Comparación de Objetos Complejos Multiversión: Una Solución Paralela

J-P Delahaye[2], G. Jomier[2], I. Meléndez[1], I. Ramos[1], M. Rukoz[1]
[1] Centro de Computación Paralela y Distribuida
Facultad de Ciencias, Universidad Central de Venezuela, Venezuela, e-mail: mrukoz@conicit.ve
[2] Laboratoire LAMSADE
Université Paris- Dauphine, France e-mail: jomier@lamsade.dauphine.fr

Resumen

La existencia de versiones en una base de datos (VBD) permite representar un número cualquiera de estados del universo modelado y no sólo uno como en el caso de las bases de datos mono-versión clásicas. El manejo de diversos estados del universo, como por ejemplo la comparación de dos de ellos, necesita tratamientos que pueden involucrar un gran número de versiones de objetos, para los cuales la implantación de algoritmos paralelos parece una solución interesante. En este articulo estudiamos la comparación de dos objetos complejos multiversión, considerando objetos cuya estructura es un árbol. Proponemos una solución paralela sobre una máquina MIMD que permite no sólo indicar si los dos objetos son iguales ó no, sino establecer toda la diferencia entre ellos a través de un árbol de comparación.

1.- Introducción

El problema de versiones en las bases de datos es el centro de todos los problemas que tienden a tomar en cuenta la evolución de las aplicaciones [DL88]. La existencia de versiones en una base de datos (VBD) permite presentar un número cualquiera de estados del universo modelado y no sólo uno como en las bases de datos mono-versión clásicas. La manipulación de una base de datos multiversión se realiza con la ayuda de operaciones complejas. Por ejemplo, muchas veces es necesario comparar dos versiones de entidades, ya que ellas representan dos estados posibles de la misma entidad del mundo real. Esta operación puede ser muy pesada cuando la profundidad del grafo inducido por la relación de referencia entre versiones de objetos es elevada. De manera general la comparación de dos VBD, es decir de dos estados posibles del mundo real y por lo tanto de un gran número de versiones de objetos dentro de dos VDB diferentes plantea un problema de eficacia. En efecto, esta operación es necesaria para poder fusionar dos VBD y poner en común, por ejemplo, el trabajo de dos concepciones de bases de datos.

En este articulo estudiamos la comparación de dos objetos complejos multiversión, considerando objetos cuya estructura es un árbol. La comparación en profundidad de dichos objetos requiere descender hasta el nivel de las hojas en el árbol lo cual implica un costo considerable en tiempo de ejecución. Después de definir y analizar la comparación de dos objetos complejos multiversión, proponemos una solución paralela sobre una máquina MIMD que permita no sólo indicar si los dos objetos son iguales ó no sino establecer toda la diferencia entre ellos a través de un árbol de comparación. Este árbol de comparación permitirá eventualmente responder rápidamente a todas las preguntas del usuario en cuanto a diferencias y similitudes de los objetos comparados.

El articulo esta estructurado de la manera siguiente: en la sección 2, exponemos los aspectos teóricos necesarios para la comprensión del mismo, como son el de base de datos multiversión, objetos complejos multiversión, y las estructuras de datos necesarias para la

manipulación de los mismos. En la sección 3, presentamos la comparación de objetos multiversión. En la sección 4, damos una solución paralela para la comparación de objetos multiversión, para ello presentamos las características relevantes de la arquitectura sobre la cual fue implantada dicha solución. Por último en la sección 5 damos las conclusiones de este trabajo.

2.- Aspectos Teóricos

La manipulación de versiones responde a necesidades de numerosas aplicaciones. Puede ser necesario por ejemplo, representar la evolución de las entidades en el tiempo, disponer de muchas copias de una entidad que permitan concebirla por aproximaciones sucesivas, o modelar muchas representaciones posibles de una entidad.

El enfoque de Versiones de Bases de Datos (VBD) [CJ90], se distingue de los otros al proponer una realización eficaz en el caso general, de la mañipulación de las versiones. El usuario manipula siempre una versión de las entidades en su contexto, representado por una VBD. Cada VBD contiene las versiones de las diferentes entidades que "van juntas" y representa una versión del universo modelado. El conjunto de VBD constituye la base de datos multiversión. Es posible entonces efectuar operaciones de base (lectura, creación, actualización, eliminación) tanto sobre las versiones de las entidades como sobre las VBD.

