UML for data warehouse dimensional modeling

Y. Mai[1], J. Li[1] & H. L. Viktor[2]
[1] Concordia University, Canada
[2] University of Ottawa, Canada

Abstract

Dimensional modeling is a common modeling technique in data warehousing. It reflects a simple logical view of a data warehouse system. It can be easily mapped to a physical design. Traditional dimensional modeling is data-oriented and semantically informal. From a software engineering perspective, the informal notations and *data-oriented* feature are insufficient to tackle the complexity of large data warehouse projects. UML, with its well-defined semantics, is now a standard modeling language that is used to model the entire life cycle of a software system. UML has rich and extensible semantics. The combination of the knowledge in standard object-oriented modeling and dimensional modeling add variable semantics to dimensional modeling without losing its understandability. This paper proposes a metamodel for data warehouse dimensional modeling using UML. Based on this metamodel, we illustrate how to model the business process and data marts of a large mobile telephone company.
Keywords: data warehouse, dimensional modeling, UML, metamodel.

1 Introduction and related work

In recent years, data warehouses have gained increasing popularity and are becoming a business growth strategy. A data warehouse is essentially a data container, which contains complete and historical business data from numerous operational sources. The data, as contained in a data warehouse, are used to analyze business, help predicting the organizational growth and improve customer relationships. In essence, a data warehouse is a queryable data source that exists to answer questions people have about the organization. These queries thus reflect the way that managers think about their organization and assist them to make sense of the data, form policies and to make informed decisions.

Management Information Systems, C. A. Brebbia (Editor)

From the technical perspective, data warehouse modeling is critically important in a data warehouse system design. One key factor in ensuring the success of a data warehouse is to create the right model that reflects the business needs and covers all business requirements. Dimensional modeling [1, 2], popularly known as the Star Schema approach, is a widely used technique for modeling the logical aspects of a data warehouse system according to business views. The main idea is to logically design a data warehouse as a set of incrementally designed data marts, each representing a view of a business process. A single dimensional model consists of fact tables and dimension tables. The data warehouse bus architecture [1, 2] then glues all data marts into a logical data warehouse. This is accomplished through the use of conformed facts and conformed dimensions, which ensure that the grain of the various data marts are compatible.

Recently, there are some attempts to model a data warehouse with Unified Modeling Language (UML) [3, 4, 5, 6, 7, 8]. UML is an OMG standard modeling language that has been widely used in object-oriented system modeling. It defines a common vocabulary for communications among designers and users. There also have been many discussions on modeling database systems using UML [9, 10, 11], focusing more specific on databases (especially for relational databases) rather than data warehouses. OMG defines a Common Warehouse Metamodel [12] that can be used to guide common warehouse modeling. It contains limited metamodel definitions for data warehouse design. But it does not contain a metamodel for multidimensional modeling. UML notations are rich in semantics but they may be complex for database modelers. Traditional data warehouse modeling techniques are straightforward for database teams and reflect the ways that people think about a database system. But since the traditional notations have little semantics and are only data-oriented, they are insufficient to tackle a complex data warehouse modeling. Thus, in this paper, we provide a combination of both, that is, using certain UML notations to represent dimensional modeling. The paper differs from previous work in 3 major areas:

- We propose a UML metamodel for data warehouse dimensional modeling. Although there are many object-oriented dimensional data models, we haven't seen any UML metamodels for data warehouse dimensional modeling so far.
- We believe UML is useful in modeling the entire software lift cycle of the data warehouse. One section in the paper shows the business process modeling.
- We define UML notations to model data marts, data warehouse bus matrix, table grains, foreign keys, and table relationships.

The paper is organized as follows. Section 2 defines a metamodel for UML dimensional modeling. This metamodel defines semantics for the basic data warehouse components including fact tables, dimension tables and their attributes. Based on this metamodel, Section 3 illustrates how to model a data warehouse involving a large mobile telephone company using UML. Section 4 concludes the paper and highlights some future research directions.

