Fragmentación vertical de clases en las bases de datos distribuidas orientadas a objetos

M.Sc. Elzbieta Malinowski Gajda[*]

---

* M.Sc. en Computación (San Petersburgo y Florida), Profesora en la Escuela de Ciencias de la Computación e Informática, UCR.

RESUMEN

El diseño de bases de datos distribuidas es un proceso complicado que incluye muchos factores para considerar. Además, siendo las bases de datos distribuidas orientadas a objetos una generalización de bases de datos relacionales, no sólo heredan los problemas de diseño presentes en bases de datos distribuidas relacionales, sino también presentan los problemas adicionales relacionados con el esquema que está compuesto por las clases. Este artículo presenta la descripción de las nuevas matrices desarrolladas para las bases de datos orientadas a objetos que son generalizaciones de las matrices usadas en el proceso de fragmentación en las bases de datos distribuidas relacionales. Además, se propone el esquema de fragmentación para las bases de datos orientadas a objetos de diferentes niveles de complejidad.

1. INTRODUCCIÓN

El interés generado por la automatización de procesamiento de datos dio inicio al desarrollo de bases de datos relacionales (BDR). Desde los años setentas, cuando Codd [Codd94] introdujo este concepto de almacenamiento de datos, creció el desarrollo de los sistemas que permiten su manejo eficiente (DBMS, *database management systems*, sistemas de manejo de bases de datos).

Además, los avances en la tecnología de micro-procesadores y el amplio uso de redes de computadoras impulsaron la investigación sobre la posibilidad de distribuir los datos en la red usando diferentes criterios como, por ejemplo, las frecuencias de utilización de los datos. Estas investigaciones dieron inicio a las así llamadas bases de datos distribuidas relacionales (BDDR).

El objetivo de las BDDR es mejorar la eficiencia de su desempeño físicamente distribuyendo los datos de acuerdo a los requerimientos de uso y las capacidades computacionales que disponen los diferentes usuarios, pero manteniendo la visión del sistema como un solo componente. Se desarrollaron diferentes algoritmos de distribución de datos y se implementaron las técnicas de fragmentación de dos tipos: fragmentación horizontal, donde se distribuyen las tuplas (registros) de las tablas de la base de datos entre diferentes nodos de la red o fragmentación vertical donde se distribuyen los atributos de acuerdo a la frecuencia de su uso. Se puede también usar la fragmentación híbrida que incluye una o varias secuencias de aplicación de las fragmentaciones anteriormente mencionadas.

Este creciente desarrollo de BDDR disminuyó su velocidad cuando se intensificó el uso de programación orientada a objetos. No solo los lenguajes de programación, sino también las bases de datos, encontraron las utilidades y facilidades de desarrollo por medio de este paradigma. Sin embargo, los científicos encontraron la inquietud sobre la posibilidad de fragmentar los objetos. Estas nuevas formas de almacenamiento no son fáciles de fragmentar debido a su compleja estructura. Para asegurar el diseño óptimo tanto en BDDR como en bases de datos distribuidas orientadas a objetos (BDDOO) de acuerdo a Özsu y Valduriez se necesita la siguiente información [Özsu94]:

- Esquema global de la base de datos.
- Predicados de uso para la fragmentación horizontal o matriz de uso de atributos por diferentes transacciones y las frecuencias de las transacciones para la fragmentación vertical.
- Costos de comunicación en la red.
- Las computadoras usadas.

Para los aspectos de la fragmentación se considera los dos primeros puntos. Debe ser evidente que para BDDOO el esquema conceptual puede ser mucho más complejo que para BDDR.

Aunque existe una diferencia significativa entre estos dos tipos de bases de datos, debe ser claro que el enfoque relacional puede ser visto (ignorando por la simplicidad la existencia de los métodos) como un caso especial de los sistemas orientados a objetos sin la jerarquía de clases y atributos complejos (atributos que contienen la referencia a otros objetos). Si este es el caso, los algoritmos desarrollados para BDDR con respecto a la distribución de los datos podrían ser generalizados y aplicados para BDDOO. Uno de los ejemplos en que es posible esta generalización es el evaluador de particiones desarrollado para la BDDR [Chak93] y generalizado para la BDDOO [Mali96].

Uno de los componentes muy importantes usados en el desarrollo de los algoritmos de fragmentación presentados para BDDR es la matriz del uso de los atributos (AUM, *attribute usage matrix*) o la matriz de la afinidad de los atributos (AAM, *attribute affinity matrix*) que en general reflejan la información de las aplicaciones que operan sobre los datos. Para poder generalizar los algoritmos de BDDR y aplicarlos a BDDOO se necesita analizar cuales son las matrices en ambiente orientado a objetos que corresponden a las matrices mencionadas anteriormente.

En el artículo se describen los aspectos considerados en la fragmentación en BDDR y BDDOO, mencionando los trabajos desarrollados en esta área. Posteriormente se justifica la necesidad del desarrollo de las nuevas matrices y se describe su presentación que permite "aplanar" la estructura jerárquica de las clases presentada en BDDOO. Finalmente, basándose en la información contenida en las matrices y los resultados de fragmentación, se presentan las formas de propagación de la fragmentación en la jerarquía de las clases.

