Base de Datos II

Unidad 5: Bases de Datos Orientada a Objetos

Objetivos

- Conceptos de OO y BD.
- ODMG.
- Object Definition Language(ODL). Object Query Language (OQL).

Problemática

Los modelos de bases de datos tradicionales han sido capaces de satisfacer con éxito las necesidades, en cuanto a bases de datos, de las aplicaciones de gestión tradicionales. Sin embargo, presentan algunas deficiencias cuando se trata de aplicaciones más complejas o sofisticadas como, por ejemplo, el diseño y fabricación en ingeniería (CAD/CAM, CIM), los experimentos científicos, los sistemas de información geográfica o los sistemas multimedia. Los requerimientos y las características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de gestión: la estructura de los objetos es más compleja, las transacciones son de larga duración, se necesitan nuevos tipos de datos para almacenar imágenes y textos, y hace falta definir operaciones no estándar, específicas para cada aplicación.

Problemática

Desarrollo de aplicaciones OO

- Lenguajes OO poseen aceptación en la práctica
- La programación en el paradigma OO sobre DBMS relacionales requiere de un mapeo entre los paradigmas o materialización de objetos a partir de la BD o desmaterialización hacia la BD o manejo de referencias entre objetos

Las bases de datos orientadas a objetos se crearon para tratar de satisfacer las necesidades de estas nuevas aplicaciones. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales. Una característica clave de las bases de datos orientadas a objetos es la potencia que proporcionan al diseñador al permitirle especificar tanto la estructura de objetos complejos, como las operaciones que se pueden aplicar sobre dichos objetos.

Ejemplos de ODBMS

- **ObjectStore (Adquirida por Progress)**
- **Objectivity/DB (Objectivity) http://www.objectivity.com/**
- **ONTOS (Ontologic)**
- **VERSANT (Versant Object Technology) http://www.versant.com/**
- **GemStone (GemStone) http://www.gemstone.com/**
- **Jasmine (CA) http://www.cai.com/**
- **DB4Object**
- **...**

ODMG (*Object Database Management Group)*

Durante los últimos años se han creado muchos prototipos experimentales de sistemas de bases de datos orientadas a objetos y también muchos sistemas comerciales. Conforme éstos fueron apareciendo, surgió la necesidad de establecer un modelo estándar y un lenguaje. Para ello, los fabricantes de los SGBD orientadas a objetos formaron un grupo denominado ODMG (*Object Database Management Group),* que propuso el estandar *ODMG–93 y que ha* ido evolucionando hasta el ODMG 3.0, su última versión. El uso de estándares proporciona portabilidad, permitiendo que una aplicación se pueda ejecutar sobre sistemas distintos con mínimas modificaciones. Los estándares también proporcionan interoperabilidad, permitiendo que una aplicación pueda acceder a varios sistemas diferentes. Y una tercera ventaja de los estándares es que permiten que los usuarios puedan comparar entre distintos sistemas comerciales, dependiendo de qué partes del estándar proporcionan.

Formas Básicas de proveer OO

- ODBMS: son SGBD basados completamente en el modelo orientado a objetos
- ORDBMS: son SGBD relacionales que permiten almacenar objetos en sus relaciones (tablas).
- Software para transformación OO-Relacional

Conceptos básicos de OO

Tradicionalmente, los datos y los procedimientos se han almacenado separadamente: los datos y sus relaciones en la base de datos y los procedimientos en los programas de aplicación. La OO, combina los procedimientos de una entidad con sus datos.

Esta combinación se considera como un paso adelante en la gestión de datos. Las entidades son unidades autocontenidas que se pueden reutilizar con relativa facilidad. En lugar de ligar el comportamiento de una entidad a un programa de aplicación, el comportamiento es parte de la entidad en sí, por lo que en cualquier lugar en el que se utilice la entidad, se comporta de un modo predecible y conocido.

El modelo orientado a objetos también soporta relaciones de muchos a muchos, siendo el primer modelo que lo permite.

No parece que las bases de datos orientadas a objetos vayan a reemplazar a las bases de datos relacionales en todas las aplicaciones del mismo modo en que éstas reemplazaron a sus predecesoras.

