Estructuras de datos multidimensionales: un análisis de desempeño

Luis Hernando Silva Flórez y Ana Clara Vélez Torres***

Resumen

En los sistemas manejadores de datos multidimensionales es necesario construir índices para agilizar las consultas. Debido a la existencia de múltiples estructuras definidas para representar los índices, se dificulta la decisión acerca de cuál estructura utilizar. Este artículo muestra los resultados de una evaluación del proceso de consulta por rangos sobre varias estructuras de datos multidimensionales, con base en el comportamiento del número de accesos a disco. Las estructuras de datos evaluadas fueron: el GRID FILE, el árbol KDB y el árbol R. Los experimentos revelan que para rangos pequeños, independientemente de la *escalabilidad*, el comportamiento del número de accesos a disco mostrado por el árbol R es similar al del GRID FILE. A medida que la *extensibilidad* aumenta, el árbol R muestra un menor número de accesos a disco. Para el caso del GRID FILE, el número de accesos a disco crece linealmente con una pendiente alta, a medida que aumenta la *extensibilidad*, lo que limita su uso a rangos pequeños. En el caso del árbol KDB, el comportamiento del número de accesos a disco no depende de la *extensibilidad*. Del resultado del experimento se deduce que de las tres estructuras evaluadas, la más recomendable, para el propósito de disminuir el número de accesos a disco en consultas por rango, es la estructura del árbol R.

---------- *Palabras clave*: estructuras de datos multidimensionales, procesamiento de consultas, GRID FILE, árbol KDB, árbol R.

Abstract

In multidimensional data management systems is necessary to build indexes for improving the response time during user queries. So far, several structures have been defined for index representation. However, there is a lack of experimental evidence to support the selection of one of the structures for index representation. In this study, an experimental comparison of the performance of range query operations on three spatial data structures was conducted. The performance was measured through the number of disk accesses during the query operations. The spatial data structures considered were the GRID-FILE, the KDB-TREE, and the R-TREE. The experiments show that for queries of small range size, the

* Profesor Asociado. Departamento Ingeniería de Sistemas. Universidad de Antioquia. E-mail: hsilva@udea.edu.co.

** Profesora Asociada. Departamento Ingeniería de Sistemas. Universidad de Antioquia. E-mail: acvelez@udea.edu.co-

number of disk accesses for the R-Tree is similar to the number of disk accesses for the GRID-FILE. These results were persistent for all the values of scalability. When the extensibility grows, the queries on the R-Tree show a lower number of disk accesses. The number of disk accesses on the GRID-FILE data structure grows linearly with a high slope as the extensibility grows. This result suggests that the GRID-FILE should only be employed for queries with small ranges. For queries on the KDB-Tree data structure, the number of disk accesses does not depend on the extensibility. From the results of the experiments we conclude that, among the three structures evaluated, the R-Tree structure is the most recommendable, when the goal is to reduce the number of disk accesses during range query operations.

---------- *Key words*: Spatial Data Structures, Query Processing, GRID-FILE, KDB-Tree, R-Tree.

Introducción

El presente informe muestra los resultados de la primera investigación sobre comparación del comportamiento de tres estructuras de datos multidimensionales con respecto al número de accesos a disco, en procesos de consulta.

Se pretende con este trabajo hacer un aporte al conocimiento sobre el manejo de datos espaciales con estructuras no tradicionales ya que el desarrollo de las bases de datos se ha centrado en el manejo de datos de tipo numérico y de tipo texto y sus consultas siempre se han limitado a los atributos de la base de datos en una dimensión. Aunque para su conformación se incluyan varios atributos, éstos se concatenan para formar uno sólo y con base en él se realiza la consulta. Las consultas se pueden hacer por valores exactos de las claves o por rangos lineales únicamente, esto es, no se permiten consultas sobre los índices en los que se incluyan rangos de varios atributos.

En los últimos años, debido a la necesidad de manejar datos multidimensionales en distintas áreas, como: cartografía, diseño asistido por computador, robótica, sistemas expertos, GIS (Sistemas de Información Geográfica), etc., se han desarrollado múltiples estructuras que permiten su manejo.

De acuerdo con los tipos de métodos que se han definido se seleccionaron las siguientes estructuras para ser comparadas:

- Como método de acceso de hashing multidimensional, se optó por el GRID FILE de un nivel.
- Como método de acceso jerárquico se implementó el árbol KDB.
- Como método basado en solapamiento de regiones se desarrolló el árbol R.

La investigación se basó en determinar cuál de los tres métodos de acceso a datos espaciales –arriba descritos– se comporta mejor en consultas por rangos con respecto a los parámetros de *escalabilidad* y *extensibilidad*.

Dado que el desempeño de un método de acceso espacial es usualmente evaluado por su habilidad para responder a consultas por rangos –definida por el número de accesos a páginas de disco– se utilizó este parámetro para la evaluación, lo que significa que en el análisis no se tuvieron en cuenta los resultados del desempeño para consultas puntuales. En las consultas por rango se incluyeron las de coincidencia parcial.

Los accesos a la base de datos se realizaron exclusivamente por la componente espacial –objeto de estudio de esta investigación– ya que para los datos tradicionales existen métodos de acceso suficientemente probados e implementados que los soportan en forma óptima.

Una base de datos espacial consiste en una colección de tuplas que representan objetos espaciales, en donde cada tupla tiene un único identificador de tipo espacial que está constituido por el conjunto de valores asociados a las d-dimensiones del espacio euclidiano donde pertenece el objeto.

