Automatically Generating Database Schemas into OLAP Tools from Object-Oriented Conceptual Models*

Juan Trujillo and Sergio Luján-Mora
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Ap. correos 99. E-03080. Spain
{jtrujillo,slujan}@dlsi.ua.es

Abstract

Graphical conceptual models for OLAP applications should semi-automatically generate the database schema and the multidimensional (MD) model for a specific target commercial OLAP tool. However, this generation process is not immediate as the semantics represented by these conceptual models are different from those considered by the underlying MD models of OLAP tools. In this paper, we present an overview of this generation process from graphical conceptual modeling into target commercial OLAP tools. The conceptual model is accomplished by using an object-oriented approach, based on the Unified Modeling Language (UML), that allows us to represent both the MD model and initial users' requirements at the conceptual level. In this generation process, some semantics represented in the conceptual model are transformed into those considered by the underlying MD model of the target OLAP tool.

1. Introduction

OLAP tools, based on the multidimensional (MD) model, are the most popular front-end tools to analyze data in data warehouses. These tools consider the implementation of the MD model from two different perspectives: the structural part and the dynamic part. The former refers to the structures that form the database schema to house MD data and, the underlying MD model that provides the OLAP tool to consider the MD semantics (e.g. facts, fact attributes, dimensions, hierarchy paths, etc.). The dynamic part refers to the definition of final users' requirements and OLAP operations[1] to further analyze data.

These OLAP tools are mainly MOLAP (Multidimensional OLAP) or ROLAP (Relational OLAP) depending on the kind of structures used to implement the database schema of the MD model. Each commercial OLAP tool provides its own MD model to consider the main semantics and concepts of MD modeling. As a consequence, different OLAP tools consider different semantics and properties of the MD model. These tools provide a graphical user interface to define the MD model from the structures (multidimensional vectors or relational tables) that form the MD database schema. Therefore, they first require the database schema be defined. Once both the database schema and the MD model have been defined, an easy point-and-click graphical user interface allows to define initial users' requirements.

On the other hand, several proposals have lately been made to accomplish the graphical conceptual design of OLAP applications [1, 3, 7, 9]. Ideally, within the context of OLAP applications, these graphical proposals should semi-automatically generate the implementation of the MD model to be directly queried in an OLAP commercial tool. To do this, the generation process should generate the implementation of the needed elements regarding the structural and dynamic parts as above-described.

To the best of our knowledge, the work presented by Hahn *et al.* in [4] is the only one in considering this semi-automatic generation process with outstanding results. In this process, only the implementation of the underlying MD model into the target OLAP tool is taken into consideration. As pointed out in [4], the semantics and concepts considered by the different MD models of OLAP commercial tools are different from those considered by the graphical conceptual approaches above-presented. Therefore, it is necessary to transform some semantics and properties in the

*This paper has been partly supported by the Spanish Ministery of Science and Technology, project number TIC2001-3530-C02-02.

[1] Each OLAP tool will provide the set of OLAP operations that can be applied from initial users' requirements.

generation process trying to preserve their initial semantics as much as possible.

In this context, the GOLD model [9] is an object-oriented (OO) conceptual model to accomplish the conceptual design of both the structural and dynamic parts of OLAP applications. To facilitate the use of the modeling constructors, the model provides a Unified Modeling Language (UML) [6] compliant graphical notation in which each modeling constructor has its corresponding graphical notation. This fact allows the designer to accomplish a correct conceptual design without the need of parsing the graphical notation.

In this paper, we present how to semi-automatically generate the implementation of the structural part from a GOLD model into Informix Metacube[2] (IM). The process first generates the star schema that will house the MD data and, secondly, the corresponding MD model of IM from the GOLD modeling constructors used in the conceptual design. Nevertheless, some of the constructors do not have their corresponding representation into IM and, therefore, some are ignored while others are transformed trying to preserve their initial semantics as much as possible. Due to space constraints, we have left the dynamic part for a future paper.