Conceptos: Objeto

- Es un elemento autocontenido utilizado por el programa. Los valores que almacena un objeto se denominan atributos, variables o propiedades. Los objetos pueden realizar acciones, que se denominan métodos, servicios, funciones, procedimientos u operaciones.
- Los objetos tienen un gran sentido de la privacidad, por lo que sólo dan información sobre sí mismos a través de los métodos que poseen para compartir su información. También ocultan la implementación de sus procedimientos, aunque es muy sencillo pedirles que los ejecuten. Los usuarios y los programas de aplicación no pueden ver qué hay dentro de los métodos, sólo pueden ver los resultados de ejecutarlos. A esto es a lo que se denomina encapsulamiento de datos.
- Cada objeto presenta una interface pública al resto de objetos que pueden utilizarlo. Una de las mayores ventajas del encapsulamiento es que mientras que la interface pública sea la misma, se puede cambiar la implementación de los métodos sin que sea necesario informar al resto de objetos que los utilizan. Para pedir datos a un objeto o que éste realice una acción se le debe enviar un mensaje.
- Un programa orientado a objetos es un conjunto de objetos que tienen atributos y métodos. Los objetos interactúan enviándose mensajes. La clave, por supuesto, es averiguar qué objetos necesita el programa y cuáles deben ser sus atributos y sus métodos.

Conceptos: Objetos

- OID: identidad única de cada objeto independiente almacenado en la BD
- OID es inmutable: el valor no cambia (como lo puede hacer una clave de la BD)
- OID es único para ese objeto, si el objeto es removido se lleva el OID con él y ningún otro objeto lo puede usar
- El objeto se identifica con un triple (i, c, v)
  - i = OID
  - c = constructor de tipo
  - v = valor corriente que tiene el objeto
  - el modelo puede incluir varios constructores de tipo:
    - átomo, tupla, set, lista, bag, array
    - set/bag: conjunto de objetos sin orden
    - lista: conjunto de objetos con orden sin tamaño definido
    - array: conjunto de objetos con orden con tamaño definido
    - tupla: tipo estructurado (struct de C ó C++)

Conceptos: Clase

- Es un patrón o plantilla en la que se basan objetos que son similares. Cuando un programa crea un objeto de una clase, proporciona datos para sus variables y el objeto puede entonces utilizar los métodos que se han escrito para la clase. Todos los objetos creados a partir de la misma clase comparten los mismos procedimientos para sus métodos, también tienen los mismos tipos para sus datos, pero los valores pueden diferir.
- Una clase también es un tipo de datos. De hecho una clase es una implementación de lo que se conoce como un tipo abstracto de datos. El que una clase sea también un tipo de datos significa que una clase se puede utilizar como tipo de datos de un atributo.

Conceptos: Tipos de métodos

Hay varios tipos de métodos que son comunes a la mayoría de las clases:

- Constructores. Un contructor es un método que tiene el mismo nombre que la clase. Se ejecuta cuando se crea un objeto de una clase. Por lo tanto, un constructor contiene instrucciones para inicializar las variables de un objeto.
- Destructores. Un destructor es un método que se utiliza para destruir un objeto. No todos los lenguajes orientados a objetos poseen destructores.
- Accesores. Un accesor es un método que devuelve el valor de un atributo privado de otro objeto. Así es cómo los objetos externos pueden acceder a los datos encapsulados.
- Mutadores. Un mutador es un método que almacena un nuevo valor en un atributo. De este modo es cómo objetos externos pueden modificar los datos encapsulados.

Conceptos OO

Herencia:

- simple: un supertipo puede ser heredado por uno o más subtipos
- múltiple: un subtipo puede heredar de más de un supertipo
- selectiva: un subtipo hereda solo una parte del supertipo (cláusula EXCEPT)

Polimorfismo:

- Sobrecarga de funciones y operadores

Versiones y configuraciones

- En algunas aplicaciones se necesitan generar nuevas versiones de algunos objetos sin descartar los existentes (diseño), muchos ODBMS tienen algunas sistemas que facilitan la generación de versiones del objeto

Persistencia de Objetos

Cuestión:

- ¿Cómo proveer persistencia a una aplicación?

Problemas en la implementación de persistencia

- ¿Cómo representar objetos que pueden ser removidos y objetos que deben ser almacenados?
- ¿Cómo acceder a objetos eficientemente?
- ¿Dónde almacenar el código de una aplicación?
- ¿Cómo excluir objetos?

