Administrador Libre de Bases de Objetos

Autores

Alejandro Ferrer (ale_ferrer@yahoo.com)
Elizabeth Jiménez Rey (ejimenezrey@yahoo.com.ar)
María Delia Grossi (mdg7501@yahoo.com.ar)
Arturo Carlos Servetto (aserve@gmail.com)
Gregorio Perichinsky (gperi@movi.com.ar)

Laboratorio de Bases de Datos y Sistemas Operativos
Tel. (011) 4343-0891 Int. 142
Departamento de Computación, Facultad de Ingeniería, Universidad de Buenos Aires

Introducción

Este trabajo se refiere a un proyecto de investigación y desarrollo tecnológico cuyo objetivo final es la liberación de un administrador de bases de objetos, bajo los principios del Software Libre:

- Libertad de ejecutar programas, para cualquier propósito
- Libertad de estudiar cómo funcionan los programas y adaptarlos a necesidades particulares
- Libertad de redistribuir copias
- Libertad de mejorar los programas y publicar las mejoras de manera que toda la comunidad se beneficie

El proyecto se lleva a cabo con la participación de alumnos de grado de la carrera de Ingeniería en Informática de la FIUBA, que cursan la materia Taller de Programación II o que desarrollan su Trabajo Profesional para graduarse.

Antecedentes

Las bases de objetos soportan en forma óptima la persistencia de objetos complejos e integran la tecnología de bases de datos con el paradigma de objetos. Tanto las bases de objetos como los entornos de programación orientados a objetos utilizan un mismo modelo eliminando los desajustes de impedancia entre modelos conceptuales y lenguajes de programación; por eso mismo y debido a la posibilidad de versionar clases y objetos, las bases de objetos y los sistemas basados en ellas son naturalmente flexibles y se pueden extender con facilidad, al tiempo que las bases de objetos casi no requieren administración.

Las características fundamentales que debe observar un sistema de bases de datos para calificar como orientado a objetos se delinean en *"The Object-Oriented Database System Manifesto"*, de Malcolm Atkinson (University of Glasgow)**,** François Bancilhon (Altaïr), David DeWitt (University of Wisconsin), Klaus Dittrich (University of Zurich), David Maier (Oregon Graduate Center) y Stanley Zdonik (Brown University):

- Reglas de Oro (características que deben soportar obligatoriamente)
  - Objetos Complejos
  - Identidad de Objetos
  - Encapsulación
  - Tipos y Clases
  - Jerarquías de Tipos o Clases
  - Sobreescritura, Sobrecarga y Enlace Dinámico

  - Completitud Computacional del Lenguaje de Manipulación de Datos
  - Extensibilidad del Conjunto de Tipos de Datos
  - Persistencia
  - Manejo de Almacenamiento Secundario
  - Concurrencia
  - Recuperación ante Fallas de Hardware o Software
  - Posibilidad de Realización de Consultas Ad Hoc
- Características Opcionales
  - Herencia Múltiple
  - Chequeo e Inferencia de Tipos
  - Distribución
  - Transacciones de Diseño (transacciones de larga duración, con transacciones parciales)
  - Manejo de Versiones de Objetos

Objetivos

Desarrollar un manejador de bases de objetos siguiendo estándares del OMG (Object Management Group).

Versionar componentes con soluciones alternativas para problemas tecnológicos subyacentes y realizar evaluaciones comparativas con experimentos de simulación.

Metodología

Prototipación Incremental Iterativa con procesos ágiles de desarrollo.

Herramientas de Desarrollo

Lenguajes C++ (módulos del servidor y bibliotecas de clientes) y Java (bibliotecas de clientes) en entorno Eclipse.