2. REVISIÓN DE TRABAJOS RELACIONADOS CON LA FRAGMENTACION EN BASES DE DATOS RELACIONALES Y BASES DE DATOS ORIENTADAS A OBJETOS

Usando la fragmentación vertical en BDR se puede agrupar los atributos de cada relación en registros más pequeños para mejorar la eficiencia de las transacciones en las bases de datos. Sin embargo, como el número de las particiones posibles es muy grande en una BDR real [Nava84], en la mayoría de las investigaciones se usa un enfoque heurístico para fragmentar. Una de las suposiciones más usadas en este enfoque es que el particionamiento debe considerar que cada fragmento que contiene los atributos debe estar muy cerca de la transacción que lo ocupa [Nava84]. Esto significa que, en el caso ideal, el fragmento tiene solo los atributos que la transacción local necesita y esta transacción no necesita accesar ningún atributo de otro nodo remoto. Sin embargo, en BDDR esto es casi imposible de lograr. La mejor forma de partir los atributos para una transacción no es necesariamente la mejor forma para la otra transacción. La meta factible es maximizar el desempeño de todas las transacciones tomando en cuenta todas ellas como un solo grupo. Pues el diseñador debe hacer un compromiso, asegurando el acceso mínimo a los atributos remotos para todas las transacciones ejecutadas en cada uno de los sitios. El seguimiento de estos conceptos se puede ver en las investigaciones presentadas sobre la fragmentación vertical en BDDR.

Actualmente existe un número considerable de algoritmos para fragmentación vertical en BDR. Los trabajos presentados por Hoffer y Severance [Hoff75], Hammer y Niamir [Hamm79], Navathe et al. [Nava84], Cornell y Yu [Corn87], Navathe y Ra [Nava89], Chu y Ieong [Chu93] son muy conocidos por la sociedad de investigadores. Estos autores utilizan en sus algoritmos las matrices AUM o AAM. Sin embargo, en este artículo se enfatiza más en el enfoque presentado por Chakravarthy et al. [Chak93] siguiendo su argumentación de usar la matriz AUM en lugar de la matriz AAM. Chakravarthy et al. opinan que AAM solamente señala la "afinidad" entre las parejas de los atributos y las relaciones entre más de dos atributos no se pueden medir en esta matriz. En este artículo se apoya la opinión de los autores y se desarrolla la matriz que corresponde a AUM para el ambiente orientado a objetos. Se puede notar que en las investigaciones referentes a BDDOO no existe acercamiento de este tipo.

La investigación de BDOO abarca muchos aspectos, entre los cuales también se puede encontrar la fragmentación. Así, en el desarrollo de la fragmentación horizontal, Ezeife y Barker [Ezei94] analizan cuatro casos separados de atributos simples y métodos simples, atributos simples y métodos complejos, atributos complejos y métodos simples, atributos complejos y métodos complejos. Ellos tratan las fragmentaciones primarias y derivadas en su algoritmo de varios pasos e incluyen la definición del grafo que enlaza las clases (*link graph*). Finalmente, combinan estos fragmentos primarios y derivados, mientras tanto encuentran la mejor localización para los fragmentos.

Otro enfoque ha sido presentado por Karlapalem y Li [Karl94a] donde ellos tratan de establecer algunos esquemas comunes para la fragmentación en BDOO; no presentan ningún algoritmo específico, pero proponen algunos pasos iniciales para cada esquema de fragmentación. Los autores mencionados categorizan tres tipos de clases: basadas en valores, basadas en objetos y mixtas. Adicionalmente, ellos analizan las posibilidades de usar los esquemas de fragmentación vertical, horizontal y *path*. Para la fragmentación horizontal para las clases basadas en objetos, ellos distinguen los casos de fragmentación horizontal derivada (cuando las clases son fragmentadas basándose en el valor del objeto en la jerarquía) y la fragmentación horizontal asociativa (cuando las clases son fragmentadas debido a su división natural en subclases).

Un acercamiento diferente describen Gruber y Amsaleg [Grub94]. Ellos proponen un agrupamiento especial de los objetos basado en la importancia de las ligas, sin embargo, no lo enfocan hacia las BDOO. Especialmente, ellos analizan las ligas de jerarquía y las ligas entre diferentes objetos y basándose en esta información desarrollan el agrupamiento de los objetos, implementándolo en el sistema EOS. Ellos consideran solamente la necesidad de poner los objetos padres e hijos juntos sin analizar la posibilidad de diferente frecuencia del uso de las clases y subclases.

Karlapalem et al. [Karl94b] describen algunos problemas necesarios para considerar en el ambiente distribuido orientado a objetos, como son el modelo de datos usado, invocación de métodos importantes para los propósitos de localización, tipos de transparencia, "cercanía" de los métodos y objetos, herencia en la jerarquía, y otros. Ellos no presentan ningún algoritmo para la fragmentación y afirman que los algoritmos propuestos por Navathe et al. Para BBDR [Nava84] pueden ser usados para el caso del método simple y atributos basados en valores. Para los métodos complejos y atributos basados en valores ellos proponen "aplanar" los objetos y aplicar para el esquema desarrollado los conceptos desarrollados por Pernul at al. [Pern91] para la fragmentación con las vistas sobrepuestas.

Aunque los autores anteriormente mencionados consideran diferentes aspectos presentes en el ambiente de orientación a objetos, no es conocido por el autor de este artículo enfoque basado directamente en el desarrollo de la matriz AUM.

3. ESPECIFICACION DE LAS MATRICES PARA LA BDOO DISTRIBUIDA

En el enfoque relacional los algoritmos de la fragmentación vertical distribuyen los atributos. Sin embargo, en el enfoque orientado a objetos el esquema conceptual puede ser mucho más complejo y lejos de los factores usados en el enfoque relacional (frecuencias de las transacciones, uso de los atributos, tipo de predicado y otros). Adicionalmente deben ser considerados unos aspectos nuevos, como son los métodos, la estructura jerárquica y los atributos complejos.

3.1 Motivación para el desarrollo de las nuevas matrices

Se puede ver que en un enfoque orientado a objetos "puro" los atributos son accesados por los métodos y las transacciones invocan los métodos en lugar de atributos directamente. Así, se necesita las matrices que pueden reflejar esta situación como la matriz de uso transacción-método (TMUM, *transaction-method usage matrix*) y la matriz de uso método-atributo (MAUM, *method-attributes usage matrix*). También, se puede tener casos donde un método invoca otro método y esta información debería ser también recopilada por medio de la matriz de uso método-método (MMUM, *method-method usage matrix*). En consecuencia se debe considerar la existencia de nuevas matrices diferentes de las presentadas en el enfoque relacional.

