DeepL
Suscríbete a DeepL Pro para poder traducir archivos de mayor tam
Más información disponible en www.DeepL.com/pro.

Estructuras de datos multidimensionales: Revisión y perspectivas

S. SITHARAMA IYENGAR
N. S. V. RAO

Departamento de Informática Louisiana Slade
University
Baton Rouge, Luisiana 70803

R. L. KASHYAP

Departamento de Ingeniería Eléctrica
Universidad de Purdue
West La fayette, Indiana 47907

V. K. VAISHNAVI

Departamento de Sistemas Informáticos
Georgia State University University
Plaza
Atlanta, Georgia 30303

' Parcialmente financiado por la NSF con la subvención IST 8405052.

ADvANCES EN COUPUTERS. VOL. 27

Derechos de autor 1988 por
Academic Press. Inc. Reservados todos los derechos
de reproducción en cualquier formato.
ISBN 0-12-0 l21 27-1

1. Introducción

El diseño y análisis de estructuras de datos y algoritmos es un área importante de la informática. En los últimos años se han producido varios avances significativos en este , desde el algoritmo de búsqueda binaria hasta el sorprendente descubrimiento del algoritmo polinómico para el problema de la programación lineal (PL). Estos resultados han producido soluciones eficientes y óptimas para muchos problemas de la vida real. Como consecuencia, se ha generado un gran interés en el campo de las estructuras de datos y los algoritmos, que han florecido hasta situarse en la vanguardia de la investigación informática.

La búsqueda de una colección ordenada de datos es un paradigma estándar en el ámbito de los algoritmos. El problema de la búsqueda en una sola ha tratado tradicionalmente utilizando las estructuras de datos de árboles binarios, árboles multidireccionales, tablas hash, etc. Sin embargo, en muchas aplicaciones que surgen en la recuperación de información, la geometría com- putacional, la robótica y la estadística, el problema central consiste en determinar los puntos que caen en límites de rango especificados en diferentes dimensiones. Estos problemas han despertado un enorme interés en el diseño y análisis de estructuras de datos multidimensionales eficientes. Además, las estructuras de datos multidimensionales se dirigen a una clase de problemas más amplia que las estructuras de datos unidimensionales. Véase, para más detalles, Horowitz y Sahni (1976). Pero las estructuras de datos multidimensionales no son extensiones naturales de las estructuras de datos unidimensionales. Por el contrario, son más complejas y están más orientadas a los problemas. Por ejemplo, consideremos la estructura de datos de árbol binario equilibrado. El coste de construir una estructura de datos de este tipo es $O(N \log N)$, donde N es el número de puntos de datos de la estructura. El coste de almacenamiento correspondiente *es O{N)*. Esta estructura de datos puede responder a consultas completas de coincidencia y rango en tiempo 0(log N) y $O(I \text{ og } N+ Kt)$ respectivamente,

donde *K* es el número de puntos de datos cualificados que satisfacen la consulta. No existen estructuras de datos multidimensionales con un rendimiento similar.

Recientemente se han estudiado una serie de estructuras de datos multidimensionales como árboles k-d, quadtrees, árboles de rango, árboles poligonales, árboles quintarios, árbol de atributos múltiples (MAT), árboles multidimensionales equilibrados y ponderados, etc. para problemas de rango y búsqueda relacionados.

Un árbol de búsqueda k-dimensional es una generalización k-dimensional de un árbol de búsqueda. Puede utilizarse para elementos que sean k-tuplas. Cuando los elementos de datos (k-dimensionales) tienen pesos (números reales positivos) asociados, los elementos de datos se denominan elementos ponderados. Un árbol k-dimensional para tales elementos ponderados es un árbol ponderado de k-dimensiones. En este artículo también exploraremos la relación entre los árboles ponderados y los árboles multidimensionales equilibrados.