Estructura del Manejador

- Gestor de Solicitudes (ORB - Object Request Broker): controla todos los objetos del sistema (manejador) y los tipos de servicios que pueden proporcionar; permite que objetos de aplicación (solicitantes) generen y reciban solicitudes y respuestas de un proveedor (manejador).
- Gestor de Metadatos: gestiona el acceso a los catálogos de las bases de objetos (definiciones de clases y relaciones) y se encarga de mantenerlos.
- Gestor de Seguridad: se encarga de controlar las autorizaciones de usuario para realizar operaciones; después de comprobar que el usuario tiene autoridad para llevar a cabo una operación (solicitud), la redirige al módulo de Operaciones.
- Gestor de Operaciones: verifica que la operación satisfaga las restricciones de integridad necesarias (según el catálogo), determina la estrategia para ejecutarla (en caso de consultas declarativas) en término de servicios del módulo de almacenamiento y redirige cada solicitud de servicio al gestor de transacciones.
- Gestor de Transacciones: realiza el procesamiento requerido para las operaciones y garantiza que las operaciones concurrentes se realicen sin entrar en conflicto unas con otras.

Controla el orden relativo en que se ejecutan operaciones de distintas transacciones y redirige solicitudes al gestor de buffers.

- Gestor de Buffers: es responsable de la transferencia de objetos entre la memoria principal y el almacenamiento secundario en el servidor, y del control de acceso a los objetos. Interactúa con el módulo de almacenamiento.
- Almacenamiento: recupera y mantiene objetos en el almacenamiento persistente.

Avance

Se han construido los primeros prototipos de cada componente, que están en etapa de pruebas individuales. A continuación se describe las características de cada componente:

- Gestor de Solicitudes (ORB - Object Request Broker): Implementa un patrón de concurrencia Monitor para el registro de los módulos activos y un Productor Consumidor para el despacho de mensajes. Los servicios están soportados por dos colas de mensajes para cada módulo (una para recibir –CR- y otra para enviar –CE-), una tabla de módulos activos (para verificar la existencia del destinatario y sus colas asignadas) y un administrador de hilos de procesamiento.

  Para enviar un mensaje el módulo origen crea un objeto Mensaje que debe contener la constante que lo identifica, un número para el Hilo de Procesamiento (HP) del módulo que está operando este envío, la constante identificadora del módulo destino y, en caso de ser dirigido a un hilo de procesamiento en particular del módulo destino, un identificador de HP; este objeto es depositado en su cola de envíos. Si la comunicación es de tipo sincrónica, el Gestor de Solicitudes (GS) bloqueará al HP hasta que el mensaje de respuesta se encuentre en la CE del módulo.
- Gestor de Metadatos: Recibe solicitudes de definición, modificación y eliminación de clases, relaciones y jerarquías, y de recuperación de definiciones. Las modificaciones de una definición se realizan a través de versiones, y el estado de una definición incluye las colecciones de cambios para pasar de cada versión a la siguiente. Los objetos definición se usan como argumentos de mensajes (solicitudes de servicio de otros componentes), por lo que, dado que el ORB sólo transmite estados de objetos, la instanciación de una definición –el constructor de definiciones- se incluye en una biblioteca compartida por todos los componentes que requieran interactuar con definiciones).
- Gestor de Seguridad: Realiza las validaciones de permisos de las operaciones que cada usuario quiera realizar en el sistema, ya sea de Ingreso, Consulta o Modificación.
- Gestor de Operaciones: Recibe solicitudes de operaciones de navegación (acceso a objetos por identificador) y de consulta (en OQL –Object Query Language), en forma transaccional (como parte de una transacción) o no transaccional. Para realizar operaciones de consulta debe pedir al gestor de metadatos las definiciones de las clases y relaciones involucradas para determinar la estrategia de resolución en término de servicios del gestor de almacenamiento. Puede requerir recuperación de objetos, que direcciona al gestor de transacciones, o recuperación de listas de identificadores de objetos (para resolver consultas), que direcciona al gestor de almacenamiento.
- Gestor de Transacciones: Implementa un patrón de concurrencia Lector-Escritor para el acceso a un archivo de registro de operaciones según un protocolo WAL (Write-Ahead Loggin) y un patrón Singleton para la asignación de Identificadores de Transacciones.

  Espera se le envíe un pedido de inicio de transacción desde el Gestor de Operaciones, registra el pedido y el identificador asignado, se lo entrega a dicho módulo y, luego, toda operación realizada por un módulo que recibe en el mensaje el identificador informa lo que

hará al Gestor de Transacciones (GT) para que registre los estados previos a la modificación para permitir la reconstrucción del estado original.