Descripción de las estructuras

EL GRID FILE (Nievergelt, Hinterberger y Sevcik 1981)

El GRID FILE superpone una malla ortogonal d-dimensional sobre el espacio. Debido a que la malla no es necesariamente regular, las celdas resultantes pueden ser de diferente forma y tamaño. Un *directorio de malla* asocia una o más de estas celdas con los buckets de datos, los cuales son almacenados sobre cada página de disco. Cada celda está asociada con un bucket, pero un bucket puede contener los datos pertenecientes a varias celdas adyacentes. Dado que el directorio crece considerablemente, se almacena en memoria secundaria. Para garantizar que los elementos de datos sean siempre encontrados con no más de dos accesos al disco, en consultas por exactitud, la malla se almacena en memoria principal, representada por d arreglos unidimensionales denominados *escalas*.

La figura 1 muestra un ejemplo de un GRID FILE para puntos en dos dimensiones, donde los c_i y

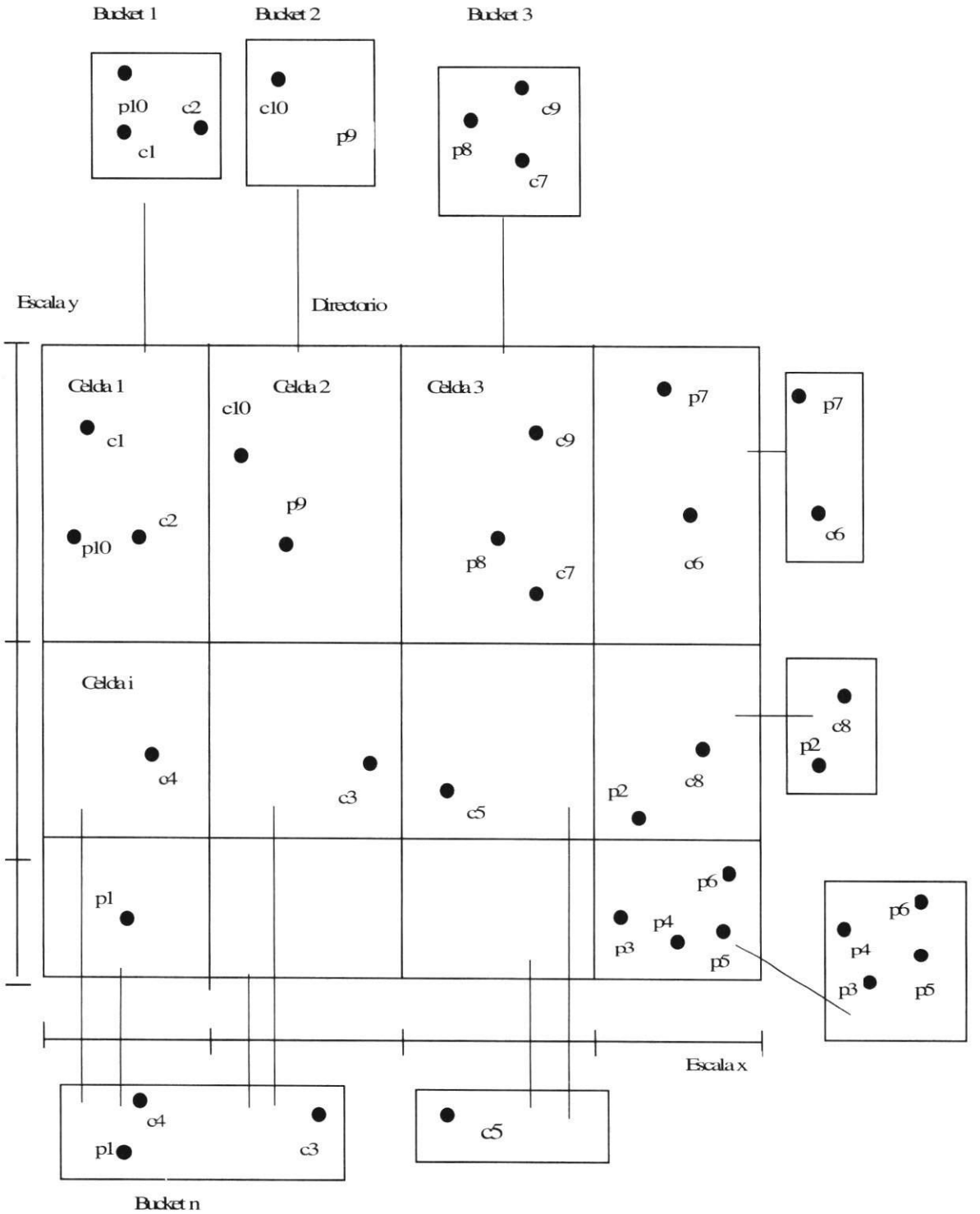


Figura 1 GRID FILE

los p_i representan los puntos de datos. Se asume una capacidad del bucket de cuatro puntos. El centro de la figura muestra el directorio con las escalas en los ejes *x* y *y*. Los puntos de datos son desplegados en el directorio solo con el fin de mostrarlos, pero no se almacenan allí. En la parte inferior izquierda, cuatro celdas son reunidas en un bucket, representadas por cuatro apuntadores a una sola página. Por tanto, existen cuatro entradas de directorio para la misma página. El área de bucket que contiene el punto c_5, ha sido mezclada con una adyacente, para una mejor utilización del espacio.