A continuación presentamos las estructuras de datos, asi como la definición de objetos complejos multiversión usados en este trabajo. Cada objeto perteneciente a cualquier versión está compuesto por:

Objetos Atómicos: son objetos tales como enteros, string, caracteres.

Objetos Complejos: son objetos que se componen de objetos atómicos, aplicándoles constructores de tipos o clases. Entre estos constructores se encuentran: N-uplas y Conjuntos. Cada objeto complejo tiene su propia identidad, representada por un identificador único (Oid) dentro de una Versión específica, es decir, no existen Oid's con el mismo valor dentro de una misma Versión. Por lo tanto, cada objeto complejo se representa con el par (Oid,Vid) donde Vid es el identificador de una versión. Los Oid's son generados por el sistema y se mantienen invariables a través del ciclo de vida de la Base de Datos.

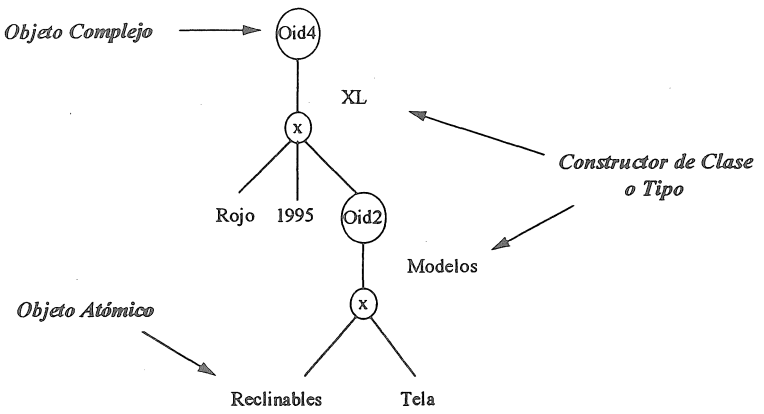
Los constructores de objetos deben ser ortogonales, es decir, cualquier constructor puede aplicarse a cualquier objeto. Esto significa que el constructor de clase Conjunto o el constructor de clase N-upla puede ser aplicado tanto a elementos atómicos como a objetos complejos (N-uplas o conjuntos). Esto define la recursividad en los tipos de constructores; por ejemplo:

N-upla[Conjunto{entero}, string] o también

Conjunto{N-upla[string, N-upla[entero, Conjunto{string}]]}.

donde: [ ] denota una n-upla y { }denota un conjunto

Un objeto complejo se puede representar como un árbol general como se muestra en la siguiente gráfica, en donde existen tres tipos de nodos: los nodos redondos con un identificador (Oid), denotan un objeto complejo, estos nodos son etiquetados con el nombre del constructor de la clase o tipo. Los nodos redondos con "x" y "*", los cuales indican el constructor de tipo, n-upla y conjunto respectivamente, del objeto complejo; y los nodos string, que denotan objetos atómicos: enteros, reales ó cadenas de caracteres.



Para almacenar la información relacionada a cada objeto, de acuerdo a su constructor de tipo o clase, se disponen de tres tablas: la tabla de objetos, la tabla de n-uplas y la tabla de conjuntos.

Tabla de Objetos: Mantiene la información de cada objeto perteneciente a las versiones de bases de datos que se estén tratando. Se almacenan únicamente los objetos complejos.

Nombre	Identificador del Objeto Complejo
Clase	Clase a la cual pertenece el Objeto
Constructor	Constructor de clase (N-upla o Conjunto)
Versión	Versión a la cual pertenece el Objeto

Tabla de N-uplas: Mantiene la información de todos los elementos o atributos de cada objeto complejo cuyo constructor de clase es N-upla.

Nombre	Identificador del Objeto Complejo
Clase	Clase a la cual pertenece el Objeto
Versión	Versión a la que pertenece el Objeto
Valor	Valor del elemento o atributo
Tipo	Tipo del elemento (N-upla, Conjunto, string, entero)

Tabla de Conjuntos: Mantiene la información de todos los elementos o atributos de cada objeto complejo cuyo constructor de clase es Conjunto.