1.1 Organización y alcance del documento

El objetivo de la parte inicial de este artículo es presentar y analizar estructuras de árbol multidimensionales equilibradas, sencillas y eficientes, útiles para aplicaciones en tiempo real. La utilidad de tales estructuras para datos muy grandes y entornos concurrentes es una preocupación importante de la investigación futura en estructuras de datos multidimensionales. Un tema importante de esta parte del artículo es la presentación de un paradigma para generalizar una de árbol equilibrada a multidimensiones, junto con un conjunto de estrategias para soportar violaciones de estructura en las estructuras generalizadas. En la última parte del artículo se tratan diversos aspectos de los árboles de atributos múltiples.

La sección 2 presenta una visión general del rendimiento de varias estructuras de datos multidimensionales. La sección 3 describe los árboles multidimensionales equilibrados y ponderados, así como sus aplicaciones en la organización de bases de datos físicas, la recuperación de información y los problemas de geometría computacional. La sección 4 revisa la literatura para generar un concepto de equilibrio para árboles multidimensionales. La sección 5 describe un paradigma que modela las estructuras multidimensionales y ponderadas equilibradas y semilibradas disponibles. Se espera que estos paradigmas guíen futuras investigaciones en esta área. La sección 6 describe estructuras de árbol de atributos múltiples para consultas de coincidencia parcial, coincidencia completa y rango. También se analiza una comparación del rendimiento de la estructura MAT con otras estructuras multidimensionales. La sección 7 analiza los paradigmas para modelar el tratamiento destacado de los árboles de atributos múltiples. Por último, la sección 8 presenta algunas conclusiones.

2. ¿Qué es una estructura de datos multidimensional ?

Se dice que un dato o elemento de datos *es sin{ffe-dimensional* si se caracteriza por un único atributo o valor clave. Ejemplos de unidimensionalidad

Las estructuras de datos son árboles binarios equilibrados, árboles B, árboles AVL, tablas hash, árboles multidireccionales, etc.

Se dice que un elemento de datos es multidimensional si se caracteriza por una serie de atributos. Un elemento de datos multidimensional especificado por k atributos puede imaginarse como un "punto" en un espacio k-dimensional. Ejemplos de estructuras de datos multidimensionales son los árboles k-d, los quadtrees, los árboles de atributos múltiples (MAT), los árboles B multidimensionales, los árboles multidimensionales equilibrados y ponderados, etc. Un problema multidimensional requiere una estructura de datos eficaz que organice estos puntos de tal forma que el acceso sea eficiente. El problema de recuperar todos los registros que satisfacen una consulta que implica una multiplicidad de atributos, conocido como *recuperación asociativa*, es una cuestión importante en la organización de bases de datos físicas. Los factores que afectan al rendimiento de la organización física de datos son las características lógicas de los datos, la complejidad de la consulta y los parámetros del dispositivo, y se ha demostrado que ninguna estructura de índice puede ser óptima en todas las circunstancias. Para más información, véase Kriegel (1981).

El rendimiento de una estructura de datos multidimensional se mide en función de las tres magnitudes siguientes (en nuestras discusiones asumimos *N* puntos, cada uno con *k* atributos):

- *P(N), coste de preprocesamiento*, que es el tiempo necesario para construir la estructura de datos;
- *S(N), coste de almacenamiento*, que es la cantidad de almacenamiento necesaria para guardar la estructura de datos;
- *Q N), coste de acceso*, que es el tiempo que se tarda en acceder a la estructura de datos según las necesidades. Puede ser el coste de consulta cuando se responde a una consulta sobre los datos. En general, el coste que conlleva puede atribuirse a la inserción, eliminación o cualquier otra modificación.

Una estructura de datos *estática* se construye una sola vez inicialmente. Las consultas pueden responderse en esta estructura de datos, pero no admiten inserciones ni eliminaciones. En una estructura de datos dinámica, las inserciones y supresiones se admiten dinámicamente. Como se verá en secciones posteriores, el mantenimiento de la estructura de datos en la forma óptima de una estática es una tarea muy difícil. Esto se debe principalmente a que las inserciones y supresiones destruirán, en general, la naturaleza equilibrada de la estructura de datos que puede obtenerse en el caso estático. Las estructuras de datos dinámicas son indispensables para un gran número de aplicaciones.