Management Information Systems, C. A. Brebbia (Editor)

2 A UML metamodel for dimensional modeling

A metamodel is a model that describes the syntax and semantics of a modeling [13]. It contains descriptions that define the structure and semantics of a model. Metamodels [12, 13, 14, 15] have been widely used to define models. The metamodel in this paper is defined under the UML definitions and is derived from the concepts of adaptive object model [14, 15]. We follow the structure and semantics of data warehouse dimensional modeling as described in [1, 2].

Figure 1 shows the data warehouse core metamodel. `Table` defines the basic construct in the data warehouse. The `Table` defined in the meta-level contains `Attribute` that describes the properties of the `Table`. The metamodel for `Table` and `Attribute` is defined in the meta-level layer. Any data warehouse table is an instance of `Table`. For example, tables such as Product, Customer, and Date are all instances of `Table`. A `Table` instance consists of a set of concrete properties such as name and location. Each property has a type, such as the `ConcreteAttribute` shown in the figure. The `ConcreteAttribute` is an instance of `Attribute`. The relationship between a concrete table and its attributes is shown in the base-level layer.

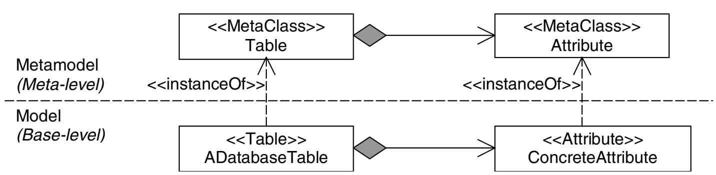

Figure 1: Data warehouse core metamodel.

`Fact` tables and `Dimension` tables are two basic kinds of tables in the data warehouse. `Aggregate Fact` tables or `Factless Fact` tables are two special cases of `Fact` table [2, 3, 16]. They differ from a normal fact table in that an aggregate table contains derived facts (or measures) for performance improvement purpose, while a factless fact table is a kind of fact table without facts. The relationships among these tables can be organized into a hierarchy tree as shown in Figure 2. Both `Fact` table and `Dimension` table inherit from the `Table` metaclass. `Aggregate Fact` table and `Factless Fact` table are subclasses of `Fact` table. UML stereotypes can be used to describe table types. We use ≪Fact≫, ≪Dimension≫, ≪Aggregate Fact≫, and ≪Factless≫ to represent tables Fact, Dimension, Aggregate Fact, and Factless Fact, respectively.

Each table consists of attributes. An attribute has three members: a string representation name, a data type, and a boolean indicating whether this attribute represents a grain or not. More details about the data type can be found in the UML metamodel [17]. A grain is the level of granularity (detail) of all properties in a table. It is crucial that every row in a fact table be recorded at exactly the same

Management Information Systems, C. A. Brebbia (Editor)

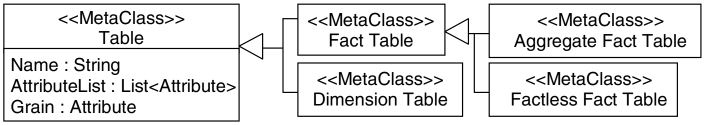

Figure 2: Metamodel hierarchy of the data warehouse tables.

level of detail at which measures will be recorded. Figure 3 shows the meta-level description of the attributes in the dimensional data warehouse.

