A Standard for Representing Multidimensional Properties: The Common Warehouse Metamodel (CWM)

Enrique Medina and Juan Trujillo

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante Spain
{emedina,jtrujillo}@dlsi.ua.es

Abstract. Data warehouses, multidimensional databases, and OLAP tools are based on the multidimensional (MD) modeling. Lately, several approaches have been proposed to easily capture main MD properties at the conceptual level. These conceptual MD models, together with a precise management of metadata, are the core of any related tool implementation. However, the broad diversity of MD models and management of metadata justifies the necessity of a universally understood standard definition for metadata, thereby allowing different tools to share information in an easy form. In this paper, we make use of the Common Warehouse Metamodel (CWM) to represent the main MD properties at the conceptual level in terms of CWM metadata. Then, CWM-compliant tools could interoperate by exchanging their CWM-based metadata in a commonly understood format and benefit of the expressiveness of the MD model at the conceptual level.
Keywords: Conceptual MD modeling, CWM, DW, metadata integration, OLAP.

1 Introduction

Historical information is a key issue available to enterprises for the decision making process. Within a decision support system, enterprises make use of data warehouses (DW), OLAP tools and multidimensional databases (MDDB), based on the multidimensional (MD) modeling to facilitate the analysis of such huge amount of historical data. In the last years, there have been several proposals to accomplish the conceptual MD modeling of these systems; due to space constraints, we refer the reader to [1] for detailed comparison and discussion about these models. We will use the Object-Oriented (OO) conceptual MD modeling approach presented in [10,11], based on the Unified Modeling Language (UML) [8], as it considers many MD issues at the conceptual level such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, or non-strict and complete hierarchies.

Regardless the MD model, the management of metadata has also been identified as a key success factor in DW projects [6]. Metadata is basically defined

Y. Manolopoulos and P. Návrat (Eds.): ADBIS 2002, LNCS 2435, pp. 232–247, 2002.

as data about data, so it captures all kinds of information about complex data structures and processes in a DW. Nevertheless, the heterogeneity between MD models provided by the different OLAP applications leads to the existence of a broad diversity of metadata. In the practice, tools with dissimilar metadata are integrated through the building of complex *metadata bridges*. Such a bridge needs to have detailed knowledge of the metadata structures and interfaces of each tool involved in the integration process. However, a certain amount of information loss occurs when translating from one form of metadata to another. Therefore, the necessity of a globally and universally understood standard definition for metadata should be addressed in order to ensure interoperability, integration and spreading of metadata use in DW projects.

Lately, two industry standards developed by multi-vendor organizations have arisen with respect to the centralized metadata management problem: the Open Information Model (OIM) [5], developed by the Meta Data Coalition (MDC) group, and the Common Warehouse Metamodel (CWM) [7], owned by the Object Management Group (OMG). Both of them specify metamodels which could be seen as conceptual schemas for metadata incorporating application-specific aspects of data warehousing. However, in September 2000, given the support for CWM building within the industry, the MDC membership joined ranks with the OMG in favor of the continued development of the CWM standard. Due to space constraints, we refer the reader to [12] for a deeper comparison of the two competing specifications.

In this paper, we will use the OO conceptual MD modeling approach by [10] because it has been successfully used [11] to represent main MD properties at the conceptual level. For every MD property, we will discuss its representation using the CWM specification [7], thereby allowing the instances of our MD models to be expressed as CWM-based metadata. To the best of our knowledge, no other related works have been done in this context. Instead, only comparison studies have been presented in order to discuss the main aspects of the metadata integration proposals [12][2].

The remainder of this paper is structured as follows: Section 2 briefly summarizes the OO conceptual MD modeling approach used to consider main relevant MD properties. Once this MD model is presented, Section 3 gives an overview of the CWM, as the standard metamodel for data warehouse metadata integration. In this sense, both architectural and organizational issues are discussed in order to give a precise knowledge of the CWM metamodel. Section 4 is the core section of the paper where every particular MD issue is discussed by means of its representation using the CWM specification. To achieve this goal, we enhance the overall discussion by means of specific real-world examples applied to every particular MD property. Finally, conclusions and future works are depicted in Section 5.

2 Conceptual Multidimensional Modeling

Several conceptual MD models have been lately presented to provide an easy set of graphical structures to facilitate the task of conceptual MD modeling, as commented previously in the introduction. In this paper, we will use the OO approach based on the UML notation presented in [10,11], as it considers many relevant MD aspects at the conceptual level. In this section, we will briefly summary how our approach represents both the structural and the dynamic part of MD modeling.