En caso de una falla del equipo, de la operación o del sistema la transacción que se realizaba no fue terminada, por lo tanto no registró su finalización, y serán ejecutadas todas las operaciones de ella registradas en proceso inverso, para reconstruir el estado anterior.

- Gestor de Buffers: Implementa un patrón de concurrencia Lector-Escritor en los objetos almacenados, un patrón Singleton para la asignación de identificadores de iterardores y un Recolector de basura para liberar memoria.

  Almacena los objetos en listas, que corresponden a cada Iterador, que se indexan bajo dos tablas de Hash, una para los objetos de consultas OQL y otra para los objetos de Definiciones. La primera tiene por entradas los identificadores de iteradores que son entregados como respuesta a cada consulta; la segunda los identificadores de clase de los objetos definición almacenados.

  Los objetos de datos se almacenan solo una vez, si son requeridos por más de una consulta se los relaciona con más de un Iterador.

  Los objetos de definiciones son almacenados en estructuras que los relacionan con los Identificadores de archivos abiertos correspondientes, necesarios para el manejo de estos datos.

  El Recolector de basura funciona bajo dos condiciones, tiempo de permanencia excedido y necesidad de memoria para otro proceso. La eliminación se realiza bajo un criterio de tiempos, se quita el más viejo, midiendo esto por las marcas de tiempo que tiene cada objeto. Esta marca se actualiza con cada operación que se realiza sobre él o sobre el Iterador al que pertenece.

- Almacenamiento: Se desarrollaron dos prototipos diferentes con la misma interfaz. Uno de ellos organiza los objetos en árboles B+, ordenados por identificador, e implementa los índices usando identificadores para referenciar a objetos (índices secundarios) también organizados en árboles B+. El otro organiza los objetos en bloques de archivos multiindexados, e implementa todos los índices en árboles B+ referenciando a los objetos a través de números de bloque de alojamiento (índices primarios).

  Para implementar el acceso secuencial, todo requerimiento de servicio de acceso a un objeto o a un índice en este modo debe efectuarse con la definición de la clase de objetos involucrados, como toda otra solicitud, y con un objeto testigo que contiene información sobre el último bloque o nodo del objeto accedido previamente y la identificación de ese objeto.

  Todos los objetos se almacenan con el número de versión de la definición de su clase en la que fueron almacenados; para devolver un objeto, el gestor de almacenamiento debe hacerlo según la última versión de su clase, y lo hace a través de servicios que presta el objeto definición provisto por el gestor de metadatos (como el gestor de solicitudes sólo transfiere estados de objetos, la instanciación de definiciones se hace en cada componente que requiera interactuar mediante una biblioteca compartida por los componentes).

  Para aprovechar el buffering del sistema operativo del servidor y proveer el acceso secuencial mediante el sistema de objetos testigo, el gestor de buffers mantiene los estados de definiciones en buffers ad hoc, junto con los descriptores de archivos correspondientes a los archivos de almacenamiento y de índices. Si el gestor de almacenamiento recibe una solicitud con descriptores nulos es porque los archivos correspondientes a la clase definida no están abiertos; entonces abre los archivos y devuelve los descriptores que quedarán temporalmente en un buffer. El gestor de buffers elimina los buffers por “time out”, y al hacerlo el destructor correspondiente cierra los archivos garantizando la coherencia.

Cada uno de los prototipos fue desarrollado por un grupo de alumnos de la materia Taller de Programación II.

Más allá de las pruebas que cada grupo realice a cada componente con herramientas de software de desarrollo propio de inspección y monitoreo, otro grupo se encarga del diseño de datos de prueba y del desarrollo de software para pruebas de caja negra para cada componente.

Perspectivas

En el año en curso se desarrollarán bibliotecas para aplicaciones cliente del manejador para los lenguajes de programación C++ y Java. Estas bibliotecas implican fundamentalmente la sobrecarga de constructores y destructores de objetos y de iteradores para clases persistentes.