The remainder of this paper is organized as follows. Section 2 summarizes how to accomplish the conceptual modeling of the structural part of OLAP applications with the GOLD model. Section 3 presents how IM stores information about its underlying MD model. Section 4 describes the generation process of the structural part from a GOLD model into IM. Finally, in Section 5, we present the conclusions and sketch some works that are currently being carried out.

2. OO Multidimensional Modeling

In this section, we summarize[3] how an OO MD model, based on the UML, can represent main structural and dynamic MD properties at the conceptual level. Most of the MD features considered by this approach such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, and non-strict and complete hierarchies are misunderstood by most of the conceptual MD models. In this approach, the main structural properties of MD modeling are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions.

Facts and dimensions are represented by *fact classes* and *dimension classes*, respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of *n* dimension classes. The flexibility of shared aggregation in the UML allows us to represent *many-to-many* relationships between facts and particular dimensions by indicating the 1..* cardinality on the dimension class role. For example, in Figure 1 (a), we can see how the fact class Sales has a many-to-many relationship with the dimension class Product and a one-to-many relationship with the dimension class Time.

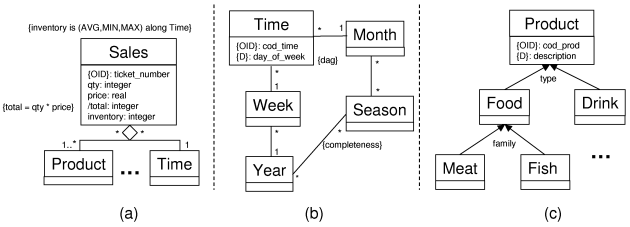

Figure 1. Multidimensional modeling using the UML

By default, all measures in the fact class are considered additive. For nonadditive measures, additive rules are defined as constraints and are included in the fact class. Furthermore, derived measures can also be explicitly considered (indicated by /) and their derivation rules are placed between braces near the fact class, as shown in Figure 1 (a).

This OO approach also allows us to define identifying attributes in the fact class, by placing the constraint $\{OID\}$ next to an attribute name. In this way we can represent *degenerate dimensions* [2][5], thereby providing other fact features in addition to the measures for analysis. For example, we could store the ticket number (ticket_number) as a degenerate dimension, as reflected in Figure 1 (a).

With respect to dimensions, every *classification hierarchy* level is specified by a class (called a *base class*). An association of classes specifies the relationships between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the dimension class (constraint $\{dag\}$ placed next to every dimension class). The DAG structure can represent both alternative path and multiple classification hierarchies. Every classification hierarchy level must have an *identifying* attribute (constraint $\{OID\}$) and a *descriptor* attribute[4] (constraint $\{D\}$). These attributes are necessary for an automatic generation process into commercial relational OLAP (ROLAP) tools, as these tools need to store these attributes in their metadata. The multiplicity *1* and *1..** defined in the target associated class role addresses the concepts of *strictness* and *non-strictness* respectively. Strictness means that an object at a hierarchy's lower level

[2] We have chosen this tool as it is one of the most leading ROLAP products.

[3] We refer the reader to [9] for a complete description of this approach.

[4] A descriptor attribute will be used as the default label in the data analysis.

belongs to only one higher-level object (e.g., as one month can be related to more than one season, the relationship between both of them is non-strict). Moreover, defining the {*completeness*} constraint in the target associated class role addresses the completeness of a classification hierarchy (see an example in Figure 1 (b)). By completeness we mean that all members belong to one higher-class object and that object consists of those members only. For example, all the recorded seasons form a year, and all the seasons that form the year have been recorded. Our approach assumes all classification hierarchies are non-complete by default.

The *categorization of dimensions*, used to model additional features for class's subtypes, is considered by means of generalization-specialization relationships. However, only the dimension class can belong to both a classification and specialization hierarchy at the same time. An example of categorization for the Product dimension can be observed in Figure 1 (c).