2.1 MD Modeling with UML

In this approach, main MD modeling structural properties are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions.

Dimensions and facts are considered by *dimension classes* and *fact classes* respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of n dimension classes. Thanks to the flexibility of shared aggregation relationships that UML provides, *many-to-many* relationships between facts and particular dimensions can be considered by indicating the 1..* cardinality on the dimension class role. For example, on Fig. 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.

By default, all measures in the fact class are considered additive. For non-additive measures, additive rules are defined as constraints and are also placed in somewhere around the fact class. Furthermore, *derived measures* can also be explicitly considered (constraint /) and their derivation rules are placed between braces in somewhere around the fact class, as can be seen in Fig. 1.a.

Our approach also allows us to define identifying attributes in the fact class, if convenient, by placing the constraint {*OID*} next to a measure name. In this way we can represent *degenerate dimensions* [3,4], thereby providing other fact features in addition to the measures for analysis. For example, we could store the ticket and line numbers as other ticket features in a fact representing sales tickets, as reflected in Fig. 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 (constraint {*D*}). These attributes are necessary for an automatic generation process into commercial OLAP tools, as these tools store this information in their metadata. The multiplicity *1* and *1..** defined in the target associated class role addresses the concepts of *strictness* and *non-strictness*. In addition, defining the {*completeness*} constraint in the target

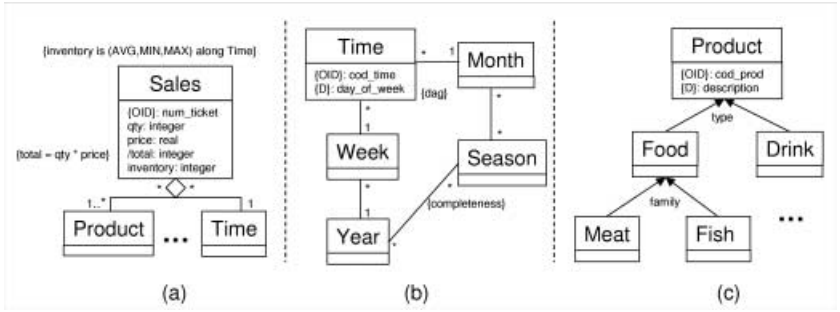

Fig. 1. Multidimensional modeling using UML

associated class role addresses the completeness of a classification hierarchy (see an example on Fig. 1.b). Our approach considers all classification hierarchies non-complete by default.

The *categorization of dimensions*, used to model additional features for an entity'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 on Fig. 1.c.

3 Overview of the CWM

The CWM [7,9] is an open industry standard of the OMG for integrating data warehousing and business analysis tools, based on the use of shared metadata. This standard is based on three key industry standards:

- MOF (*Meta Object Facility*), an OMG metamodeling standard that defines an extensible framework for defining models for metadata, and providing tools with programmatic interfaces to store and access metadata in a repository
- UML (*Unified Modeling Language*), an OMG modeling standard that defines a rich, OO modeling language that is supported by a considerable range of graphical design tools
- XMI (*XML Metadata Interchange*), an OMG metadata interchange standard that allows metadata to be interchanged as streams or files with an XML format

These three standards provide the CWM with the foundation technology to perfectly represent the semantic of data warehousing. The former serves as the foundation model used to specify the CWM metamodel, thereby allowing the latter, i.e. (XMI), to be used to transfer instances of warehouse metadata that conform to the CWM metamodel as XML documents. We will focus on the relationship between MOF and CWM next.

Finally, the UML is used in three different roles: firstly, together with the UML notation and Object Constraint Language (OCL) which are used as the modeling language, graphical notation and constraint language, respectively for defining and representing the CWM; secondly, the UML metamodel is used as the foundation of CWM from which classes and associations are inherited, specifically a subset of the Object Model package; finally, the UML metamodel, specifically its Object Model package, is used as the OO metamodel for representing OO data resources in the CWM.

3.1 CWM and the MOF

The CWM[1] has been designed to conform to the "MOF model". This abstract syntax is a model for defining metamodels, i.e. a meta-metamodel, and is placed in the top level of the four layer architecture shown in Table 1:

Table 1. OMG metadata architecture

Meta-level	MOF Terms	Examples
M3	meta-metamodel	The "MOF Model"
M2	metamodel, meta-metadata	UML Metamodel, CWM Metamodel
M1	model, metadata	UML models, CWM metadata
M0	object, data	Modeled systems, Warehouse data

This layered architecture is a classification of the OMG and MOF metadata terminology used to describe issues in terms of their level in the meta-stack. For example, using the construction metaphor taken from [9], a filing cabinet would represent the role played by the M3-level. As a consequence, the drawers in this filing cabinet containing collections of plans for specific kind of buildings would be M2-level objects. Therefore, the building plans would be M1-level objects. Finally, details of individual bricks and specific customers would occupy the lower level in the OMG metadata architecture, i.e. the M0-level. In this sense, we can describe the CWM as M2-level within this architecture, as observed in Table 1.

