I Taller de Almacenes de Datos y Tecnología OLAP (ADTO'2001), VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2001), p. 12-17, Almagro (Spain), November 22 2001.

Extensión de UML para el modelado multidimensional

Juan Trujillo, Sergio Luján, and Enrique Medina

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

Resumen Los almacenes de datos (AD), las bases de datos multidimensionales (BDM) y las aplicaciones de Procesamiento Analítico en Línea (*On-Line Analytical Processing*, OLAP) están basadas en el modelado multidimensional (MD). En este artículo presentamos cómo el Lenguaje de Modelado Unificado (*Unified Modeling Language*, UML) se ha extendido para llevar a cabo el diseño conceptual de un modelo MD. La estructura del modelo se especifica mediante un diagrama de clases UML que considera las principales propiedades del modelado multidimensional por medio de un número reducido de restricciones y extensiones sobre UML. Además, proponemos una notación gráfica de clases (clases cubo) compatible con UML para representar los requisitos OLAP iniciales de usuario. El comportamiento del sistema se modela a través de los diagramas de estados e interacciones. Finalmente, presentamos brevemente las investigaciones que se están llevando a cabo actualmente a partir de esta propuesta.

Palabras clave: Almacenes de datos, bases de datos multidimensionales, OLAP, modelado conceptual, UML, orientación a objetos

1 Introducción

Se acepta ampliamente que los AD, BDM y aplicaciones OLAP están basados en el modelado multidimensional (MD). El beneficio de utilizar este modelado MD es doble. Por un lado, el modelo MD está cercano a la forma de pensar de los analistas y, por tanto, ayuda a los usuarios a entender los datos. Por otro, a partir del modelo MD se puede predecir las acciones que llevarán a cabo los usuarios finales y, permitir de esta forma aumentar el rendimiento del sistema.

Últimamente se han propuesto algunas aproximaciones para llevar a cabo el diseño conceptual de estos sistemas[1]. Estas propuestas intentan representar las principales propiedades MD a nivel conceptual, aunque hasta el momento ninguna de ellas ha sido aceptada como un estándar.

En este artículo presentamos un resumen de nuestra propuesta Orientada a Objetos (OO) para llevar a cabo el modelado conceptual de los AD, BDM y aplicaciones OLAP. Dicha propuesta introduce un número reducido de extensiones y restricciones sobre UML [2] necesarias para una adecuada representación de las propiedades del modelado MD. Estas extensiones se basan en mecanismos estándares que proporciona el propio lenguaje UML para adaptarlo a un método o lenguaje específico (ej. restricciones y propiedades[2]). Las propiedades estructurales del modelado multidimensional (hechos, dimensiones, aditividad, etc.) se especifican mediante un diagrama de clases UML [7] (sección 2). En cuanto a las propiedades dinámicas y de comportamiento [6], proponemos una notación gráfica de clase (clases cubo) compatible con UML para representar los requisitos OLAP iniciales de usuario y un conjunto de operaciones OLAP. A partir de estas clases cubo utilizamos los diagramas de estados e interacciones para modelar el comportamiento del sistema en función de las operaciones OLAP que se apliquen (sección 3). La información representada en estos diagramas se puede aprovechar para cuestiones relativas al mantenimiento de vistas (sección 4).

2 Propiedades estructurales del modelado multidimensional

El modelado MD estructura la información principalmente en *hechos* y *dimensiones*. Supongamos para todo el artículo como ejemplo un sistema de ventas de productos en el que los hechos son los tickets

[1] En [1] se presentan las aproximaciones MD más significativas propuestas hasta el momento.
[2] *constraints* y *tagged values*.

emitidos en las cajas registradoras de una cadena de grandes almacenes. Se desea analizar estas ventas de productos a lo largo de las siguientes dimensiones: producto vendido, cliente que realizó la compra, almacén donde se emitió y, finalmente, la fecha en que fue emitido.

En nuestra aproximación UML, los hechos se especifican como clases compuestas en una relación de agregación de n clases de dimensión. La cardinalidad mínima en el "role" de las clases de dimensión es *1* para indicar que todo hecho ha de estar siempre relacionado con todas las dimensiones. Las relaciones *"muchos a muchos"* entre un hecho y una dimensión en particular se especifican mediante la cardinalidad *1..** en el role de la clase de dimensión correspondiente.

