APPLYING UML FOR DESIGNING MULTIDIMENSIONAL DATABASES AND OLAP APPLICATIONS

Name: Juan Trujillo

Affiliation: Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante

Address: Campus de San Vicente del Raspeig, Ap. Correos 99, E-03080 Alicante, Spain

Phone number: +34 965 90 34 00 ext. 2967

Fax number: +34 965 90 93 26

Email: jtrujillo@dlsi.ua.es

Name: Il-Yeol Song

Affiliation: College of Information Science and Technology, Drexel University

Address: PA 19104, USA

Phone number: +1 (215) 895-2489

Fax number: +1 (215) 895-2494

Email: songiy@drexel.edu

Name: Sergio Luján-Mora

Affiliation: Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante

Address: Campus de San Vicente del Raspeig, Ap. Correos 99, E-03080 Alicante, Spain

Phone number: +34 965 90 34 00 ext. 2962

Fax number: +34 965 90 93 26

Email: slujan@dlsi.ua.es

APPLYING UML FOR DESIGNING MULTIDIMENSIONAL DATABASES AND OLAP APPLICATIONS

Multidimensional (MD) modeling is the basis for Data warehouses (DW), multidimensional databases (MDB) and On-Line Analytical Processing (OLAP) applications. In this chapter, we present how the Unified Modeling Language (UML) can be successfully used to represent both structural and dynamic properties of these systems at the conceptual level. The structure of the system is specified by means of a UML class diagram that considers the main properties of MD modeling with minimal use of constraints and extensions of the UML. If the system to be modeled is too complex, thereby leading us to a considerable number of classes and relationships, we sketch out how to use the *package* grouping mechanism provided by the UML to simplify the final model. Furthermore, we provide a UML-compliant class notation (called cube class) to represent OLAP initial user requirements. We also describe how we can use the UML state and interaction diagrams to model the behavior of a data warehouse system. We believe that our innovative approach provides a theoretical foundation for simplifying the conceptual design of multidimensional systems and our examples illustrate the use of our approach.

Keywords: Data warehouses, multidimensional databases, OLAP, conceptual modeling, UML, object orientation

INTRODUCTION

It is widely accepted that DW, MDB and OLAP applications are based on multidimensional modeling. The benefit of using this MD modeling is two-fold. On one hand, the MD model is close to data analyzers' way of thinking, therefore, it helps users understand data. On the other hand, the MD model supports performance improvement as its simple structure allows us to predict final users' intentions.

Some approaches have been proposed lately (presented in Section 3) to accomplish the conceptual design of these systems. Unfortunately, none of them has been accepted as a standard for DW conceptual modeling. These proposals try to represent main MD properties at the conceptual level with special emphasis on MD data structures. A conceptual modeling approach for DW, however, should also concern other relevant aspects such as initial user requirements, the behavior of the system (e.g. main operations to be accomplished on MD data structures), available data sources, and specific issues for automatic generation of the database schema and so on. We claim that object orientation with the UML provides an adequate notation for modeling every aspect of a DW system (MD data structures, the behavior of the system, etc.) from user requirements to implementation.

In this chapter, we present an object-oriented (OO) approach to accomplish the conceptual modeling of DW, MDB and OLAP applications. Our approach introduces a set of minimal constraints and extensions of the UML (Booch, 1998; OMG, 2001) needed for an adequate representation of MD modeling properties. These extensions are based on the standard mechanisms provided by the UML to adapt to a specific method or model (e.g. constraints, tagged values). We also present how to group classes into *packages* to simplify the final model in case that the model becomes too complex due to

the high number of classes. Furthermore, we provide a UML-compliant class notation to represent OLAP initial user requirements (called *cube class*). From these cube classes, we then describe the use of state and interaction diagrams to model the behavior of the system based on the applied OLAP operations. We also discuss issues such as identifying attributes and descriptor attributes that set the basis for an adequate semi-automatic generation of a database schema and user requirements in a target commercial OLAP tool. Finally, we present a set of case studies to show the elegant way in which our proposal represents both structural and dynamic properties of MD modeling.

The UML can also be used with powerful mechanisms such as the Object Constraint Language (OCL) (Warmer, 1998; OMG, 2001) and the Object Query Language (OQL) (Cattell, 2000) to embed DW constraints (e.g. additivity and derived attributes) and initial user requirements in the conceptual model. In this way, when we model a DW system, we can obtain simple yet powerful extended UML class diagrams that represent main MD properties at a conceptual level. We believe that our innovative approach provides a theoretical foundation for the possible use of OODB and ORDB for DW and OLAP applications.

The remainder of this chapter is organized as follows: Section 2 details the major features of MD modeling that should be taken into account for a proper MD conceptual design. Section 3 summarizes the most relevant conceptual approaches proposed so far by the research community. In Section 4, we summarize how we use the UML to consider main MD properties at the conceptual level. In Section 5, we present a set of case studies taken from Kimball (Kimball, 2002) to show the benefit of our approach.

Finally, Section 6 draws some conclusions and sketches some work that is currently being carried out.

MULTIDIMENSIONAL MODELING PROPERTIES

In MD modeling, information is structured into **facts** and **dimensions**. A fact is an item of interest for an enterprise, and is described through a set of attributes called **measures** or **fact attributes** (**atomic** or **derived**), which are contained in cells or points in the data cube. This set of measures is based on a set of dimensions that determine the granularity adopted for representing facts (i.e. the context in which facts are to be analyzed). Moreover, dimensions are also characterized by attributes, which are usually called **dimension attributes**. They are used for grouping, browsing, and constraining measures.

Let us consider an example in which the fact is the *product sales* in a large store chain and the dimensions are as follows: *product*, *store*, *customer* and *time*. On the left hand side of Figure 1, we can observe a data cube typically used for representing a MD model. In this particular case, we have defined a cube for analyzing measures along the *product*, *store* and *time* dimensions.

We note that a fact usually represents a *many-to-many* relationship between any of two dimensions. For example, a *product* is sold in many *stores* and a *store* sells many *products*. We also assume that there is a *many-to-one* relationship between a fact and each particular dimension. For example, for each *store* there are many *sale tickets*, but each *sale ticket* belongs to only one *store*.

Nevertheless, there are some cases in which a fact may be associated with a particular dimensions as a ***many-to-many*** relationship. For example, the fact *product_sales* is considered as a particular *many-to-many* relationship to the *product* dimension as one ticket may consist of more than one *product* even though every ticket is still purchased in only one *store* by one *customer* and at one *time*.

