An Object Oriented Multidimensional Data Model for OLAP

Thanh Binh Nguyen, A Min Tjoa, and Roland Wagner

Institute of Software Technology (E188), Vienna University of Technology
Favoritenstrasse 9-11/188, A-1040 Vienna, Austria
{binh,tjoa}@ifs.tuwien.ac.at
Institute of Applied Knowledge Processing, University of Linz
Altenberger Strasse 69, A-4040 Linz, Austria
wagner@ifs.uni-linz.ac.at

Abstract. Online Analytical Processing (OLAP) data is frequently organized in the form of multidimensional data cubes each of which is used to examine a set of data values, called measures, associated with multiple dimensions and their multiple levels. In this paper, we first propose a conceptual multidimensional data model, which is able to represent and capture natural hierarchical relationships among members within a dimension as well as the relationships between dimension members and measure data values. Hereafter, dimensions and data cubes with their operators are formally introduced. Afterward, we use UML (Unified Modeling Language) to model the conceptual multidimensional model in the context of object oriented databases.

1. Introduction

Data warehouses and OLAP are essential elements of decision support [5], they enable business decision makers to creatively approach, analyze and understand business problems [16]. While data warehouses are built to store very large amounts of integrated data used to assist the decision-making process [9], the concept of OLAP, which is first formulated in 1993 by [6] to enable business decision makers to work with data warehouses, supports dynamic synthesis, analysis, and consolidation of large volumes of multidimensional data [7]. OLAP systems organize data using the multidimensional paradigm in the form of data cubes, each of which is a combination of multiple dimensions with multiple levels per dimension. Summarized data is pre-aggregated and stored with the main purpose to explore the relationship between independent, static variables, *dimensions*, and dependent, dynamic variables, *measures* [3]. Moreover, dimensions always have structures and are linguistic categories that describe different ways of looking at the information [4]. These dimensions contain one or more natural hierarchies, together with other attributes that do not have a hierarchy's relationship to any of the attributes in the dimensions [10]. Having and handling the predefined hierarchy or hierarchies within dimensions provide the foundation of two typical operations like *rolling up* and *drilling down*. Because unbalanced and multiple hierarchical structures (Fig. 1,2) are the common structures of dimensions, the two current OLAP technologies, namely ROLAP and MOLAP, have limitations in the handling of dimensions with these structures [15].

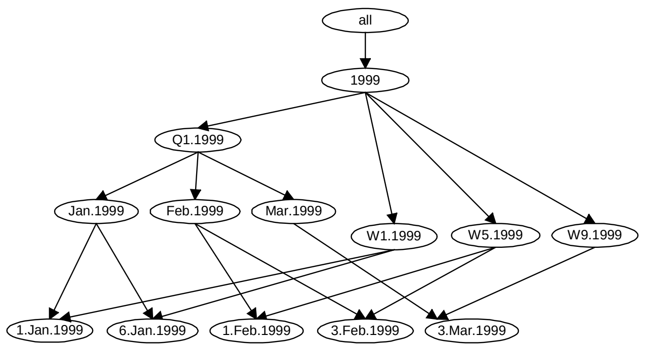

Fig. 1. An instance of the dimension *Time* with unbalanced and multiple hierarchical structure

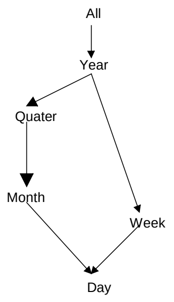

Fig. 2. A schema of the dimension *Time* with multihierarchical structure.

ROLAP (Relational OLAP) products are set on top of existing relational database management systems (RDBMS), which are well standardized and meet the needs of storing large amounts of data. Dimensions and facts are mapped into relational tables, called fact and dimension tables, organized as *Star Schema* and/or *Snowflake Schema* [10]. Therefore in many cases, ROLAP products are not suitable for handling dimensions with multihierarchical and unbalanced structures. Furthermore, existing relational query languages (e.g. SQL) are not sufficiently powerful or flexible enough to support true OLAP capabilities [19]. [3] clearly demonstrated the mismatch between multidimensional operations and SQL.

Although MOLAP (Multidimensional OLAP) easily supports dimensions with multiple and unbalanced hierarchical structures and MOLAP queries are very powerful and flexible in terms of OLAP processing [14], there are still several challenges for these products. First, the underlying data structures are limited in their ability to support multiple subject areas and to provide access to detailed data. Navigation and analysis of data is limited because the data is designed according to previously determined requirements [7]. In addition, with products that require complete pre-calculation, the dimensional explosion could result in physical database that is unmanageable [14].

The first goal of this paper is the introduction of a conceptual multidimensional data model that facilitates a precise rigorous conceptualization for OLAP. First, the model is able to represent and capture natural hierarchical relationships among members within a dimension. Therefore, dimensions with complex structures, such as: unbalanced and multihierarchical structures, can be handled. Moreover, the data model is able to represent the relationships between dimension members and measure data values by mean of cube cells. Hereafter, the data cubes, which are basic components in multidimensional data analysis, are formally introduced. Furthermore, cube operators (e.g. *jumping*, *rollingUp* and *drillingDown*) are defined in a very elegant manner.

The second goal is the modeling of the conceptual multidimensional data model in term of classes by using UML. Based on the formal representation of the class specifications in UML, the design and implementation of the data model for object oriented databases are straightforward.