Además, la matriz AUM usada en la fragmentación vertical (en los trabajos previos) en BDDR solamente considera el caso cuando las frecuencias de acceso de los atributos por una transacción específica son iguales para todos los atributos. Sin embargo, en el caso de BDDOO la matriz resultante que corresponde a la matriz AUM puede tener diferentes frecuencias de los atributos para la misma transacción; por ejemplo, en el caso de atributos complejos, cuando el objeto hacia el cual dicho atributo tiene referencia puede ser accesado

"indirectamente" por medio de este objeto complejo o directamente desde alguna otra transacción.

¿Por qué no fragmentar los métodos y después asignarlos a los fragmentos adecuados? La razón es obvia. Los métodos como las piezas de programas pueden ser fácilmente copiados o duplicados sin ningún problema (espacio, control de concurrencia y otros). La adecuada distribución de datos, la cual optimiza el espacio de almacenamiento (sin duplicación innecesaria) y da un desempeño bueno de las transacciones ejecutadas, es la meta principal del diseño en sistemas de bases de datos distribuidos, relacionales u orientadas a objetos.

Tomando en cuenta lo anterior se puede presentar las matrices más formalmente.

3.2 Presentación de matrices

La meta del diseño es obtener TAUM que corresponde a AUM en el enfoque relacional, la cual sirve de base de numerosos algoritmos de fragmentación. La única diferencia entre estas dos matrices consiste en que en TAUM existen diferentes frecuencias de acceso de atributos por la transacción y en AUM estas frecuencias son iguales.

Se consideran las siguientes suposiciones:

1. Se utiliza el enfoque basado en objetos no valores.

2. Se tiene la información sobre las clases incluyendo los métodos y los atributos que estos métodos usan.

3. Se accesan los atributos solamente usando los métodos de acuerdo a los principios de encapsulamiento.

4. Se dispone de la información sobre acceso de los métodos por las transacciones.

5. Se especifica las frecuencias de acceso de transacciones.

6. Se ignora la existencia de los tipos tupla, conjunto y lista.

Tomando en cuenta las consideraciones anteriores, se puede presentar la información necesaria para el particionamiento en la forma de TMUM, MMUM, MAUM.

1. *Transaction-Method Usage Matrix* (TMUM): la matriz de frecuencias que indica si la transacción invoca un método específico. Los valores usados de cero o uno representan la situación que la transacción no llama o llama el método respectivamente:

trans/método	$m^1_{i,j}$	$m^2_{i,j}$	...	$m^m_{i,j}$
tr_1	$fm_{1,1}$	$fm_{1,2}$	...	$fm_{1,m}$
tr_2	$fm_{2,1}$	$fm_{2,2}$	...	$fm_{2,m}$
...	...	...	...	...

tr_T	$fm_{T,1}$	$fm_{T,2}$	...	$fm_{T,m}$

donde

m El número total de métodos en el sistema considerado para la partición

$m^k_{i,j}$ El método j de la clase i, para k = 1 hasta m.

T El número total de transacciones.

tr_p Transacción p, para p = 1 hasta T.

$fm_{p,r}$ La frecuencia que la transacción p accesa el método r (igual 0 o 1).

2. *Method-Method Usage Matrix* (MMUM): la matriz de frecuencias que indica el número de veces que el método invoca otro método (en el caso de métodos simples esta matriz no existe). Parecido a TMUM los valores 0 o 1 indican la existencia de llamadas anidadas de los métodos.

método/método	$m^1_{i,j}$	$m^2_{i,j}$	...	$m^m_{i,j}$
$m^1_{i,j}$	1	$fmm_{1,2}$	...	$fmm_{1,m}$
$m^2_{i,j}$	$fmm_{2,1}$	1	...	$fmm_{2,m}$
...	...	...	...	...
$m^m_{i,j}$	$fmm_{m,1}$	$fmm_{m,2}$	...	1

donde

m El número total de métodos en el sistema considerado para la partición.

$m^k_{i,j}$ El método j de la clase i, para k = 1 hasta m.

$fmm_{s,r}$ La frecuencia que el método s accesa el método r (igual 0 o 1).

3. *Method-Attributes Usage Matrix* (MAUM): la matriz que representa el número de invocaciones de los atributos específicos en una ejecución de método. Tenemos posibilidad de tener los siguientes valores:

- cero - indica que el método no accesa ningún atributo,

- uno - indica que el método lee el valor del atributo (recuperar),

- dos - indica que el método lee y escribe el valor del atributo (modificar).

método/atributo	$at^1_{i,j}$	$at^2_{i,j}$	...	$at^n_{i,j}$
$m^1_{i,j}$	$fat_{1,1}$	$fat_{1,2}$	...	$fat_{1,n}$

$m^2_{i,j}$	$fat_{2,1}$	$fat_{2,2}$	...	$fat_{2,n}$
...	...	...	...	...
$m^m_{i,j}$	$fat_{m,1}$	$fat_{m,2}$	...	$fat_{m,n}$

donde

n El número total de atributos en el sistema considerado para la partición.

$at^p_{i,j}$ Atributo j de la clase i, para p = 1 hasta n.

$fat_{k,l}$ La frecuencia de acceso de atributo k por el método k (igual 0, 1 o 2).

La información especificada arriba puede ser dada por el diseñador del sistema.