With reference to measures, the concept of **additivity** or summaribility (Blaschka, 1998; Golfarelli, 1998; Kimball, 2002; Trujillo, 2000; Tryfona, 1999) on measures along dimensions is crucial for MD data modeling. A measure is *additive* along a dimension if the SUM operator can be used to aggregate attribute values along all hierarchies defined on that dimension. The aggregation of some fact attributes (roll-up in OLAP terminology), however, might not be semantically meaningful for all measures along all dimensions. A measure is *semi-additive* if SUM operator can be applied to some dimensions, but not all the dimensions. A measure is *non-additive* if SUM operator cannot be applied to any dimension. In our example, *number of clients* (estimated by counting the number of purchased receipts for a given *product*, *day* and *store*) is not additive along the *product* dimension. Since the same ticket may include other *products*, adding up the *number of clients* along two or more *products* would lead to inconsistent results. However, other aggregation operators (e.g. SUM, AVG and MIN) could still be used along other dimensions such as *time*. Thus, *number of clients* is semi-additive. An example of non-additive measures is *interest rate*.

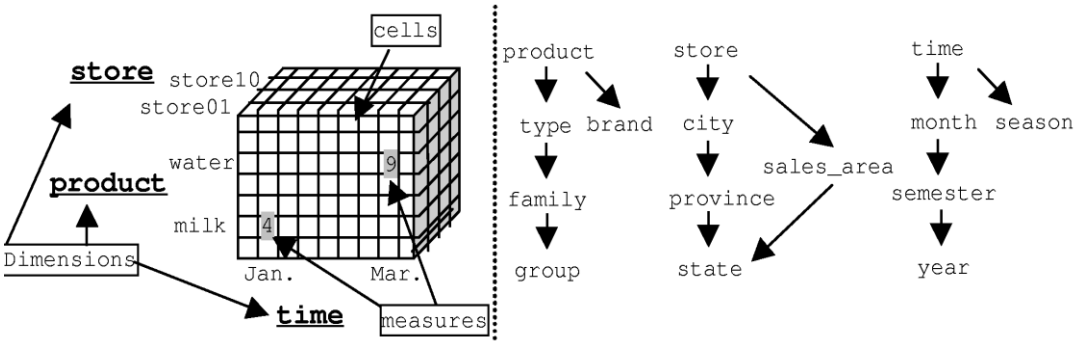

Figure 1: A data cube and classification hierarchies defined on dimensions

Regarding dimensions, the **classification hierarchies** defined on certain dimension attributes are crucial because the subsequent data analysis will be addressed by these classification hierarchies. A dimension attribute may also be aggregated (related) to more than one hierarchy, therefore, **multiple classification hierarchies** and **alternative path hierarchies** are also relevant. For this reason, a common way of representing and considering dimensions with their classification hierarchies is by means of Directed Acyclic Graphs (DAG).

On the right hand side of Figure 1, we can observe different classification hierarchies defined on the *product*, *store* and *time* dimensions. On the *product* dimension, we have considered a multiple classification hierarchy to be able to aggregate data values along two different hierarchy paths: (i) *product*, *type*, *family*, *group* and (ii) *product*, *brand*. There may exist attributes that are not used for aggregating purposes and provide features for other dimension attributes (e.g. *product name*). On the *store* dimension, we have defined an alternative classification hierarchy with two different paths that converge into the same hierarchy level: (i) *store*, *city*, *province*, *state* and (ii) *store*, *sales_area*, *state*. Finally, we have also defined another multiple classification hierarchy

with the following paths on the *time* dimension: (i) *time*, *month*, *semester*, *year* and (ii) *time*, *season*.

Nevertheless, classification hierarchies are not so simple in most cases. The concepts of **strictness** and **completeness** are quite important, not only for conceptual purposes, but also for further design steps of MD modeling (Tryfona, 1999). "Strictness" means that an object of a lower level in a hierarchy belongs to *only* one in a higher level, e.g. a *province* is only related to one *state*. "Completeness" means that all members belong to one higher-class object and that object consists of those members only. For example, suppose that the classification hierarchy between the *state* and *province* levels is "complete". In this case, a *state* is formed by *all* the *provinces* recorded and all the *provinces* that form the *state* are recorded.

OLAP scenarios sometimes become very large as the number of dimensions increases significantly, and therefore, this fact may lead to extremely sparse dimensions and data cubes. In this way, there are attributes that are normally valid for all elements within a dimension while others are only valid for a subset of elements (also known as the **categorization of dimensions** (Lehner, 1998; Tryfona, 1999)). For example, attributes *alcohol percentage* and *volume* would only be valid for *drink products* and will be "null" for *food products*. Thus, a proper MD data model should be able to consider attributes only when necessary, depending on the categorization of dimensions.

Furthermore, let us suppose that apart from a high number of dimensions (e.g. 20) with their corresponding hierarchies, we have a considerable number of facts (e.g. 8) sharing dimensions and classification hierarchies. This system will lead us to a very complex

design, thereby increasing the difficulty in reading the modeled system. Therefore, a MD conceptual model should also provide techniques to **avoid flat diagrams**, allowing us to group dimensions and facts to simplify the final model.

Once the structure of the MD model has been defined, OLAP users usually define a set of initial user requirements as a starting point for the subsequent data analysis phase. From these initial requirements, users can apply a set of operations (usually called OLAP operations (Chaudhuri, 1997) to the MD view of data for further data analysis. These OLAP operations are usually as follows: roll-up (increasing the level of aggregation) and drill-down (decreasing the level of aggregation) along one or more classification hierarchies, slice-dice (selection and projection) and pivoting (re-orienting the MD view of data which also allows us to exchange dimensions for facts; i.e. **symmetric** treatment of facts and dimensions).

Star Schema

In this sub-section, we will summarize the star schema popularized by Kimball (Kimball, 2002) since it is the most well-known schema to represent MD properties in relational databases.

Kimball claims that the star schema and its variants fact constellations schema and the snowflake schema are logical choices for MD modeling to be implemented in relational systems. We will briefly introduce this well-known approach using Sales Dimensional Model.

Figure 2 shows an example of Kimball's Sales Dimensional Model. In this model, the fact is the name of the middle box (Sales fact table). Measures are the non-foreign keys in the fact table (dollars_sold, units_sold, and dollars_cost). Dimensions are the boxes connected to the fact table in a one-to-many relationship (Time, Store, Product, Customer, and Promotion). Each dimension contains relevant attributes: day_of_week, week_number, and month in Time; store_name, address, district, and floor_type in Store, and so on.