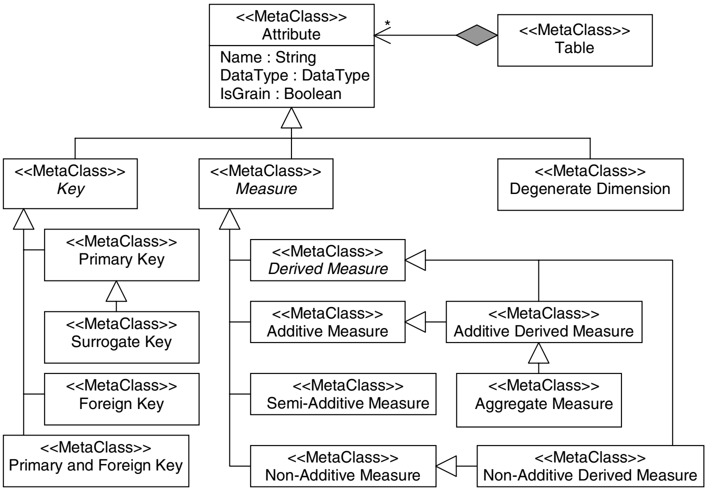

Figure 3: UML meta-level description of data warehouse attributes.

In addition to normal attributes in a database table, there are three special kinds of attributes: (1) An attribute can be a key: primary key, surrogate key, foreign key, or a combination of primary and foreign key. The left-hand side inheritance tree in Figure 3 shows the attribute key metaclasses. (2) An attribute in a fact table can be a measure: additive, semi-additive, non-additive, both derived and additive, aggregate, or both derived and non-additive. (3) In a fact table, an attribute can be a degenerated dimension. A degenerated dimension is a special attribute that is lack of properties and can be combined with a fact table. Figure 3 shows the entire attribute hierarchy tree. We use stereotypes (enclosed in “≪≫”) to represent items (1) and (3); Constraints (enclosed in “{}”) to represent item (2) except derived measure which is adorned with “/”.

This section presented the UML metamodel for data warehouse dimensional modeling. The metamodel describes the object-oriented model that we propose for dimensional modeling. In the next section, we will illustrate how to model the data warehouse based on this metamodel.

Management Information Systems, C. A. Brebbia (Editor)

3 Data warehouse modeling with UML

Data warehouse is designed to answer questions being asked throughout the business on an every day basis. These questions do not focus on individual transactions but on the overall process and are mainly used to determine trends and bottlenecks in the organization. To answer these questions, the design of the data warehouse should directly reflect the way that managers perceive their business [16]. That is, it should capture the measurements of importance to the business, and the parameters by which these measurements are viewed. We will first discuss the business process modeling, and then the data mart modeling.

Throughout this section, we will use a mobile telephone company data warehouse as a running example. The purpose of this data warehouse is to provide a repository of data regarding the customers and the telecommunications network, including diverse aspects such as call duration, peak call times, sales of contracts, sales of mobile phones and customer profiles, amongst others.

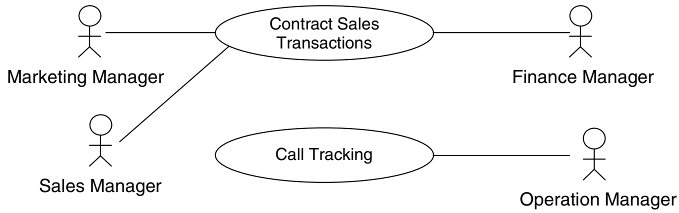

Figure 4: Modeling business process functionality.

3.1 Business process modeling

Business process modeling contains 3 steps:

- To capture the stakeholders. In UML, stakeholders are modeled as actors, i.e. roles that are outside the system but that closely interact with the system [18]. Actors are not part of the system. In a data warehouse system, they can be organization executives, different levels of managers, data warehouse administrators, etc. They can also be data sources including operational or transactional systems.
- To capture business requirements. A business process can be represented as a use case in UML. Each business process can have a number of actors interacting with it. The use case (business process) provides necessary functionality for the actors to fulfill some specific tasks. All use cases put together comprise the entire functionality that the data warehouse provides. We use UML Use Case diagrams to model the business processes (and requirements) in a data warehouse system. Figure 4 shows an example. On the other hand, a business process can be *realized* by a data mart, a collection of data

Management Information Systems, C. A. Brebbia (Editor)

used by this single key business process. we model data marts with UML Collaborations and apply Realization relationship to modeling the relationship between the data mart and the business process. Figure 5 shows that `Customer Call Activity` data mart, which is used to model the business process use case `Customer Call Activity`.