En la Figura 1 se presenta un fotograma de la herramienta CASE que implementa nuestra propuesta. En dicho fotograma se incluye el diagrama de clases UML que representa el sistema Ventas de productos. Se puede observar en la Figura 1 la clase de hechos Ventas de productos y las clases de dimensión Producto, Almacén, Cliente y Tiempo. Se ha definido una relación *"muchos a muchos"* entre la clase de hechos y la clase de dimensión producto ya que un ticket puede contener más de un producto.

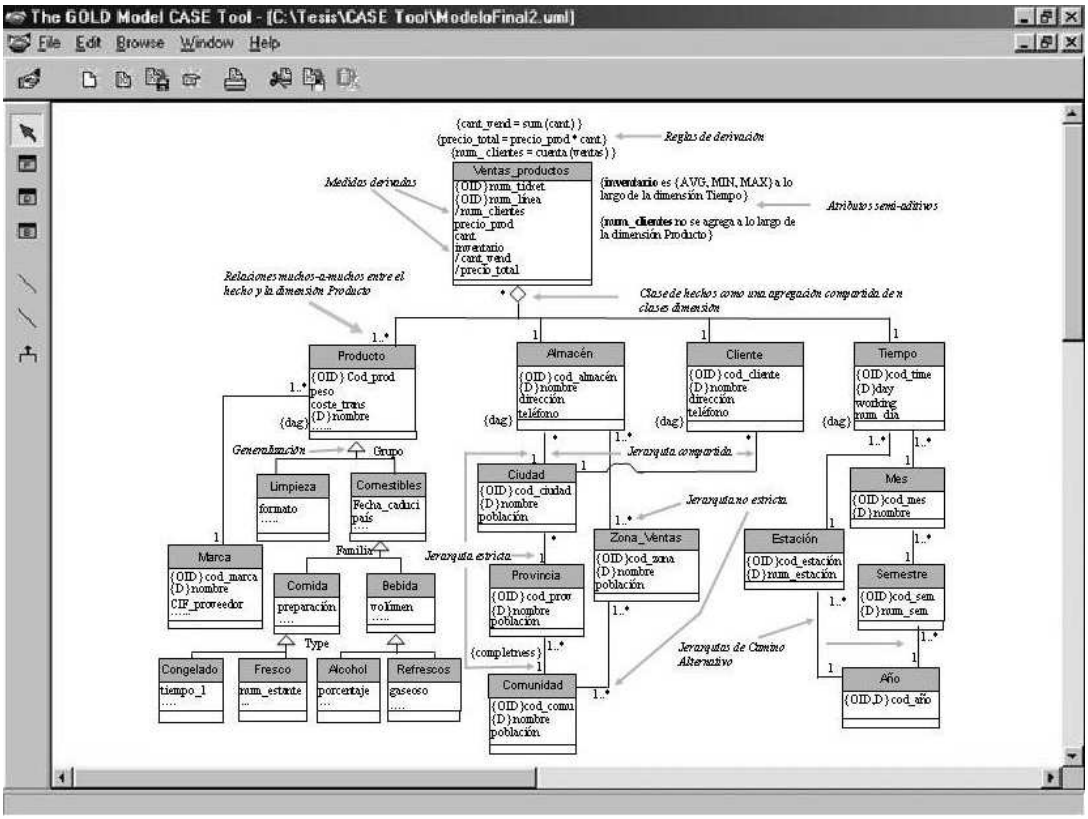

Figura 1. *Diagrama de clases UML para representar el sistema Ventas de productos*

Las medidas o atributos en los que deseamos centrar nuestro análisis se definen como atributos de la clase de hechos (ej. núm_clientes, cant, etc.) Además de las medidas atómicas, también podemos representar medidas derivadas antecediendo la restricción "/" al nombre de la medida. Las reglas de derivación se sitúan entre llaves alrededor de la clase de hechos (ver Figura 1).

En cuanto a la aditividad [3] de las medidas, consideramos por defecto que todas las medidas son aditivas. Las medidas semi-aditivas y no aditivas se especifican mediante restricciones entre llaves en una sintaxis inglesa no formal. Dichas restricciones especifican las operaciones de agregación permitidas sobre las dimensiones que no son aditivas. Por ejemplo, en la Figura 1 podemos ver que el atributo num_clientes no se puede agregar utilizando ningún operador de agregación a lo largo de la dimensión producto.

