Administración de Objetos con Almacenamiento Adaptativo

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
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 con almacenamiento adaptativo, 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

Antecedentes

En todo proceso de desarrollo de sistemas, la definición de bases de datos continúa siendo un problema fundamental, que requiere la participación de especialistas, entraña riesgos importantes y que generalmente condiciona la elección de tecnologías de implementación. Es muy común que los responsables del diseño de bases de datos comiencen directamente con el diseño lógico, desaprovechando principios de abstracción del modelado conceptual en desmedro de la calidad e inteligibilidad de las soluciones. La brecha entre el diseño conceptual y el físico continúa siendo un problema que provoca encontrar nuevas soluciones o mejorar existentes.

Otro problema de importancia en los sistemas de bases de datos tradicionales es que normalmente existe otro desajuste o "impedancia" entre las construcciones típicas provistas por los modelos de datos y las provistas por los ambientes de programación basados en objetos. La evolución tecnológica en los ambientes de programación intenta subsanar este problema sin abandonar las bases de datos relacionales bien privilegiando las construcciones del ambiente mediante *"frameworks"* de mapeo objeto-relacional o bien privilegiando las construcciones de las bases de datos proveyendo estructuras con equivalencia directa en las bases, como los *"datasets"*.

A medida que los desarrolladores se expanden con nuevos tipos de aplicaciones con estructuras de datos complejas y amplían las existentes, sus intentos por utilizar manejadores de bases de datos relacionales chocan con limitaciones de desempeño y funcionalidad, lo mismo que cuando se requieren operaciones complejas en ambientes distribuidos. A medida que la complejidad de una aplicación aumenta las tablas se multiplican, provocando no sólo que aumenten los tiempos de ejecución sino la dificultad de cambio de la base de datos y de la aplicación: se pierde flexibilidad; también se sacrifica la extensibilidad.

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; 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 el *"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 datos orientado a objetos adaptativo, que permita cambiar estructuras de objetos *"on the fly"*, detecte patrones de navegación y reestructure el almacenamiento de los objetos para optimizar el rendimiento, y tienda a "cero administración".

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 y documentación en UML 2.0. En la programación participan alumnos de grado de las carreras de Licenciatura en Análisis de Sistemas y de Ingeniería en Informática, como parte de los trabajos prácticos que deben realizar en las materias Organización de Datos y Taller de Programación II. También se prevé la participación de alumnos con mayor grado de participación y responsabilidad en realización del Trabajo Profesional para graduarse de Ingenieros en Informática.

Cronograma

- Primera etapa: prototipación y evaluación comparativa de repositorios de objetos (dos años).
- Segunda etapa: prototipación y evaluación comparativa de servidores de objetos (dos años).
- Tercera etapa: prototipación y evaluación comparativa de administradores de objetos (dos años).

Avance

El proyecto se encuentra en su primer etapa. Se está investigando y evaluando experimentalmente organizaciones de archivos para el almacenamiento de objetos y organización de índices para la recuperación eficiente.

A los efectos de considerar la organización óptima de archivos para almacenar objetos e implementar índices, se considera la siguiente clasificación de archivos y sus patrones de acceso característicos:

- *De Datos Maestros*: datos de un sistema de información que representan entidades de existencia real o ideal, por ejemplo productos o servicios, o valores de referencia para determinar características o atributos de otros datos (dominios de atributos definidos por extensión). Se caracterizan por ser actualizables con poca frecuencia y por patrones de recuperación secuencial para búsquedas por aproximación, particularmente los dominios. Se experimenta con organizaciones secuenciales indexadas con organización de registros en bloques y agrupación por clave de recuperación principal (no por identificador de objeto sino por identificador semántico).

  Por ejemplo, considérese un archivo para registrar artículos en un negocio con la siguiente estructura lógica:

  ```
  Artículo((código)i, ((nombre)ie, (marca)ie)i, presentación(descripción,
          existencia, precio de venta unitario)*)
  ```

  podría organizarse secuencial indexado por nombre y marca, dados los requerimientos de acceso secuencial por este criterio de recuperación.

- *De Datos Transaccionales*: registros de hechos o eventos relacionados con datos maestros, por ejemplo de ventas de productos o de prestaciones de servicios. Se distinguen los hechos o eventos pasados, que en general no son actualizables, y los hechos o eventos programados a futuro, que pueden ser actualizables; para estos últimos es factible utilizar índices selectivos. Se caracterizan por patrones de acceso secuencial. Se experimenta con organizaciones secuenciales indexadas con organización de registros en bloques y agrupación en función de las dependencias de otros objetos.