- To model data warehouse bus matrix. Table 1 shows a simplified data warehouse bus matrix for the mobile phone company, created for the use case diagram shown in Figure 4. The first column in the table represents the data marts, and the others represent dimension tables. A cross represents that a dimension participates in a data mart. A corresponding UML representation is showed in Figure 6.

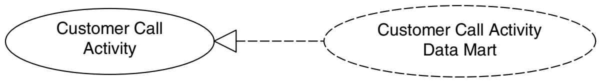

Figure 5: Modeling relationship between the use case and the data mart.

Table 1: Data warehouse bus matrix.

	Date	Customer	Product	Sales Rep	Store	Promotion	Transaction
Contract Sales	X	X	X	X	X	X	X
Call tracking	X	X	X				
... ...							

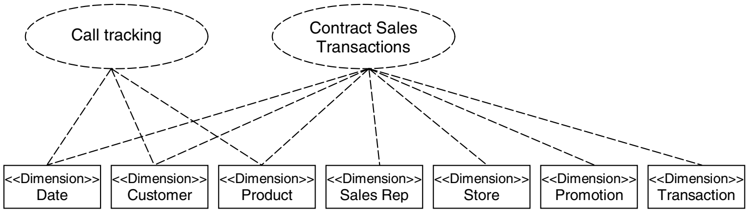

Figure 6: Modeling data warehouse bus matrix.

3.2 Modeling the data mart

A data mart reflects a process view of the data warehouse. It implements a data warehouse use case. A data mart contains a set of tables, for example, a fact table,

Management Information Systems, C. A. Brebbia (Editor)

aggregate tables (if any), and dimension tables, that are organized together into a star schema to represent a specific business process. Since the modeling of the fact and dimension tables in the data mart are straightforward based on the metamodel definition, we only show an example of basic modeling of the data mart and then discuss a few special cases in the data mart modeling.

Figure 7 shows an example of modeling of the `Contract Sales` data mart. The fact table is modeled as a composite class in the diagram. It has shared-aggregation relationships with the corresponding dimension classes. The role of a dimension class represents a reference from the fact class to a dimension class. The role has a name, which is the name of the surrogate key of the dimension class (or the foreign key name in the fact table for that dimension table). Fact class `Sales` links to `Time` dimension class which plays the role of "*time_key*" in the fact table. The "*time_key*" is also the surrogate key name in the `Time` dimension. The grains of the dimension and fact tables are shown in separate compartments of the class icons. The cardinality for the relationships between the fact class and the dimension classes are marked besides the table links. "1" represents exact one. "*" represents zero or more.

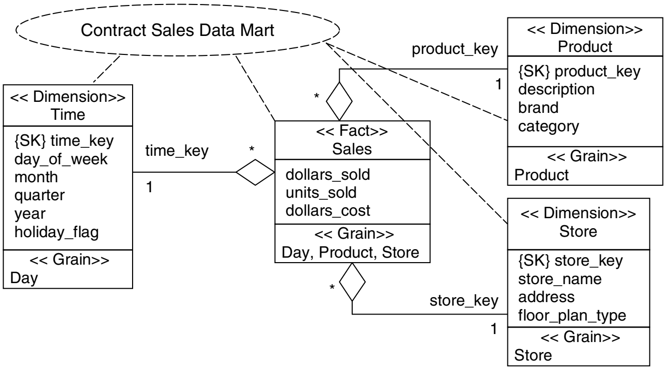

Figure 7: Modeling contract sales data mart.