A continuación presentamos algunas estructuras de datos multidimensionales que surgen con frecuencia en diversas aplicaciones. La gran cantidad de literatura y la variada naturaleza de las estructuras de datos propuestas impiden un tratamiento exhaustivo. Por lo tanto, este documento cubre, en detalle, estructuras de datos multidimensionales y multidimensionales equilibradas.

árboles ponderados, árboles de atributos múltiples (MAT) y, en menor medida, otras estructuras de datos multidimensionales. Los paradigmas para modelar árboles multidimensionales y ponderados equilibrados, así como MAT, se presentan en detalle en la sección 7.

2.1 Clasificación de los problemas de búsqueda multidimensional

Una clase importante de problemas multidimensionales *es el problema de búsqueda.* Sean T" T2 *T* conjuntos de consultas, elementos y respuestas respectivamente. Un problema de búsqueda *P* sobre *T, ,* T2 +3 es una función

$$P : \mathrm{T}, \times 2''' \quad -- \quad Tz .$$

El problema de búsqueda *P* toma un objeto de consulta - *T* y un conjunto de dominios i_2 e 2 ' y produce una respuesta t, e *T* . Asignando distintas propiedades a *Th ,* To y *T ,* se obtienen las distintas clases de problemas de búsqueda.

En un problema de *búsqueda de miembros T* -- To y T-3 (sí, no}. El problema de búsqueda multidi- mensional se puede convertir en uno unidimensional, mediante concatenando todos los atributos para formar una única clave. Por lo tanto, los enfoques estándar para casos unidimensionales, como en Aho, Hopcroft, U llman (1974) y Knuth (1974) se pueden aplicar para obtener *Q[N) -- 0[k* log *N), P(N)= O(kN log N), S[N) = OWN).* Se pueden obtener las mismas estimaciones de complejidad para buscar el elemento r" o para encontrar el rango de un elemento en un conjunto ordenado como en Overmars (1984).

Otro problema de búsqueda importante *es la búsqueda de rango.* En este , T_2 contiene puntos de un espacio k-dimensional, y cada *q e T* especifica un rango para cada dimensión. *T* es un subconjunto de Toy contiene los puntos que satisfacen q. El quadtree de puntos propuesto por Finkel y Bentley (1974) y Lee y Wong (1977) tiene las estimaciones de complejidad de *Q(N) -- 0\ N' !' + K), P(N) - 0) N* log *N), S(N)= O(N),* donde N, es el número de puntos que satisfacen la consulta. Un quadtree de puntos para un conjunto dado de puntos es un árbol 2'-ario, que divide recursivamente el espacio en 2' cuadrantes. Aunque el tiempo en el peor de los casos no es muy impresionante, en la práctica el quadtree se muestra eficiente en muchos problemas prácticos. Los detalles pueden encontrarse en Finkel y Bentley (1974), Bentley y Stanat (1975) y Alt *et al.* (1981).

Una estructura de datos importante para resolver el problema de búsqueda de rango se llama *árbol k-d.* Esta estructura de datos fue introducida por Bentley (1975). Un árbol k-d es un tipo de árbol binario construido como sigue: Se toma un punto de datos como nodo raíz; el espacio de datos se divide en dos subconjuntos en función del primer valor de atributo; cada uno de estos conjuntos se representa como un subárbol de la raíz; cada subárbol se divide en función del segundo valor de atributo. Este proceso se realiza recursivamente en los subconjuntos. Tras la división con al atributo *k",* se vuelve a considerar el primer atributo. Bentley (1975) y Lee y Wong (1977)

resolver el problema de búsqueda de rango con las siguientes medidas de rendimiento: *Q(N) -- OWN ' '+ K), P(N) -- OWN* log *N), S(N) -- O(N).* Mehlhorn (1984b) analiza los árboles dd, que son muy similares a los árboles k-d. En cada paso, el espacio de datos se divide en tres subconjuntos, en contraste con los dos del árbol k-d. El primer subconjunto contiene todos los puntos cuyo valor de atributo correspondiente es menor que el de la raíz. El primer subconjunto contiene todos los puntos cuyo valor de atributo correspondiente es menor que el de la raíz. El segundo subconjunto contiene todos los puntos que tienen el mismo valor de atributo que la raíz. El tercer subconjunto contiene todos los puntos cuyo valor de atributo es mayor que el de la raíz. Ambas estructuras tienen características similares.