Para responder a una consulta por exactitud, primero se utiliza la escala para localizar la celda que contiene el punto de búsqueda. Si la celda apropiada no se encuentra en memoria principal, se necesita un acceso al disco. Las celdas cargadas contienen una referencia a la página donde se podrá encontrar posiblemente el punto que se busca. Recuperar esta página puede requerir otro acceso a disco. En total, no se necesitan más de dos accesos a las páginas para responder a una consulta puntual. Para una consulta de rangos, se examinan todas las celdas que se solapan con la región de búsqueda. Después de eliminar los duplicados se buscan las páginas de datos correspondientes en memoria principal para una inspección más detallada.

Árbol KDB

El árbol KDB de Robinson combina las propiedades del árbol KD adaptivo (Bentley y Friedman 1979) y del árbol B (Bayer y McCreight 1970) para manejar puntos mul-tidimensionales. Particiona el espacio en la forma de un árbol KD adaptivo y asocia los sub-espacios resultantes con los nodos del árbol. Cada nodo interno corresponde a un intervalo de la región. Las regiones correspondientes a los nodos en un mismo nivel son mutuamente disjuntas. Los nodos hojas almacenan los puntos de datos que son localizados en la partición correspondiente. Similar al árbol B, el árbol KDB es un árbol perfectamente balanceado que se adapta bien a la distribución de los datos. En la figura 2 se muestra un ejemplo del árbol KDB.

Similar al árbol B, el árbol KDB consiste en una colección de páginas y una variable que contiene un apuntador a la página raíz (*ID raíz*).

El siguiente conjunto de propiedades define la estructura del árbol KDB:

1. Considere cada página como un nodo y cada *ID página* como un apuntador a un nodo, la estructura resultante es un árbol multicamino con raíz *ID raíz*, por tanto, una página región contiene un conjunto de puntos y son los nodos hojas del árbol.
2. La longitud de la trayectoria, desde la página raíz a las páginas hoja, es la misma para todas las páginas hoja.
3. En cada página región, las regiones contenidas en ella son disjuntas y su unión es una región.
4. Si la página raíz es una página región (puede no existir o si existe, ser una sola página en el árbol y será entonces una página hoja) la unión de las regiones conforma todo el dominio de la región o el universo.
5. Si (*región, ID hijo*) se presenta en una página región y la página hijo referida por *ID hijo* es una página región, la unión de las regiones en la página hijo es una región.
6. Si la página hijo es una página hoja, entonces todos los objetos de la página conformarán una región.

Estructura del árbol KDB

Se define como hiperplano al elemento perteneciente a un dominio *i* que divide a una región en dos subregiones. Las regiones deben ser disjuntas y un punto no puede pertenecer a más de una región.