3. Informix Metacube

In this section, we will present how Informix Metacube (IM) represents the structural part of multidimensional modeling and the tools it provides for these tasks.

IM works with both the star and snowflake schema. However, the snowflake schema is partial in the sense that the tables that represent different levels of hierarchy are not related between them. Thus, these tables are related to the one that represents the minimum level of hierarchy. Therefore, in our generation process, we will only generate the database schema that corresponds to the star schema.

On the other hand, the underlying MD model of IM is called Decision Support System (*DSS*). The content of the DSS can be defined through an easy graphical user interface with the tool Data Warehouse Manager (*DWM*). To do this, it is necessary to have the database schema (star or snowflake) previously implemented. The information about the MD models defined in IM is stored in relational tables which contain information about the MD elements defined in the DSS (e.g. facts, dimensions, hierarchy levels, etc.) and, the logical information on these MD elements (e.g. fact tables, primary key of the fact tables, attributes in the relational tables to identify instances of hierarchy levels, etc.).

To clarify the main MD properties considered by the DSS, we have modeled the DSS model with the UML (see Figure 2). Thus, the relational tables of the DSS have been modeled with classes and relationships between them. The name of the classes and relationships are the same as their corresponding relational tables in the DSS. It can be observed that not only information on the defined MD concepts is considered (facts, dimensions, etc.) but also logical information (primary keys, table column that corresponds to one measure, etc.).

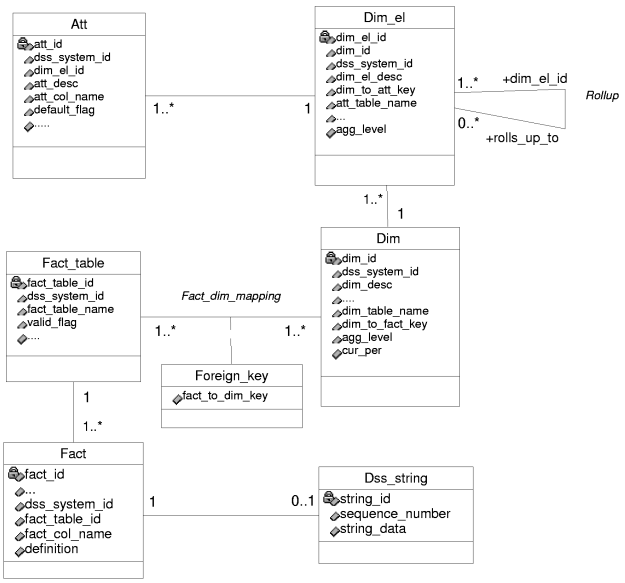

Figure 2. Modeling the DSS model of Informix Metacube with the UML

The classes Fact_table and Dim store information about the facts and dimensions defined in the DSS respectively. The relationship Fact_dim_mapping represents the information about which dimensions are related to which facts through the corresponding foreign key defined in the fact table (information considered in the associated class Foreign_key) as several facts and dimensions can be defined in the DSS. The class Fact stores information on the measures defined in the DSS, so that every measure must always be contained in a fact (see cardinality of the relationship). The derivation rules of derived measures are represented in the class Dss_string.

In the DSS, hierarchy levels are called dimensional groups and considered in the class Dim_el. Every dimensional group only belongs to one dimension (see cardinality of the relationship). This means that even though two or more dimensions share the same dimensional group, the same dimensional group has to be defined per each dimension. The relationship Rollup stores information about what dimensional group is connected to (rolls_up_to) which dimensional group. These dimensional groups can be connected to form multiple and alternative hierarchy paths (see cardinality of the relationship). Finally, the class Att stores information on the attributes defined in each dimensional group. An attribute must only belong to one dimensional group. For example, if the attribute name has been defined in both the City and Community classes, the attribute name must be defined twice as in the DSS they will be considered as different attributes (name in the City class and name in