The remainder of this paper is organized as follows. In section 2, we discuss about related works. Then in section 3, we introduce a conceptual data model that will be mapped into object-oriented database by means of UML in section 4. The paper concludes with section 5, which presents our current and future works.

2. Related works

Since Codd's [6] formulated the term Online Analytical Processing (OLAP) in 1993, many commercial products, like Arborsoft (now Hyperion) Essbase, Cognos Powerplay or MicroStrategy's DSS Agent have been introduced on the market [2]. But unfortunately, sound concepts were not available at the time of the commercial products being developed. The scientific community struggles hard to deliver a common basis for multidimensional data models ([1], [4], [8], [11], [12], [13], [21]). The data models presented so far differ in expressive power, complexity and formalism. In the followings, some research works in the field of data warehousing systems and OLAP tools are summarized.

In [12] a multidimensional data model is introduced based on relational elements. Dimensions are modeled as "dimension relations", practically annotating attributes with dimension names. The cubes are modeled as functions from the Cartesian product of the dimensions to the measure and are mapped to "grouping relations" through an applicability definition.

In [8] n-dimensional tables are defined and a relational mapping is provided through the notation of completion. Multidimensional database are considered to be composed from set of tables forming denormalized star schemata. Attribute hierarchies are modeled through the introduction of functional dependencies in the attributes of dimension tables.

[4] modeled a multidimensional database through the notations of dimensions and f-tables. Dimensions are constructed from hierarchies of dimension levels, whereas f-tables are repositories for the factual data. Data are characterized from a set of roll-up functions, mapping the instance of a dimension level to instances of other dimension level.

In statistical databases, [17] presented a comparison of work done in statistical and multidimensional databases. The comparison was made with respect to application areas, conceptual modeling, data structure representation, operations, physical organization aspects and privacy issues.

In [3], a framework for Object-Oriented OLAP is introduced. Two major physical implementations exist today: ROLAP and MOLAP and their advantages and disadvantages due to physical implementation were introduced. The paper also presented another physical implementation called O3LAP model.

[20] took the concepts and basic ideas of the classical multidimensional model based on the Object-Oriented paradigm. The basic elements of their Object Oriented

Multidimensional Model are dimension classes and fact classes. They also presented cube classes as the basic structure to allow a subsequent analysis of the data stored in the system.

In this paper, we address a suitable mutidimensional data model for OLAP. The main contributions are: (a) the introduction of a formal multidimensional data model; (b) the very elegant manners of definitions of three cube operators, namely *jumping*, *rollingUp* and *drillingDow*n; (c) the modeling of the conceptual multidimensional data model in term of classes by using UML.

3. A Conceptual Data Model

In our approach, a multidimensional data model is constructed based on a set of dimensions $\mathcal{D} = \{D_1,..,D_x\}, x \in \mathbf{N}$, a set of measures $\mathcal{M} = \{M_1,..,M_y\}, y \in \mathbf{N}$ and a set of data cubes $\mathcal{C} = \{C_1,..,C_z\}, z \in \mathbf{N}$. The following sections formally introduce the descriptions of dimensions with their structures, measures and data cubes.

3.1. The Concepts of Dimensions

First, we introduce hierarchical relationships among dimension members by means of one hierarchical domain per dimension. A hierarchical domain is a set of dimension members, organized in hierarchy of levels, corresponding to different levels of granularity. It allows us to consider a dimension schema as a partially ordered set of levels. In this concept, a hierarchy is a path along the dimension schema, beginning at the root level and ending at a leaf level. Moreover, the recursive definitions of two dimension operators, namely *ancestor* and *descendant*, provide abilities to navigate along a dimension structure. In a consequence, dimensions with any complexity in their structures can be captured with this data model.

Definition 3.1.1. [*Dimension Hierarchical Domain*] A hierarchical domain of a dimension D is a non-empty set and denoted by $dom(\mathrm{D}) = \{all\} \cup \{dm_1,..,dm_n\}$, where:

- Each dimension member dm_i is a data item within a dimension. E.g. *1999*, *Q1.1999*, *Jan.1999*, and *1.Jan.1999*, etc are dimension members within the dimension *Time* (Fig. 1).
- Such that the graph $G_{\prec_M} = (V, E)$, defined as the representation over the binary relation over the $dom(\mathrm{D})$, is a tree and defined as follows:
  $V = dom(\mathrm{D})$,
  $E \subset dom(\mathrm{D}) \times dom(\mathrm{D})$. $\forall (dm_i, dm_j) \in E : dm_i \prec_M dm_j$ is an edge in $G_{\prec_M}$. The edge is given when there is an ordered relationship in the sense of hierarchy.
- And the two operators {+,-}: $\forall dm_i \in dom(\mathrm{D})$:
  $-(dm_i) = \{dm_j \in dom(\mathrm{D}) : dm_j \prec_M dm_i\}$

$+(dm_i) = \{dm_k \in dom(\mathrm{D}) : dm_i \prec_M dm_k\}$

- The all or root member: $(\exists! all \in dom(\mathrm{D}))(\neg\exists dm \in dom(\mathrm{D}) : dm \prec_M all)$.
- Leaf members: $(\forall dm_i \in dom(\mathrm{D}))(\neg\exists dm_j \in dom(\mathrm{D}), i \neq j : dm_i \prec_M dm_j)$.