Nombre	Identificador del Objeto Complejo
Clase	Clase a la cual pertenece el Objeto
Versión	Versión a la que pertenece el Objeto
Valor	Valor del elemento o atributo
Tipo	Tipo del elemento (N-upla, Conjunto, string, entero)

3.- Comparación de Objetos Complejos Multiversión

La comparación de dos objetos complejos multiversión permite conocer cuales son los cambios que ha sufrido una misma entidad del mundo real entre dos versiones diferentes. Para esto es necesario comparar un mismo objeto dentro de dos VBD diferentes, lo cual puede significar la comparación de dos objetos de identificadores diferentes dentro de una misma VBD y/o dos objetos de identificadores diferentes dentro de dos VBD diferentes según los objetos que componen a los objetos complejos a comparar. La comparación debe responder de manera eficaz a la diversidad de las necesidades de interrogación del usuario teniendo en cuenta el modelo de datos del enfoque de Versiones de Bases de Datos..

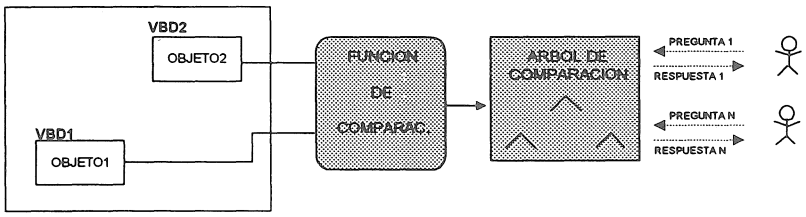
Existen diversos conceptos que resaltan en el proceso de comparación. En lo que sigue se describirán cada uno de ellos [Delahaye92].

La igualdad :Comparar el contenido de dos objetos, es decir, comparación por valor. Si los valores de los atributos son iguales, los objetos son iguales. La igualdad de dos objetos complejos puede ser en Superficie o en Profundidad. En Superficie significa que no se desciende dentro del árbol representado por el atributo complejo a fin de reemplazar las identidades de los objetos por su valor. Sólo se toma en cuenta el identificador. En profundidad significa que se reemplaza todos los identificadores de objetos por sus valores o atributos descendiendo dentro del árbol representado por el atributo complejo y sus elementos.

La identidad :Comparar la identidad de dos objetos, lo cual significa realizar una comparación por valor sobre el identificador de los objetos. Su igualdad significa que los objetos comparados son el mismo objeto. Dos objetos cualesquiera Oid1 y Oid2 existentes respectivamente en VBDx y VBDy son idénticos si y sólo si:

(Oid1,identificador de VBDx) = (Oid2,identificador de VBDy).

Función de Comparación: La función de comparación de versiones retorna una estructura de datos, denominada árbol de comparación, la cual especifica la información que el usuario desea obtener. El objetivo de esta estructura es ser la fuente de información para responder a las preguntas sucesivas del usuario evitando así solicitar accesos físicos a las versiones de la base de datos cada vez que el usuario solicite una nueva información sobre la comparación realizada [Delahaye92]. Esta interacción se muestra en la siguiente figura



La idea principal es conservar de la comparación tanta información como sea posible, a fin de minimizar las necesidades de acceder a los discos para recuperar una información útil para otras visiones de la comparación.

Argumentos de la función de comparación: Los parámetros de entrada de la función de comparación son:

- El Identificador del primer objeto (objeto 1) a comparar

- La VBD seleccionada para el objeto 1
- Identificador del segundo objeto (objeto 2) a comparar
- La VBD seleccionada para el objeto 2

Proceso de Comparación: La Comparación de dos objetos pertenecientes a dos VBD diferentes depende de su tipo, así:

Comparación de objetos atómicos : La igualdad de dos objetos atómicos se determina por el valor de cada Objeto.

Comparación de objetos tipo N-upla: Para los objetos cuyo constructor de clase es N-upla, la comparación se determina tratando objetos simples (atómicos) o complejos de la misma clase. La comparación puede ser en Superficie o en Profundidad.

Superficie: Los objetos a comparar deben tener el mismo constructor de clase; de no ser así el proceso de comparación finaliza. Si poseen el mismo constructor se debe tener en cuenta lo siguiente:

- Los objetos deben tener el mismo número de elementos. En caso de ser diferentes el proceso termina.
- Se verifica el tipo de cada elemento. Cada uno de estos elementos deben ser de igual tipo que su correspondiente en cada objeto; si al menos uno es diferente la comparación termina para el subárbol tratante. En caso de ser iguales los tipos de cada elemento entonces verifica la igualdad de sus elementos si éstos son iguales se dice que los objetos son iguales en superficie.