[1] For the sake of simplicity, we will refer to the CWM metamodel as simply the CWM throughout the rest of the paper.

Table 2. CWM metamodel layering and its packages

Management	Warehouse Process			Warehouse Operation		
Analysis	Transformation	OLAP	Data Mining	Information Visualization	Business Nomenclature	
Resource	Object	Relational	Record	Multidimensional	XML	
Foundation	Business Information	Data Types	Expressions	Keys and Indexes	Software Deployment	Type Mapping
Object Model	Core	Behavioral		Relationships	Instance	

3.2 Organization of the CWM

The CWM is a set of extensions to the OMG metamodel architecture that customize it for the needs and purposes of the DW and business intelligence domains. The CWM has a modular, or package, architecture built on an OO foundation. As a consequence, it was organized in 21 separate packages which they were grouped into five stackable layers by means of similar roles, as seen in Table 2. One of the basic principles of CWM is that metamodels residing at one particular layer are dependent only on metamodels residing at a lower layer in order to avoid *package coupling* between the same level, or from a lower level to a higher level.

The reason for constructing the model this way was to maximize the use of the CWM. The CWM committee understood from the outset that no single tool would support all the concepts in CWM. In order to make the use of the CWM as easy as possible, the package structure was built with no horizontal coupling and as little vertical coupling as possible. In addition, no dependencies exist along any horizontal plane of the packages. This means that someone implementing a tool with the CWM would only need the vertical packages germane to his individual tool, i.e. the accompanying implementation of all others metamodel packages that it depends on, but no others.

Following these considerations, the CWM is a complete M2-level layered metamodel actually divided in a number of different but closely related metamodels. Within the block diagram describing the overall organization of the CWM presented in Table 2, the five layers shown are:

– **Object Model Layer**. This UML subset layer is used by the CWM as its base metamodel. Many Object Model classes and associations are intentionally corresponded to UML classes an associations, as UML heritage provides a widely used and accepted foundation for the CWM. Therefore, the Object

Model layer contains packages that define fundamental metamodel services required by the other CWM packages.

- **Foundation Layer**. This layer provides CWM-specific services to other packages residing at higher layers. The main difference with the previous layer is that the packages in this layer are not general-purpose and are specifically designed for the CWM. In this sense, the BusinessInformation package owns classes that provide access to business information services. The DataTypes package provides the infrastructure required to support the definition of both primitive and structured data types. As a complement to this package, the TypeMapping package allows the mapping of data types between type systems. Closely related to the mapping concept, there is the Expression package, where an expression is an ordered combination of values and operations that can be evaluated to produce a value, set of values, or effect. Because keys and indexes are used by several CWM packages, the KeysIndexes package has been included to provide classes supporting them. To conclude, the SoftwareDeployment package records how software and hardware in a DW are used.

- **Resource Layer**. CWM packages in the Resource layer describe the structure of data resources that act as either sources or targets of a CWM-mediated interchange. The layer contains metamodel packages that allow descriptions of OO databases and applications, relational database management systems, traditional record-oriented data sources such as files and record model database management systems, multidimensional databases created by OLAP tools, and XML streams or files.

- **Analysis Layer**. This layer supports warehouse activities not directly related to the description of data sources and targets. Rather, it describes services that operate on the data sources and targets described by the previous layer. The layer includes a Transformation package supporting extraction, transformation and loading (ETL), and data lineage services, an OLAP model for viewing warehouse data as cubes and dimensions, a data mining support metamodel, a foundation for storing visually displayed objects, and a terminology package supporting the definition of logical business concepts that cannot be directly defined by Resource layer packages.

- **Management Layer**. This layer provides service functions that can support the day-to-day operation and management of a DW by means of information flows, i.e. the WarehouseProcess package, and events, i.e. the WarehouseOperation package, in a DW. With respect to events, three types can be recorded: transformation executions, measurements and change requests. In addition, packages within this layer can serve as a foundation upon which more elaborate warehouse management activities can be built using CWM extension mechanisms, such as stereotypes, tagged values and inheritance.