Example: Figure 1 shows a representation in tree term of the dimension *Time*. Hereafter, we have:

dom(*Time*)={*all,1999,Q1.1999,..,3.Mar.1999*},
all $\prec_M$ *1999,1999* $\prec_M$ *Q1.1999,...,Mar.1999* $\prec_M$ *3.Mar.1999,*
-(*1999*)=*all*; +(*1999*)={*Q1.1999,W1.1999,W5.1999,W9.1999*}

Definition 3.1.2. [*Dimension Levels*] Let $Levels(\mathrm{D}) = All \cup \{l_1,..,l_h\}, h \in \mathbf{N}$ be a finite set of levels of a dimension D, where:

- The collection of subsets $\{dom(l_1),..,dom(l_h)\}$ is a partition of *dom*(D),
- The *All* or root level: $\exists! All \in Levels(\mathrm{D}) : dom(All) = \{all\}$,
- Leaf levels: $\{l_i \in Levels(\mathrm{D}) \mid \forall dm_j \in dom(l_i) : dm_j \text{ a } leafmember\}$.

Example: The dimension *Time* has three levels *Levels*(*Time*)={*All,Year,Quarter, Month,Week,Day*}. And:

dom(*All*)= {*all*}, *dom*(*Year*)={*1999*}, *dom*(*Quarter*)= {*Q1.1999*}
dom(*Month*)= {*Jan.1999,Feb.1999,Mar.1999*},
dom(*Week*)={*W1.1999,W5.1999,W9.1999*},
dom(*Day*)= {*1.Jan.1999,6.Jan.1999,1.Feb.1999,3.Feb.1999,3.Mar.1999*}

Definition 3.1.3. [*Dimension Schema*] A schema of a dimension D, denoted by *DSchema*(D)=$\langle Levels(\mathrm{D}), \prec_L \rangle$, is a partially ordered set of levels:

- *Levels*(D) is a finite set of dimension levels,
- And $\prec_L$ is an ordered relation over the levels and satisfies the following condition:
  $l_i \prec_L l_j$ if $(\exists dm_t \in dom(l_i))$ *and* $(\exists dm_u \in dom(l_j)) : dm_t \prec_M dm_u$.

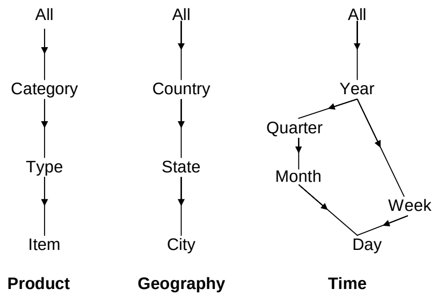

Fig. 3. Schemas of three dimensions *Product, Geography* and *Time*

Example: Figure 3 is used to describe schemas of three dimensions *Product, Geography,* and *Time*.

DSchema(*Product*)={*All* $\prec_L$ *Category, Category* $\prec_L$ *Type, Type* $\prec_L$ *Item*}

DSchema(*Geography*)={*All* $\prec_L$ *Country, Country* $\prec_L$ *State, State* $\prec_L$ *City*}
DSchema(*Time*)={*All* $\prec_L$ *Year ,Year* $\prec_L$ *Quarter, Quarter* $\prec_L$ *Month, Month* $\prec_L$ *Day ,*
All $\prec_L$ *Year, Year* $\prec_L$ *Week, Week* $\prec_L$ *Day*}

Definition 3.1.4. [*Dimension Path*] A path within a dimension schema is a linear, totally ordered list of levels and can be defined as follows:

$\forall l_i, l_j \in Levels(\mathrm{D})$:

$$path(l_i,l_j)=\begin{cases}\{l_i \prec_L l_j\} & \text{If } l_i \prec_L l_j\\ \{l_i \prec_L l_t,..,l_u \prec_L l_j\} & \text{Else if } \exists l_t,..,l_u \in Levels(\mathrm{D}): l_i \prec_L l_t,..,l_u \prec_L l_j\\ \phi & \text{Else}\end{cases}$$

Definition 3.1.5. [*Dimension Hierarchy*] A hierarchy is defined by a $path(All, l_{leaf})$ within the schema of a dimension D. The path begins at *All* (*root*) level and ends at a leaf level.

Let $H(\mathrm{D})=\{h_1,..,h_m\}, m \in \mathbf{N}$ be a set of hierarchies of a dimension D. If *m*=1 then the dimension has single hierarchical structure, else the dimension has multihierarchical structure.

Definition 3.1.6. [*Dimension Operators*] Two dimension operators (DO), namely *ancestor* and *descendant*, are defined recursively as follows:

$\forall l_i, l_a, l_d \in Levels(\mathrm{D})$ and $\forall dm \in dom(l_i) \subset dom(\mathrm{D})$:

$$ancestor(dm,l_a,\mathrm{D})=\begin{cases} undefined & \text{If } (path(l_a,l_i)=\phi)\\ dm^- \in dom(l_a): dm^- \in -(dm) & \text{Else If } (l_a \prec_L l_i)\\ ancestor(dm^-,l_a,\mathrm{D}), \text{where}: & \text{Else}\\ \quad dm^- = ancestor(dm,l_p,\mathrm{D}): l_p \prec_L l_i & \end{cases}$$

$$descendant(dm,l_d,\mathrm{D})=\begin{cases} undefined & \text{If } (path(l_i,l_d)=\phi)\\ \{dm^+ \in dom(l_d): dm^+ \in +(dm)\} & \text{Else If } (l_i \prec_L l_d)\\ descendant(dm^+,l_d,\mathrm{D}), \text{where}: & \text{Else}\\ dm^+ \in descentdant(dm,l_n,\mathrm{D}): l_i \prec_L l_n & \end{cases}$$

Example: *ancestor*(*Q1.1999,Year,Time*)=*1999*,
descendant(*Q1.1999,Month,Time*)= {*Jan.1999,Feb.1999,Mar.1999*},

3.2. The Concepts of Measures

In this section we introduce measures, which are the objects of analysis in the context of multidimensional data model.

