KA=\text{sales_code} \)
  - \( FA=\text{sales_number, value} \),
  - \( ARR=() \)
  - If \( ARR \) is an empty relation we will not apply the functions domroll-up and domdrill-down on fact class attributes \((FA)\).
  - \( E=\text{new, delete} \)

4 Cube classes

Once we have defined the different object classes that we will have in our system, we proceed to the definition of the cube classes \((CC)\) to permit a subsequent data analysis. The
CC can be considered as the classical cube and the objects belonging to the CC as the data contained in the cube on which the analysis operations will be applied.

Concerning the CC, their definitions are always based on a fact class, and therefore, they will contain data from dimension classes. This means that we need n-dimension classes and one fact class to build this basic CC. With reference to the operations permitted on the objects of the CC our approach is twofold. On the one hand, we have the operations that can create or destroy an object of the class ("new"/"delete"). On the other hand, those that permit different analysis on the data contained in the CC.

**Definition 12** Let DC1, DC2,...,DCn be a n dimension classes and FC be a fact class built from these n dimension classes, we define a cube class (CC) as a tuple (DC,FC,A,C,E,CE), where

- DC is the set of dimension classes that have been used for constructing the fact class
- FC is the fact class from which the cube class has been built
- A = KA ∪ CFA ∪ CDA where,
  - KA is the set of key attributes of the FC.
  - CFA is a subset of FA from the FC
  - CDA is a subset of the DA from DCi

**Note.** By defining both dimension attributes and fact attributes while constructing the CC, we use the following format: class_name.attribute_name to allow us to know which object classes every attribute belongs to.

- C is a condition n-tuple (a1=v1, a2=v2,..., an=vn) where ai are dimension attributes and vi the set of values that must fulfill each attribute ai to select the objects that will integrate this CC.

**Note.** If ai=ai with vi ≠ vi both kinds of different objects will be selected.

- E is the set of operations allowed on the objects of the CC, i.e. "new" and "delete".
- CE is the set of events (operations) permitted on the cube class, i.e. operations that are applied to the set of all the cube objects contained in the class. These operations according to [1] and [5] will allow the subsequent analysis of the data contained in the cube achieving an intuitive navigation on the cube class.

**Example** Following the class definition provided in the previous section, we will show an example of a CC construction. We wish to analyze the sold vehicle number where the group of vehicle is "four wheels" and the store country is "Spain" grouped by the vehicle family and brand and by the store area and name. According to this specification, we will construct the following cube class.

A CC on the sales fact class will be the tuple (DC, FC, A, C, E, CE), where

- DC = store dimension class, vehicle dimension class
- FC = sales fact class
- A = KA ∪ CFA ∪ CDA where,
  - KA = sales_code
- CFA=(sales.number)
- CDA=(vehicle.group, vehicle.family, vehicle.brand, store.country, store.area, store.name)
- C=(vehicle.group="four wheel vehicles", store.country="Spain")
- E=(new, delete).
- CE will be the operations commented in the previous definition

In Table 1., we show in a more intuitive way (the classical multidimensional view of data) the result of the previous example CC construction. The condition C is printed in bold letters, whereas the result of C (objects contained/selected in this CC) is printed in normal font. On the other hand, the attributes shown in this figure are CFA and CDA, i.e. the KA’s are not shown. Finally, the name of this cube class (sales) is printed in cursive.

| Sales | Vehicle.group="4 wheel vehicles"
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Man</td>
<td>BMW</td>
</tr>
<tr>
<td>east</td>
<td>10</td>
</tr>
<tr>
<td>Court</td>
<td>20</td>
</tr>
<tr>
<td>south</td>
<td>Vals</td>
</tr>
<tr>
<td>Court</td>
<td>41</td>
</tr>
</tbody>
</table>

**Table 1.**

### 5 Conclusions and future work

We have presented a first revolutionary OO approach to MDB conceptual modeling. We have firstly defined the two basic elements of our OOMD, i.e. dimension classes and fact classes. We have then defined the cube class (from dimension and fact classes) to encapsulate both data and operations allowed on it, which will allow us to accomplish a subsequent data analysis. From our point of view, the star model only considers relational tables and therefore, basic elements of MDB’s such as the classification hierarchy on attributes along dimensions cannot be expressed. Our approach, however, provides mechanisms (ARR and domain functions) to achieve this issue.

This revolutionary approach provides a higher level of abstraction (encapsulates both data and operations in one structure) than the models proposed until now as well as a more restrictive way to a subsequent analysis of the data contained in the cube. Unlike other models which are either extensions to the classical multidimensional model or mappings from the classical database conceptual models (such as the ER model), our OOMD is an independent and revolutionary approach since it does take into consideration neither any of these assumptions nor any subsequent implementation.

We are currently extending the cube class set of operations to allow us to accomplish a subsequent data analysis as well as being able to construct a cube class hierarchy, i.e. being able to build a cube from others. We are also extending the cube class definition to allow us to define the cube class from n fact classes instead of only one. On the
other hand, we wish to provide a rich conceptual model and its graphical user interface to facilitate the definition of our OOMD structures (to make the model more intuitive) as well as to provide an easy set of point-and-click operations to accomplish a subsequent data analysis. By finishing these and other extensions that are currently being carried out, we wish to extend this conceptual model to a formal logical model. In conclusion, we fairly believe that this first OOMD approach is a solid basis for solving the MDB conceptual modeling problems derived from the lack of a formal and independent general conceptual model. Our attempt is to provide a more intuitive and complete conceptual model than the “star model” providing the necessary mechanisms to consider all relevant aspects of MDB’s.

Acknowledgements

We want to thank Dr. J. Samos and the anonymous reviewers of the DOLAP workshop (CIKM’98) for their detailed comments, which helped us improve this paper.

6 References