Dominio: 0, 1, 2, …, i, ..., *d-1*

Punto : $X_0, X_1, X_2, ..., X_i, ..., X_{d-1}$

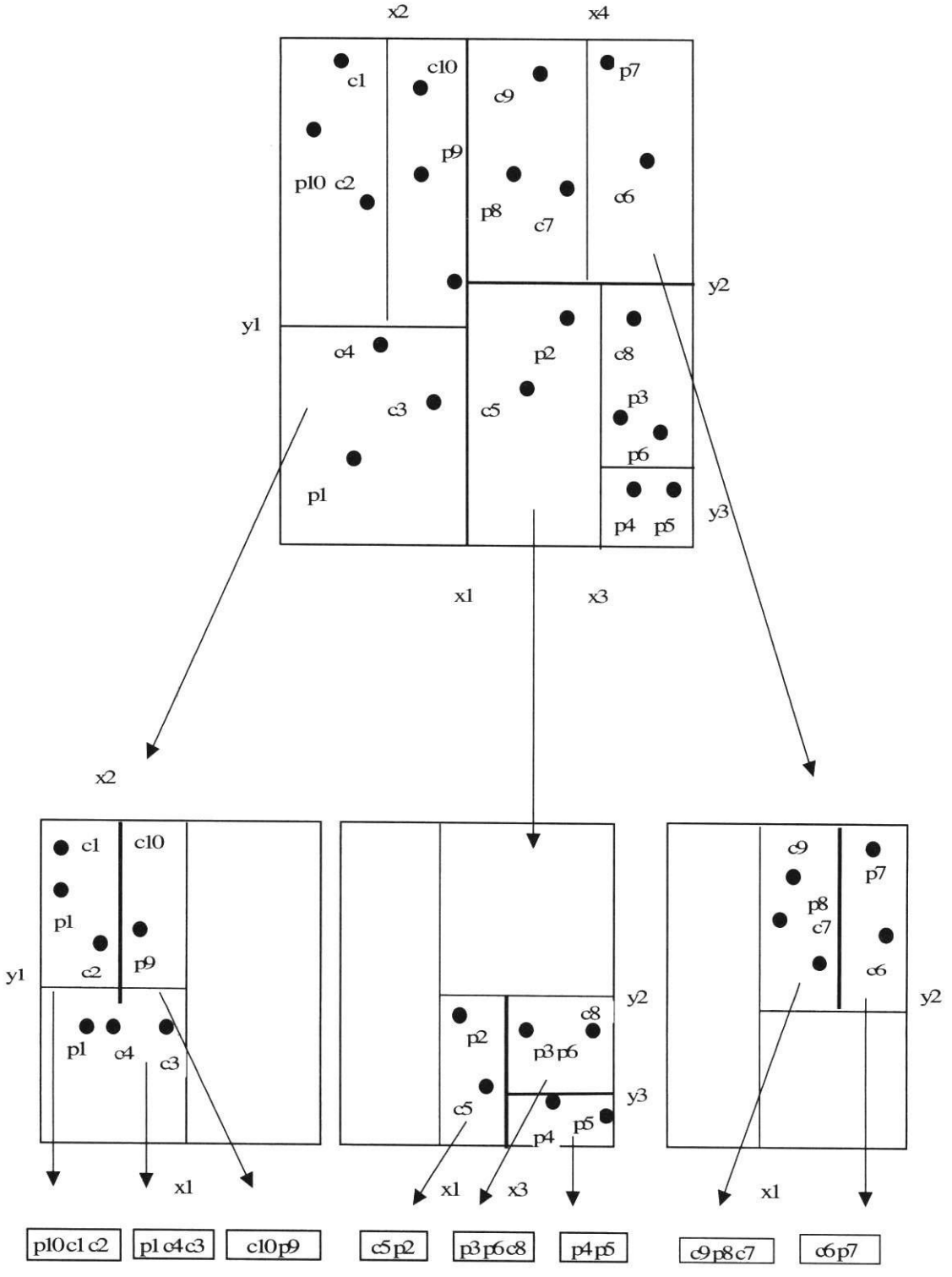


Figura 2 Árbol KDB

Región: $X_0, X_1, X_2, ..., X_i, ..., X_{d-1}$

$Y_0, Y_1, Y_2, ..., Y_i, ..., Y_{d-1}$

$Z_0, Z_1, Z_2, ..., Z_i, ..., Z_{d-1}$

El árbol KDB consiste en una colección de páginas y un apuntador a la raíz del árbol y posee dos tipos de nodos denominados *Nodo Índice* y *Nodo Hoja*.

Un *Nodo Índice* consiste en una colección de apuntadores a otros nodos *Hoja* o *Índice*, una colección de elementos de un dominio, una colección de dominios y un contador:

N: Cantidad de hiperplanos en el nodo.

d: Conjunto de *N* hiperplanos.

D: Conjunto de *N* dominios (dimensiones).

A: Conjunto de *N+1* apuntadores.

Cada hiperplano está unido a una dimensión y a dos apuntadores.

Los hiperplanos para un mismo dominio se encuentran ordenados.

$P_i < P_j$ donde $i \neq j$ e $i < j$, $i >= 0$

Un *Nodo Hoja* consiste en una colección de puntos y un contador:

N: Cantidad de puntos almacenados en el nodo.

P: Conjunto de *N* puntos de *d* dimensiones.

Algoritmo de consulta

Parámetros: Raíz, Nivel Actual, Rango de la consulta.

Paso 1

Si Raíz = 0 entonces Mensaje "no puntos insertados"

Terminar. Fin si

Paso 2

Si Nivel Actual es igual a la Altura del Árbol entonces

Leer Nodo Hoja diseccionado por la Raíz

Buscar puntos dentro del nodo hoja que coincida con la búsqueda. Terminar

Fin si

Ir a paso 3

Paso 3

Leer un nodo índice direccionado por Raíz

Para todo hiperplano del nodo índice haga

Consultar los valores apropiados para determinar si se continúa la búsqueda de las hojas por su hijo derecho, su hijo izquierdo o por ambos.

Si se debe continuar por la izquierda y el subárbol no ha sido MARCADO.

Llamar recursivamente a la función Consultar con parámetros Raíz del subárbol del lado izquierdo y el nivel actual más uno.

Si no, Desmarcar el subárbol.

Fin si

Si se debe continuar por la derecha entonces

Llamar recursivamente a la función consultar con parámetros Raíz del subárbol del lado derecho y el nivel actual más uno.

MARCAR el subárbol

Fin si

Fin ciclo para

Terminar

Algoritmo de búsqueda de los nodos hoja

Este algoritmo permite determinar el subárbol por donde se debe continuar la búsqueda de las hojas donde se hallan los puntos que satisfacen la consulta.

Parámetros: Nodo índice, Posición del hiperplano padre dentro del nodo índice

Valores de retorno: Raíz Izquierda, Raíz Derecha

Li1 = 0, Li2 = 0, Ld1 = 0, Ld2 = 0

Seleccionar uno de los casos siguientes para el operador relacional del primer operador de consulta correspondiente al dominio al cual pertenece el hiperplano padre.

Caso MAYOR QUE:

Si la clave del hiperplano es mayor que Valor del primer operador de consulta entonces

Li1 = Raíz del hijo izquierdo del hiperplano. Fin Si

Ld1 = Raíz del hijo derecho del hiperplano

Caso MENOR QUE:

Si la clave del hiperplano es menor que Valor del primer operador de consulta entonces

Ld1 = Raíz del hijo derecho del hiperplano. Fin Si

Li1 = Raíz del hijo izquierdo del hiperplano

Caso MAYOR IGUAL QUE:

Si la clave del hiperplano es mayor que Valor del primer operador de consulta entonces

Li1 = Raíz del hijo izquierdo del hiperplano. Fin Si

Ld1 = Raíz del hijo derecho del hiperplano

Caso MENOR IGUAL QUE:

Si la clave del hiperplano es menor o igual que Valor del primer operador de consulta entonces

Ld1 = Raíz del hijo derecho del hiperplano. Fin Si

Li1 = Raíz del hijo izquierdo del hiperplano

Caso IGUAL A:

Si la clave del hiperplano es mayor que Valor del primer operador de consulta entonces

Li1 = Raíz del hijo izquierdo del hiperplano. Si no,

Ld1 = Raíz del hijo derecho del hiperplano

Fin si

Caso DIFERENTE:

Li1 = Raíz del hijo izquierdo del hiperplano

Ld1 = Raíz del hijo derecho del hiperplano

Caso NULO:

Li1 = Raíz del hijo izquierdo del hiperplano

Ld1 = Raíz del hijo derecho del hiperplano

Fin Selección

Seleccionar uno de los casos siguientes para el operador relacional del primer operador de consulta correspondiente al dominio al cual pertenece el hiperplano padre.