Definition 3.2.1. [*Measure Schema*] A schema of a measure M is a tuple $MSchema(\mathrm{M}) = \langle Fname, \mathrm{O} \rangle$, where:

- *Fname* is a name of a corresponding fact,
- $\mathrm{O} \in \Omega \cup \{\mathrm{NONE}, \mathrm{COMPOSITE}\}$ is an operation type applied to a specific fact [2]:
  - Ω={SUM, COUNT, MAX, MIN} is a set of aggregation functions,
  - COMPOSITE is an operation (e.g. average), where measures cannot be utilized in order to automatically derive higher aggregations,
  - NONE measures are not aggregated. In this case, the measure is the fact.

Definition 3.2.2. [*Measure Domain*] Let $\mathcal{N}$ be a numerical domain where a measure value is defined (e.g. **N**, **Z**, **R**). The domain of a measure M is a subset of $\mathcal{N}$. We denote by $dom(\mathrm{M}) \subset \mathcal{N}$.

3.3. The Concepts of Data Cubes

A multidimensional cube is constructed based on a set dimensions and a set of measures, and consists a collection of cells. Each cell is an intersection among a set of dimension members and measure data values. Furthermore, cells are grouped into granular groupbys, each of which expresses a mapping from the domains of *x*-tuple of dimension levels (independent variables) to *y*-numerical domains of *y*-tuple of numeric measures (dependent variables).

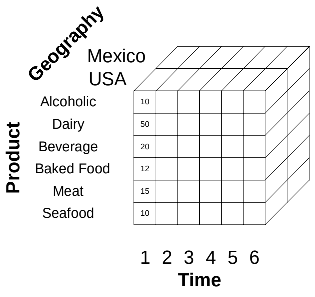

Fig. 4. *Sale* cube includes dimensions: *Geography, Product* and *Time* and a fact: *Sale amount*.

Given *x* dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x \in \mathbf{N}$, and *y* measures $\mathrm{M}_1,..,\mathrm{M}_y, y \in \mathbf{N}$.

Definition 3.3.1. [*Cube Schema*] A cube schema is tuple $CSchema(\mathrm{C}) = \langle Cname, DSchemas, MSchemas \rangle$:

- *Cname* is the name of a cube,
- *DSchemas* are the schemas of *x* dimensions, denoted by $DSchemas = < DSchema(\mathrm{D}_1),.., DSchema(\mathrm{D}_x) >$,
- *MSchemas* are the schemas of *y* measures, denoted by $MSchemas = < MSchema(\mathrm{M}_1),.., MSchema(\mathrm{M}_y) >$.

Definition 3.3.2. [*Cube Domain*] Given a function:

$$f : dom(\mathrm{D}_1)\times..\times dom(\mathrm{D}_x)\times dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y) \to \{true,\ false\},$$

A cube domain, denoted by $dom(\mathrm{C}) = \{c_1,..,c_k\}, k \in \mathbf{N}$ is determined as follows:

$$dom(\mathrm{C}) = \{c = (dms,\ fms) | dms \in dom(\mathrm{D}_1)\times..\times dom(\mathrm{D}_x),$$
$$fms \in dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y) : f(dms,\ fms) = true\}$$

3.4. Operating Group by, Rolling Up, Drilling Down in our model

Let a cube C be constituted from x dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x \in \mathbf{N}$, and y measures $\mathrm{M}_1,..,\mathrm{M}_y, y \in \mathbf{N}$. We define groupby and three operators, namely *jumping*, *rollingUp* and *drillingDown* as follows:

Definition 3.4.1. [*Groupby*] A groupby is triple $\mathrm{G}=\langle Gname, GSchema(\mathrm{G}), dom(\mathrm{G})\rangle$ where:

- *Gname* is the name of this groupby,
- $GSchema(\mathrm{G})=\langle GLevels(\mathrm{G}), GMSchemas(\mathrm{G})\rangle$:

  $GLevels(\mathrm{G}) =< l_{\mathrm{D}_1},..,l_{\mathrm{D}_x} >\in Levels(\mathrm{D}_1)\times..\times Levels(\mathrm{D}_x)$ is a x-tuple of levels of the x dimensions $\mathrm{D}_1,..,\mathrm{D}_x, x \in \mathbf{N}$.

  $GMSchemas(\mathrm{G}) =< MSchema(\mathrm{M}_1),.., MSchema(\mathrm{M}_y) >$ is a y-tuple of measure schemas of the y measures $\mathrm{M}_1,..,\mathrm{M}_y, y \in \mathbf{N}$.
- $dom(\mathrm{G}) = \{c =< dms,\ fms >\in dom(\mathrm{C}) \mid dms \in dom(l_{\mathrm{D}_1})\times..\times dom(l_{\mathrm{D}_x}),$
  $fms \in dom(\mathrm{M}_1)\times..\times dom(\mathrm{M}_y)\}$

Let h_i be a number of levels of each dimension D_i ($1 \le i \le x$). The total set of groupbys over a cube C is defined as $Groupbys(\mathrm{C}) = \{\mathrm{G}_1,..,\mathrm{G}_p\}, p = \prod_{i=1}^{x} h_i$ [18].

