ODMG - Estandarizando Bases de datos OO

Nieves Carralero Colmenar.
IES Pedro Mercedes. Junta de Comunidades de Castilla-La Mancha. España.

RESUMEN

En este artículo se esboza la propuesta de ODMG (Object Data Management Group) para estandarizar las bases de datos orientadas a objetos. En particular se centra en ODL como lenguaje de definición de objetos.

1. ODMG (Object Data Management Group)

El modelo de objetos ODMG (Object Data Management Group) permite que los diseños OO y las implementaciones usando lenguajes OO sean portables entre los sistemas que lo soportan. El modelo de datos dispone de unas primitivas de modelado. Estas primitivas subyacen en la totalidad de los lenguajes orientados a objetos puros (Eifflel, Smalltalk, etc.) y en mayor o menor medida en los híbridos (Java, C++, etc.):

Las primitivas básicas de una base de datos orientada a objetos son los *objetos* y los *literales*.

- Un *objeto* es una *instancia* de una entidad de interés del mundo real. Los *objetos* necesitan de un identificador único (OID- Identificador de Objeto).
- Un *literal* es un valor específico. Los literales no tienen identificadores. Un literal no tiene que ser necesariamente un solo valor, puede ser una estructura o un conjunto de valores relacionados que se guardan bajo un solo nombre (por ejemplo, enumeraciones).

Los objetos se dividen en *tipos*. Sin ser estrictos en la definición, un *tipo* se puede entender como una *clase* en POO. Los objetos de un mismo tipo tienen un mismo comportamiento y muestran un rango de estados común:

- El comportamiento se define por un conjunto de operaciones que pueden ser ejecutadas por un objeto del tipo (métodos en POO).

- El estado de los objetos se define por los valores que tienen para un conjunto de propiedades. Las propiedades pueden ser:
    - *Atributos*: Los atributos toman *literales* por valores y son accedidos por operaciones del tipo *get_value* y *set_value* (como exige la OO pura, y nunca se accede a ellos directamente)
    - *Relaciones* entre el objeto y uno o más objetos: Son propiedades que se definen entre tipos de objetos, no entre instancias. Las relaciones pueden ser 1-a-1, 1-a-muchos o muchos-a-muchos.

Un tipo tiene una *interfaz* y una o más implementaciones. La *interfaz* define las propiedades visibles externamente y las operaciones soportadas por todas las instancias del tipo. La *implementación* define la representación física de las instancias del tipo y los métodos que implementan las operaciones definidas en la interfaz.

Los tipos pueden tener las siguientes propiedades:

- *Supertipo*. Los tipos se pueden jerarquizar (*herencia simple*). Todos los atributos, relaciones y operaciones definidas sobre un *supertipo* son heredadas por los *subtipos*. Los *subtipos* pueden añadir propiedades y operaciones adicionales para proporcionar un comportamiento especializado a sus instancias. El modelo contempla también la *herencia múltiple*, y en el caso de que dos propiedades heredadas coincidan en el subtipo, se *redefinirá* el nombre de una de ellas.
- *Extensión*: es el conjunto de todas las instancias de un tipo dado. El sistema puede mantener automáticamente un índice con los miembros de este conjunto incluyendo una declaración de extensión en la definición de tipos. El mantenimiento de la extensión es opcional y no necesita ser realizada para todos los tipos.
- *Claves*: propiedad o conjunto de propiedades que identifican de forma única las instancias de un tipo (OID). Las claves pueden ser *simples* (constituidas por una única propiedad) o *compuestas* (constituidas por un conjunto de propiedades).

2. ODL (Lenguaje de Definición de Objetos)

ODL (Lenguaje de Definición de Objetos) es un lenguaje para definir la especificación de los tipos de objetos en sistemas compatibles con ODMG. ODL es el equivalente de DDL (lenguaje de definición de datos) de los SGBD relacionales. ODL define los atributos y las relaciones entre tipos y especifica la signatura de las operaciones. ODL se utiliza para expresar la estructura y condiciones de integridad sobre el esquema de la base de datos: mientras que en *una base de datos relacional* DDL define las tablas, los atributos en la tabla, el dominio de los atributos y las restricciones sobre un atributo o una tabla, en *una base de datos orientada a objetos* ODL define los objetos, métodos, jerarquías, herencia y el resto de elemento del modelo OO.

Una característica importante que debe cumplir (según ODMG) un ODL es ofrecer al diseñador de bases de datos un sistema de tipos semejantes a los de otros lenguajes de programación OO. Los tipos permitidos son:

- Tipos *básicos*: incluyen los tipos *atómicos* (Boolean, Float, Short, Long, Double, Chart, etc.) y las *enumeraciones*.
- Tipos de *interfaz* o estructurados: son tipos complejos obtenidos al combinar tipos básicos por medio de los siguientes constructores de tipos:
    - Conjunto (Set<*tipo*>) denota el tipo cuyos valores son todos los conjuntos finitos de elementos del *tipo*.
    - Bolsa (Bag<*tipo*>) denota el tipo cuyos valores son bolsas o multi-conjuntos de elementos del *tipo*. Una bolsa permite a un elemento aparecer más de una vez, a diferencia de los conjunto, por ejemplo {1, 2, 1} es una bolsa pero no un conjunto
    - Lista (List<*tipo*>) denota el tipo cuyos valores son listas ordenadas finitas conteniendo 0 o más elementos del *tipo*. Un caso especial lo constituye el tipo *String* que es una abreviatura del tipo List<char>.
    - Array (Array<tipo,i>) denota el tipo cuyos elementos son arrays de i elementos del *tipo*.

Por tanto con la ayuda de ODL se puede crear el esquema de cualquier base de datos en un SGBDOO que siga el estándar ODMG. Una vez creado el esquema, usando el propio gestor o un lenguaje de programación se pueden crear, modificar, eliminar y consultar objetos que satisfagan ese esquema.

El siguiente ejemplo muestra la definición de un esquema usando ODL para el SGBDOO *Matisse*. En el ejemplo se definen dos tipos complejos llamados *Libro* y *Autor*:

- Un Libro tiene como atributos: *título* de tipo básico *String*, *año* y *páginas* de tipo básico *Integer*.
- Un Autor tiene como atributos: apellidos, nombre y nacionalidad de tipo *String* y edad de tipo *Short.*
- Entre ambos tipos hay relaciones definidas como conjuntos *Set*: un *Libro* es *escrito_por* un conjunto de Autores y un *Autor escribe* un conjunto de libros.

```
interface Libro
{
        /* Definición de atributos
attribute string título;
attribute integer año;
attribute integer paginas;
attribute enum PosiblesEncuadernaciones (Dura,Bolsillo) tipo;

        /* Definición de relaciones */

relationship Set<Autor> escrito_por inverse Autor::escribe;
}


interface Autor
{
/* Definición de atributos */
attribute string apellidos;
attribute string nombre;
attribute string nacionalidad;
attribute short edad;

/* Definición de relaciones */
relationship Set<Libro> escribe inverse Libros::escrito_por;
```