From Figure 2, we can easily see that there are many MD features that are not reflected in the Dimensional Model: Which are the classification hierarchies defined on dimensions? Can we use all aggregation operators on all measures along all dimensions? What are these classification hierarchies like? Non-strict, strict, complete, …? And many more. Therefore, we argue that for a proper DW and OLAP design, a conceptual MD model should be provided to better reflect user requirements and then, this conceptual model could be translated into a logical model for a later implementation. In this way, we can be sure that we are analyzing the real world as users perceive them.

RELATED WORK

Lately, several MD data models have been published. Some of them fall into the logical level (such as the well-known star-schema by R. Kimball (Kimball, 2002)). Others may be considered as formal models as they provide a formalism to consider main MD properties. A review of the most relevant logical and formal models can be found in (Blaschka, 1998; Abello, 2001).

In this section, we will only make a brief reference to the most relevant models that we consider "pure" conceptual MD models. These models provide a high level of abstraction for the main MD modeling properties presented in Section 2 and are totally independent from implementation issues. These are as follows: The Dimensional-Fact (DF) model by Golfarelli (1998), The Multidimensional/ER (M/ER) model by Sapia (1998, 1999) and The starER model by Tryfona (1999).

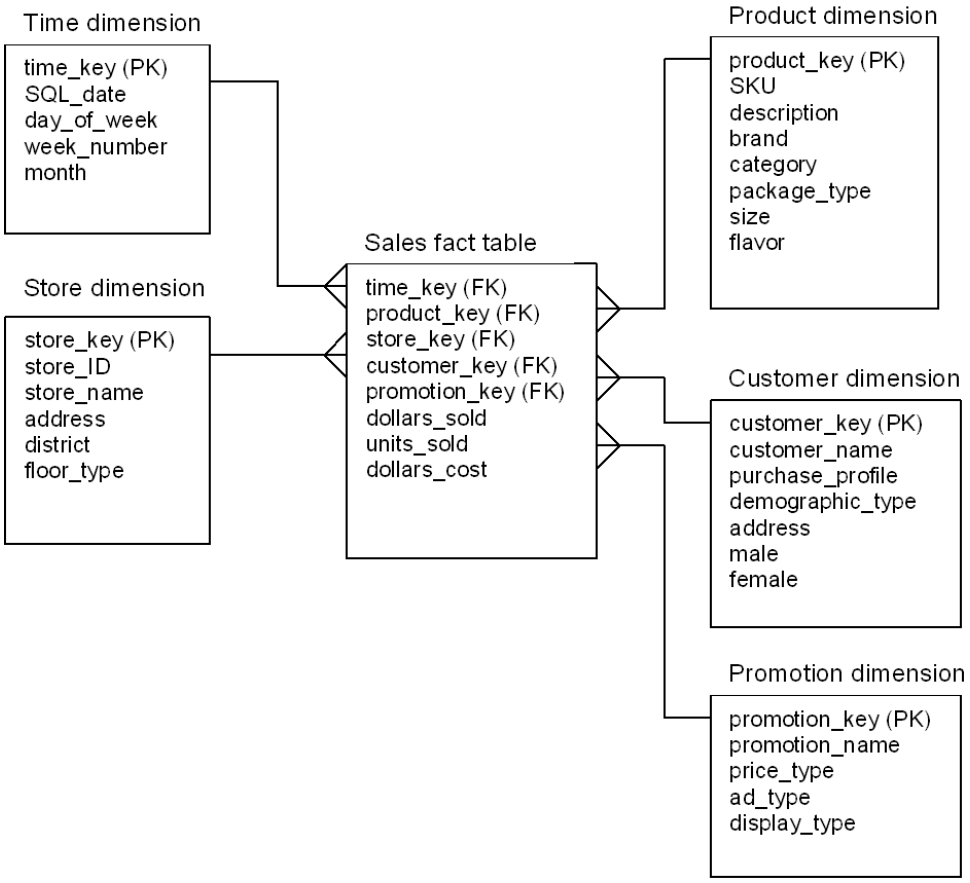

Figure 2: Sales Dimensional Model

In Table 1, we provide the coverage degree of each above-mentioned conceptual model regarding the main MD properties described in the previous section. To start with, to the best of our knowledge, no proposal provides a grouping mechanism to avoid flat

diagrams and to simplify the conceptual design when a system becomes complex due to a high number of dimensions and facts sharing dimensions and their corresponding hierarchies. Regarding facts, only the starER model considers *many-to-many* relationships between facts and particular dimensions by indicating the exact cardinality (multiplicity) between them. None of them considers derived measures or their derivation rules as part of the conceptual schema. The DF and the starER models consider the additivity of measures by explicitly representing the set of aggregation operators that can be applied on non-additive measures. With reference to dimensions, all of the models consider multiple and alternative path classification hierarchies by means of Directed Acyclic Graphs (DAG) defined on certain dimension attributes. However, only the starER model considers non-strict and complete classification hierarchies by specifying the exact cardinality between classification hierarchy levels. As both the M/ER and the starER models are extensions of the Entity Relationship (ER) model, it is easy for them to consider the categorization of dimensions by means of *Is-a* relationships.

Multidimensional modeling properties	Model		
	DF	**M/ER**	**StarEr**
Structural level			
Facts			
Many-to-many relationships with particular dimensions	No	No	Yes
Atomic measures	Yes	Yes	Yes
Derived measures	No	No	No
Additivity	Yes	No	Yes

Dimensions			
Multiple and alternative path classification hierarchies	Yes	Yes	Yes
Nonstrict classification hierarchies	No	No	Yes
Complete classification hierarchies	No	No	Yes
Categorization of dimensions	No	Yes	Yes
Dynamic level			
Specifying initial user requirements	Yes	Yes	No
OLAP operations	No	Yes	No
Modeling system behavior	No	Yes	No
Graphical notation	Yes	Yes	Yes
Automatic generation into a target OLAP commercial tool	No	Yes	No

Table 1: Comparison of conceptual multidimensional models

With reference to the dynamic level of MD modeling, the starER model is the only one that does not provide an explicit mechanism to represent initial user requirements. On the other hand, only the M/ER model provides a set of basic OLAP operations to be applied from these initial user requirements, and it models the behavior of the system by means of state diagrams.

Finally, we note that all the models provide a graphical notation that facilitates the conceptual modeling task to the designer. On the other hand, only the M/ER model provides a framework for an automatic generation of the database schema into a target commercial OLAP tool (particularly into Informix Metacube and Cognos Powerplay).

From Table 1, one may conclude that none of the current conceptual modeling approaches considers all MD properties at both the structural and dynamic levels. Therefore, we claim that a standard conceptual model is needed to consider all MD modeling properties at both the structural and dynamic levels. We argue that an OO approach with the UML is the right way of linking structural and dynamic level properties in an elegant way at the conceptual level.