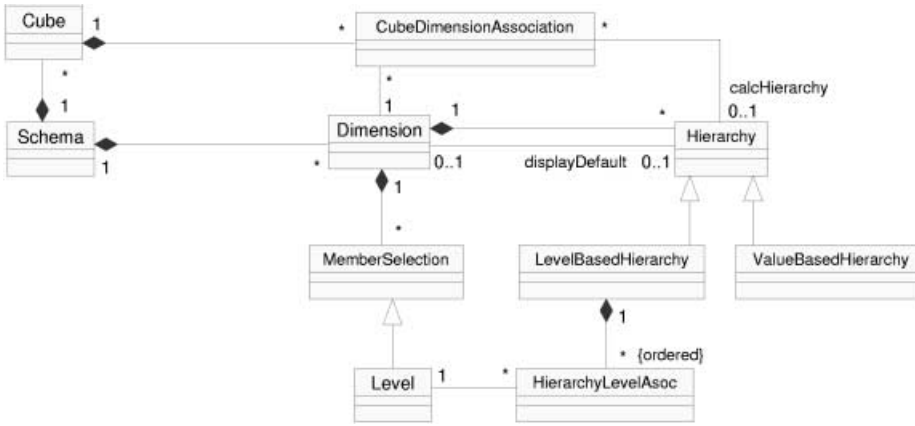

Fig. 2. The OLAP package metamodel

From this organization, we will mainly focus our work on the Analysis layer and, more precisely, on the OLAP package as a metamodel to describe conceptual MD models in terms of cubes and dimensions. Nevertheless, other packages will be discussed throughout this paper, as they will also be needed to represent the expressiveness of the MD model, e.g. the Transformation package.

4 Using the OLAP Package to Represent MD Properties

To the best of our knowledge, every main MD property can be mainly represented using the OLAP package metamodel, besides some specific features of other packages owned by the Analysis layer. The OLAP metamodel is structured into a Schema class that owns all elements of an OLAP model, i.e. Dimensions and Cubes. The UML class diagram corresponding to the OLAP package metamodel is shown in Fig. 2.

In the OLAP metamodel, each Dimension is a collection of Members representing ordinal positions along the Dimension. The Members are not part of the metamodel because they are treated as data itself. Dimensions are a type of Classifier that describe the attributes of their Members, which can be used to identify individual Members. The MemberSelection class supports limiting the portions of a Dimension that are currently viewed. Dimensions can also contain multiple and diverse hierarchical arrangements of Members including two specialized hierarchies that support ordering Members by hierarchy levels (LevelBasedHierarchy class) and by values (ValueBasedHierarchy), as can be seen from Fig. 2. In addition, Cubes are used to store Measures and they are related to the Dimensions through the CubeDimensionAssociation class. The OLAP metamodel uses the Core package to define attributes as Features within dimension levels and cubes as Classifiers.

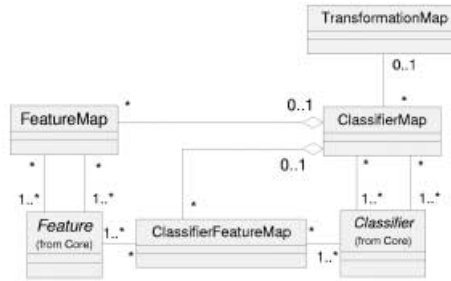

Fig. 3. The Transformation package metamodel

At the Analysis layer, a particular subset of the Transformation package, represented in Fig. 3, allows features to be transformed into other features by means of "white box" transformations. In this subset of transformations, a specific piece of a data source and specific piece of a data target are related to each other through a specific part of the transformation at a fine-grain level, i.e. feature level in our case. One such transformation is the *transformation map* which consists of a set of *classifier maps* that in turn consists of a set of *feature maps* or *classifierfeature maps*. We will also use this kind of transformations together with the OLAP package to represent the MD properties of the MD model.

From a higher level of abstraction, the main elements of the MD model are fact classes, dimension classes and levels (base classes), as introduced in 4.1, together with the attributes that define them. The following diagram in Fig. 4 illustrates the inherent semantic equivalence between classes of the MD model and the CWM. The semantic correspondence is illustrated by the associations mapping the equivalent metaclasses. Notice that these associations are neither a part of the CWM, nor the MD model; instead, they can be viewed as being "external" to both the CWM and the MD model. However, from the OMG metadata architecture point of view (Table 1), they are also at the M2-level.