Por otro lado, también podemos definir atributos identificadores (OID) en la clase de hechos mediante la restricción $\{OID\}$. Estos atributos se emplean para identificar las instancias de hechos de forma

unívoca si existe alguna relación *"muchos a muchos"* con alguna clase de dimensión en particular y la posterior implementación del modelo se va a llevar a cabo en un sistema relacional [4]. En nuestro ejemplo hemos definido los atributos identificadores num_ticket y num_linea para reflejar la línea concreta que un producto ocupa en un determinado ticket.

Con respecto a las dimensiones, cada nivel de una jerarquía de clasificación se representa mediante una clase. Estas clases deben contener un atributo identificador ($\{OID\}$) para poder identificar unívocamente las instancias de un nivel de jerarquía y un atributo descriptor ($\{D\}$) que será el utilizado por defecto en la fase de análisis en la herramienta OLAP final donde se implemente el modelo diseñado. Estos dos atributos son necesarios en un proceso de generación semi-automática de la implementación del modelo en una herramienta OLAP concreta [5].

Estas clases que representan jerarquías de clasificación forman un Grafo Acíclico Dirigido (GAD) (restricción $\{dag\}$ situada al lado de cada clase de dimensión) a partir de cada clase de dimensión. Los GAD nos permiten considerar de una forma sencilla jerarquías de clasificación múltiple y de camino alternativo así como compartir caminos de jerarquía por varias dimensiones (ej. dimensión almacén en la Figura 1).

Gracias a la flexibilidad de UML, podemos considerar las particularidades de las jerarquías de clasificación como las *jerarquías no estrictas* (un objeto de un nivel inferior pertenece a más de uno de un nivel superior) y las *jerarquías completas* (todos los miembros pertenecen a un único objeto de una clase superior y ese objeto está compuesto por esos objetos exclusivamente). Estas características se especifican mediante la cardinalidad en los "roles" de las asociaciones y la restricción $\{completeness\}$ respectivamente, como se puede ver en la dimensión almacén de la Figura 1. Finalmente, la categorización de dimensiones se considera mediante las jerarquías de generalización/especialización, como se puede observar en la dimensión producto de la Figura 1.

3 Propiedades dinámicas del modelado multidimensional

Proponemos una nueva notación gráfica de clase compatible con UML denominada clase cubo para especificar los requisitos iniciales de los usuarios. Las secciones de estas clases cubo son:

- Encabezado (*Head*) (H): nombre de la clase cubo.
- Medidas (*Measures*) (M): medidas que se desean analizar.
- Slice (S): restricciones que han de satisfacer los datos analizados.
- Dice (D_C): dimensiones y niveles de jerarquía sobre los que se desean analizar las medidas.
- Operaciones Cubo (*Cube Operations*, CO): operaciones OLAP permitidas para el posterior análisis de los datos

Supongamos como ejemplo el siguiente requisito inicial de usuario: "Cantidad vendida de productos donde el grupo de productos sea "Alimentos" y la comunidad donde se hayan vendido sea "Valencia" y que estén agrupados por la familia y marca de los productos, y las provincias y ciudades donde fueron vendidos."

En la Figura 2 podemos observar la notación gráfica de la clase cubo que se corresponde con el requisito anterior donde se pueden identificar fácilmente las diferentes secciones. La sección *Measures* contiene la medida SUM(cant_vendida) objeto de análisis, Slice las restricciones definidas sobre las dimensiones almacén y producto y, en la sección Dice las dimensiones y los niveles de jerarquía de las dimensiones almacén y producto a lo largo de los que se desean analizar los datos.

A partir de las clases cubo iniciales, los usuarios pueden aplicar ciertas operaciones OLAP[3] para analizar la posible evolución de estos requisitos. Esta evolución se modela mediante los diagramas de estados e interacciones de UML en función del tipo de operación OLAP aplicada. Así, para cada clase cubo inicial definimos un diagrama de estados en el que cada estado representa una clase cubo que se puede alcanzar aplicando las operaciones OLAP que permiten a los usuarios navegar por los distintos niveles de jerarquía (*roll-up* y *drill down*) o introducir restricciones más finas que las actuales.