The GOLD Model	The DSS Model
Dimensions	
Dimension	Dimension
Time dimension	Attribute current period
Dimension class	Base dimensional group
Base class	Dimensional group (D.G.)
Class identifying attribute {OID}	D.G. identifying attribute
Class attribute	D.G. attribute
Class descriptor attribute {D}	D.G. default attribute
Classification hierarchies	Roll-up relationships between D.G.
Alternative path and multiple hierarchies	Alternative path and multiple hierarchies
Non-strict and complete hierarchies	**NOT CONSIDERED**
Specialization hierarchies	**TRANSFORM** into roll-up relationships between D.G.
Specialization concept	D.G. "specialization concept"
Specialized class attribute	Attribute of the D.G. just created
Specialization relationships	Roll-up relationships
Facts	
Fact class	Fact
Many-to-many relationships between a fact and one dimension	**NOT CONSIDERED** Ask for identifying attribute {*OID*}
Shared aggregation with a class	Fact-dimension relationship
Fact class identifying attribute {OID}	**TRANSFORM** Create dimension, D.G. and the only attribute of that D.G.
Fact attribute	Measure
Derivation rule	Derivation rule
Additivity of measures	**NOT CONSIDERED**

Table 1. Correspondence between the GOLD model and the DSS model

the Community class).

After this brief review of the main MD features considered by the DSS, we will summary some important MD properties that cannot be considered by the DSS model:

- *"Many-to-many"* relationships between a fact and one dimension cannot be considered as the primary key of the fact table is only composed by the foreign keys of the dimension tables to which the fact table is related. This would require more attributes to be part of the primary key of the fact table or additional relational tables to represent these *"many-to-many"* relationships.

- Additivity cannot be considered, i.e. there is not way of indicating that a certain fact attribute cannot be aggregated along a dimension. Not either it is possible to restrict the set of aggregation operators that can be applied on a fact attribute (e.g. it is not possible to specify that only MAX and MIN can be applied on a specific fact attribute).

- The relationships between dimensional groups (relationship Rollup) are considered strict by default and, therefore, aspects about the cardinality of these relationships are not considered. In the star schema managed by IM, an instance of a dimensional group is only related to one instance of a higher hierarchy level. That is, non-strict and completeness classification hierarchies are not considered.

- The standard star schema does not allow the consideration of the categorization of dimensions as all attributes that correspond to all possible categories of a dimension are defined as attributes within the same relational table in the star schema.

4. From the GOLD model into the DSS model

In this section, we will present the main tasks of the semi-automatic generation process that from a GOLD model obtains its corresponding DSS model to implement it in IM.

The algorithm of the structural part reads a GOLD conceptual model and generates two files with SQL sentences:

- the first file contains the SQL sentences needed to create the relational tables that correspond to the star schema that will form the database schema and,

- the second one, contains the SQL sentences to register the MD concepts in the DSS that correspond to the modeling constructors used in a GOLD model.

In this generation process, we have to handle that certain modeling constructors of the GOLD model do not have their

