El modelo de objetos

La característicasticas que distingue a una base de datos orientada a objetos es que proporciona soporte al modelo de objetos.

Ambito del modelo

El modelo de objetos: *Object Database Management Group* (ODMG).

- El ODMG es un consorcio de vendedores de DMBSs constituido para desarrollar un estándar que asegure la portabilidad de las aplicaciones
- Eventualmente, también la interoperabilidad entre los diferentes productos.

El trabajo del ODMG evoluciona continuamente: se acaban de publicar las especificaciones del ODMG 3.0.

- Importante en la medida es que está soportado por productos ODBMS comerciales. Las compañías se comprometieron a dar soporte al modelo de objetos ODMG-93.
- El nivel actual de soporte del modelo evolucionará necesariamente hacia ese estándar.
- El ODMG-93 también define características (definición de tipos lenguaje de consulta de objetos) que no dependen del lenguaje de programación utilizado.

Interface e implementación

El concepto de objeto es el bloque básico de construcción del software. El interior de un objeto es privado mientras que su exterior es público y disponible para que otros objetos lo inspeccionen y utilicen.

La separación entre interface e implementación es esencial y la base del concepto de encapsulación fundamental en la tecnología orientada a objetos.

Los elementos semánticos básicos del modelo de objetos del ODMG son:

- La primitiva fundamental de diseño es el objeto. Los términos objeto e instancia se utilizan indistintamente.
- Cada objeto es unívocamente identificable mediante un identificador de objeto (*Object IDentifier*, OID). El OID no se puede modificar durante el periodo de vida del objeto.
- Los objetos se clasifican en una jerarquía de *tipos* y *subtipos*. Todos los objetos de un determinado tipo presentan características comunes (estado y comportamiento). Un *subtipo* hereda las características de sus *supertipos*.
- El *estado* de un objeto viene definido por un conjunto de propiedades, que pueden ser tanto *atributos* como *relaciones* con uno o más objetos.
- EL *comportamiento* está definido por el conjunto de *operaciones* que son aplicables al objeto.

El modelo de objetos del ODMG define un objeto como una abstracción que puede implementarse mediante diferentes vías. Los modelos de ODMG y del lenguaje de programación de objetos específico utilizado son diferentes pero trabajan conjuntamente.

El modelo de objetos C++

En el lenguaje de programación C++:

- Un objeto es una instancia de una clase.
- El programador define la clase especificando sus elementos (con nombre y tipo) y sus funciones miembro (cada una con su nombre, el tipo de sus argumentos y el tipo de retorno).
- C++ cuenta con un conjunto predefinido de tipos de datos simples (por ejemplo `char`, `int`, `float`, `double`, etc...), un conjunto predefinido de tipos estructudados (por ejemplo `array`) y la habilidad para definir nuevas clases que son agregados de estos elementos y sus operaciones.

En el lenguaje de programación C++ cada variable tiene un tipo bien definido.

- El compilador comprueba todas las asignaciones de valores para asegurar que coinciden con los tipos de las variables designadas.
- La comprobación de tipos se considera una buena práctica, ya que previene la aparición de ciertos tipos de errores. Cuando se produce un error de este tipo, el compilador puede aplicar una regla de conversión de tipo o señalar el error.
- El lenguaje de programación C++ se dice que es un lenguaje *strongly tiped*.

Objetos modificables y objetos constantes

En el modelo de objetos ODMG existen dos tipos de objetos: mutables e inmutables.

- Los objetos mutables son aquellos cuyo estado puede ser modificado. Los valores de las propiedades de un objeto mutable pueden cambiarse.
- Los objetos inmutables se conocen con el nombre de *literales* y su valor es constante y no se puede cambiar.

La mutabilidad introduce una restricción importante:

- Si un ODBMS sabe que un determinado objeto es inmutable, puede impedir que las aplicaciones lleven a cabo modificaciones sobre el objeto.
- El conocimiento de la mutabilidad es también importante en la determinación de la estrategia de implementación más adecuada para un objeto.

Por ejemplo, un objeto que sea el conjunto de los nombres de los estados de la Comunidad Europea es inmutable.

Identificador único

En ODMG, todos los objetos tienen un identificador único inmutable.

- Cada objeto cuenta con una existencia separada y pude distinguirse de todos los demás objetos.
- Cuando se crea un objeto, el *sistema* le asigna un identificador de objeto, conocido normalmente como OID.
- El ODBMS utiliza el OID para identificar unívocamente al objeto y para comprobar la igualdad entre objetos. Dos objetos son el mismo sí y sólo sí tienen el mismo OID.