Por otro lado, para cada diagrama de clases definimos un diagrama de interacciones en el que se especifican cómo ciertas operaciones OLAP permiten interactuar a dos clases cubo. En este sentido, las operaciones OLAP permiten introducir condiciones más "gruesas" y niveles de jerarquía que corresponden a otras dimensiones no consideradas en la clase cubo sobre la que se aplica la operación.

[3] El conjunto de operaciones propuesto se puede consultar en [6].

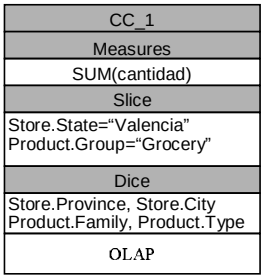

Figura 2. *Ejemplo de una clase cubo*

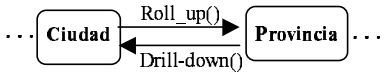

Figura 3. *Ejemplo de diagrama de estados*

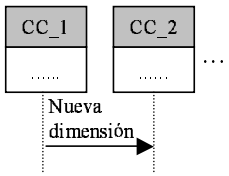

Figura 4. *Ejemplo de diagrama de interacciones*

En la Figura 3 se puede ver un ejemplo de diagrama de estados en el que la clásica operación *roll-up* nos permite navegar a lo largo de los niveles de jerarquía ciudad y provincia de la dimensión almacén. Por otro lado, en la Figura 4 se puede ver un diagrama de interacciones que muestra cómo se obtiene otra clase cubo introduciendo una nueva dimensión de análisis sobre la clase cubo inicial CC_1.

4 Conclusiones y trabajos futuros

En este artículo hemos presentado cómo se puede extender el lenguaje UML para considerar las propiedades estructurales y dinámicas y de comportamiento del modelado multidimensional a un nivel conceptual. Esta propuesta se ha plasmado en una herramienta CASE que, además, permite generar de una forma semi-automática la implementación del modelo en la herramienta OLAP Informix Metacube [5].

Actualmente estamos considerando los siguientes trabajos futuros:

- Llevar a cabo la implementación en sistemas de gestión de bases de datos objeto-relacionales y orientados a objetos. Además, se pretende diseñar una política de mantenimiento de vistas adecuada a partir de los requisitos iniciales de usuario especificados en las clases cubo y utilizando la información contenida en los diagramas de estados e interacciones.
- Introducir más elementos estándares en nuestra propuesta como puede ser la especificación de las restricciones mediante el Lenguaje de Restricciones de Objetos (*Object Constraint Language*, OCL) y de las consultas iniciales de usuario mediante el Lenguaje de Consulta a Objetos (*Object Query Language*, OQL).
- Integrar en nuestra herramienta CASE las operaciones OLAP de consulta que hemos definido para poder seguir la evolución de los requisitos iniciales de usuario. Además, nos planteamos generar datos de ejemplo a partir del modelo definido con lo que pretendemos integrar aspectos de modelado conceptual con aspectos de consulta de datos.

Referencias

[1] A. Abelló, J. Samos, and F. Saltor. A data warehouse multidimensional data models classification. Technical report, Universidad de Granada, Diciembre 2000.
[2] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.
[3] R. Kimball. *The data warehousing toolkit*. John Wiley, 2 edition, 1996.
[4] V. Poe, P. Klauer, and S. Brobst. *Building a Data Warehouse for Decision Support*. Prentice-Hall, 1997.
[5] J. Trujillo. *El modelo GOLD: un modelo conceptual orientado a objetos para el diseño de aplicaciones OLAP*. Tesis. Dept. Lenguajes y Sistemas Informáticos. Universidad de Alicante, June 2001.
[6] J. Trujillo, J. Gómez, and M. Palomar. Modeling the Behavior of OLAP Applications Using an UML Compilant Approach. In T. Yakhno, editor, *Proceedings of the First International Conference On Advances in Information Systems (ADVIS'00)*, volume 1909 of *Lecture Notes in Computer Science*, pages 14–23. Springer-Verlag, 2000.
[7] J. Trujillo, M. Palomar, and J. Gómez. Applying Object-Oriented Conceptual Modeling Techniques To The Design of Multidimensional Databases and OLAP Applications. In H. Lu and A. Zhou, editors, *Proceedings of the First International Conference On Web-Age Information Management (WAIM'00)*, volume 1846 of *Lecture Notes in Computer Science*, pages 83–94. Springer-Verlag, 2000.