Definition 3.4.2. [*Cube Operators*] The three basic navigational cube operators (*CO*), namely *jumping*, *rollingUp* and *drillingDown*, which are applied to navigate along a data cube C, corresponding to a dimension D*ᵢ*, are defined as follows:

Given a current groupby G_c, associated with a level l_c of a dimension D_i, and three other levels $l_j, l_r, l_d \in Levels(\mathrm{D}_i)$.

- *Jumping:*

  $$jumping(\mathrm{G}_c, l_j, \mathrm{D}_i) = \mathrm{G}_j =< GLevels(\mathrm{G}_j), GMSchemas(\mathrm{G}_j) >$$

  Where:

  $$GMSchemas(\mathrm{G}_j) = GMSchemas(\mathrm{G}_c),$$
  $$GLevels(\mathrm{G}_j)(i) = l_j,\ GLevels(\mathrm{G}_j)(k) = GLevels(\mathrm{G}_c)(k), \forall k \neq i.$$
- *Rolling Up:* $\forall dm \in dom(l_c)$, $\mathrm{G}_r = jumping(\mathrm{G}_c, l_r, \mathrm{D}_i)$:

$$rollingUp(G_c, dm, l_r, D_i) = G_r^{sub} =< GSchema(G_r^{sub}), dom(G_r^{sub}) >$$

Where:

$$GSchema(G_r^{sub}) = GSchema(G_r),$$

$$dom(G_r^{sub}) = \{c_r \in dom(G_r) \mid \exists c \in dom(G_c) : c.dms(i) = dm,$$

$$c_r.dms(i) = ancestor(dm, l_r, D_i),\ c_r.dms(j) = c.dms(j), \forall j \neq i\}$$

- *Drilling Down:* $\forall dm \in dom(l_c)$, $G_d = jumping(G_c, l_d, D_i)$:

$$drillingDown(G_c, dm, l_d, D_i) = G_d^{sub} =< GSchema(G_d^{sub}), dom(G_d^{sub}) >$$

Where:

$$GSchema(G_d^{sub}) = GSchema(G_d),$$

$$dom(G_d^{sub}) = \{c_d \in dom(G_d) \mid \exists c \in dom(G_c) : c.dms(i) = dm,$$

$$c_d.dms(i) \in descendant(dm, l_d, D_i),\ c_d.dms(j) = c.dms(j), \forall j \neq i\}$$

4. Modeling Multidimensional Data Model

In this section, UML is used to model dimensions, measures and data cubes in context of an object oriented data model. All conceptual components, which are introduced in section 3, are mapped as classes. Figure 5 illustrates the modeling for our data model in term of class diagrams by using UML.

4.1. The Modeling of Dimensions

The dimension concepts, such as: dimension members, levels, dimension schemas, hierarchy and dimension, are modeled in term of class diagrams by using UML. First, a hierarchical domain of dimension members within a dimension is handled by means of the *DMember* class. The two dimension member operators {+} and {-} are mapped into the two methods *getFathers* and *getChildren*, built-in every instance of this class. Hereafter, The *Level*, *DSchema*, *Hierarchy* classes are defined to describe the concepts of level, dimension schema and dimension hierarchy. Afterwards, each instance of the *Dimension* class describes a dimension. The dimension operators, namely *ancestor* and *descendant* are mapped as two methods with the same names.

4.1.1. *DMember.* Each instance of this class describes a dimension member within a hierarchical domain of a dimension.

- *Attributes:*
  description (String) - That is a data item within a dimension.
- *Relationships:*
  fathers (Set<*DMember*>) - A set of referred *DMember* objects.
  children (Set<*DMember*>) - A set of referred *DMember* objects
- *Main methods:*
  getFathers() (return Set<*DMember*>) - This method describes the operator {-}.
  getChildren() (returns Set<*DMember*>) - This method describes the operator {+}.