It is also possible to generate instances (M1-level) of both the MD and the CWM models in which the equivalence associations still hold true. That is, the equivalence associations have their own corresponding instances, or projections, at the M1-level. Notice that, in Fig. 4, neither model is "generated" or "instantiated" from the other. Rather, the two models are equivalent representations of the same concepts.

The following class diagram in Fig. 5 illustrates a particular instantiation of the MD model. Therefore, this class diagram is a M1-level model. Whether we instantiate this M1-level model, we will obtain objects at the M0-level. For example, Sales Fact is a M1-level instance of the M2-level metaclass FactClass. Furthermore, Sales Fact "describes" many possible fact values, i.e. the content of MD cells, which are M0-level objects (data in the OMG hierarchy). We will use this M1-level example model to improve clearness and comprehension about how every MD property is represented using the CWM.

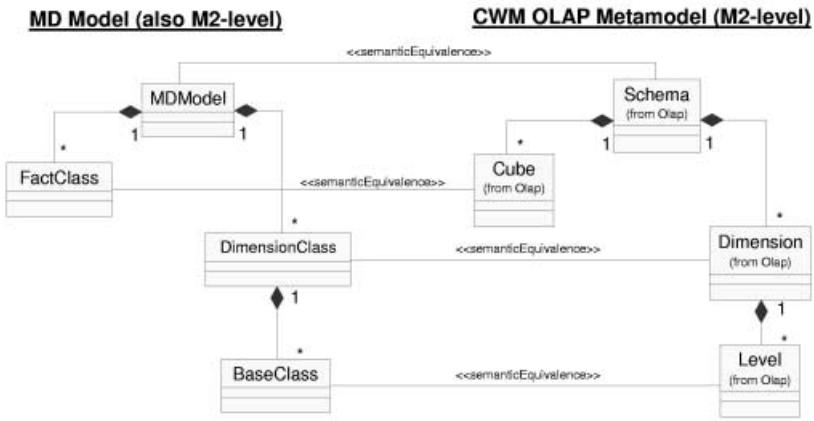

Fig. 4. Semantic equivalence between classes of the MD model and the CWM.

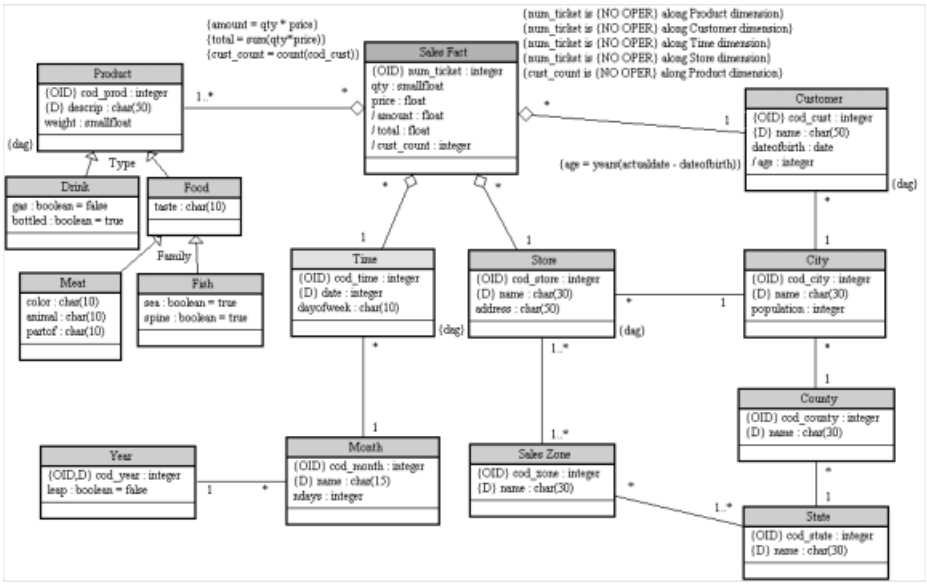

Fig. 5. Example of a M1-level instantiation of a M2-level MD model representing a sales system

More specifically, this example deals with sales of products in stores by means of tickets. Every ticket has information about who causes the sale (Customer), what item is sold (Product), and where and when is the sale produced (Store and Time, respectively). Furthermore, the ticket will store information about

what we need to measure, i.e. `quantity` and `price`. With respect to dimensions, this example MD model defines both classification hierarchies, i.e. dimensions Customer, Store, and Time, and categorization of dimensions, as can be seen in the dimension Product.

4.1 From the MD Model into the CWM

To correctly map the MD model into the CWM specification, we will describe the correspondence between the structural issues of the MD model and the OLAP metamodel. A summary of these issues is presented in Table 3.