Adicionalmente, en la misma manera como para el enfoque relacional, se puede considerar la frecuencia de uso de las transacciones por las aplicaciones (ftr):

Transacción	tr_1	tr_2	tr_3	...	tr_T
Frecuencia	ftr_1	ftr_2	ftr_3	...	ftr_T

Como consecuencia se puede decir que, si se multiplica la TMUM que contiene la información sobre métodos usados por las transacciones específicas, y la MAUM que contiene la información sobre las frecuencias de uso de los atributos por los métodos específicos, se puede obtener la matriz que representa las frecuencias de uso de los atributos por las transacciones. Estas frecuencias pueden servir como base para el esquema de fragmentación y corresponden a AUM del enfoque relacio-nal. Sin embargo, es necesario considerar los mismos cuatro casos que fueron analizados por Ezeife y Barker [Ezei94] de atributos simples y métodos simples, atributos simples y métodos complejos, atributos complejos y métodos simples, atributos complejos y métodos complejos.

4. DESCRIPCIÓN DE DISEÑO DE JERARQUÍAS Y PROPAGACIÓN DE LA FRAGMENTACIÓN DE OBJETOS

En el proceso de desarrollo del modelo de fragmentación se presentan los métodos para el particionamiento de las clases en BDDOO calculando primero los valores de la matriz TAUM que corresponde a la matriz AUM del enfoque relacional. Además, se seleccionan las particiones por medio de la búsqueda exhaustiva aplicando el evaluador de particiones para BDDOO (PEOO, *Partition Evaluator for Object-Oriented Databases*) [Mali96]. Los métodos presentan variaciones dependiendo de la complejidad del sistema y la granularidad de fragmentación. Por la granularidad se entiende el nivel de profundidad de jerarquía de las clases hasta el cual se necesita aplicar la fragmentación.

4.1 Atributos simples y métodos simples

En esta categoría se dispone solamente de una estructura jerárquica muy simple, donde los métodos pueden usar los atributos de su propia clase o sus superclases. No se tiene los métodos anidados o atributos que apuntan a otros atributos.

En el algoritmo que se propone se debe:

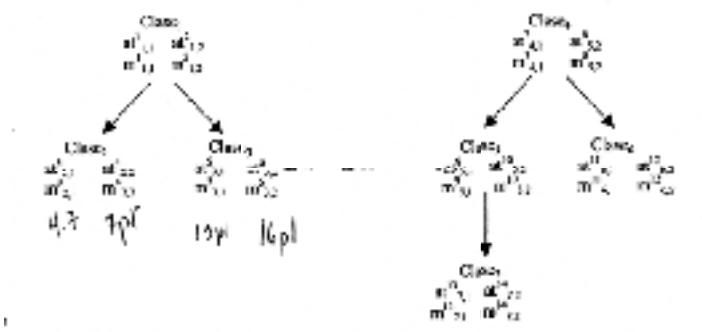
1. Especificar en TMUM y MAUM todas las frecuencias del uso de los métodos y atributos usados desde la raíz hasta el nivel de granularidad i establecido. Si existen las clases de un nivel mayor que i, entonces para cada una ellas se representan las frecuencias de acceso de los atributos y métodos como la suma de los valores correspondientes. Nótese aquí, que si la clase tiene sus subclases, la suma debe incluir todas las frecuencias de los métodos y atributos de esta clase y todas sus sub-clases.

2. Multiplicar cada fila de TMUM con los valores correspondientes de las frecuencias de las transacciones.

3. Multiplicar TMUM y MAUM.

4. Aplicar la búsqueda exhaustiva y PEOO para seleccionar el mejor esquema de fragmentación. Esta búsqueda considerará los atributos de las subclases del nivel mayor que i como indivisibles.

5. Propagar la fragmentación.



Como ejemplo considérese la siguiente presentación abstracta de las clases:

Para hacer el gráfico más fácil de entender no se repiten los métodos y atributos en sus subclases, asumiendo que todos los métodos y atributos de las clases pueden ser accesados por sus subclases. Aquí, si el nivel de granularidad es igual a uno para las dos clases raíces ($Clase_1$ y $Clase_4$) tendremos TMUM de la siguiente forma:

trans / método	$m^1_{1,1}$	$m^2_{1,2}$	$m^7_{4,1}$	$m^8_{4,2}$	M_2	M_3	M_5	M_6
tr_1	$fm_{1,1}$	$fm_{1,2}$	$fm_{1,3}$	$fm_{1,4}$	$fm_{1,5}$	$fm_{1,6}$	$fm_{1,7}$	$fm_{1,8}$
tr_2	$fm_{2,1}$	$fm_{2,2}$	$fm_{2,3}$	$fm_{2,4}$	$fm_{2,5}$	$fm_{2,6}$	$fm_{2,7}$	$fm_{2,8}$

...	...	...	...	...	...	...	...	...
tr_T	$fm_{T,1}$	$fm_{T,2}$	$fm_{T,3}$	$fm_{T,4}$	$fm_{T,5}$	$fm_{T,6}$	$fm_{T,7}$	$fm_{T,8}$

Los valores en las columnas M_i representan la suma de las frecuencias del acceso de los métodos por las transacciones correspondientes. Por ejemplo, el valor $fm_{1,5}$ es la suma de las frecuencias del acceso de los métodos $m^3_{2,1}$ y $m^4_{2,2}$ de la $Clase_2$ por la transacción tr_1. Para la $Clase_5$ los valores correspondientes incluyen no solamente las frecuencias de acceso de los métodos de esta clase, sino también las frecuencias de accesos de los métodos de sus subclases: $Clase_7$.