Profundidad: Se comparan los objetos tomando en cuenta los atributos o elementos con sus valores. Los objetos a comparar deben tener el mismo constructor de clase; en caso de no serlo el proceso de comparación en el subárbol que se esté tratando finaliza. Si poseen el mismo constructor se debe tener en cuenta lo siguiente:

- Los objetos deben tener el mismo número de elementos. En caso de ser diferentes el proceso de comparación por el subárbol tratante termina.
- Se verifica el tipo de cada elemento. Cada uno de estos elementos deben ser de igual tipo que su correspondiente en cada objeto; si al menos uno de estos elementos es diferente la comparación por el subárbol tratante termina. En caso de ser iguales los tipos de cada elemento se verifica la igualdad de los valores de cada uno de ellos.
- Si al menos uno de los elementos es otro objeto complejo se desciende en el árbol representado por el mismo, y se repite el mismo procedimiento de comparación, hasta que se comparen solo objetos atómicos.
- Al encontrar que los valores y los tipos de los elementos de los objetos son iguales se dice que los objetos son iguales en profundidad. Si al menos difieren en uno, la comparación por el subárbol recorrido termina.

Comparación de objetos tipo Conjunto

Para los objetos cuyo constructor de clase es conjunto, la comparación se determina tratando objetos simples (atómicos) o complejos de la misma clase. La comparación puede ser en Superficie o en Profundidad.

Superficie: Los objetos a comparar deben tener el mismo constructor de clase; en caso de no serlo el proceso de comparación finaliza por el subárbol tratante. Si poseen el mismo constructor se comparan los elementos de los conjuntos, si estos son iguales se dice que los objetos son iguales en superficie.

Profundidad: Los objetos a comparar deben tener el mismo constructor de clase; de no ser así el proceso de comparación para el subárbol tratante finaliza. Si son del mismo tipo, el valor de cada elemento de un objeto se compara con los valores de cada uno de los elementos del otro objeto, considerando la igualdad de los tipos de cada elemento. Si los elementos de un objeto se encuentran en el conjunto formado por el otro objeto, se dice que los objetos son iguales en profundidad.

Cabe destacar que si al menos un elemento de cualquier objeto es otro objeto complejo se desciende en el árbol representado por el mismo, y se repite el mismo procedimiento de comparación, hasta que se comparen sólo objetos atómicos. Al encontrar que los valores y tipos de cada uno de los elementos de los objetos son iguales se dice que éstos son iguales en profundidad. Si al menos difieren en un elemento el proceso de comparación por el subárbol tratante termina.

En lo que sigue consideraremos sólo la comparación en profundidad.

Como se ha mencionado, la función de comparación de versiones de bases de datos retorna un árbol de comparación, el cual es el resultado de comparar dos Versiones de Bases de Datos. La representación gráfica del árbol de comparación utiliza dos tipos de nodos: los nodos redondos con "x" y "*", los cuales denotan el tipo del objeto complejo (n-upla y conjunto respectivamente), y los nodos cuadrados, los cuales denotan los tipos atómicos: enteros, reales, cadenas de caracteres. Los objetos complejos son representados por un redondo conteniendo el símbolo Oid, etiquetado con el nombre de la clase correspondiente.
Sea la VBD4 con las siguientes clases:

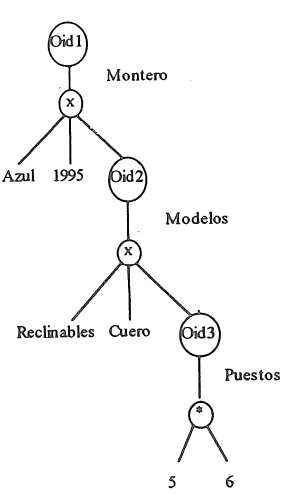
add class Montero
type n-upla (Color:**String**, Año:**Entero**, Asientos:**Modelos**)
add class Modelos
type n-upla (Tipo:**String**, Material:**String**, Ptos:**Puestos**)
add class Puestos
type conjunto (P:**Entero**)