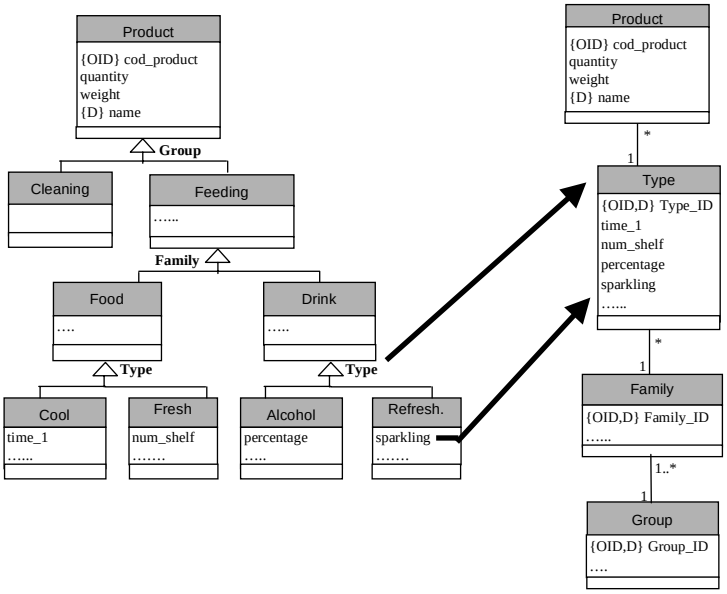

Figure 3. Transformation of specialization hierarchies into classification hierarchies

corresponding representation into the DSS model. In some cases, it has been possible to carry out a minimal semi-automatic transformation (the designer has to decide if this transformation is carried out in some cases) of the modeling constructors to be able to represent them in the DSS trying to preserve their initial semantics. In other cases, such a transformation is not possible and, therefore, those modeling constructors have been ignored with the corresponding lack of expressiveness in the final representation of a GOLD model. Due to the lack of space, in this paper we will only describe the transformations accomplished for some modeling constructors.

The Table 1 shows the correspondence between the modeling constructors of the GOLD model and the MD concepts considered by the DSS model. From now on, we will only mention the modeling constructors ignored as well as we will only remark the transformations accomplished.

To start with, every class that represents a hierarchy level is defined as a dimensional group. Then, the process reads all associations for every one of these classes and defines a Rollup relationship between the two associated classes. This means that even though two or more dimensions share the same hierarchy level, this level is defined for each dimension as a dimensional group. This is required by the DSS model where every dimensional group must only belong to one dimension.

On the other hand, non-strict and complete classification hierarchies are ignored as in the database schema managed by the DSS model an instance of a hierarchy level can only refer to a one instance of a higher level of the classification hierarchy. Thus, we have to ignore those properties in the generation algorithm.

Finally, the additivity of measures is not considered by the DSS model and, therefore, this property is ignored in the generation process. In the follow, we will describe how to accomplish the transformations described in Table 1.

4.1. Specialization hierarchies

Specialization hierarchies are **transformed** into strict classification hierarchies. Every concept of the specialization hierarchy is transformed into one dimensional group (level) of the classification hierarchy. Every attribute within a class defined under this specialization concept is considered as an attribute of the new dimensional group. These new dimensional groups are related by means of strict classification hierarchies. Finally, every new dimensional group will have defined as identifying and default attribute the attribute specialization_name_ID.

In Figure 3, we can see an example of this transformation accomplished for the Product dimension from a conceptual point of view. The specialization levels Group, Family and Type will be transformed into their corresponding classification levels. The identifying and default attribute will be called specialization_name_ID of type String that will have the possible values of the name of the classes defined under the specialization concept that represents. For example, the different values the attribute Type_ID can have are Cool, Fresh, Alcohol and Refreshments. Finally, the arrows in the Figure 3 show how all attributes defined under a specialization concept are included in the corresponding new classification level.

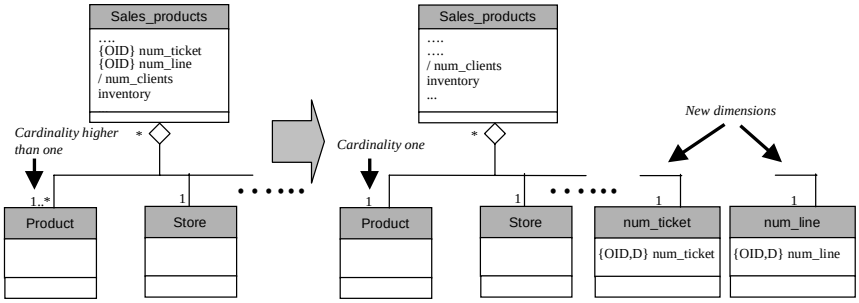

Figure 4. Transformation of identifying attributes {OID} of fact classes into new dimensions