Caso MAYOR QUE:

Si la clave del hiperplano es mayor que Valor del segundo operador de consulta entonces

Li2 = Raíz del hijo izquierdo del hiperplano. Fin Si

Ld2 = Raíz del hijo derecho del hiperplano

Caso MENOR QUE

Si la clave del hiperplano es menor que Valor del segundo operador de consulta entonces

Ld2 = Raíz del hijo derecho del hiperplano. Fin Si

Li2 = Raíz del hijo izquierdo del hiperplano

Caso MAYOR IGUAL QUE:

Si la clave del hiperplano es mayor que Valor del segundo operador de consulta entonces

Li2 = Raíz del hijo izquierdo del hiperplano. Fin Si

Ld2 = Raíz del hijo derecho del hiperplano

Caso MENOR IGUAL QUE:

Si la clave del hiperplano es menor o igual que Valor del segundo operador de consulta entonces

Ld2 = Raíz del hijo derecho del hiperplano. Fin Si

Li2 = Raíz del hijo izquierdo del hiperplano

Caso IGUAL A:

Si la clave del hiperplano es mayor que Valor del segundo operador de consulta entonces

Li2 = Raíz del hijo izquierdo del hiperplano. Si no,

Ld2 = Raíz del hijo derecho del hiperplano. Fin si

Caso DIFERENTE:

Li2 = Raíz del hijo izquierdo del hiperplano

Ld2 = Raíz del hijo derecho del hiperplano

Caso NULO:

Li2 = Raíz del hijo izquierdo del hiperplano

Ld2 = Raíz del hijo derecho del hiperplano

Fin Selección

Si Li1 <> 0 y Li2 <> 0 Entonces

Hacer Raíz izquierda igual a Li1. Fin Si

Si Ld1 <> 0 y Ld2 <> 0 Entonces

Hacer Raíz Derecha igual a Ld1. Fin Si

Terminar

Árbol R

Un árbol R (figura 3) es un árbol de altura balanceada similar al árbol B, con registros de índice en los nodos hoja que contienen apuntadores a objetos de datos. Los nodos corresponden a páginas de disco; el índice, por su exten-sibilidad, reside en el disco y la estructura está diseñada de tal forma que una búsqueda espacial requiera visitar solamente un pequeño número de nodos. El índice es completamente dinámico, las inserciones y los borrados pueden ser entremezclados con las búsquedas sin requerir de una reorganización periódica.

Los árboles R son una técnica común de indexamiento para datos multidimensionales y son ampliamente usados en bases de datos espaciales y multidimensionales. Almacenando los límites de las cajas de objetos geométricos arbitrarios como: puntos, polígonos y objetos más complejos, los árboles R pueden ser usados para determinar cuáles objetos intersectan una región de consulta dada.

Los árboles R almacenan una colección de rectángulos que puede cambiar a través del tiempo por inserciones y borrados. Objetos geométricos son representados por su rectángulo de mínima frontera, Minimal Bounding Rectangle (MBR), esto es, el rectángulo con la más pequeña esquina superior derecha que contiene al objeto.

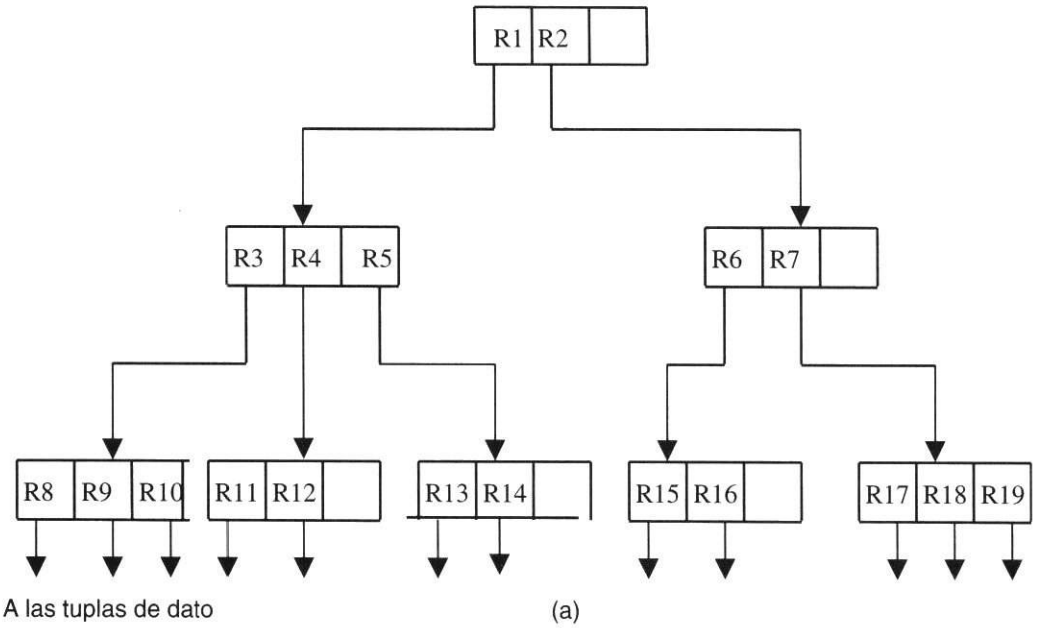
Una ruta hacia abajo en el árbol corresponde a una secuencia de rectángulos anidados, el último de los cuales contiene objetos de datos. Nótese también que los rectángulos en cualquier nivel pueden solaparse y que un árbol R creado para un conjunto de objetos en particular no siempre será el mismo, esto es, su estructura depende de la secuencia de entrada de datos.