  Por ejemplo, un archivo de facturas con la siguiente estructura lógica:

  ```
  Factura((número)i, fecha(año, mes, día), forma de pago('CO' | 'CH' | 'TD'
         | 'TC' | 'CC'), (referencia pago)?, descuento, (componente(código
         producto, cantidad, precio de venta unitario))+)
  ```

  sería un caso de eventos pasados, sin actualización, por lo que se puede prescindir de la organización en bloques. En cambio un archivo para registrar reservas y estadías en un hotel

  ```
  Estadía(((idH)ie, desde)i, hasta, (alojamiento((idHab)ie, ((idH)ie)*,
         (consumo((idServicio)ie, fecha, hora, precio, cantidad ))*)+)
  ```

  donde la estadía se identifica por el huésped responsable y la fecha de inicio, y puede involucrar más de una habitación, con uno o más huéspedes asignados (entre los que puede estar o no el responsable) y con servicios consumidos con cargo a la habitación, es un archivo actualizable cuyas operaciones críticas son las consultas de disponibilidad de habitaciones para reservar en función de dos fechas (se debe recorrer secuencialmente para descartar habitaciones ocupadas), el registro de reservas indicando el responsable y las

habitaciones, el check in (se registran todos los huéspedes y se asignan a las habitaciones, mediando o no reservas), el registro de consumos con cargo a una habitación, y el check out (se calcula lo que debe pagar el responsable de una estadía antes de retirarse). La organización más apropiada para este archivo sería la secuencial indexada con registros agrupados en bloques por fecha de inicio de estadía y huésped responsable.

- *De Reporte: información editada para su presentación al usuario (en general en formatos pdf, html o de texto). Organización secuencial.*
- *De Trabajo*: resultados parciales o intermedios de procesamiento, o datos de intercambio entre programas. Organización secuencial o directa según el caso.
- *De Control de Datos*: para almacenar metadatos (definiciones de datos), administrar espacios libres, registrar identificadores de registro vacantes o acceder al contenido de otro archivo (índices y tablas de acceso). Organizaciones secuencial, de árbol B+ o directas, según el caso.
- *De Intercambio de Datos*: para representar datos en formatos estándar de manera que puedan ser procesados libremente conociendo el estándar. Generalmente son archivos de texto, con alguna convención para rotular o delimitar datos, que pueden incluir o no definiciones sobre la estructura de la información contenida (un estándar actual es el XML: eXtended Markup Language). Organización secuencial.
- *De Unidades Grandes de Información:* imágenes, audio, vídeo.

En cuanto a los índices para el acceso a los objetos, se tiene en cuenta la siguiente clasificación:

- *Según la cantidad de referencias a registros de datos que acompañan a la clave en un registro de índice*
  - De identificación o clave única: cada clave refiere a un único registro (el índice sirve para identificar registros dentro de un archivo):
    `Indice(clave, referencia)`
  - De clasificación o clave redundante: cada clave refiere a una colección de registros (el índice sirve para clasificar –determinar subconjuntos de pertenencia– registros dentro de un archivo):
    `Indice(clave, (referencia)*)`
- *Según el índice refiera a todos o a parte de los registros de un archivo*
  - Exhaustivo: todos los registros del archivo están referidos en el índice
  - Selectivo: sólo algunos registros del archivo están referidos, ya sea porque la inserción de registros en el índice está condicionada (índice selectivo condicional) o sea porque cada registro de índice refiere a una secuencia de registros ordenados (índice selectivo de agrupación)
- *Según el tipo de referencias que acompañan a las claves*
  - Primario o de referencias directas a registros: las referencias son a posiciones físicas de registros o de bloques de registros
  - Secundario o de referencias indirectas a registros: las referencias son claves de identificación principales de registros
- *Según la jerarquía o importancia del índice*
  - Principal: primario de identificación. Todo archivo indicado debe tener obligatoriamente uno para verificar la unicidad de los registros en el archivo, y la clave de este índice es la que se utiliza en otros archivos como clave foránea para referir a los registros del archivo indicado.

  - Alternativo: provee un medio para acceder a registros por una clave distinta a la principal.

- *Según el número de posibilidades de acceso o búsqueda de registros*
  - De acceso único: los registros tienen una única clave de acceso, sea simple (con un solo atributo) o compuesta (con varios atributos)
  - De accesos alternativos: los registros tienen más de una clave de acceso (en general, son índices principales y selectivos de agrupación sobre archivos secuenciales ordenados; por ejemplo, para un archivo de facturas, se puede definir un índice que tenga como claves de acceso alternativas el número de factura y la fecha de la factura, que sólo indique la primer factura de cada fecha).

La notación para las definiciones lógicas se basa en la siguiente convención. Para cada atributo se indica la identidad, mediante un nombre, la estructura, listando los atributos componentes entre paréntesis, y la cardinalidad, mediante los calificadores

? : opcionalidad (0 ó 1 valor)
* : ninguno o varios valores (* seguido de un número implicaría ninguno o hasta ese número de valores)
+ : uno o varios valores (+ seguido de un número implicaría uno hasta ese número de valores)

También se especifican valores por extensión, cuando fuera pertinente, como en el caso

```
forma de pago('CO' | 'CH' | 'TD' | 'TC' | 'CC')
```

(**CO**ntado, **CH**eque, **T**arjeta de **D**ébito, **T**arjeta de **C**rédito o **C**uenta **C**orriente) donde la barra vertical (|) indica alternativa excluyente.

Y también se indica cualesquiera combinaciones de atributos que puedan identificar un objeto en el sistema de información, como en el caso de los artículos, con el calificador `i`:

```
(código)i, ((nombre)ie, (marca)ie)i
```

El calificador `ie` significa identificador externo.