La MAUM se representa de la siguiente forma:

método/atributo	$at^1_{1,1}$	$at^2_{1,2}$	$at^7_{4,1}$	$at^8_{4,2}$	AT_2	AT_3	AT_5	AT_6
$m^1_{1,1}$	$fat_{1,1}$	$fat_{1,2}$	$fat_{1,3}$	$fat_{1,4}$	$fat_{1,5}$	$fat_{1,6}$	$fat_{1,7}$	$fat_{1,8}$
$m^2_{1,2}$	$fat_{2,1}$	$fat_{2,2}$	$fat_{2,3}$	$fat_{2,4}$	$fat_{2.5}$	$fat_{2,6}$	$fat_{2.7}$	$fat_{2.8}$
$m^7_{4,1}$	$fat_{3,1}$	$fat_{3,2}$	$fat_{3.3}$	$fat_{3.4}$	$fat_{4,5}$	$fat_{3,6}$	$fat_{3.7}$	$fat_{3,8}$
$m^8_{4,2}$	$fat_{4,1}$	$fat_{4,2}$	$fat_{4,3}$	$fat_{4,4}$	$fat_{4,5}$	$fat_{4,6}$	$fat_{4,7}$	$fat_{4,8}$
M_2	$fat_{5,1}$	$fat_{5,2}$	$fat_{5,3}$	$fat_{5,4}$	$fat_{5,5}$	$fat_{5,6}$	$fat_{5.7}$	$fat_{5,8}$
M_3	$fat_{6,1}$	$fat_{6,2}$	$fat_{6,3}$	$fat_{6,4}$	$fat_{6,5}$	$fat_{6,6}$	$fat_{6,7}$	$fat_{6,8}$
M_5	$fat_{7,1}$	$fat_{7,2}$	$fat_{7,3}$	$fat_{7,4}$	$fat_{7,5}$	$fat_{7,6}$	$fat_{7,7}$	$fat_{7,8}$
M_6	$fat_{8,1}$	$fat_{8,2}$	$fat_{8,3}$	$fat_{8,4}$	$fat_{8,5}$	$fat_{8,6}$	$fat_{8,7}$	$fat_{8,8}$

En esta matriz las columnas representadas A_{tk} son las sumas de las frecuencias de acceso de atributos por los métodos correspondientes y son representadas en la manera similar como en TMUM. Obsérvese que la intersección de la fila M_i y la columna A_{tk} tiene los valores que representan la suma de las frecuencias de todos los atributos de A_{tk} accesados por los métodos de M_i.

Para la búsqueda exhaustiva (el paso 4 del algoritmo presentado previamente), los atributos de las subclases $Clase_2$, $Clase_3$, $Clase_5$ y $Clase_6$ son considerados indivisibles. Esto significa que no se va a analizar el caso cuando los atributos de alguna de estas clases pertenecen a diferentes fragmentos. Además, esta consideración implica que todos los atributos de todas las subclases presentados en el mismo camino de la jerarquía deben formar una entidad indivisible (para las clases $Clase_5$ y $Clase_7$ los atributos $at^9_{5,1}$, $at^{10}_{5,2}$, $at^3_{7,1}$ y at $at^{14}_{7,2}$ deben ser considerados como indivisibles).

Si el nivel de granularidad es dos para la clase raíz $Clase_1$ y uno para la raíz $Clase_4$ las matrices correspondientes serán:

trans / método	$m^1_{1,1}$	$m^2_{1,2}$	...	$m^8_{4,2}$	M_5	M_6
tr_1	$fm_{1,1}$	$fm_{1,2}$	...	$fm_{1,8}$	$fm_{1,9}$	$fm_{1,10}$

tr_2	$fm_{2,1}$	$fm_{2,2}$	...	$fm_{2,8}$	$fm_{2,9}$	$fm_{2,10}$			
...	...	...	...	...	...	...	...	...	...
tr_T	$fm_{T,1}$	$fm_{T,2}$	...	$fm_{T,8}$	$fm_{T,9}$	$fm_{T,10}$			
método / attributo		$at^1_{1,1}$	$at^2_{1,2}$	...	$at^8_{4,2}$	AT_5	AT_6		
$m^1_{1,1}$		$fat_{1,1}$	$fat_{1,2}$	...	$fat_{1,8}$	$fat_{1,9}$	$fat_{1,10}$		
$m^2_{1,2}$		$fat_{2,1}$	$fat_{2,2}$	...	$fat_{2,8}$	$fat_{2,9}$	$fat_{2,10}$		
...		...	...	...	...	...	...		
$m^8_{4,2}$		$fat_{8,1}$	$fat_{8,2}$	...	$fat_{8,8}$	$fat_{8,9}$	$fat_{8,10}$		
M_5		$fat_{9,1}$	$fat_{9,2}$	...	$fat_{9,8}$	$fat_{5,9}$	$fat_{9,10}$		
M_6		$fat_{10,1}$	$fat_{10,2}$	...	$fat_{10,8}$	$fat_{10,9}$	$fat_{10,10}$		

Los grupos de atributos indivisibles se definen en la forma similar del caso del nivel de la granularidad uno.

Después de obtener estas matrices multiplicativas y aplicar algoritmo para obtener el mejor esquema de fragmentación, sigue el proceso de propagación.

El paso de la propagación puede ser representado como ejemplo para la $Clase_1$, suponiendo que después de aplicar la búsqueda exhaustiva con el nivel de granularidad uno se obtiene dos fragmentos: el primero tiene el atributos $at^1_{1,1}$ y el segundo tiene atributo $at^2_{1,2}$. La propagación de esta fragmentación puede ser presentada de la siguiente manera:

Obsérvese que, para el propósito de la implementación, los atributos y métodos de las clases $Clase_2$ y $Clase_3$ pueden ser replicados como se presenta en la figura de arriba o se puede mantener solo una copia de cada subclase. Además, como se mencionó anteriormente porque los métodos son fáciles de duplicar, se propone su duplicación.