Table 3. Summary of the main structural properties of the MD model

Multidimensional modeling properties
Facts
Calculated measures
Additivity
Degenerated dimensions
Many-to-many relationships with particular dimensions
Dimensions
Non-strictness
Derived attributes
Classification hierarchy paths and merging dimensions
Generalization

For each MD property presented in Table 3, we will discuss in depth how it can be expressed using the CWM specification. To help this objective, we provide the discussion with twofold purpose figures that illustrate: on the left hand side, the class diagram corresponding to the part of the CWM metamodel being used (M2-level), and on the right hand side, an instance diagram using M1-level objects from our example model (tickets).

From a lower level of abstraction, the main issues considered by the OO conceptual modeling approach are the following:

1. **Calculated measures and derived attributes.** Attributes may be calculated using a well-formed formula where other attributes may be involved. This property can be applied to any attribute in the MD model, i.e. both to fact (measures) and dimension attributes. Fig. 6 illustrates how they can be specified by using a "white-box" transformation.
   The FeatureMap metaclass on Fig. 6.a allows us to declare a well-formed formula by means of its attribute `function` of type `ProcedureExpression`. This formula will reference the source attributes, i.e. attributes that may appear as part of the formula, and the target attribute, i.e. the derived attribute. In addition, this formula can be expressed in OCL, thereby making use of the standard language for specifying constraints in UML. As an example, Fig. 6.b instantiates metaclasses on the left hand side in order to define

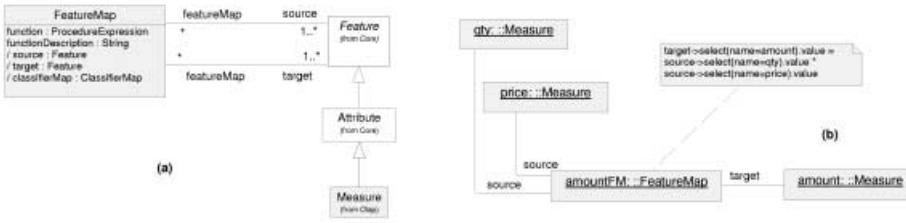

Fig. 6. Definition of derived attributes in the CWM

the calculated measure `amount` from our example model.

2. **Additivity and degenerated dimensions.** Although they are different concepts in the MD model, they share common characteristics that can be represented together in the CWM. Actually, a degenerated dimension is a fact attribute with no additivity along with a unique index represented by the constraint {OID} in the MD model.
   As previously commented, the Transformation package allows us to specify "white-box" transformations that consist of a set of *classifierfeature* maps. In this sense, Fig. 7.a represents how additivity rules can be described by means of a ClassifierFeatureMap metaclass in the CWM, as measures and dimensions involved in the additivity rule are a specialization of Features and Classifiers metaclasses in the CWM, respectively.

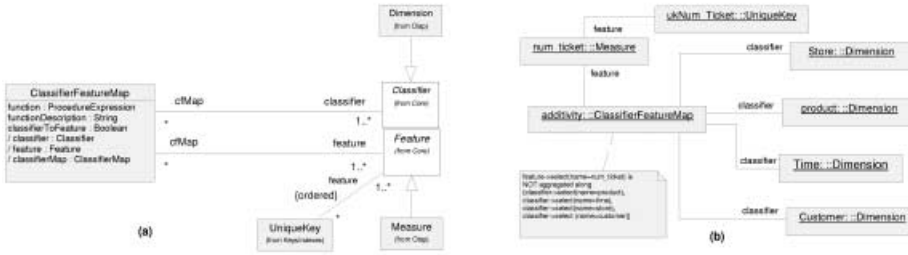

Fig. 7. Definition of additivity and degenerated dimensions in the CWM

Regarding the degenerated dimensions, Fig. 7.a also shows the use of the UniqueKey metaclass, from the KeysIndexes package, to identify a fact attribute. Fig. 7.b shows the definition of the additivity rules for the measure `num_ticket` from our example model. As this measure is actually a degenerated dimension, we use a classifierfeature map where the measure plays the role of feature and every dimension play the role of classifier in their association with the ClassifierFeatureMap metaclass of the CWM. Notice that the

constraint {OID} can also be expressed as an UniqueKey instance.

3. **Many-to-many relationships between facts and particular dimensions and non-strictness.** All these MD properties are considered together because they can be specified by means of multiplicity between associations. In fact, non-strict hierarchies are actually many-to-many relationships between levels of the hierarchy. Therefore, we will use the definition of association ends within an association relationship in the CWM, as seen in Fig. 8.a.