We cover 3 special cases of data mart modeling:

- Model multiple foreign key relationships. We represent each foreign key as a shared-aggregation relationship. Figure 8 shows the shipment fact class has two foreign keys, *shipment_date_key* and *order_date_key*, each representing as a shared-aggregation relationship linking to the `Date` dimension.
- Model many-to-many relationship between tables. Figure 9 shows an example of using a bridge table `Account-Customer` (adapted from [2]). This table contains a composite primary key with one *account_key* refers to the account table and the other *customer_key* refers to the customer table. In such way, the `Monthly Account Balance` table refers to the `Account-Customer` bridge table through the *account_key* and the *customer_key* in

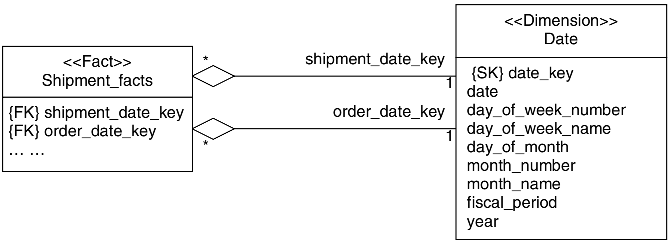

Figure 8: Modeling multiple foreign key relationships.

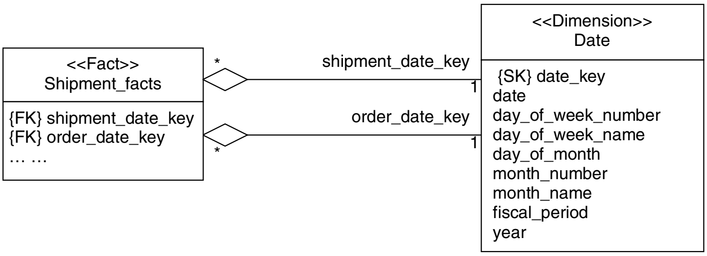

Figure 9: Modeling many-to-many relationship.

the bridge table refers to one customer in the `Customer` table. Since *account_key* in the bridge table does not uniquely identify a record (it is part of the primary key), each record in the fact table can refer more than one customer in the `Customer` dimension.

- Model aggregate fact table. A common convention in data warehouse modeling is to pre-store aggregate data in the data warehouse (for performance improvement). Figure 10 gives an example of using the aggregate fact table. First, a new dimension called `Category` is created as a shrunken table of dimension `Product`. It only contains the product key (surrogate key), category, and department. The surrogate key here is only a subset of the product keys in dimension `Product` with each representing a category product. We subsequently create an aggregate class called `Sales_by_Category`. In this class, each row contains data such as time, store, product category, as well as some measures such as dollars sold, quantity sold, and dollars cost along the time, store and product category dimensions. The directed dashed lines represent dependency relationships. This is due to the fact that `Category` is a subset of `Product`, and `Sales_by_Category` is calculated from Sales.

4 Conclusion

Modeling a data warehouse is a complex task due to a number of complex factors, including the huge size thereof, the constraints imposed by operational data

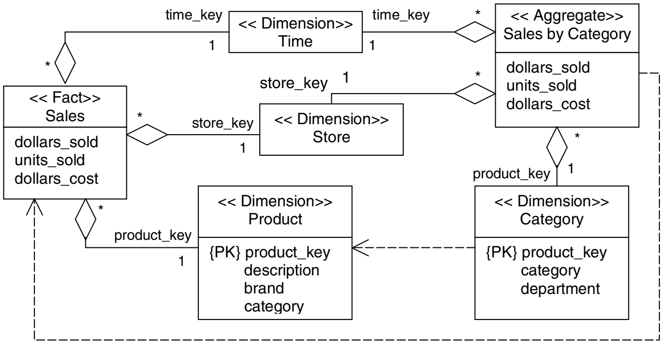

Figure 10: Using the aggregate table.

sources and the complex business requirements as obtained from numerous users, amongst others. Dimensional modeling separates business concerns and addresses one business need at a time. The purpose of dimensional modeling is to bridge the communication gap between domain users and data warehouse developers. However, a traditional dimensional model is a data-driven model. It contains little semantics for notations and it is therefore difficult to embed the inherent meaning of the data and data relationships therein. Data warehouse developers therefore have difficulty in understanding the subtle meanings of conformed dimensions and facts, which may lead to poorly designed data warehouse systems.