Por ejemplo, el núcleo del modelo OMG CORBA también incluye el concepto de OID, pero:

- Un objeto puede tener diferentes OIDs (denominados *objrefs*) en diferentes contextos.
- El mismo *objref* puede aplicarse a diferentes objetos en la medida en que se encuentren en diferentes contextos.

Esta aproximación simplifica la tarea de gestionar grandes entornos distribuidos (en donde un identificador único puede ser complicado de mantener).

Implementación de OIDs

- La representación física de los identificiadores de objetos no está especificada en el modelo.
- Los ODBMSs que implementan el ODMG utilizan una estrategia de implementación propia.
- Es frecuente que un ODBMS utilice una aproximación diferente para representar los OIDs de objetos y literales.
  - El identificador de un literal se representa por un patrón de bits que codifican el valor del literal.
  - En un objeto mutable el OID se representa por un patrón de bits único generado expresamente para ese fin.

El lenguaje C++ también cuenta con una noción abstracta de identidad:

- Emplea una técnica de representación diferente. Todos los objetos de C++ son transitorios.
- En C++ la identidad de un objeto mutable se representa mediante su dirección.
- La dirección se utiliza como puntero, permitiendo que un objeto haga referencia a otro en memoria.
- La identidad de un literal C++ es una codificación de su valor en lugar de una dirección.

Ambito del OID

- Los identificadores de C++ y de un ODBMS deben ajustarse a diferentes requisitos, ya que se aplican a objetos cuyo ciclo vital es diferente.
- Un ODBMS no puede utilizar la dirección de memoria principal para representar la identidad de un objeto, ya que el espacio de direcciones es demasiado pequeño.
- Una aplicación puede ser escalable en el tiempo sólo si los objetos que necesira pueden almacenarse y relocalizarse en diferentes nodos.

Similitud: los *objrefs* de CORBA son únicos dentro del ámbito de una red.

- Los *objrefs* de CORBA son asignados por un *Object Request Broker* (ORB) y son aplicables a cualquier objeto registrado por el ORB.
- Los *objrefs* pueden ser únicos aún entre productos de múltiples vendedores.

Diferencia: Los OIDs asignados por un producto ODBMS son únicos sólo dentro de la instalación del producto.

Objetos con nombre

Además del OID, un objeto puede tener uno o varios nombres con cierto significado para la aplicación.

- Un nombre es un literal de caracteres que puede utilizar el ODBMS para localizar al objeto.
- Los nombres facilitan que la aplicación acceda a un objeto particular.
- La aplicación es la que determina el nombre más adecuado a utilizar en cada caso.

Los nombres de objetos no son lo mismo que las claves principales del modelo relacional:

- Difieren en su ámbito de aplicación y en su mutabilidad.
- Una clave principal en el modelo relacional es el conjunto de columnas en una tabla que identifican cada una de las filas. El ámbito de la clave es la tabla en la que está definida y si la clave cambia, cambia la fila.
- Un objeto, sin embargo, puede contar con varios nombres que pueden ser modificados sin modificar la identidad del objeto.

El ODBMS proporciona un directorio de nombres para que las aplicaciones lo utilicen de acuerdo con sus necesidades y propósitos.

Características de un objeto: propiedades y operaciones

Las características de un objeto vienen definidas por su conjunto de propiedades y operaciones. Estas características son precisamente lo que se describe mediante la interface al objeto:

- Los atributos cuyos valores pueden leer y modificar las aplicaciones.
- Las relaciones que las aplicaciones pueden recorrer desde el objeto hacia otros objetos relacionados.
- Las operaciones que son aplicables sobre el objeto.

En algunos modelos de objetos, tales como OMG CORBA y Smalltalk, la interface del objeto sólo incluye operaciones.

El modelo de objetos del ODMG incluye tanto propiedades como operaciones.

Por ejemplo, consideremos un objeto persona. Tres características de este objeto son el nombre, la altura y la edad.

- El modelo ODMG considera que estas tres características son atributos (propiedades) del tipo persona.
- El modelo OMG CORBA considera que nombre, altura y edad son operaciones definidas como parte de la interface al objeto de tipo persona.

El modelo ODMG deja libertad a la implementación los detalles tales como si los atributos nombre, altura y edad se representan mediante campos de datos o como funciones que se aplican a otros valores almacenados en el estado interno del objeto (por ejemplo, la edad se podría calcular a partir de la fecha de nacimiento).