Persistencia de Objetos

Objeto persistente

- objeto que sobrevive a la ejecución del proceso que lo creó

Objeto transiente

- objeto que deja de existir cuando el proceso que lo creó termina su ejecución

LPOO

- concebido para manipular datos transientes

ODBMS

- concebido para manipular datos persistentes

Persistencia explicita e inferida

Persistencia explícita

- el programador explícitamente informa al sistema cuáles objetos son persistentes por medio de un nombre (mecanismo de nominación)

Persistencia inferida

- el sistema infiere cuáles objetos son persistentes, por el hecho de estar asociados a otros objetos persistentes
- también denominada persistencia por alcanzabilidad
  - si un objeto es persistente, todo objeto al cual el mismo referencia también es persistente
- el mecanismo comienza con un conjunto de raíces
  - todos los objetos "alcanzables" desde una raíz son persistente
  - ej.: "extent" (población) de una clase

Jerarquía de tipo - Herencia

- **Definición de nuevos tipos a partir de tipos predefinidos**
- **El mecanismo de herencia responde a la pregunta: es-un**
- **Se define un sub-tipo a partir de un supertipo,**
- **Jerarquía de generalización-especialización los sub-tipos especializan al supertipo**

Se heredan los atributos y métodos: reuso del código sólo se implementan los nuevos atributos y métodos

EXTENSION: colección de objetos del mismo tipo

Restricción: todos los objetos de una extensión que pertenece a un subtipo deben ser miembros de la extensión que pertenece al supertipo

Objetos no estructurados

- Para almacenar objetos grandes: objetos bitmap o textos largos generalmente conocidos como BLOBs
- Complejos porque requieren gran cantidad de almacenamiento, que puede recuperarse por porciones
- Complejos porque no es posible generar condición de selección directas sobre estos objetos basado en sus valores
- Se manipulan a través del comportamiento que se define al objeto
- Se dice que los ODBMS es un sistema que extiende los tipos, se pueden crear librerías de nuevos tipos que permiten la generación de una aplicación compleja particular, el DBMS debe proveer los mecanismos de almacenamiento y recuperación

Objetos complejos estructurados

- Por ejemplo una tupla que es lo más parecido a una relación o tabla que su vez puede estar compuesta por otras tuplas o sets (listas, arreglos), y así sucesivamente.
- La manera de referenciar los objetos son dos:
  - Composición: los sub-objetos son encapsulados dentro del objeto complejo y son considerados parte de él (es parte de)
  - Referencia: son objetos independientes referenciados por el objeto complejo (esta asociado a)
- En composición, los objetos que lo componen no necesitan OID, son accedidos a través de la interfase del objeto si elimina el objeto que lo contiene éstos tambien son eliminados
- Un objeto referenciado es independiente, tiene su propio OID, puede ser referenciado por + de un objeto y no se elimina automáticamente si el objeto complejo es eliminado

ODMG

- Estándar de ODBMS

Objetivo

- **integrar en forma transparente las funcionalidades de bases de datos en un lenguaje de programación OO**

Características

- **usuario no necesita aprender DML independientemente del LP que utiliza**
- **carga / descarga de objetos implícita**
- **lenguaje de consulta a bases de objetos**

Componentes de la arquitectura ODMG

Modelo de objetos

- **basado en el modelo de objetos de OMG con extensiones para BDs**

Lenguaje de definición de objetos (ODL)

- **equivalente a DDL**
- **lenguajes**
  - **ODL (Object Definition Language): definición de objetos**
  - **OIF (Object InterchangeFormat): intercambio de objetos entre OODBMS**

OQL (Object Query Language)

- **lenguaje declarativo para consultar una base de datos basado en SQL**

Componentes de la arquitectura ODMG

Enlaces (*bindings*) con LPOO

- **C++ OML (lenguaje de manipulación de objetos)**
  - **C++ ODL versión de ODL en sintaxis C++**
  - **mecanismos para ejecutar sentencias OQL**
  - **mecanismos para ejecutar operaciones sobre la base de datos y manipular transacciones**
- **Smalltalk**
- **Java**

ODMG – Modelo de objetos

- Modelo de objetos:
  - conceptos soportados por una BD ODMG
  - indica las características de los objetos ODMG: cómo se relacionan, cómo se nominan e identifican
- Un modelo específico se construye utilizando ODL (independiente del lenguaje o no)
- No todo LPOO implementa necesariamente todos los conceptos del modelo de objetos ODMG

Estado y Comportamiento

Estado de un objeto

- **definido por valores de un conjunto de propiedades que pueden ser:**
  - **atributos**
  - **relaciones**

Comportamiento

- **definido por un conjunto de operaciones (métodos)**

Modelando estados: atributos

```
class Persona {
    attribute short edad;
    attribute string name;
    attribute enum sexo{masculino, femenino};
    attribute direccion dire_personal;
    attribute set<telefono> telefonos;
    attribute departamento dept;
    }
```

- el valor de un atributo puede ser un literal ó un OID
- un atributo es distinto de una estructura de datos
- atributo:abstracto ; estructura de datos : representación física
- atributos pueden ser implementados por estructuras de datos o por métodos que calculan su valor