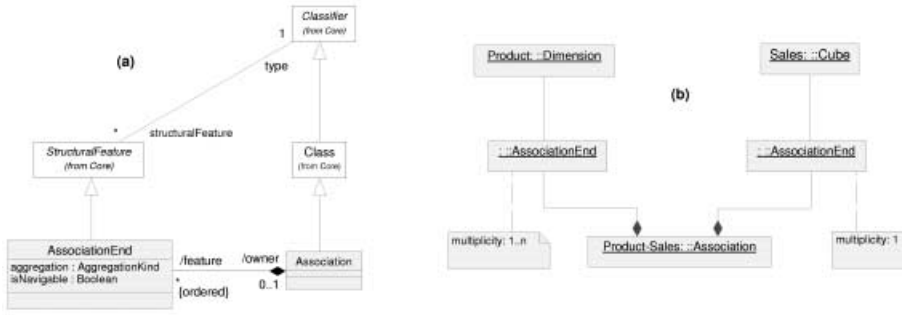

Fig. 8. Definition of many-to-many relationships in the CWM

As commented in Section 2.1, the relationship between facts and dimensions is a special kind of association called shared aggregation in the MD model. Therefore, it can be represented in Fig. 8.a as an association that owns two association ends with a specific multiplicity[2]. To clarify this concept, an instance diagram for the relationship between the fact Sales and the dimension Product in our example model is illustrated on Fig. 8.b. Notice that the cardinality of every association end is expressed by giving its respective value to the attribute `Multiplicity`.

4. **Classification hierarchy paths and merging dimensions.** A dimension may have one or more hierarchies to define both navigational and consolidation paths through the dimension. In the OLAP metamodel, the Hierarchy metaclass allows the specification of two kinds of multiple hierarchy by means of the subclasses LevelBasedHierarchy and ValueBasedHierarchy. The former describes relationships between levels in a dimension, while the latter defines a hierarchical ordering of members in which the concept of level has no significance. Therefore, we will use the LevelBasedHierarchy approach to represent hierarchical information within dimensions in the MD model.

[2] Being a specialization of the StructuralFeature metaclass from the Core package, AssociationEnd inherits the attribute Multiplicity to indicate cardinality.

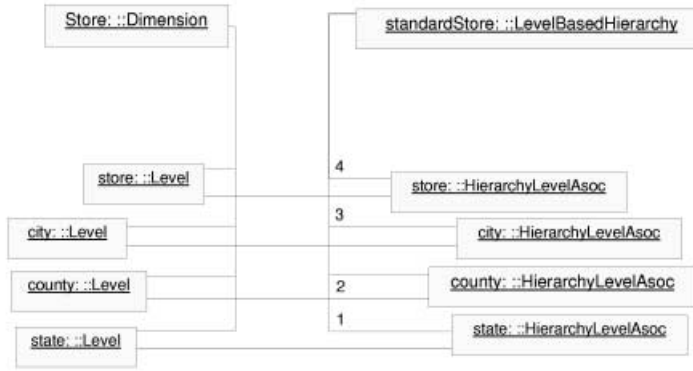

Fig. 9. Representation of the hierarchy path for the dimension Store from our example model

There is one relevant aspect that has to be considered when defining level-based hierarchy paths using the OLAP package metamodel. The association between LevelBasedHierarchy and HierarchyLevelAssociation is ordered. This ordering is accomplished from the higher to the lower level of the hierarchy. For example, to define the hierarchy path of the dimension Store from our example model, Fig. 9 indicates the correct order using numbers as labels somewhere around the corresponding HierarchyLevelAsoc metaclass.

However, levels in the CWM cannot be shared by different dimensions. A CWM Level is a subclass of the CWM MemberSelection, which is exclusively owned by a Dimension. Therefore, a Level is an exclusively owned attribute or property of a Dimension in the CWM, and cannot be shared in terms of ownership/composition. As a consequence, merging dimensions cannot be expressed by reusing Level definitions in the CWM. Instead, Levels could be, of course, be mapped between Dimensions using the transformation maps to formal model such correspondence.

5. **Generalization.** Being a special form of relationship between classes, generalization can be easily expressed using the Relationship package in the CWM. This package defines the generalization as a parent/child association between classes by means of the Generalization metaclass, as can be seen on Fig. 10.a.

   An instance diagram representing the generalization for the dimension Product of our example model is also shown on Fig. 10.b. As the Generalization metaclass is a specialized metaclass, the name of each generalization can be expressed by giving a value to the attribute Name inherited from the ModelElement metaclass from the Core package metamodel.