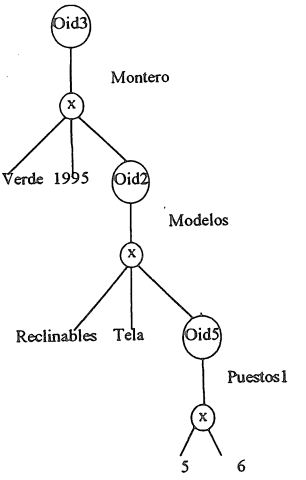
Sea la VBD5 igual a VBD4 excepto la clase Puestos que será sustituida por:
add class Puestos1
type N-upla (Cinco:**Entero**, Seis:**Entero**)

La Figura 1 muestra dos objetos pertenecientes a las versiones VBD4 y VBD5 respectivamente. Al aplicar el proceso de comparación descrito anteriormente se obtiene el árbol de comparación mostrado en la misma Figura.

4.- Una solución Paralela para la Comparación de Objetos Complejos Multiversiones

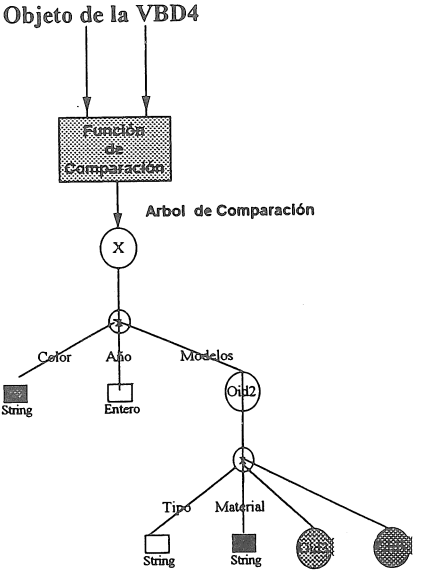
En esta sección estudiaremos la función de comparación bajo un ambiente paralelo, de forma tal de distribuir el proceso de comparación en varios procesadores, dependiendo de la topología paralela, para así agilizar dicho proceso y obtener menores tiempos de respuestas a las diversas consultas que realicen los usuarios.





Objeto de la VBD4

Objeto de la VBD5



Donde:

- Atributo Atómico, donde el valor de los objetos es diferente entre las dos versiones
- Atributo Atómico, donde el valor de los objetos es igual entre las dos versiones
- Objeto específico a la VBD4
- Objeto específico a la VBD5

Ambiente Paralelo

El ambiente paralelo considerado en este trabajo es la máquina Parsytec Multicluster-3DE (MC-3DE), la cual posee una topología de malla basada en Transputers T800 [Parsytec92]. Esta máquina está conectada a una estación de trabajo SUN Sparc Station/2. El proceso de comparación de versiones fue desarrollado y ejecutado en ésta máquina sobre ocho procesadores, no obstante está diseñado para operar sobre máquinas paralelas con n procesadores.

Para alcanzar un buen rendimiento y modelar en forma apropiada la aplicación paralela, los procesos necesitan ser distribuidos entre los procesadores de manera tal que el tiempo de comunicación entre ellos sea el menor posible. Una distribución no adecuada de los procesos puede afectar el rendimiento total de la aplicación. La topología física de interconexión de la máquina MC-3DE es una malla de m x n procesadores. Por lo tanto, existen a lo sumo cuatro conexiones o enlaces por procesador. Dada la limitación en cuanto a las conexiones físicas, el tratamiento de un árbol general de un objeto complejo multiversión con más de k hijos por nodo del árbol ($k \geq 1$), será distribuido en la malla de procesadores en forma de su árbol binario correspondiente, lo que hace que la representación y manipulación de los objetos en cada procesador se simplifique.

Arbol binario asociado a un objeto complejo

A un objeto complejo se le asociará un árbol binario que permitirá por una parte asignar los procesadores y por otra controlar la ejecución del proceso de comparación. Este árbol binario se construirá haciendo un recorrido inorder [Knuth76] del árbol general. Cada nodo del árbol binario contendrá un objeto componente del árbol general del objeto. La raíz del árbol binario contiene el el identificador del objeto complejo, su subárbol izquierdo contiene el primer componente del mismo y su subárbol derecho contiene el resto de los elementos. Cada subárbol está formado de la siguiente manera:

La raíz contiene el objeto raíz del subárbol dado. Si el objeto es atómico su subárbol izquierdo contiene sus hermanos y su subárbol derecho es vacio. Si el objeto es un identificador de objeto complejo, su subárbol izquierdo contiene todos los elementos que lo componen (sus hijos) y su subárbol derecho contiene sus hermanos.