Para ejecutar una consulta Q, todos los rectángulos que intersectan la región consultada deben ser recuperados y analizados (sin hacer caso de si ellos están almacenados en un nodo interno o en un nodo hoja). Esta recuperación es realizada usando un simple procedimiento recursivo que comienza en la raíz y que puede seguir varias rutas hacia abajo a través del árbol.

Un nodo en primer lugar es procesado por la recuperación de todos los rectángulos almacenados en ese nodo que intercepta Q. Si el nodo es interno, los subárboles correspondientes a los rectángulos recuperados son buscados recursivamente. De otro modo, el nodo es una hoja y los rectángulos recuperados simplemente se reportan.

Definición

En un nodo hoja de un árbol R, se tiene un registro índice que hace referencia al dato espacial. El registro índice es (I, Identificador de tupla). I es un rectángulo d-dimensional correspondiente a su MBR y cada entrada en un identificador de tupla son los límites superior e inferior en cada una de las dimensiones del rectángulo.



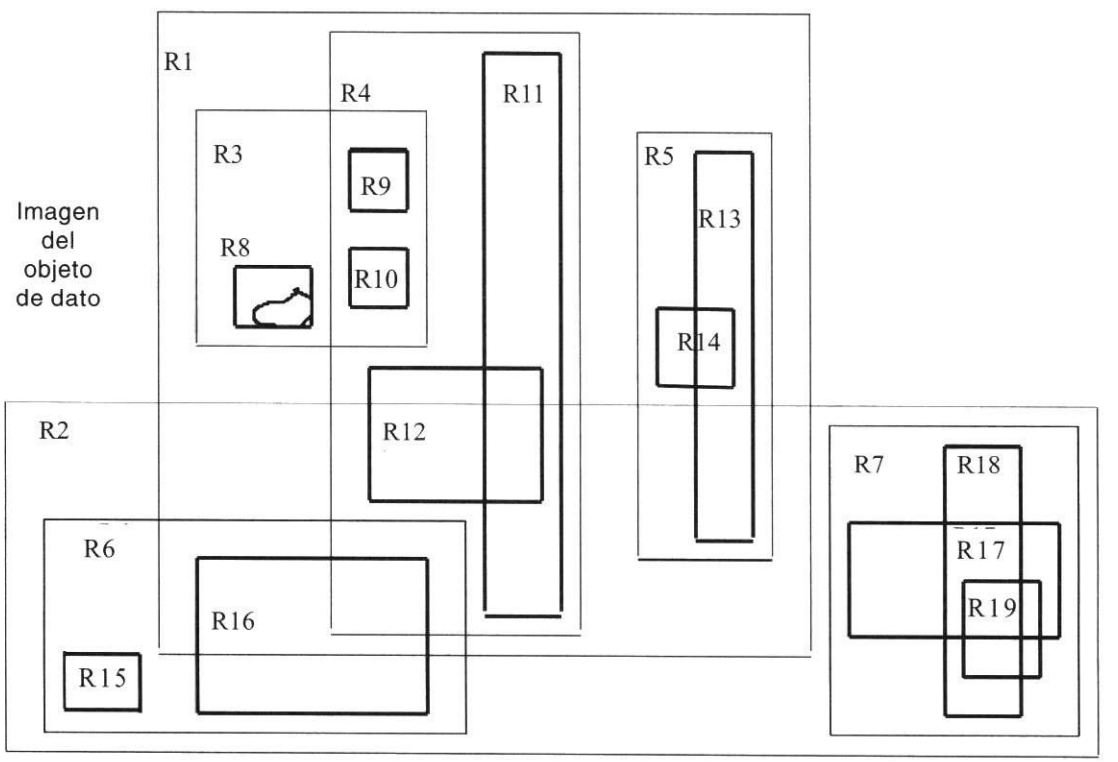


Figura 3 Estructura del árbol R

La idea fundamental de un árbol R es: árbol de altura balanceada, similar a un árbol B, donde los registros de índices se almacenan solamente en las hojas como apuntadores a objetos de datos.

Los registros índice que lo conforman son de la forma: *(I, Identificador de tupla),* donde, *Identificador de tupla* = apuntador a objeto en la base de datos, $I = (I_0, I_1, I_2, ..., I_{d-1})$, Límite de la caja del objeto en un rectángulo n-dimensional. Cada I_j está contenida en un intervalo [a, b] (puede ser $-\infty$ o $+\infty$) e I_j es el límite del objeto en la dimensión j.

Las entradas en los nodos no hoja son de la forma: *(I, Apuntador al hijo)* donde

I: Límites del rectángulo n-dimensional que contiene a todos los rectángulos n-dimensionales en los nodos hijos.

Apuntador al hijo: dirección de un nodo en un nivel inferior en el árbol.