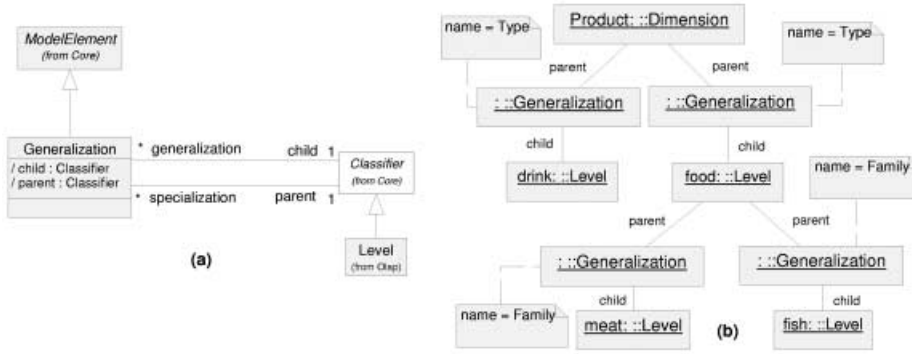

Fig. 10. Definition of generalization in the CWM

5 Conclusions and Future Work

The heterogeneity between the MD models used by the different tools leads to the existence of dissimilar metadata. As a consequence, there is the necessity for a standard metadata that allows tools to interchange their information based on the MD model. In this sense, the CWM is becoming a standard "de-facto" for representing metadata in data warehousing and business analysis.

In this paper we have discussed how to represent the main MD properties at the conceptual level by means of a semantic equivalence between the classes of the MD model and the CWM metamodel. We have presented how every structural MD property has to be mapped to conform the CWM specification. As a result, we obtain instances of MD models expressed as CWM metadata; the main advantage is that any tool could benefit from the expressiveness of the MD model through the interchange of CWM-based metadata.

Our future work will be the representation of the dynamic part of the MD model in the CWM. In this sense, we will discuss the mappings needed to represent cube classes to specify initial user requirements. We will also accomplish the implementation of a programmatic interface within a CASE tool, thereby allowing us to share and interchange MD models with any CWM-compliant tool in the market, e.g. Oracle Warehouse Builder, IBM DB2 Warehouse Manager, Hyperion Essbase, etc.

Acknowledgements

We would like to thank the CWM committee, especially David Mellor and John Poole, for their useful ideas and support in the writing of this paper.

References

1. A. Abelló, J. Samos, and F. Saltor. Benefits of an Object-Oriented Multidimensional Data Model. In K. Dittrich, G. Guerrini, I. Merlo, M. Oliva, and E. Rodriguez, editors, *Proceedings Symposium on Objects and Databases in 14th ECOOP Conference*, pages 141–152. Springer LNCS 1944, 2000.
2. P. A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders, and D. Shutt. Microsoft Repository Version 2 and the Open Information Model. *Information Systems, 24, 2*, 1999.
3. W. Giovinazzo. *Object-Oriented Data Warehouse Design. Building a star schema.* Prentice-Hall, NJ, 2000.
4. R. Kimball. *The data warehousing toolkit.* John Wiley, 2 edition, 1996.
5. Meta Data Coalition. Open Information Model Version, 1.0. Internet: `http://www.MDCinfo.com`, August 1999.
6. Meta Data Europe 99. Implementing, Managing and Integration Meta Data. Internet: `http://www.ttiuk.co.uk`, March 1999.
7. Object Management Group (OMG). Common Warehouse Metamodel (CWM). Internet: `http://www.omg.org/cgi-bin/doc?ad/2001-02-01`, 2000.
8. Object Management Group (OMG). Unified Modeling Language (UML). Internet: `http://www.omg.org/cgi-bin/doc?formal/01-09-67`, January 2001.
9. J. Poole, D. Chang, D. Tolbert, and D. Mellor. *CWM: An Introduction to the Standard for Data Warehouse Integration.* John Wiley, 2002.
10. J. Trujillo, J. Gómez, and M. Palomar. Modeling the Behavior of OLAP Applications Using an UML Compilant Approach. In *Proceedings 1st ADVIS Conference*, pages 14–23. Springer LNCS 1909, 2000.
11. J. Trujillo, M. Palomar, J. Gómez, and Il-Yeol Song. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer, special issue on Data Warehouses, 34, 12, 66-75*, 2001.
12. T. Vetterli, A. Vaduva, and M. Staudt. Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metamodel. *ACM SIGMOD Record, 23, 3*, 2000.