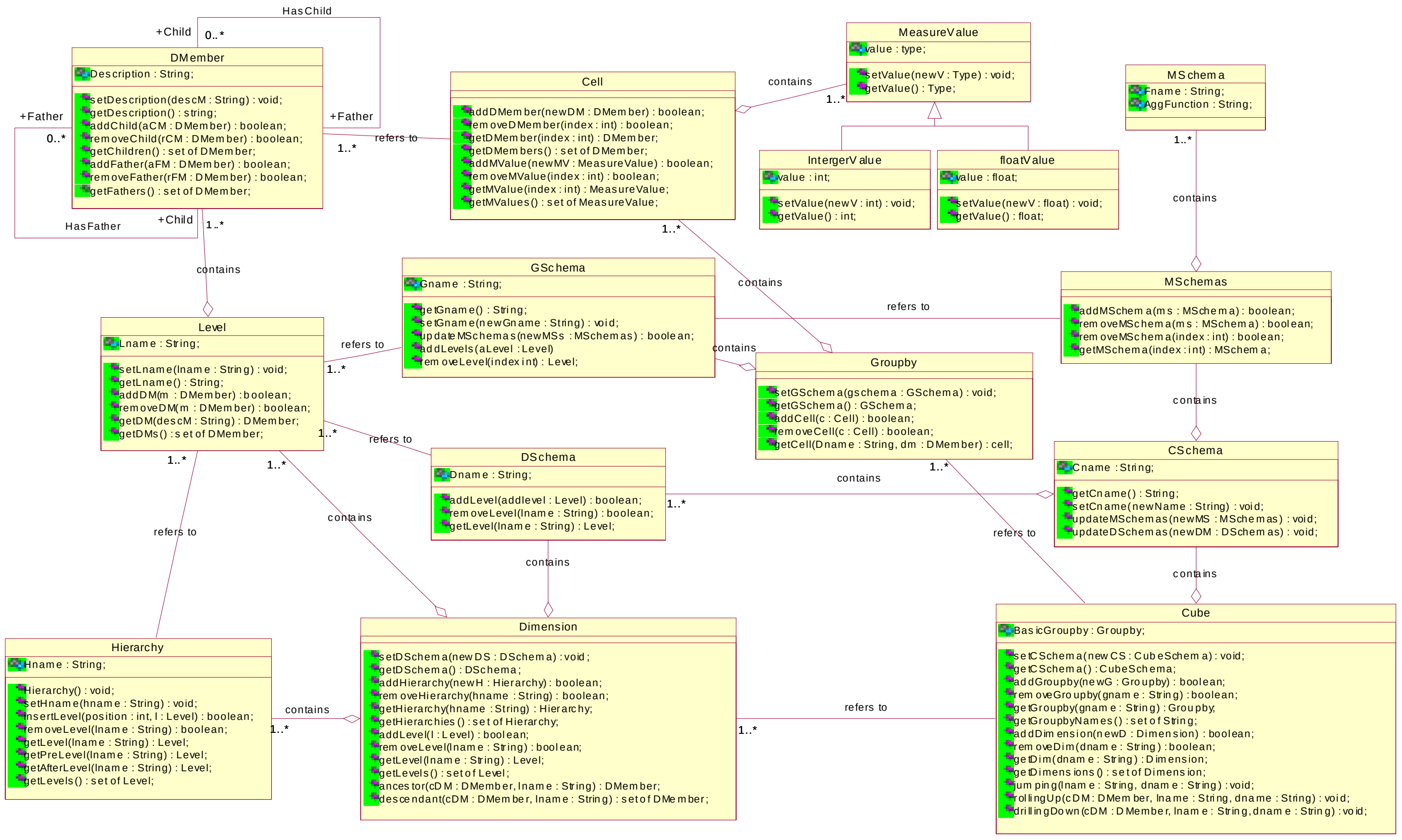
HasChild
+Child
0..*
DMember
Description : String;
setDescription(descM : String) : void;
getDescription() : string;
addChild(aCM : DMember) : boolean;
removeChild(rCM : DMember) : boolean;
getChildren() : set of DMember;
addFather(aFM : DMember) : boolean;
removeFather(rFM : DMember) : boolean;
getFathers() : set of DMember;
+Father
0..*
+Father
HasFather
+Child
1..*
refers to
1..*
Cell
addDMember(newDM : DMember) : boolean;
removeDMember(index : int) : boolean;
getDMember(index : int) : DMember;
getDMembers() : set of DMember;
addMValue(newMV : MeasureValue) : boolean;
removeMValue(index : int) : boolean;
getMValue(index : int) : MeasureValue;
getMValues() : set of MeasureValue;
contains
1..*
MeasureValue
value : type;
setValue(newV : Type) : void;
getValue() : Type;
IntergerValue
value : int;
setValue(newV : int) : void;
getValue() : int;
floatValue
value : float;
setValue(newV : float) : void;
getValue() : float;
MSchema
Fname : String;
AggFunction : String;
1..*
contains
contains
1..*
GSchema
Gname : String;
getGname() : String;
setGname(newGname : String) : void;
updateMSchemas(newMSs : MSchemas) : boolean;
addLevels(aLevel : Level)
removeLevel(index int) : Level;
contains
refers to
MSchemas
addMSchema(ms : MSchema) : boolean;
removeMSchema(ms : MSchema) : boolean;
removeMSchema(index : int) : boolean;
getMSchema(index : int) : MSchema;
contains
Level
Lname : String;
setLname(lname : String) : void;
getLname() : String;
addDM(m : DMember) : boolean;
removeDM(m : DMember) : boolean;
getDM(descM : String) : DMember;
getDMs() : set of DMember;
refers to
1..*
contains
Groupby
setGSchema(gschema : GSchema) : void;
getGSchema() : GSchema;
addCell(c : Cell) : boolean;
removeCell(c : Cell) : boolean;
getCell(Dname : String, dm : DMember) : cell;
contains
1..*
refers to
DSchema
Dname : String;
addLevel(addlevel : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
1..*
contains
CSchema
Cname : String;
getCname() : String;
setCname(newName : String) : void;
updateMSchemas(newMS : MSchemas) : void;
updateDSchemas(newDM : DSchemas) : void;
1..*
1..*
1..*
refers to
contains
refers to
contains
contains
Hierarchy
Hname : String;
Hierarchy() : void;
setHname(hname : String) : void;
insertLevel(position : int, l : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
getPreLevel(lname : String) : Level;
getAfterLevel(lname : String) : Level;
getLevels() : set of Level;
contains
1..*
Dimension
setDSchema(newDS : DSchema) : void;
getDSchema() : DSchema;
addHierarchy(newH : Hierarchy) : boolean;
removeHierarchy(hname : String) : boolean;
getHierarchy(hname : String) : Hierarchy;
getHierarchies() : set of Hierarchy;
addLevel(l : Level) : boolean;
removeLevel(lname : String) : boolean;
getLevel(lname : String) : Level;
getLevels() : set of Level;
ancestor(cDM : DMember, lname : String) : DMember;
descendant(cDM : DMember, lname : String) : set of DMember;
refers to
1..*
Cube
BasicGroupby : Groupby;
setCSchema(newCS : CubeSchema) : void;
getCSchema() : CubeSchema;
addGroupby(newG : Groupby) : boolean;
removeGroupby(gname : String) : boolean;
getGroupby(gname : String) : Groupby;
getGroupbyNames() : set of String;
addDimension(newD : Dimension) : boolean;
removeDim(dname : String) : boolean;
getDim(dname : String) : Dimension;
getDimensions() : set of Dimension;
jumping(lname : String, dname : String) : void;
rollingUp(cDM : DMember, lname : String, dname : String) : void;
drillingDown(cDM : DMember, lname : String, dname : String) : void;