A NOVEL OO CONCEPTUAL MODELING APPROACH BASED ON THE UML

In this section, we summarize how our OO MD model, based on a subset of the UML, can represent main structural aspects of MD modeling. A complete description of our approach can be found in (Trujillo, 2001). The main features considered are the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, and non-strict and complete hierarchies.

It is important to remark that if we are modeling complex and large DW systems, we are not restricted to use flat UML class diagrams. Instead, we can make use of the grouping mechanism provided by the UML called *package* to group classes together into higher level units to create different levels of abstraction, and therefore, simplifying the final model (Luján-Mora, 2002). In this way, a UML class diagram improves and simplifies the system specification accomplished by classic semantic data models such as the ER model. Furthermore, necessary operations and constraints (e.g. additivity rules) can be embedded in the class diagram by means of OCL expressions.

In this approach, the main structural properties of MD models are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions. Dimensions and facts are represented by *dimension classes* and *fact 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. In our example in Figure 3 (a), we can see how the fact class Sales has a many-to-one relationship with both dimension classes.

By default, all measures in the fact class are considered additive. For nonadditive measures, *additivity 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 3 (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* (Kimball, 2002; Giovinazzo, 2000), thereby representing other fact features in addition to the measures for analysis. For example, we could store the ticket number (ticket_num) and the line number (line_num) as degenerate dimensions, as reflected in Figure 3 (a).

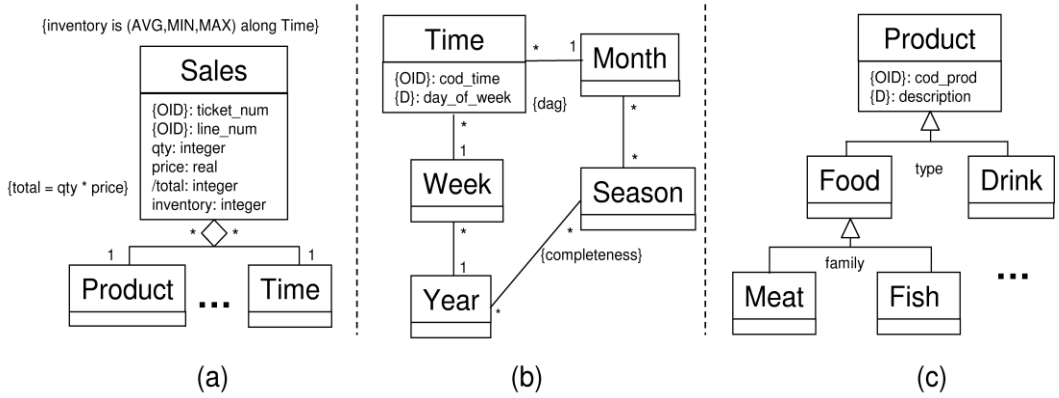

Figure 3: Multidimensional modeling using UML

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 (constraint *{D}*). A descriptor attribute will be used as the default label in the data analysis. These attributes are necessary for an automatic generation process into commercial OLAP tools, as these tools store the information 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 belongs to only one higher-level object (e.g., as one month can be related to more than one season, the relationship between 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 3 (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.

Finally, the *categorization of dimensions,* used to model additional features for a class's subtypes, is represented by means of *generalization-specialization* relationships. However, only the dimension class can belong to both a classification and a specialization hierarchy at the same time. An example of categorization for the Product dimension is shown in Figure 3 (c).

Regarding dynamic properties, this approach allows us to specify initial user requirements by means of a UML-compliant class notation called *cube class* during analysis. Then, behavioral properties are mainly related to these cube classes that represent initial user requirements. From these cube classes, final users may start a navigational process by applying certain OLAP operations (*roll-up, drill-down,* etc.) in the further data analysis phase. These operations are closed as they generate another cube class as an output. Thus, we use **state** and **interaction diagrams** to model the behavior (evolution) of these cube classes based on the applied OLAP operation. These diagrams contain information about the most probable evolution of final user requirements from the specified initial user requirement. The information contained in these diagrams can be used by OLAP designers to predict user behaviors, and therefore, help them design a proper view maintenance policy. (See first case study in next section for further details on dynamic properties).

CASE STUDIES

The aim of this section is to exemplify the usage of our conceptual modeling approach on modeling MD databases. We have selected three different examples taken from Kimball's book (Kimball, 2002), each of which introduces a new particular modeling feature: a warehouse, a large bank, and a college course. Due to the lack of space, we will only apply our complete modeling approach for the first example: we will apply all of the diagrams we use for modeling a DW (package diagrams, class diagrams, interaction diagrams, etc.). For the rest of the examples, due to space constraints, we will only focus on representing the structural properties of MD modeling by specifying the corresponding UML class diagram. This class diagram is the key one in our approach since the rest of diagrams can be easily obtained from it.

The warehouse

This example explores three inventory models of a warehouse. The first one is the *inventory snapshot*, where the inventory levels are measured every day and are placed in separate records in the database. The second model is the *delivery status model*, which contains one record for each delivery to the warehouse and the disposition of all the items is registered until they have left the warehouse. Finally, the third inventory model is the *transaction model*, which records every change of the status of delivery products as they arrive at the warehouse, are processed into the warehouse, etc.

This example introduces two important concepts: the *semiadditivity* and the *multistar model* (also known as fact constellations). The former has already been introduced in

Section 2 and refers to the fact that a measure cannot be summarized by using the *sum* function along a dimension. In this example, the inventory level (stock) of the warehouse is semiadditive, because it cannot be summed along time dimension, but it can be averaged along the same dimension. The multistar (fact constellations) concept refers to the fact that the same MD model has multiple facts.

To start with, in our approach, we model multistar models by means of package diagrams. In this way, at the first level, we create a package diagram for each one of the facts considered in the model. At this level, connecting package diagrams means that a model will use elements (e.g. dimensions, hierarchies) defined in the other package. Figure 4 shows the first level of the model formed by three packages that represent the different star schemas in the case study.

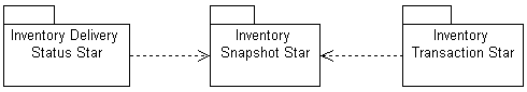

Figure 4: Level 1 of Inventory case study

Then, we explore each package diagram at the second level to define packages for each one of the facts and dimensions defined in the corresponding package diagram. Figure 5 shows the content of the package Inventory Snapshot Star at level 2. The fact package Inventory Snapshot Fact is represented in the middle of Figure 5, and the dimension packages (Product Dimension, Time Dimension, and Warehouse Dimension) are placed around the fact package. As can be seen, a dependency is drawn from the fact package to each one of the dimension packages, because the fact package comprises the whole definition of the star schema. At level 2, it is possible to create a dependency from a fact

package to a dimension package or between dimension packages (when they share some hierarchy levels), but not from a dimension package to a fact package.