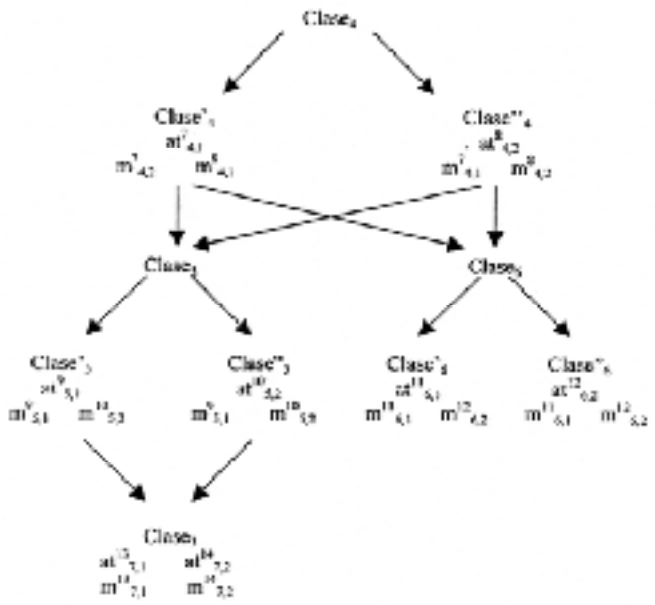
En el caso del nivel de granularidad dos, se puede representar la propagación de fragmentación de forma similar. Se supone que la búsqueda exhaustiva se aplica a la raíz de

la clase $Clase_4$ y sus subclases $Clase_5$ y $Clase_6$. Para la simplicidad de representación se asume que el mejor esquema de partición tiene dos fragmentos con los siguientes atributos:

fragmento 1: $at^7_{4,1}$, $at^9_{5,1}$ y $at^{11}_{6,1}$

fragmento 2: $at^8_{4,2}$, $at^{10}_{5,2}$ y $at^{12}_{6,2}$.

La representación gráfica de la propagación es la siguiente:



Aquí, dos fragmentos de $Clase_5$ apuntan a la misma clase $Clase_7$ evitando la duplicación de los objetos de la clase $Clase_7$.

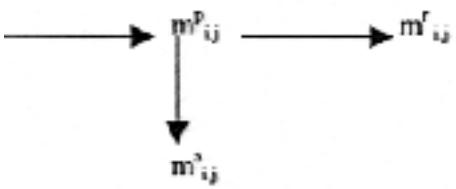
Aunque esta representación gráfica de la propagación parece muy complicada para esta simple clase, en realidad puede ser implementada usando las ideas presentadas en el sistema ORION [Kim94], donde cada clase tiene su propio descriptor. En esta implementación cada objeto tiene su único indetificador de objeto (UID, *unique identifier*) que consiste de un par de valores: el identificador de la clase y el identificador del objeto. De esta forma la información en cuál fragmento se encuentra el atributo solicitado se puede mantener en el descriptor de la clase.

Sin embargo, la representación de clases puede ser más compleja e incluir la existencia de métodos anidados.

4.2 Atributos simples y métodos complejos

La diferencia aquí con el caso previo de atributos simples y métodos simples consiste que ahora se tiene la posibilidad de las llamadas anidadas entre los métodos o de la misma

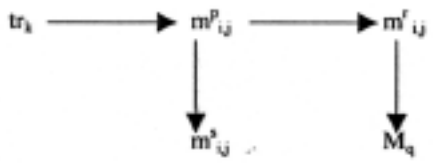
clase o entre las clases que pertenecen a los niveles antecedentes del mismo camino del árbol de la jerarquía.



Gráficamente, un nivel de llamadas anidadas se puede representar de la siguiente forma:

Las aristas del grafo (las frecuencias del uso de los respectivos métodos) se presentan en las matrices de TMUM y MMUM en las posiciones correspondientes con el valor igual a 1.

Un caso más complicado puede ser presentado como sigue:



Dependiendo del nivel de granularidad, la transacción puede accesar el método (m) o a uno de los métodos que forma la parte indivisible del grupo de los métodos (M). Estas llamadas anidadas deben ser reflejadas en MMUM.

En este caso la diferencia con el algoritmo presentado anteriormente consiste en que antes de multiplicar las matrices TMUM y MAUM, primero se necesita multiplicar TMUM y MMUM para considerar las llamadas anidadas. Teniendo la matriz TMUM modificada se puede multiplicarla con la matriz MAUM y seguir los pasos 4 y 5 del primer caso.

Si se sigue aumentando la complejidad de clases, se pueda llegar al siguiente nivel de atributos complejos y métodos simples.

4.3 Atributos complejos y métodos simples

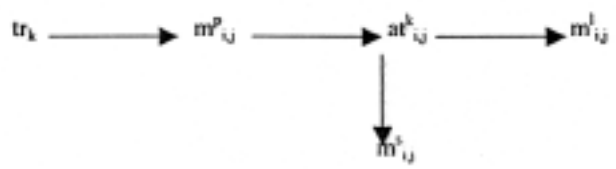
Los objetos complejos son los objetos los cuales tienen los atributos apuntando a otros objetos. En este caso, cuando el atributo complejo es accesado, en realidad lo que se accesa es este atributo y algún método de la clase hacia donde este atributo apunta. Como se sigue el principio de encapsulación, no existe la situación donde el atributo complejo accesa el atributo de otra clase en forma directa.

Ahora se tiene, no solamente la relación entre el método y el atributo (los atributos que un método específico usa presentados en MAUM), sino también una nueva relación entre los métodos.