Fig.5. The modeling of the object oriented multidimensional data model

4.1.2. Level. Each instance of the *Level* class describes a level of a dimension.
Attributes:
dmembers (Set<*DMember*>) - A set of contained *DMember* objects.

4.1.3. DSchema. Each instance of the *DSchema* describes a dimension schema.

- *Attributes:*
  dname (*String*) - That is a dimension name.
- *Relationships:*
  levels (Set<*Level*>) - A set of Level objects.

4.1.4. Hierarchy. Each instance of the Hierarchy class describes a hierarchy of levels within a dimension.

- *Attributes*:
  dname (String) - That is a hierarchy name
- *Relationships*:
  levels (Set<*Level*>) - A set of *Level* objects

4.1.5. Dimension. Each instance of this class describes a dimension.

- *Attributes:*
  dschema (*DSchema*) - A *DSchema* object.
  hierarchies (Set<*Hierarchy*>) - A set of *Hierarchy* objects.
  levels (Set<*Level*>) - A set of *Level* objects.
- *Main methods:*
  ancestor(*DMember* cDM;*lname*:String) (returns *DMember*): *ancestor* operator.
  descendant(*DMember* cDM,*lname*:String) (Set<*DMember*>): *descendant* operator.

4.2. The Modeling of Measures

Measure schemas and measure data values are mapped into classes. First, an *MSchema* describes a measure schema. Afterwards, The *MValue* is an abstract type that serves as a common super type for measure values. It is obvious that the two classes, namely *intMValue* and *floatValue*, are subclasses of *MValue*.

4.2.1. MSchema. Each instance of this class describes a measure schema.

- *Attributes:*
  fname (String) - That is a fact name.
  aggFunction (String)- An aggregation function, such as *Max, Min, Sum, Count, None* and *Composite*.

4.2.2. MValue. *MValue* is an abstract type that serves as common super type for measure values.
Attributes:
value (*Type*) – That describes a measure value.

4.2.3. intMValue. The *intMValue* is a subclass of *MValue*. Each instance of this class describes an integer measure value.

- *Specializes:*
  MValue: The *intMValue* class inherits from *MValue* class

- *Attributes:*
  value (int) – The value is overridden as integer.

4.2.4. floatMValue. The *floatMValue* is a subclass of *MValue*. Each instance of this class describes a float measure value.

- *Specializes:*
  MValue: The *floatMValue* class inherits from *MValue* class
- *Attributes:*
  value (float) – The value is overridden as float.

4.3. The Modeling of Multidimensional Components

First, each instance of the *CSchema*, which contains an *x*-tuple of *DSchemas* and a *y*-tuple of *MSchemas*, describes a cube schema. Then, a *Cell* refers to *x DMembers* of *x* dimensions and *y MValues* of *y* measures. In addition, a *GSchema* contains *x Level*s of *x* dimensions and the *y*-tuple *MSchemas*. Afterwards, a *Groupby* contains a *GSchema* and a subset of *Cell*s. As a consequence, a *Cube* contains a *CSchema* and a *BasicGroupby* (*Groupby*), is associated with a set of *Dimension*s, and a set of *Groupby*s. The three methods, i.e. *jumping*, *rollingUp* and *drillingDown*, are the mappings of the three operators with the same names.

4.3.1. CSchema. Each instance of the *CSchema* describes a cube schema.

- *Attributes:*
  dschemas (Set<*DSchema*>) - A set of *x DSchema* objects,
  mschemas (Set<*MSchema*>) - A set of *y MSchema* objects.

4.3.2. Cell. Each instance of this class describes a cube cell.

- *Relationships:*
  dmembers (Set<*DMember*>) - A set of *x DMember* objects.
  mvalues (Set<*MValue*>) - A set of *y MValue* objects.

4.3.3. GSchema. Each instance of the *GSchema* class describes a groupby schema.

- *Relationships:*
  levels (Set<*Level*>) - A set of *x Level* objects.
  mschemas (Set<*MSchema*>) - A set of *y MSchema* objects.

4.3.4. Groupby. Each instance of this class describes a groupby.

- *Attributes:*
  gschema (*GSchema*): A *GSchema* object that describes a groupby schema.
  cells (Set<*Cell*>): A set of *Cell* objects.