Asignación de Procesadores

La asignación de procesadores implica distribuir el árbol binario descrito anteriormente en la malla de procesadores. El proceso de asignación lo realizan los procesadores entre sí cada vez que lo requieran a medida que se va construyendo el árbol binario correspondiente. Antes de describir el algoritmo de asignación de procesadores, es importante definir el procesador, denominado raíz, y los estados de los procesadores. Denotaremos con P[i,j] al procesador que ocupa la fila i y la columna j dentro de la malla de procesadores.

El procesador raíz, va a contener el objeto complejo que representa el primer nodo identificador (Oid) del árbol. De acuerdo a las dimensiones de la malla m x n, el procesador raíz será el procesador que ocupa la posición indicada en la siguiente tabla:

Mallas cuadradas impares	P[(n-1)/2, (n-1)/2]
Mallas cuadradas pares	P[n/2, n/2]
Mallas no cuadradas	P[m-1/2, (n-1)/2] con m y n impar P[(m/2)-1, (n/2)-1] con m y n par P[(m/2)-1, (n-1)/2] con m par y n impar P[(m-1)/2, (n/2)-1] con m impar y n par

Un procesador puede estar en uno de los siguientes estados: ***ocioso*** cuando no está procesando ninguna información determinada y ***activo*** cuando está procesando alguna información.
Llamaremos **procesador adyacente derecho** del procesador raíz P[i,j] al procesador P[i,j+1] ó P[i+1,j] ó P[i,j-1] ó P[i,j] en ese orden de acuerdo a la disponibilidad o existencia del mismo. De la misma manera llamaremos **procesador adyacente derecho** de un procesador P[i,j] diferente al procesador raíz, al procesador P[i-1,j].ó P[i,j-1] ó P[i,j] en ese orden de acuerdo a la disponibilidad o existencia del mismo.
Similarmente llamaremos **procesador adyacente izquierdo** del procesador raíz P[i,j] al procesador P[i-1,j] ó P[i,j-1] ó P[i+1,j] ó P[i,j] en ese orden de acuerdo a la disponibilidad o existencia del mismo. Llamaremos **procesador adyacente izquierdo** de un procesador P[i,j] diferente del procesador raíz al procesador P[i+1,j] ó P[i,j-1] ó P[i,j+1], ó P[i,j] en ese orden de acuerdo a la disponibilidad o existencia del mismo.

Proceso de asignación de procesadores:

*Un proceso, cuando reciba un árbol representando a un objeto

Si es el proceso que está en el procesador raíz, almacena la información del nodo raíz del objeto.

Toma el subárbol más a la izquierda

Si quedan subárboles por procesar los envia al procesador adyacente derecho.

Del subárbol tomado, recorre la rama más a la izquierda hasta encontrar el primer objeto atómico. Por cada nodo recorrido si no está marcado, lo marca y almacena su información, si está marcado lo ignora. El objeto atómico es almacenado y eliminado del árbol. Si en el subárbol recorrido aún quedan nodos sin marcar éste es enviado al procesador adyacente izquierdo.

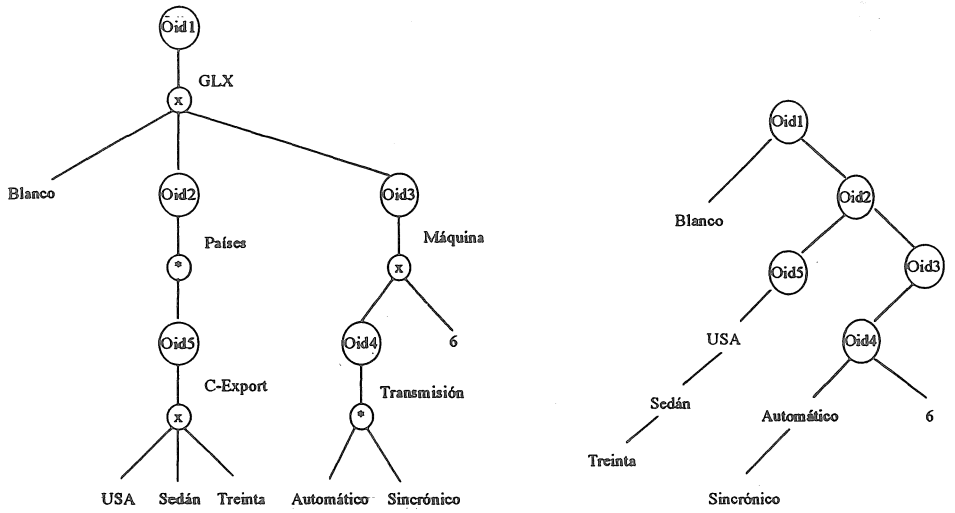
Cabe acotar que estos criterios se realizan validando que los procesadores a asignar ocupen una posición existente dentro de la dimensión de la Malla y que además se encuentren ociosos. Cuando no existan procesadores adyacentes ociosos, el procesador solicitante realizará el procesamiento de los datos a través de procesos internos o threads.