This paper proposes an approach to represent the dimensional model in a semantically rich manner. As a standard modeling language, UML provides rich and extensible semantics to a model. With the UML as the foundation, this paper proposes a metamodel for dimensional modeling, in which we define a vocabulary to model basic data warehouse concepts using UML notations. The metamodel extends the UML metamodel. It defines the object-oriented dimensional model that we propose. Based on this metamodel, we model the business processes and requirements, and the dimensional model with data warehouse bus matrix and data marts. The modeling diagrams are easy to understand without losing explicit semantics.

This paper shows how dimensional modeling can be extended using object-oriented modeling techniques, thus providing a foundation for modeling the logical core of a data warehouse system. Future work will focus on extending the model to capture the entire business life cycle of a data warehouse, including its behavioral modeling and physical modeling. In addition, the extension of UML diagrams for modeling misuse of the data warehouse will be further investigated. That is, the investigation of approaches to design our system is such a way to anticipate and model the illegal use of the data warehouse is an exciting new research direction, especially within the domain of so-called data webhouses.

Management Information Systems, C. A. Brebbia (Editor)

References

[1] Kimball, R., The Data Warehouse Toolkit. John Wiley & Sons Inc, 1998.
[2] Kimball et al., R., The Data Warehouse Lifecycle Toolkit. John Wiley & Sons Inc, 1998.
[3] Trujillo et al., J., Designing Data Warehouses with OO Conceptual Models. IEEE Computer, V 34, No 12, pp. 66–75, 2001.
[4] Abello et al., A., Understanding Analysis Dimensions in a Multidimensional Object-oriented Model. Proceedings of the international workshop on Design and Management of Data warehousing (DMDW' 2001), Interlaken, Switzerland, pp. 16–19, 2001.
[5] Abello et al., A., Understanding Facts in a Multidimensional Object-oriented Model. Proceedings of the fourth ACM international workshop on Data warehousing and OLAP, 2001.
[6] Bruckner et al., R.M., Developing Requirements for Data Warehouse Systems with Use Cases. The Seventh Americas Conference on Information Systems, 2001.
[7] Lujan-Mora et al., S., Extending the UML for Multidimensional Modeling. Proceedings of UML 2002 - The Unified Modeling Language Model Engineering, Languages, Concepts, and Tools 5th International Conference, Dresden, Germany, 2002.
[8] Lujan-Mora et al., S., Multidimensional Modeling with UML Package Diagrams. The 21st International Conference on Conceptual Modeling (ER 2002), Tampere, Finland, 2002.
[9] Ambler, S.W., A UML Profile for Data Modeling. wwwagiledataorg, 2002.
[10] Blaha, M. & Premeriani, W., Using UML to Design Database Applications. http://wwwumlchinacom/Indepth/usingumlhtm, 2002.
[11] Gornik, D., UML Data Modeling Profile. Rational Software White Paper (TP162), 2002.
[12] OMG, Common Warehouse Metamodel. Object Management Group (OMG), http://wwwomgorg/cwm/, 2000.
[13] OMG, MetaObject Facility (MOF) Specification. Object Management Group (OMG), http://cgiomgorg/docs/formal/00-04-03pdf, 2000.
[14] Johnson, R., Dynamic Object Model. Object-Oriented Programming Systems, Languages & Applications (OOPSLA' 1997), Atlanta, Georgia, 1997.
[15] Johnson, R. & Woolf, B., Type Object, Pattern Languages of Program Design 3. Addison Wesley, 1998.
[16] Adamson, C. & Venerable, M., Data Warehouse Design Solutions. John Wiley & Sons Inc, 1998.
[17] OMG, Unified Modeling Language (UML) Specification (version 1.4). Object Management Group (OMG), 2001.
[18] Booch, G., Jacobson, I. & Rumbaugh, J., The Unified Modeling Language User Guide. Addison-Wesley, 1999.

Management Information Systems, C. A. Brebbia (Editor)