Propiedades

Un árbol R satisface las siguiente propiedades:

- Es un árbol de altura balanceada y todas sus hojas están en el mismo nivel.
- El nodo Raíz tiene al menos dos hijos, a no ser que sea un nodo hoja.
- Sea M el máximo número de entradas en un nodo completo y sea $m \leq M/2$, el mínimo número de entradas en un nodo. Cada nodo no hoja contiene entre m y M hijos, a menos que sea la raíz.
- Por cada entrada *(I, Apuntador a nodo hijo)* en un nodo no hoja, *I* es el más pequeño rectángulo que espacialmente contiene todos los rectángulos en el nodo hijo.
- Toda hoja contiene entre m y M registros de índice, a menos que sea la raíz.
- Por cada registro índice *(I, Identificador de tupla)* en un nodo hoja, *I* es el más pequeño rectángulo que espacialmente contiene el objeto de dato d-dimensional representado por el identificador de tupla.

La altura de un árbol R que contiene N registros de índice es a lo sumo $|\log_m N| - 1$, debido a que el factor de bifurcación de cada nodo es al menos m.

El máximo número de nodos es $|N/m| + |N/m^2| + ... + 1$. En el peor caso, la utilización de espacio de todos los nodos, excepto la raíz, es m/M. Si los nodos tienen más de 3 ó 4 entradas, el árbol es muy ancho y casi todo el espacio es usado por los nodos hoja que contienen registros de índice. El parámetro m puede ser variado como parte de la sintonización del desempeño.

Algoritmo de búsqueda

Dado un árbol R donde el nodo raíz es T, encuentra todos los registros de índice donde los rectángulos se solapan con el rectángulo de búsqueda *S.*

S1. *Buscar subárboles*

Si T no es una hoja, chequea cada entrada E para determinar si E.I se solapa con S. Para todas las entradas que se solapan, invoca *BUSCAR* sobre el subárbol donde el nodo raíz es apuntado por E.p.

S2. *Buscar en nodos hojas*

Si T es una hoja, chequea todas las entradas de E para determinar si E.I se solapa con S. Si es así, E es un registro cualificado.

Procedimiento

Dado que el objetivo de la investigación es el desempeño de los métodos de acceso en las consultas, no se describen los algoritmos de borrado.

Para la comparación del comportamiento de los métodos de acceso se tuvieron en cuenta dos parámetros en especial, la *extensibilidad,* referida a la amplitud de los rangos de consulta, y la

escalabilidad, referida ésta a las consultas sobre el volumen de datos.

Para evaluar la componente *escalabilidad,* se construyó una base de datos con diez (10) archivos que contenían desde 100.000 hasta un millón de tuplas (con incrementos de 100.000), que representan un igual número de puntos espaciales, los cuales poseen una componente espacial, conformada por sus correspondientes coordenadas en tres dimensiones.

Con respecto a la componente *extensibilidad,* se realizaron consultas equivalentes a rangos del 5, 35 y 75%, tal como lo muestran las figuras 4, 5 y 6 sobre los diferentes archivos de la base de datos y se midió el comportamiento de las tres estructuras de datos con respecto al número de accesos a disco.

Para el análisis de comportamiento, se utilizó un bloque que tenía la capacidad de contener hasta 60 puntos de datos, para todas las estructuras. Los algoritmos para adición y consulta se codificaron utilizando compiladores de lenguaje C, C++ y Visual Basic en un computador con procesador Intel Pentium.