Tipo de un objeto

Los objetos que cuentan con la misma interface se dice que son del mismo tipo.

- La definición del tipo incluye la especificación de la interface, es decir, del conjunto de características aplicables a todas las instancias.
- En este sentido, el tipo constituye el "molde" o "patrón" del objeto (*template*).
- Mantener y gestionar la información acerca de los tipos en un sistema es responsabilidad del gestor de tipos.
- El gestor de tipos del ODBMS trabaja conjuntamente con el gestor de tipos del lenguaje orientado a objetos. Esta cooperación es necesaria para que el entorno de la base de datos y el entorno de programación se comporten como un entorno integrado.
- En el ODMG, el tipo de un objeto se determina cuando es creado y no puede modificarse posteriormente. Esta limitación es consistente con el modelo de objetos de C++.

Características de un tipo

Una de las características de un entorno orientado a objetos es la extensibilidad del sistema de tipos. Dos aspectos de este mecanismo son importantes:

- El tipo original toma el papel de *supertipo* para estos nuevos tipos.
- La interface de cada *subtipo* incluye la interface de su supertipo. La interface se dice que es heredada del supertipo.

Por ejemplo, un `Empleado_Parcial` es un tipo de `Empleado`: tiene los mismos atributos, relaciones y operaciones

Un subtipo puede introducir características adicionales (atributos, relaciones y operaciones) que no forman parte de la especificación de su supertipo.

Por ejemplo, `Empleado_Parcial` puede tener una característica adicional como `horasExtra` como parte de su interface

Un objeto se considera que es una instancia de su tipo inmediato y de todos sus supertipos.

Por ejemplo, un `Empleado_Parcial` es un tipo de `Empleado` y éste a su vez un tipo de `Persona`. De acuerdo con esto, un objeto `Empleado_Parcial` es una instancia de los siguientes tipos: `Empleado_Parcial`, `Empleado` y `Persona` y contiene las características de todos estos tipos

Características de un tipo

El modelo de objetos del ODMG es similar al de C++ en el sentido de que la jerarquía de tipos no es estricta y un tipo puede tener múltiples supertipos (herencia múltiple).

- Un subtipo puede ser substituido por su supertipo en cualquier contexto en el que este último sea válido.

  Por ejemplo, si las operaciones `get_name`, `set_phone` y `get_payroll` están definidas en `Empleado` entonces también se pueden aplicar a `Empleado_Parcial`

- Cada subtipo puede tener su propia implementación del método.

  Por ejemplo, el subtipo `Empleado_Parcial` implementa su propio método `get_payroll`

- La posibilidad de substituir un subtipo por su supertipo es un factor importante para mejorar la legibilidad, el mantenimiento y la extensibilidad del código.

Propiedades de un tipo ...

Los tipos son a su vez objetos, pueden tener atributos y participar en relaciones:

- Los atributos que el modelo ODMG define para un tipo son su *extent* y las restricciones de singularidad.
- Estas propiedades se aplican al tipo como un todo y no sobre sus instancias individuales.
- La única relación que tiene un tipo viene determinada por el grafo tipo–subtipo.

El conjunto de todas las instancias de un tipo se denomina su *extent*. La utilización más común de un *extent* es el soporte de consultas.

- Es responsabilidad del ODBMS mantener el *extent*.
- Cuando un objeto es creado, el DBMS añade una instancia al *extent* y la elimina cuando la aplicación borra el objeto.
- Para mantener la integridad del *extent* el ODBMS no permite el acceso directo al *extent*.

Un tipo puede disponer de una o más restricciones de singularidad, denominadas claves.

- Una clave está constituida por uno o más atributos que identifican unívocamente al objeto dentro del *extent*.
- Ya que el ámbito de singularidad es el *extent*, tienen un papel análogo a las claves primarias en el modelo relacional.
- El ODBMS crea y mantiene automáticamente los índices de las claves y los utiliza para asegurar la restricción de singularidad.

Organización de tipos y objetos

Formas de clasificar tipos y objetos:

- Mediante su relación tipo–subtipo.
- La ya comentada de clasificar los objetos entre mutables e inmutables.
- Distinguir entre objetos atómicos o estructurados. Un objeto atómico no puede subdividirse; sus constituyentes no son direccionables de forma independiente. Un objeto estructurado puede subdividirse y sus partes ser usadas de forma independiente.