Control de ejecución de la comparación

* Cuando un proceso ha terminado de construir su **respuesta** la envía a su padre.
* Cuando un nodo hoja termina de comparar sus elementos envía la respuesta a su padre
* Cuando un proceso distinto al raíz recibe una respuesta de comparación:
  * Si el contiene un elemento atómico su **respuesta** es el par (respuesta recibida, su resultado)
  * Si el contiene un identificador de un elemento complejo su **respuesta** es el par (composición de acuerdo a su tipo de constructor de la respuesta recibida de su subárbol izquierdo, respuesta de su subárbol derecho)
* Cuando el procesador raíz recibe una respuesta de comparación, su **respuesta** es la composición, de acuerdo a su tipo de constructor, de la respuesta recibida.
  Si su respuesta ha sido completamente construida el proceso termina

A continuación se presenta un ejemplo del proceso de asignación en la Parsytec MC-3DE con una malla de 4x2, donde se distribuirá un objeto perteneciente la siguiente Versión:

add class GLX
type n-upla (Color:**String**, País_Exporta:**Países**, Tipo:**Máquina**)
add class Países
type conjunto (export: **C-Export**)
add class C-Export
type n-upla (Nombre:**String**, Modelo:**String**, N°_Autos:**String**)
add class Máquina
type n-upla (Tipo_T:**Transmisión**, Cilindros:**Entero**)
add class Transmisión
type conjunto (Trans1:**String**, Trans2:**String**)



Árbol Binario

La información relacionada a cada objeto complejo y atómico en sus tablas respectivas es la siguiente:

Tabla de Objetos:

Nombre	Clase	Constructor	Versión
Oid1	GLX	N-upla	10
Oid2	Países	Conjunto	10
Oid3	Máquina	N-upla	10
Oid4	Transmisión	Conjunto	10
Oid5	C-Export	N-upla	10

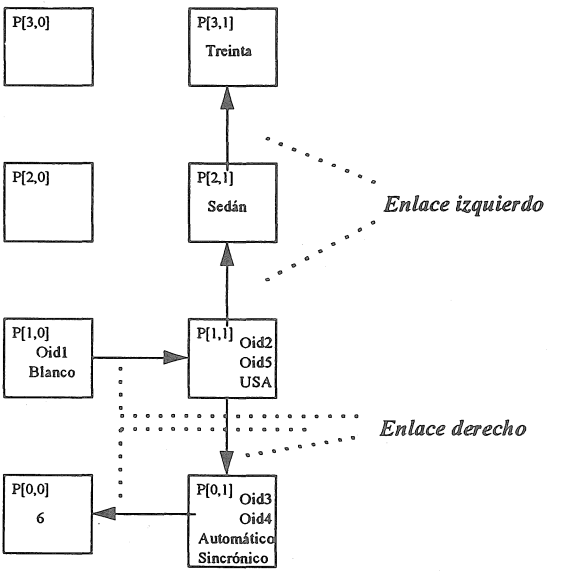
Tabla de Conjuntos:

Nombre	Clase	Versión	Valor	Tipo
Oid2	Países	10	Oid5	String
Oid4	Transmisión	10	Automático	String
Oid4	Transmisión	10	Sincrónico	String

Tabla de N-uplas:

Nombre	Clase	Versión	Valor	Tipo
Oid1	GLX	10	Blanco	String
Oid1	GLX	10	Oid2	Conjunto
Oid1	GLX	10	Oid3	N-upla
Oid3	Máquina	10	Oid4	Conjunto
Oid3	Máquina	10	6	Entero
Oid5	C-Export	10	USA	String
Oid5	C-Export	10	Sedán	String
Oid5	C-Export	10	Treinta	String

Al distribuir en la malla el objeto en forma de árbol binario, se obtiene el siguiente resultado:



Implementación Paralela

El sistema se desarrolló bajo el Paradigma Cliente/Servidor, en donde para establecer una comunicación entre estos dos entes se necesita una Interfaz de Programas de Aplicación (API), denominada Socket. Los diversos objetos de las versiones de la base de batos a utilizar son almacenados en disco.

Procesos definidos Para la implementación del sistema se va a disponer de dos tipos procesos: un proceso Servidor, residente en la SUN Sparc Station/2, y un proceso Cliente, el cual se ejecuta en

cada uno de los ocho procesadores que conforman la Parsytec MultiCluster-3DE. Entre estos procesos va a existir una comunicación bidireccional, mediante un Socket.

- Proceso Servidor: *Función*: Procesar las peticiones realizadas tanto por el Usuario como por el proceso Cliente. El Usuario se comunica con el proceso Servidor para solicitar la comparación de dos objetos complejos. El proceso Servidor establece una comunicación con el proceso Cliente para indicarle que inicie el proceso de comparación. Una vez obtenido los resultados, el Servidor se los retorna al Usuario.
- Proceso Cliente: *Función*: Realizar el proceso de comparación de versiones de bases de datos.

El proceso Cliente le solicita al proceso Servidor la información necesaria durante el proceso de comparación, con el fin de satisfacer requerimientos futuros que realice el Usuario. La ejecución del proceso Cliente implica la conexión o enlace entre los procesadores, de acuerdo al tipo de datos que le sea suministrado por parte del Servidor (Objeto Complejo y/o Objeto Atómico). A medida que esto ocurra se irá realizando la comparación y dependiendo del resultado obtenido se establecerá una comunicación con los otros procesadores, para enviar dicho resultado. El resultado final, es decir, el árbol de comparación, será construido por el procesador que ha sido asignado como Raíz, una vez que reciba todos los resultados de sus procesadores adyacentes.

5.- Conclusiones

Hemos presentado un algoritmo paralelo de comparación de objetos complejos multiversión. Este algoritmo permite no sólo indicar si dos objetos son iguales o no sino que además genera un árbol de comparación donde se establecen todos los puntos de diferencias entre los objetos comparados.

La implantación del algoritmo sobre una máquina paralela con topología de malla requiere tomar en cuenta los enlaces físicos de la misma. Para ello, se propuso un algoritmo de asignación de procesadores que se basa en el árbol binario correspondiente a los arboles de los objetos complejos a comparar. Este algoritmo de asignación puede ocasionar que existan procesadores con mas información que otros (ver ejemplo de la sección 4). Actualmente se está realizando un una evaluación de nuestro algoritmo de comparación de objetos complejos multiversión para diversas estructuras de objetos complejos.

Referencias

[CJ90] Cellary W, y Jomier G. Consistency of versions in object-oriented databases. In Proc. 16th VLDB, Brisbane (Australia), 1990.

[Delahaye92] Delahaye J-P. Gestion de version de bases de données conception et maquettage des fonctions de comparaison et de fusion de versions de base de données". Rapport de stage Université Paris-Dauphine Septembre 92.

[DL88] Dittrich K. y Lorie R.. Version support for engineering database systems. IEEE Transaction on Software Enginering Vol. 14 N4, pp 429-437.

[GJ95] Gancarski y Jomier Ä framework for the Manipulation of a Multiversion Database. DEXA'95 International Conference, Workshop proc. pp 247-256. London (U.K.) Septiembre 1995.

[Gancarski94] Gancarski S. Versions et bases de données: modèle formel, supports de langage et dínterface-utilisateur. PhD Thesis, Université Paris 11, Dic. 1994.

[Knuth76] Knuth D. Fundamental Algorithms. The art of computer programming. Addison Wesley, 1976.

[Parsytec92] Parsytec Computer GmgH. PARIX Documentaction. Release 1.1, Septiembre 1992.