Esta nueva relación tiene que ser capturada en la matriz de MMUM dando el caso de atributos simples y métodos complejos. De otro lado, la relación entre el método y el atributo complejo puede ser ignorada, después de reflejar en MMUM la relación entre los métodos. Esto ocurre porque el atributo complejo puede ser visto como un tipo de atributo "virtual". La asignación de este atributo en algún fragmento específico, después de aplicar el algoritmo de fragmentación, no da la información necesaria concerniente, como particionar los atributos de la clase donde este atributo está apuntando. En este acercamiento se propone eliminar el atributo "virtual" del esquema de particionamiento. Este atributo siempre puede ser puesto y duplicado en cualquier fragmento donde sea necesario. Eliminando este atributo del algoritmo de fragmentación, se obtiene el caso de los atributos simples y métodos complejos, en lugar de los atributos complejos y métodos simples.

Para entender mejor esta situación, se presenta un ejemplo.

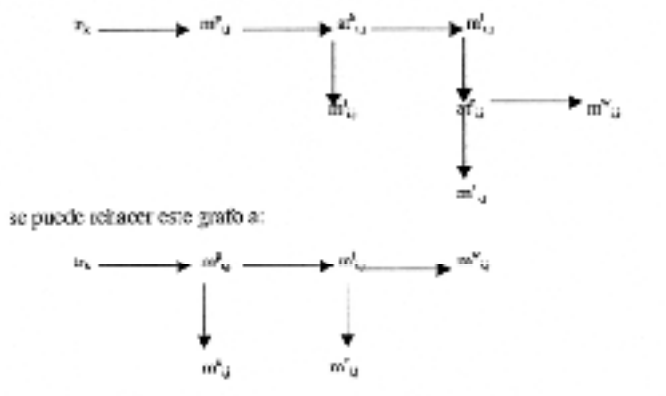
Supóngase que el atributo complejo $at^k_{i,j}$ se usa con los siguientes métodos: $m^l_{i,j}$ y $m^s_{i,j}$. Esto significa que en algún método se puede encontrar la llamada como: $at^k_{i,j}.m^l_{i,j}$ o $at^l_{i,j}.m^s_{i,j}$ (como se está siguiendo el principio de encapsulación, no se puede encontrar en la transacción, la llamada directa de dos atributos complejos). Así, se puede representar la situación en la forma gráfica:



Si alguna transacción k llama el método que utiliza este atributo complejo, se propone eliminar este atributo del grafo. Esto resulta en la situación donde los métodos $m^l_{i,j}$ y $m^s_{i,j}$ son llamados en el método $m^p_{i,j}$. Esta relación entre los métodos ya se reflejó anteriormente en las posiciones (p,l) y (p,s) y no se necesita hacer modificaciones.

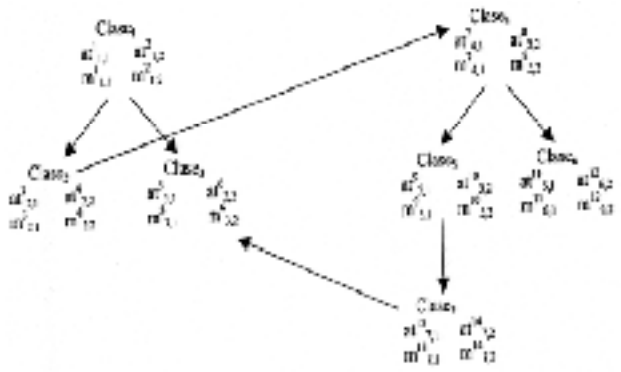
En el caso en que el acceso a los atributos complejos es más complicado (algún método al cual este atributo complejo apunta, accesa otro atributo complejo) se necesita hacer las modificaciones para el grafo de la misma forma como en el caso anterior.

Por ejemplo, si se tiene la siguiente situación:



y además, se puede presentar de forma adecuada los valores en la matriz MMUM.

También se puede considerar diferentes niveles de granularidad en la forma muy similar en que fue hecho en el caso anterior para los atributos simples y métodos simples. Considérese la siguiente jerarquía de los objetos:



Si el nivel de la granularidad es uno para las dos clases raíces, los métodos $m^1_{1,2}$, $m^2_{1,2}$, $m^7_{4,1}$ y $m^8_{4,2}$ se representan en las matrices correspondientes. Para las clases $Clase_2$, $Clase_3$, $Clase_5$ junto con la clase $Clase_7$, y $Clase_6$, las frecuencias de acceso de los métodos se presentarán como las sumas de las frecuencias de los métodos de las clases mencionados anteriormente. Basándose en la explicación dada para el caso de los atributos simples y métodos simples, se tiene, adicionalmente a los métodos especificados al principio del párrafo, los siguientes grupos de métodos:

M_2 contiene los métodos de la clase $Clase_2$

M_3 contiene los métodos de la clase $Clase_3$

M_5 contiene los métodos de la clases $Clase_5$ y $Clase_7$

M_6 contiene los métodos de la clase $Clase_6$

La misma consideración se puede aplicar a los atributos.

Después de preparar TMUM, MMUM (si es necesario) y MAUM, aplicar la búsqueda exhaustiva con un nivel de granularidad específico y obtener el esquema de fragmentación de los atributos, la pregunta que viene es cómo propagar la fragmentación. La forma de hacerlo es exactamente la misma como en el primer caso de atributos simples y métodos simples: comenzando desde la raíz para cada fragmento se analiza los atributos propios de cada clase. Después se propaga esta fragmentación hacia las subclases. En cada subclase se refleja primero la fragmentación heredada y solo después se analiza la propia fragmentación de esta subclase. Finalmente, se aplica la fragmentación a estos atributos propios de la subclase en la misma manera como se hizo con los atributos en la raíz. Este proceso se repite recursivamente hasta el nivel i de granularidad dado. Para las clases de nivel mayor que i, se aplica el concepto del grupo de atributos indivisibles, el cual se explicó anteriormente.

De los casos vistos anteriormente, queda por ver el último, de atributos complejos y métodos complejos.

4.4 Atributos complejos y métodos complejos

Se puede ver que este caso ya fue incluido en los casos anteriores.