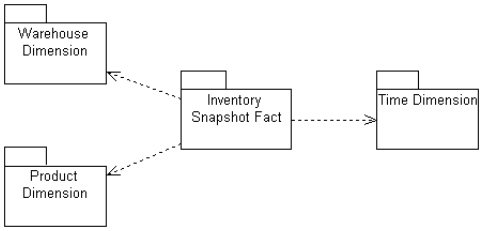

Figure 5: Level 2 of Inventory Snapshot Star

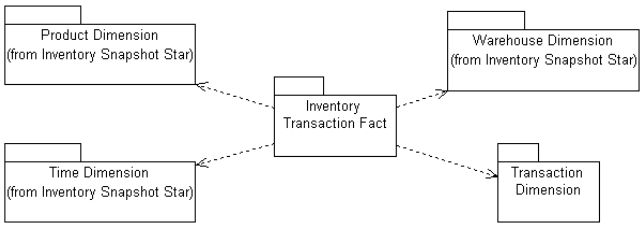

Figure 6: Level 2 of Inventory Transaction Star

Figure 6 shows the content of the package Inventory Transaction Star at level 2. As in the Inventory Snapshot Star, the fact package is placed in the middle of the figure and the dimension packages are placed around the fact package in a star fashion. Three dimension packages (Product Dimension, Time Dimension, and Warehouse Dimension) have been previously defined in the Inventory Snapshot Star (Figure 5), and they are imported in this package. Therefore, the name of the package where they have been previously defined appears below the package name (from Inventory Snapshot Star).

The content of the dimension and fact packages is represented at level 3. The diagrams at this level are only comprised of classes and associations among them. For example, Figure 7 shows the content of the package Warehouse Dimension at level 3. In a dimension package, a class is drawn for the dimension class (Warehouse) and a class for each classification hierarchy level (ZIP, City, County, State, SubRegion, and Region). For the sake of simplicity, the methods of each class have not been depicted in the figure. As can be seen in Figure 7, Warehouse presents alternative path classification hierarchies: (i) ZIP, City, County, State, and (ii) SubRegion, Region, State.

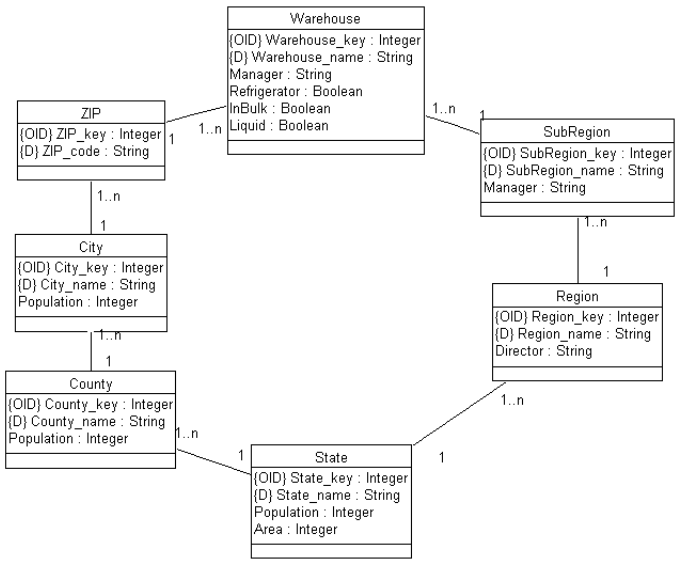

Figure 7: Level 3 of Warehouse Dimension

Finally, Figure 8 shows the content of the package Inventory Snapshot Fact. In this package, the whole star schema is displayed: the fact class (Inventory Snapshot) is defined and the dimensions with their corresponding hierarchy levels are imported from the dimension packages. To avoid unnecessary details, we have hidden the attributes and methods of dimensions and hierarchy levels, but the measures of the fact are shown as attributes of the fact class: four atomic measures (quantity_on_hand, quantity_shipped, value_at_cost, and value_at_LSP), and three derived measures (number_of_turns, gross_profit, and gross_margin). The definition of the derived measures is included in the model by means of derivation rules. Regarding the additivity of the measures, only quanty_on_hand is semiadditive; because of this, an additivity rule has been added to the model. Finally, Warehouse presents alternative path classification hierarchies and Time and Product present multiple classification hierarchies, as can be seen in Figure 8.