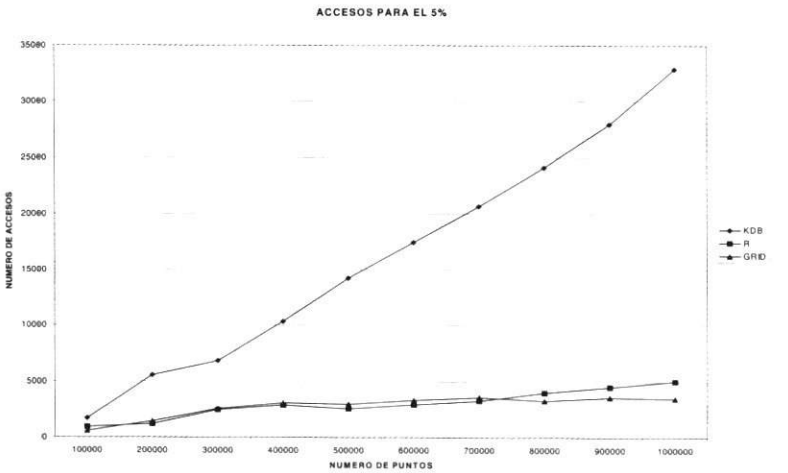


Figura 4 Componente *extensibilidad* consulta equivalente al rango 5%

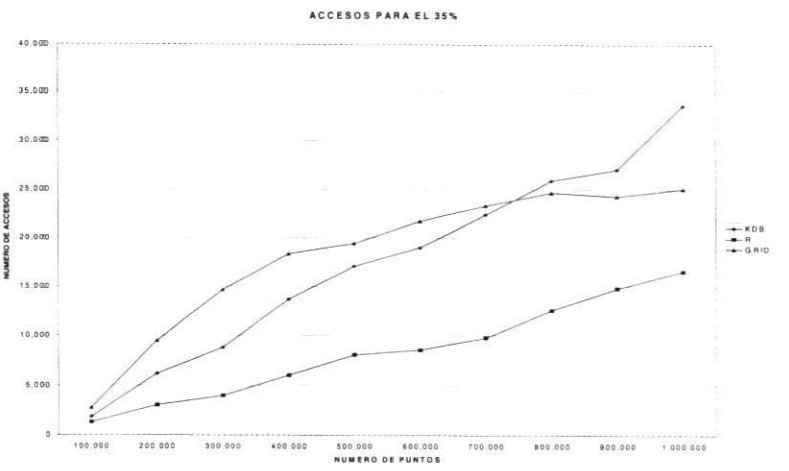


Figura 5 Componente *extensibilidad* consulta equivalente al rango 35%

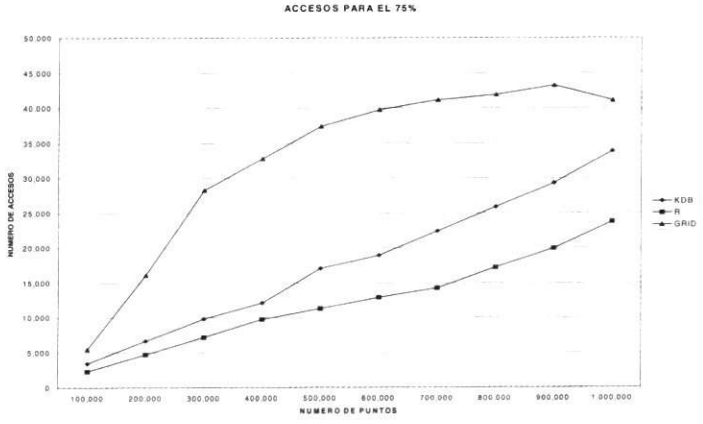


Figura 6 Componente *extensibilidad* consulta equivalente al rango 75%

Análisis de resultados

- A medida que aumenta el número de tuplas (escalabilidad), todas las estructuras tienden a aumentar el número de accesos, como se observa en los gráficos de resultados.
- Para consultas de rangos pequeños (5%), independiente de la escalabilidad, tanto el GRID FILE como el árbol R presentan un comportamiento aceptable, lo que no ocurre con el árbol KDB.
- Para consultas de rangos medianos (35%), el GRID FILE muestra un comportamiento directamente proporcional a la escalabilidad. Por el contrario, el árbol R, demuestra un crecimiento aceptable de acuerdo con la escalabilidad, con un aumento lineal y pendiente pequeña dependiendo del tamaño del archivo. En este rango de consultas, el árbol KDB aunque muestra, en escalas pequeñas, un comportamiento mejor que el GRID FILE, éste desmejora a medida que aumenta la escala; de todos modos, su comportamiento es inferior al del árbol R, en todas la escalas.
- Para consultas de rangos grandes (75%), el GRID FILE muestra, al igual que en rangos medianos, un comportamiento directamente proporcional a la escalabilidad. El árbol R, sigue demostrando un crecimiento aceptable de acuerdo con la escalabilidad, con un aumento lineal y pendiente pequeña dependiendo del tamaño del archivo. En este rango de consultas, el árbol KDB muestra mejor desempeño que en rangos medianos, con crecimiento lineal pero con pendiente aceptable.

Como conclusión, aunque el árbol KDB es completamente balanceado, su comportamiento no es aceptable debido a que cuando se hace acceso a un nodo de éste, lo que se obtiene es un árbol KD, cuyos nodos son hiperplanos en una sola dimensión y por tanto en las consultas por rangos se debe hacer recorrido de varios subárboles del árbol KDB con lo que se aumenta el número de accesos a disco.

El mal comportamiento demostrado por el GRID FILE respecto a la extensibilidad, se presenta debido a que a medida que aumenta el rango de búsqueda, el número de celdas, y por tanto de buckets a que se debe hacer acceso, aumenta en forma exponencial, así solo se aumente el rango en una dimensión.

El árbol R presenta un excelente desempeño independiente de la escalabilidad y la extensibilidad, lo que lo hace ideal como método de acceso para consultas por rangos en tres dimensiones.

Se observó que con el método de división de nodos utilizado para el árbol KDB, si los datos se sesgan, la varianza calculada puede, en un caso dado, ser 0; por tanto el hiperplano que se debe trazar generará una región que contendrá todos los puntos menos 1 y otra región con el punto restante.

Bibliografía

1. Beckmann N., *et al.* "The R*-tree: an efficient and roust access method for points and rectangles". *ACM SIGMOD*, mayo de 1990. pp 322-331.
2. Guttman A. "R-trees: a dynamic index structure for spatial searching". Proc. *ACM SIGMOD*, Junio de 1984. pp. 47-57.
3. Kamel, I., *et al.* "On packing R-trees". In *Proc. 2nd International Conference on Information and Knowledge Management*. Arlington, VA, noviembre de 1993. pp. 490-499.
4. Kamel, I., *et al.* "Hilbert R-tree Using Fractals". In *Proc 20th International Conference on VLDB*. Santiago Chile, 1994. pp. 500-509.
5. Sellis, T., *et al.* "The R+-tree: a dynamic index for multi-dimensional objects". In *Proc 13th International Conference on VLDB*. England, September, 1987. pp. 507-518.
6. Roussopoulos, N., *et al.* "Direct Spatial on Pictorial Databases Using Packed R-trees". Proc. *ACM SIGMOD,* May, 1985.
7. Hinrichs, K. "The Grid File System: implementation and case studies for applications". *Dissertation No. 7734*, Eidgenossische Technische Hochschule, Zuerich. 1985.
8. Robinson, J.T. "The KDB Tree: Search Structure for Large Multidimensional Dynamic Indexes". Proc. *ACM SIGMOD* 1981. Int *Conference Management of Data.* Abril de 1981. pp. 10-18.
9. NievergelT, J., *et al.* "The Grid File: An adaptable, symmetric multikey file structure". *ACM trans. Database System. 9*, 1. Marzo de 1984. pp 38-71.
10. Gaede, V. *et al. Multidimensional Access Methods.* Berlin. 1997.