Para todos los casos, el algoritmo aplicado es el mismo que el presentado para atributos simples y métodos simples excepto el siguiente refinamiento para el caso de las llamadas anidadas de los métodos: Si las llamadas anidadas existen, se debe preparar MMUM y multiplicarla por TMUM antes de realizar el segundo paso.

5. CONCLUSIONES

Son muchos aspectos que se debe considerar en las BDDR que permiten mejorar el desempeño en comparación con un sistema relacional no distribuido. Uno de estos aspectos es la adecuada forma de fragmentar los datos. Los algoritmos de la fragmentación vertical proponen la distribución de atributos en diferentes nodos, lo que asegura el mejor desempeño en un ambiente dado. Estos algoritmos usan matrices AUM o AAM. La matriz AUM recopila la información sobre la frecuencia de uso de los atributos por las transacciones y AAM señala la "afinidad" de parejas de atributos.

Aunque se desarrollaron investigaciones para encontrar formas de distribuir los datos en un ambiente orientado a objetos, éstos no usan ninguna de estas matrices. La razón principal es que, aunque se considera BDOO como el caso general de BDR, todavía no se han encontrado los correspondientes elementos generalizados para poder ser usados en este ambiente de orientación a objetos. Como la matriz AUM presenta una información más general, en este artículo se expuso la forma de obtenerla en BDOO, basándose en nuevas matrices del uso de métodos por las transacciones (TMUM), del uso de atributos por los métodos (MAUM) y del uso de los métodos por métodos (MMUM). Además, se especificó un algoritmo general de fragmentación tomando en cuenta la existencia de cuatro tipos de ambientes: atributos simples y métodos simples, atributos simples y métodos complejos, atributos complejos y métodos simples y atributos complejos y métodos complejos. Por falta

de algoritmos heurísticos de fragmentaciones, se propuso el uso de búsqueda exhaustiva y aplicación de evaluador de particiones para la selección del mejor esquema de particionamiento. Dependiendo del nivel de "granularidad" (profundidad de jerarquía de clases que se desea fragmentar), se puede modificar los datos en las matrices para respetar los requerimientos del usuario con respecto a la distribución de los atributos en diferentes fragmentos.

Como las BDOO en general se representan en forma de las estructuras jerárquicas, adicionalmente a la existencia de las matrices y algoritmos de particionamiento, ellas necesitan resolver el problema de la propagación de la fragmentación de clases. Aunque este aspecto se puede solucionar en teoría, presentando las clases en forma de grafos, las implementaciones reales señalarán si se justifica incorporar las posibilidades de fragmentar en un manejador de bases de datos distribuidas orientadas a objetos y no empeorar el desempeño del sistema.

6. BIBLIOGRAFIA

Chakravarthy, S., Muthural, J., Varadarajan, R., y Navathe, S.B. "An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis". *Distributed and Parallel Databases*, Vol. 2 , No. 1, 1993.

Chu Pai-Cheng. "A transaction Oriented Approach to Attribute Partitioning". *Information Systems*, Vol. 17, No.4, 1992.

Chu, W. y Ieong, I.T. "A Transaction Based Approach to Vertical Partitioning for Relational Database Systems". *IEEE Transactions on Software Engineering*, Vol. 19, No. 8, Agosto 1993.

Cornell, D. y Yu, P. "A Vertical Partitioning Algorithm for Relational Databases". *Proc. Third International Conference on Data Engineering*, Febrero 1987.

Ezeife, C.I. y Barker, K. "A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System". *Technical Report*, Advanced Database Systems Laboratory, Department of Computer Science, University of Manitoba, Canada, October 1994.

Gruber, O. y Amsaleg, L. "Object Grouping in EOS" en M.T. Ozsu, U.Dayal y P.Valduriez (eds.) *Distributed Object Management*. Morgan Kaufman Publishers, San Mateo, Ca 1994.

Hammer, M. y Niammir, B. "A Heuristic Approach to Attribute Partitioning". *Proc. ACM SIGMOD International Conference on Management of Data*. Boston, MA, 1979.

Hoffer, J.A. y Severeance, D.G. "The Use of Cluster Analysis in Physical Database Design". *Proc. First Intenational Conference on Very Large Data Bases*. Fragingham, MA, Setiembre 1975.

Karlapalem, K. y Li, O. "Partitioning Schemes for Object Oriented Databases". *Technical Report*, University of Science and Technology, Department of Computer Science Clear Water Bay, Kawloon, Honk Kong, Agosto 1994.

Karlapalem, K. Navathe, A.B. y Morsi, M. "Issues in Distribution Design" en M.T. Ozsu, U. Dayal y P. Valduriez (eds.) *Distributed Object Management*. Morgan Kaufman Publishers, San Mateo, Ca. 1994.

Kim, W. "Architecture of the ORION Next-Generation Database Systems" en M. Stonebarker *Readings in Database Systems*. Morgan Kaufman Publishers, 1994.

Lin, X., Orlowska, M. y Zhang, Y. "A Graph Based Cluster Approach for Vertical Partitioning in Database Systems". *Data & Knowledge Engineering*, Vol. 11, 1993.

Malinowski, E. "Fragmentation Techniques for Distributed Object-Oriented Databases". *Tesis de Maestría*. Universidad de Florida en Gainesville, 1996.

Navathe, S., Ceri, G., Wiederhold y Dou, J. "Vertical Partitioning Algorithm for Database Design". *ACM Transaction on Database Systems*, Vol. 9, No.4, Diciembre 1984.

Navathe, S. y Ra, M. "Vertical Partitioning for Database Design: A Graphical Algorithm". *ACM SIGMOD*, Portland, June 1989.

Pernul, G., Karlapalem, K. y Navathe, S.B. "Relational Database Organization Based on Views and Fragments". *Proc. of the Second International Conference on Data and Expert Systems Applications*, 1991.