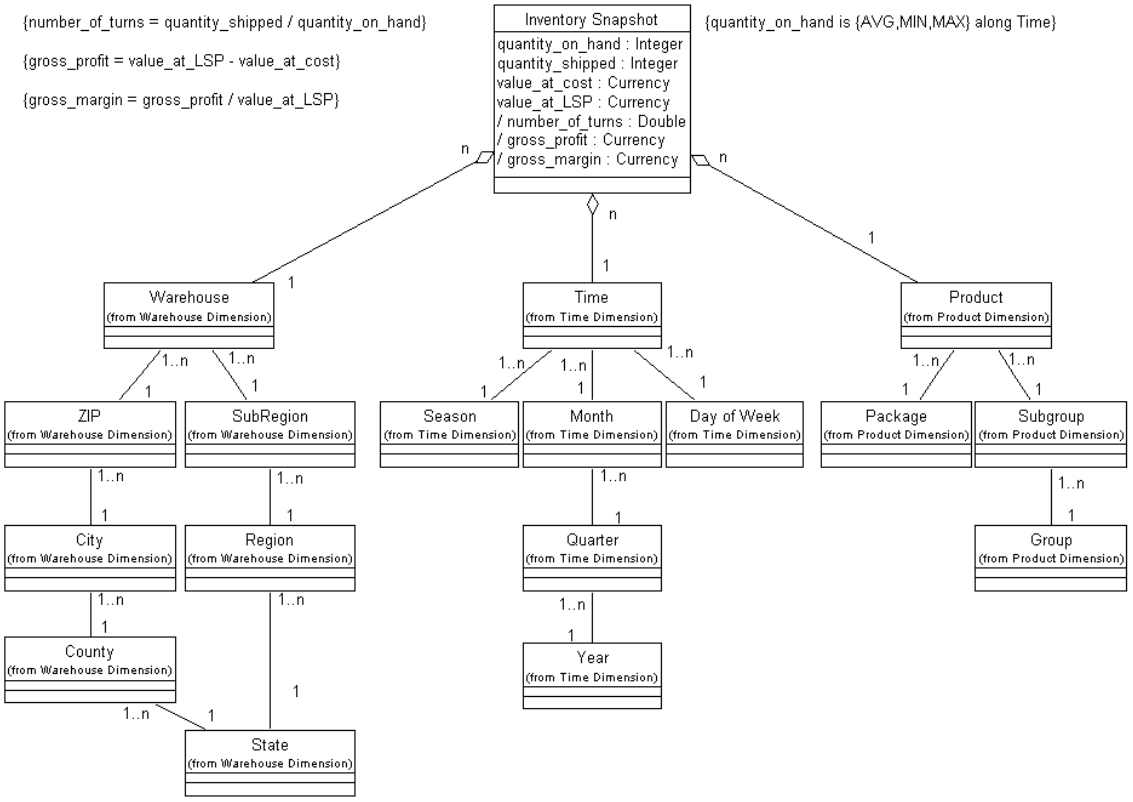

Figure 8: Level 3 of Inventory Snapshot Fact

Regarding the dynamic part of the model, initial user requirements are specified by means of a UML-compliant class notation called *cube class* (CC). A CC is structured into three sections: *measures*, to specify which fact attributes are analyzed; *slice*, to express constraints in terms of filters; and *dice*, to define grouping conditions of the data.

Let us suppose the following initial user requirement on the MD model specified by the UML class diagram of Figure 8: '*We wish to analyze the quantity_on_hand of products* ***where*** *the group of products is "Grocery" and the warehouse state is "Valencia"* ***grouped*** *according to the product subgroup* ***and*** *the warehouse region and subregion'*. On the left hand side of Figure 9, we can observe the graphical notation of the CC that corresponds to this requirement. The measure to be analyzed (quantity_on_hand) is

specified in the measure area. Constraints defined on dimension classification hierarchy levels (group and state) are included in the slice area, while classification hierarchy levels along which we are interested in analyzing measures (subgroup, region, and subregion) are included in the dice area. Finally, the available OLAP operations are specified in the CO (Cube Operations) section (in this example the CO are omitted to avoid unnecessary detail).

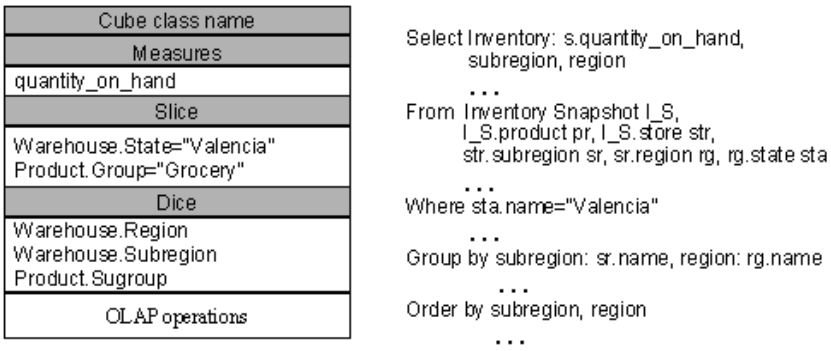

Figure 9: An example of initial user requirement

We should point out that this graphical notation of the cube class aims at facilitating the definition of initial user requirements to non-expert UML or databases users. In a more formal way, every one of these cube classes has its underlying OQL specification. Moreover, an expert user can directly define cube classes by specifying the OQL sentences. On the right hand side of Figure 9, we have sketched how the cube class example definition could be specified in OQL syntax.

As discussed in Section 4, we use the **state** and **interaction** diagrams provided by the UML to model the behavior (evolution) of these cube classes based on the applied OLAP operation. Regarding **state diagrams**, one state diagram is defined for each initial cube class. The diagram specifies that certain OLAP operations lead users to

cube classes that allow them to analyze the same data (the same measures along the same dimensions) in different ways. In these diagrams, each classification hierarchy level defined on a dimension included in the Dice area is considered as a valid state. Every one of these valid states will be a new cube class. Then, the provided OLAP operations allow us to navigate along the states to define new cube classes. For example, in Figure 10 we can see the corresponding state diagram of the cube class definition of Figure 9. It may be observed, for example, that roll-up and drill-down operations applied on the classification hierarchies levels defined on the Warehouse and Product dimensions will allow us to navigate up and down along the classification hierarchies defined in both dimensions.

On the other hand, an **interaction diagram** can also be defined for each UML class diagram. In our approach, we have adopted sequence diagrams (Booch, 1998; OMG, 2001) for their clarity and low complexity. This interaction diagram shows interactions among cube classes, changed by OLAP operations such as rotate, pivot, slice, or dice. Thus, we can specify that certain OLAP operations (e.g. dice) lead users to cube classes which will show completely different data. Then, these new cube classes represent the most probable new requirements a final user wish to execute. In Figure 11, we can see an example of interaction diagram, in which we have considered three cube classes that specify initial user requirements. Then, we have defined the OLAP operations needed to switch between these cube classes.