Modelando estados:relaciones

- Se modelan relaciones binarias solamente
- Se implementan entre objetos, no entre literales
- Cardinalidad: 1:1, 1:N, N;M
- Se definen explícitamente y se declaran de a pares

```
class Profesor { . . .
relationship set <Curso> enseña inverse Curso::es_enseñado_por;
}
class Curso { . . .
relationship Profesor es_enseñado_por inverse Profesor::enseña;
}
```

- los caminos de acceso pueden ser ordenado o desordenados dependiendo el tipo de colección que se especifica

Modelando estados: relaciones

- ODBMS es el responsable por mantener la integridad referencial, si el objeto es borrado, todo camino que accede al objeto también es borrado
- Asegura que las aplicaciones no puedan desreferenciar caminos de acceso a objetos no existentes
- Atributos (contenidos) vs. referencias
- Atributos de un objeto referencian otro objeto, sin caminos de acceso inversos o integridad referencial: "relaciones unidireccionales"

Modelando comportamiento: operaciones

- Firma de las operaciones: nombre, nombre y tipo de cada argumento, tipo de valores retornados, nombre de las excepciones que se pueden "lanzar"
- Una operación debe estar definida sobre un tipo simple
- El nombre de una operación debe ser único dentro de la definición de un tipo: no existe polimorfismo estático (sobrecarga nombres función)
- Se pueden sobrecargar operaciones sobre distintos tipos
- Las operación a emplear (en la sobrecarga de operaciones) se resuelven basado en el tipo de objeto más especifico definido en el primer argumento de la llamada.

Herencia de comportamiento

- Esta herencia se define por medio del símbolo :
- El supertipo siempre tiene que ser una interface
- Provista por el mecanismo de tipo y subtipo (tipificación)
  - interface Empleado { ... };
  - interface Profesor : Empleado { ... };
  - class ProfesorTitular : Profesor { ... };
- Admite herencia múltiple
- El modelo restringe
  - Clases pueden heredar de clases e interfaces
  - Interfaces sólo pueden heredar de interfaces

Herencia de estado y comportamiento

- Se emplea la palabra clave EXTENDS
- Se emplea para heredar estado y comportamiento entre clases
- El supertipo y el subtipo siempre tiene que ser una clase
- NO admite herencia múltiple
- Ejemplo:
  - class Person {...}
  - class Profesor extends Person {...}
  - class Estudiante extends Person {...}

Extensión de un tipo ("extend")

- Se puede declarar un extent para cualquier tipo definido via una declaración de clase (class)
- Es un nombre que va a contener todos los objetos persistentes de esa clase
- La extensión de un tipo, es el conjunto de todas las instancias de ese tipo que existen en la BD
- Equivalente al concepto de tabla en BDs relacionales
- La extensión es una lista mantenida automáticamente por el ODBMS cuando un objeto de ese tipo es creado, el mismo se inserta en la extensión del tipo
- cuando un objeto es destruido, el mismo se elimina de la extensión del tipo

Clave

- **Una clase con una extensión puede contener una o más claves cuyos valores son construidos para ser únicos dentro de cada extent.**
- **Concepto útil para un DBMS**
  - **el modelo OO no contiene el concepto de clave**
- **Si la clave es compuesta se debe especificar entre paréntesis.**

Nombre de Objeto

- Es posible asociar uno (o más) nombres a un objeto
  - Operación bind del Objeto Database en ODMG
- El nombre es un identificador provisto por el usuario (el OID es creado por el ODBMS)
- Existe una función que retorna el objeto identificado por un nombre dado
- Se utiliza para dar nombres a objetos "raíz", que sirven como puntos de entrada para navegar por la BD
- Ejemplo
  - en un sistema de puntos de ventas, el objeto raíz puede ser una registradora o una casa comercial
  - No es una clave (la clave es un atributo del objeto)
  - No es un OID (el OID es creado por el OODBMS)

Resumen del Modelo de Objetos

- Diferencia objetos (con identidad) y literales
- Literales: atómicos, colecciones, estructurados
- Clases con atributos, métodos, subtipos, extensiones
- Relaciones uni/bidireccionales, integridad referencial mantenida por el DBMS para las bidireccionales
- Objetos estructurados, colección de objetos
- Objetos persistentes y transientes
- Se pueden asignar nombres a los objetos
- Repositorio de metadatos (las veces de DD)
- Manejo de excepciones
- Bloqueo tradicional para el manejo de concurrencia

Objetivos

- Conceptos de OO y BD.
- ODMG.
- Object Definition Language(ODL). Object Query Language (OQL).