4.3.5. Cube. Each instance of the *Cube* class describes a data cube.

- *Attributes:*
  cschema (*CSchema*) – A *CSchema* describes a cube schema of a cube.
  basicgroupby (*Groupby*) - A *Groupby* object.
- *Relationships:*
  dimensions (Set<*Dimension*>) - *Dimensions* express for *x* dimensions of a cube.
  groupbies (Set<*Groupby*>) – A set of *Groupby* objects,

- *Main methods:*
  jumping(*lname*:String;*dname*:String) (void) - *jumping* operator.
  rollingUp(*cDM*:*DMember*;*lname*:String;*dname*:String) (void) - *rollingUp* operator.
  drillingDown(*cDM*:*DMember*;*lname*:String;*dname*:String) (void) - *drillingDown* operator.

5. Conclusion and future works

In this paper, we have introduced the conceptual multidimensional data model, which facilitates even sophisticated constructs based on multidimensional data units or members such as dimension members, measure data values and then cells. The model is able to represent and capture natural hierarchical relationships among dimension members. Dimensions with complexity of their structures, such as: unbalanced and multihierarchical structures, can be modeled in an elegant and consistent way. Moreover, the data model represents the relationships between dimension members and measure data values by mean of cube cells. In consequence, data cubes, which are basic components in multidimensional data analysis, and their operators are formally introduced in a very elegant manner. We have also proposed a modeling of the conceptual multidimensional data model in term of classes by means of UML, which is an object oriented standard analysis and design notation.

In context of future works, we are investigating two approaches for implementation: pure object-oriented orientation and object-relational approach. With the first model, dimensions and cube are mapped into an object-oriented database in term of classes. In the other alternative, dimensions, measure schema, and cube schema are grouped into a term of metadata, which will be mapped into object-oriented database in term of classes. Some useful methods built in those classes are used to give the required Ids within those dimensions. The given Ids will be joined to the fact table, which is implemented in relational database.

Acknowledgment

This work is partly supported by the EU-4th Framework Project GOAL (Geographic Information Online Analysis)-EU Project #977071 and by the ASEAN European Union Academic Network (ASEA-Uninet), Project EZA 894/98.

References

1. Agrawal, R., Gupta, A., Sarawagi, A.: Modeling Multidimensional Databases. IBM Research Report, IBM Almaden Research Center, September 1995.
2. Albrecht, J., Guenzel, H., Lehner, W.: Set-Derivability of Multidimensiona Aggregates. First International Conference on Data Warehousing and Knowledge Discovery. DaWaK'99, Florence, Italy, August 30 - September 1.

3. Buzydlowski, J. W., Song, II-Y., Hassell, L.: A Framework for Object-Oriented On-Line Analytic Processing. DOLAP 1998
4. Cabibbo, L., Torlone, R.: A Logical Approach to Multidimensional Databases. EDBT 1998
5. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record Volume 26, Number 1, September 1997.
6. Codd, E. F., Codd, S.B., Salley, C. T.: Providing OLAP (On-Line Analytical Processing) to user-analysts: An IT mandate. Technical report, 1993.
7. Connolly, T., Begg, C.: Database system: a practical approach to design, implementation, and management. Addison-Wesley Longman, Inc., 1999.
8. Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases, Proc. VLDB'97.
9. Hurtado, C., Mendelzon, A., Vaisman, A.: Maintaining Data Cubes under Dimension Updates. Proc IEEE/ICDE '99.
10. Kimball, R.: The Data Warehouse Lifecycle Toolkit. John Wiley & Sons, Inc., 1998.
11. Lehner, W.: Modeling Large Scale OLAP Scenarios. 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, 23-27, March 1998.
12. Li, C., Wang, X.S.: A Data Model for Supporting On-Line Analytical Processing. CIKM 1996.
13. Mangisengi, O., Tjoa, A M., Wagner, R.R.: Multidimensional Modelling Approaches for OLAP. Proceedings of the Ninth International Database Conference "Heterogeneous and Internet Databases" 1999, ISBN 962-937-046-8. Ed. J. Fong, Hong Kong, 1999
14. McGuff, F., Kador, J.: Developing Analytical Database Applications. Prentice Hall PTR, 1999.
15. Nguyen, T.B., Tjoa, A M., Wagner, R.R.: *Conceptual Object Oriented Multidimensional Data Model for OLAP*. Technical Report, IFS, Vienna 1999.
16. Samtani, S., Mohania, M.K., Kumar, V., Kambayashi, Y.: Recent Advances and Research Problems in Data Warehousing. ER Workshops 1998.
17. Shoshani, A.: OLAP and Statistical Databases: Similarities and Differences. Tutorials of PODS 1997.
18. Shukla A., Deshpande, P., Naughton, J. F., Ramasamy, K.: Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies. VLDB 1996: 522-531
19. Thomsen, E.: OLAP solutions: Building Multidimensional Information Systems. John Wiley& Sons, Inc., 1997.
20. Trujillo, J., Palomar, M.: An Object Oriented Approach to Multidimensional Database. Conceptual Modeling (OOMD), DOLAP 1998.
21. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube operations. In Proc. 10th Scientific and Statistical Database Management Conference (SSDBM '98), Capri, Italy, June 1998.
22. Wang, M., Iyer, B.: Efficient roll-up and drill-down analysis in relational database. In 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.