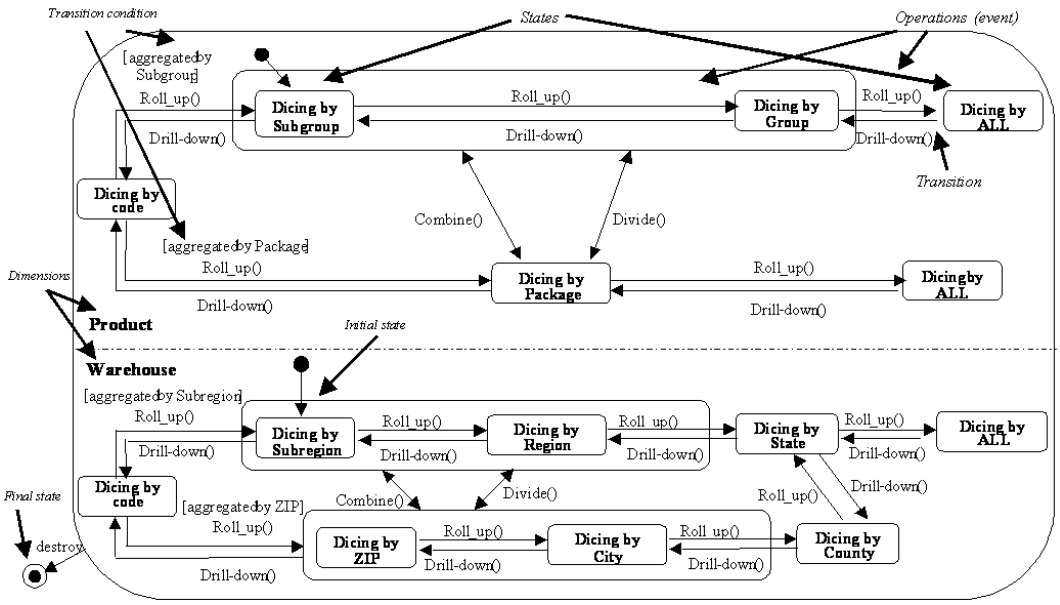

Figure 10: An example of state diagram

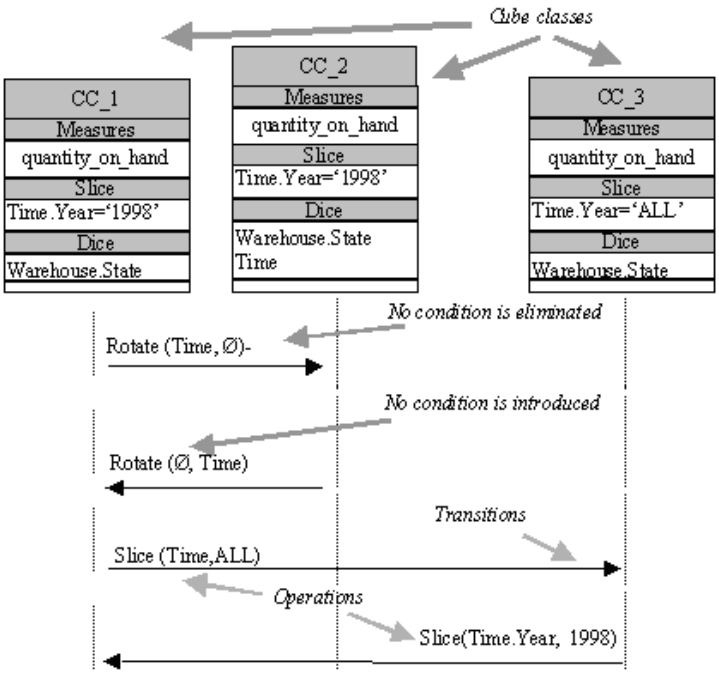

Figure 11: An example of interaction diagram

A large bank

In this example, a DW for a large bank is presented. The bank offers a significant portfolio of financial services: checking accounts, savings accounts, mortgage loans, safe deposit boxes, and so on.

This example introduces the following concepts:

- Heterogeneous dimension: a dimension that describes a large number of heterogeneous items with different attributes. Kimball's recommended technique is "to create a core fact table and a core dimension table in order to allow queries to cross the disparate types and to create a custom fact table and a custom dimension table for querying each individual type in depth". However, our conceptual MD approach can provide an elegant and simple solution to this problem, thanks to the categorization of dimensions.
- Categorization of dimensions: it allows us to model additional features for a dimension's subtypes.
- Shared classification hierarchies between dimensions: our approach allows two or more dimensions to share some levels of their classification hierarchies.

Figure 12 represents the level 1, which comprises five star packages: Saving Accounts Star, Personal Loans Star, Investment Loans Star, Safe Deposit Boxes Star, and Mortgage Loans Star. From now, we will only center on the Mortgage Loans Star. The corresponding level 2 of this star package is depicted in Figure 13.

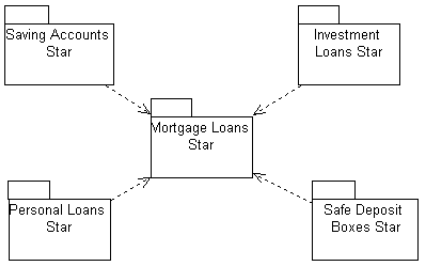

Figure 12: Level 1

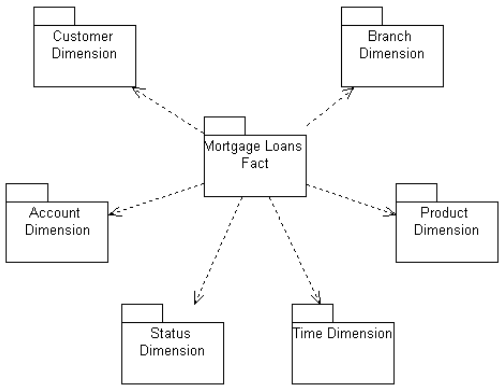

Figure 13: Level 2 of Mortgage Loans Star

Level 3 of Mortgage Loans Fact is shown in Figure 14. To avoid unnecessarily complicating the figure, three of the dimensions (Account, Time, and Status) with their corresponding hierarchies are not represented. Moreover, the attributes of the represented hierarchy levels have been omitted. The fact class (Mortgage Loans) contains four attributes that represent the measures: total, balance, and payment_number are atomic; whereas debt is derived (the corresponding derivation rule is placed next to the fact class). Regarding the additivity of the measures, none of the measures is additive, consequently, the additivity rules are also placed next to the fact class.

In this example, the dimensions present two special characteristics. On one hand, Branch and Customer share some hierarchy levels: ZIP, City, County, and State. On the other hand, Product dimension has a generalization-specialization hierarchy. This kind of hierarchy allows us to easily deal with heterogeneous dimensions: the different items can be grouped together in different categorization levels depending on their properties.