Existen dos tipos fundamentales de objetos estructurados:

- *Estructuras*. Una estructura puede tener elementos heterogéneos. La estructura cuenta con un número fijo de posiciones, cada una de las cuales con su tipo y que pueden hacer referencia a objetos mutables o inmutables, atómicos o estructurados. La inserción y la eliminación de objetos en las posiciones *slots* se lleva a cabo referenciándolos por su nombre, por ejemplo `direccion.codigo_postal = 46014`.
- *Colecciones*. Los elementos de una colección deben ser homogéneos (todos deben ser del mismo tipo). Una colección puede contener un número arbitrario de elementos. Los tipos predefinidos de colección son *Set*, *Bag*, *List* y *Array* (estos últimos son unidimensionales y de dimensión variable).

Organización de tipos y objetos ...

El modelo de objetos incluye la especificación de las operaciones que son aplicables a cada una de las colecciones. Las colecciones son importantes en la medida en que cualquier ODBMS que cumple las especificaciones del ODMG los implementa (portabilidad).

Nombre del tipo	Mismo tipo	Orden	Duplicados
Estructura	No		
Colección	Si		
Set	Si	No	No
Bag	Si	No	Si
List	Si	Si	No
Array	Si		Si

Relaciones

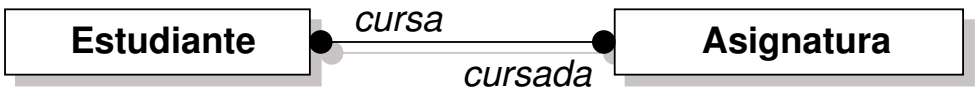
Otra de las características de un objeto son sus relaciones.

- Las relaciones se definen sobre los tipos, pero no son propiedades del tipo ya que son las instancias las que participan en la relación y no el tipo en sí mismo.
- Cualquier objeto puede participar en una relación con otros objetos.
- En el modelo ODMG, las relaciones no son lo mismo que los punteros, aun cuando un puntero establece un camino entre un objeto y otro. Una relación es una abstracción que representa la asociación entre objetos mientras que un puntero es una estructura física.

Una relación se diseña mediante:

- Una pareja de *firmas*, cada una de las cuales define el tipo del otro objeto implicado en la relación.
- El nombre de una expresión de recorrido (*traversal path*) usado para referirse a los objetos relacionados.

Por ejemplo, en la figura se representa la relación entre un estudiante que *cursa* una serie de asignaturas y una asignatura que es *cursada* por un conjunto de estudiantes.



Relaciones ...

La expresión de recorrido se declara en la inteface de definición de los tipos de los objetos que participan en la relación:

- La expresión de recorrido *cursa* se define en la especificación de la interface del tipo Estudiante.
- La expresión del recorrido *cursada* se define en la interface de especificación de Asignatura.
- La cardinalidad de la relación se incluye en la especificación de la expresión de recorrido.

El ejemplo anterior se especifica, mediante el ODL de ODMG, del siguiente modo:

```
interface Estudiante
( extent estudiantes)
{ attribute String nombre;
  attribute Short  id_estudiante;
  relationship Set<Asignatura> cursa
     inverse Asignatura::cursada;
}
interface Asignatura ()
{ Attribute String nombre;
  relationship Set<Estudiante> cursada
     inverse Estudiante::cursa;
}
```

Relaciones ...

El ODBMS proporciona las siguientes operaciones, entre otras, para manipular las relaciones:

- `set`. Establece una relación uno-a-uno entre una pareja de objetos. Establecer la relación supone crear la pareja de expresiones de recorrido entre los objetos.
- `clear`. Elimina una relación uno-a-uno. Eliminar la relación supone eliminar las expresiones de recorrido entre objetos.
- `insert_element`. Se aplica sólo a las relaciones uno-a-muchos y muchos-a-muchos. Añadir un elemento significa crear la pareja de expresiones de recorrido.
- `remove_element`. También se aplica sólo a las relaciones uno-a-muchos y muchos-a-muchos. Eliminar el elemento supone eliminar las expresiones de recorrido.
- `get`. Devuelve una referencia a un único objeto en una relación uno-a-uno o en una relación múltiple desde el extremo muchos al extremo uno. `traverse` devuelve una referencia a un conjunto de objetos en el caso de relaciones muchos-a-muchos o en relaciones uno-a-muchos desde el extremo uno al extremo muchos. El programador puede iterar sobre el conjunto para obtener todos los objetos individuales.
- `create_iterator`. Devuelve un iterador empleado para iterar sobre el conjunto de objetos obtenido mediante una expresión de recorrido.