4.2. Many-to-many relationship between a fact and one dimension

The DSS model does not consider the *many-to-many* relationship between a fact and one dimension. If the algorithm reads a cardinality higher than *1* in a shared aggregation on the role of a dimension class, this property will be ignored if no identifying attribute has been defined in the fact class. Let us remind (see section 2) that the GOLD model allows the designer to define identifying attributes $\{OID\}$ in the fact class that may be needed to represent *many-to-many* relationships between a fact and one dimension.

Therefore, in our generation process, if the designer has defined identifying attributes in the fact class, every one of theses attributes will be transformed into a new dimension with only the base dimensional group with only one attribute (identifying and descriptor at the same time).

In Figure 4, we can see the transformation accomplished from a conceptual point of view. The identifying attributes $\{OID\}$ num_ticket and $\{OID\}$ num_line will be transformed into two new dimensions with only one base dimensional group for each new dimension. The only attribute defined within these new dimensional groups is the same as the one defined in the fact class with the properties of $\{OID, D\}$ (i.e. identifying and descriptor attribute).

5. Conclusions

The GOLD model is an OO conceptual model to represent the structural and dynamic part of OLAP applications. To facilitate the conceptual design, the model provides an easy UML graphical notation that will be used by the designer in the CASE tool that gives support to the model.

In this paper, we have presented how to semi-automatically generate all the information needed to implement the structural part of a GOLD model into Informix Metacube (IM). The generation of the structural part consists of generating both the star schema to house the MD data and the MD concepts of the DSS model that correspond to the GOLD modeling constructors used in the design. In this process, it has been necessary to transform some modeling constructors that do not have their corresponding representation in IM.

We have developed "The GOLD Model CASE Tool" [8], that gives support to the GOLD model. Moreover, the generation process described throughout the paper has also been implemented in this CASE tool.

Regarding the dynamic part, we will present the transformation process of initial users' requirements defined in a GOLD model into IM requirements in a future work.

References

[1] L. Cabibbo and R. Torlone. From a Procedural to a Visual Query Language for OLAP. In *Proc. of the 10th Intl. Conf. on Scientific and Statistical Database Management (SSDM'98)*, pages 74–83, 1998.

[2] W. Giovinazzo. *Object-Oriented Data Warehouse Design. Building a star schema*. Prentice-Hall, 2000.

[3] M. Golfarelli and S. Rizzi. A methodological Framework for Data Warehouse Design. In *Proc. of the ACM 1st Intl. Workshop on Data warehousing and OLAP (DOLAP'98)*, pages 3–9, 1998.

[4] K. Hahn, C. Sapia, and M. Blaschka. Automatically Generating OLAP Schemata from Conceptual Graphical Models. In *Proc. of the ACM 3rd Intl. Workshop on Data warehousing and OLAP (DOLAP'00)*, 2000.

[5] R. Kimball. *The data warehousing toolkit*. John Wiley, 1996.

[6] Object Management Group (OMG). Unified Modeling Language Specification 1.4. Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67, September 2001.

[7] C. Sapia, M. Blaschka, G. Hfling, and B. Dinter. Extending the E/R Model for the Multidimensional Paradigm. In *Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98)*, volume 1552 of *Lecture Notes in Computer Science*, pages 105–116. Springer-Verlag, 1998.

[8] J. Trujillo, S. Luján-Mora, and E. Medina. The GOLD Model CASE Tool: an environment for designing OLAP applications. In *Proc. 4th Intl. Conf. on Enterprise Information Systems (ICEIS 2002)*, pages 699–707, 1986.

[9] J. Trujillo, M. Palomar, J. Gómez, and I.-Y. Song. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer, special issue on Data Warehouses*, 34(12):66–75, 2001.