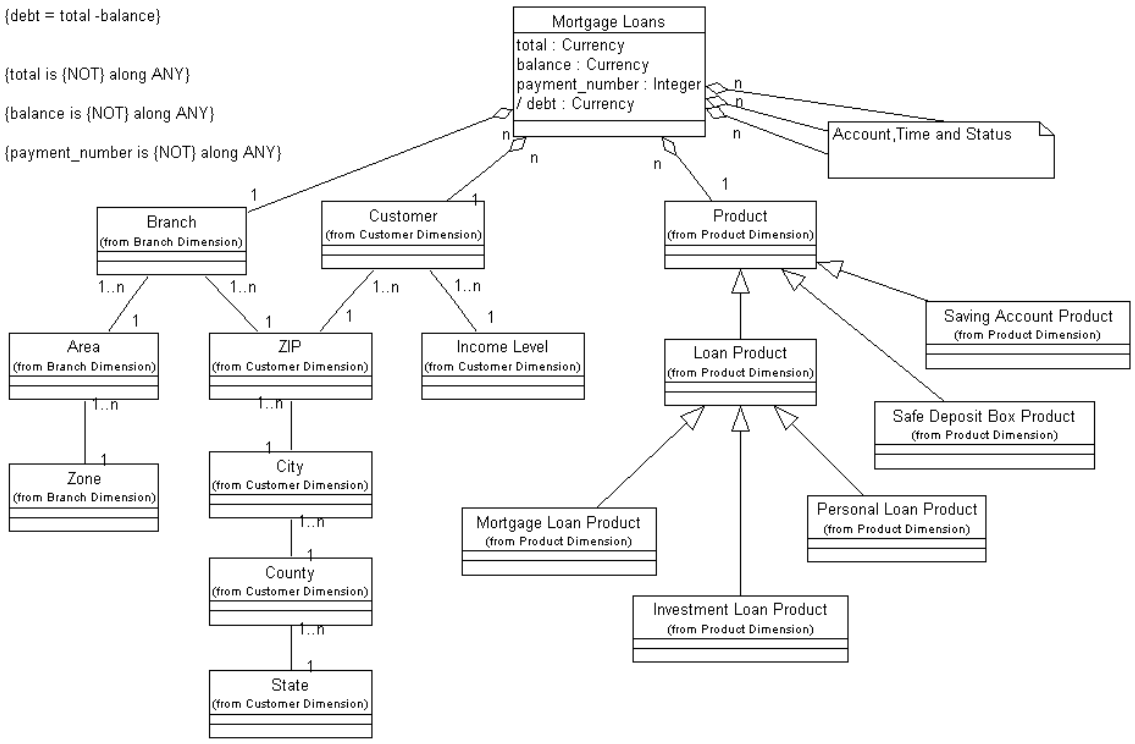

Figure 14: Level 3 of Mortgage Loans Fact

The college course

This example introduces the concept of *factless fact table* (FFT): fact tables for which there are no measured facts. Kimball distinguishes two major variations of FFT: *event tracking tables* ant *coverage tables*. In this example we will focus on the first type.

Event tracking tables are used when a large number of events need to be recorded as a number of dimensional entities coming together simultaneously. In this example, we will model daily class attendance at a college. In Figure 15 and Figure 16, level 1 and level 2 of this model are depicted respectively. In this case, level 1 only contains one star package.

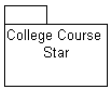

Figure 15: Level 1

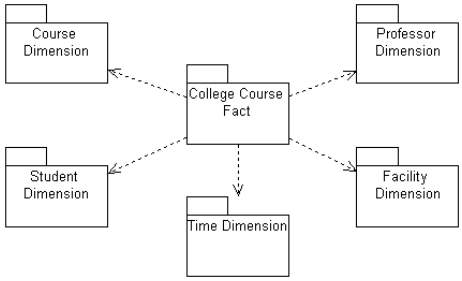

Figure 16: Level 2 of College Course Star

Figure 17 shows level 3 of College Course Fact. For the sake of simplicity, the attributes and methods of every class have not been depicted in the figure. As it can be seen, the fact class College Course contains no measures because it is a FFT. In FFT, the majority of the questions that users create imply counting the number of records that satisfy a constraint, such as which facilities were used most heavily? Or, which courses were the least attended?

Regarding the dimensions, Course and Time present multiple classification hierarchies, Professor and Student share some hierarchy levels, and Facility presents a categorization hierarchy.

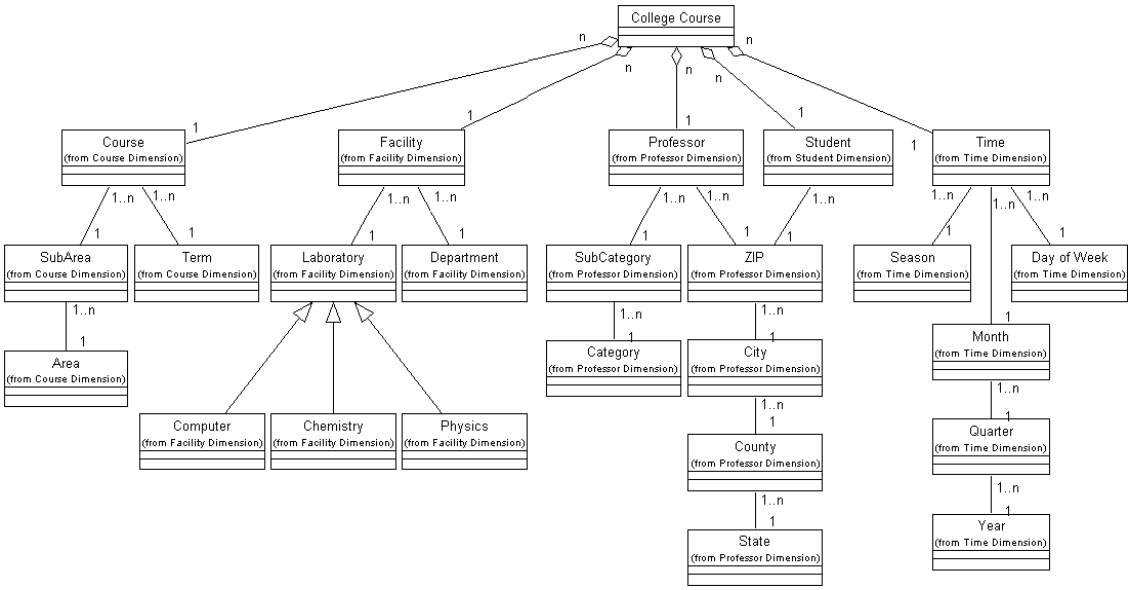

Figure 17: Level 3 of College Course Star

CONCLUSIONS

In this chapter, we have presented an OO conceptual modeling approach, based on the UML, to properly design DWs, MD databases and OLAP applications. Structural aspects of MD modeling are easily specified by means of a UML class diagram in which classes are related through association and shared aggregation relationships. In this context, thanks to the flexibility of the power of the UML, all the semantics required for a proper MD conceptual modeling are considered, such as *many-to-many* relationships between facts and particular dimensions, multiple path hierarchies of dimensions, the strictness and completeness of classification hierarchies, and categorization of dimension attributes. Regarding dynamic aspects, we provide a UML-compliant class graphical notation (called *cube classes*) to specify initial user requirements at the conceptual level. We have also described how we use state and interaction diagrams to model the behavioral aspects of the system regarding these cube

classes based on the set of the applied OLAP operations. Finally, we have selected three case studies from Kimball's book and modeled them following our approach. This shows that our approach is a very easy-to-use yet powerful conceptual model that represents main structural and dynamic properties of MD modeling in an easy and elegant way.

We are currently working on formally extending the UML with our conceptual modeling constructors. Furthermore, we are also considering the automatic implementation of a MD model from our approach into object-oriented and object-relational databases.