Existe otra estructura de datos *para* la consulta de rangos que presenta un mejor coste de consulta en el peor de los casos, a costa de elevados costes de preprocesamiento y almacenamiento. Esta estructura de datos ha sido propuesta independientemente por varios autores, y cada uno difiere del otro sólo ligeramente en los aspectos estructurales. Las medidas de rendimiento de esta estructura de datos son *Q(N)* -- 0(log' *N+ K), P(N) - OWN* log' *' N), S(N) - O(N* log' *' N).* La estructura de datos básica que subyace a todas estas versiones, denominada aquí *árbol de rangos,* puede definirse como sigue: Para el caso unidimensional, el árbol de rangos se reduce a un árbol de búsqueda de hojas binario equilibrado en el que los puntos se almacenan en los nodos hoja. Además, las hojas están enlazadas en ese orden. Para realizar una consulta de rangobuscamos los límites. A continuación, los puntos que se encuentran dentro de los límites del rango se recuperan utilizando la estructura de listas enlazadas. Un árbol de rangos k-dimensional consiste en un árbol de búsqueda de hojas binario equilibrado *T* en el que se almacenan los puntos y se ordenan según el primer atributo. A cada nodo interno " le asocia un árbol de rango *(k* - 1)-dimensional *T.* Este subárbol enraizado en " sólo tiene en cuenta los atributos segundo a k" para todos los puntos de los datos. La estructura básica de datos subyacente es un árbol de búsqueda binario unidimensional. Conceptualmente, se puede pensar en utilizar como estructura de datos subyacente otras estructuras de datos unidimensionales como el árbol AVL, el árbol B, el árbol 2-3, el árbol BB[nd, etc. Las estimaciones de orden para la consulta de rango siguen siendo las mismas, pero el número exacto de pasos para una consulta concreta difiere. Willard (1979b) utiliza el *árbol* B, y la estructura de datos multidimensional correspondiente se denomina ***superárbol*** *B.* Lueker (1978) utiliza el árbol BB be] como estructura de datos subyacente, y la versión multidimensional se denomina ***árbol*** *d-fold.* El árbol ***quintario*** de Lee y Wong (1980) también pertenece a este tipo. Utilizan un árbol quíntuple como estructura subyacente. Las demás estructuras de datos unidimensionales también pueden utilizarse para obtener nuevas estructuras de datos multidimensionales. Sin embargotales extensiones no prometer estructuras más eficientes que las existentes. Willard (1985) y Hart (198 l) reducen el coste de consulta a *Q(N)* -- O(log' *N+ K j* utilizando una técnica inteligente. Willard (1985) denomina a esta estructura de datos *árbol k-fold.* Se han propuesto otras estructuras de datos, como *rangos de k solapados, rangos de k* no *solapados,* etc. Estas estructuras ilustran las diferencias en el coste de las consultas. Estas estructuras ilustran las diferentes compensaciones entre el tiempo de consulta, el tiempo de preprocesamiento y el tiempo de almacenamiento.

coste. Las medidas de rendimiento para los rangos k solapados son: *Q(N)* - O(log *N+ K), P(N) -- O(N)*, y *S(N) - O(N' "'j, para e>* 0. Los rangos k no solapados tienen las siguientes medidas de rendimiento: *Q[N)= O(N' + K,), P(N) - O(N* log *N),* y *S iN) -- O(N).* La reducción de uno de los costes se refleja en un aumento de los demás. Bentley y Friedman (1979) ofrecen un excelente estudio sobre las estructuras de datos para la búsqueda de rangos. Se puede encontrar más información sobre el rango y cuestiones relacionadas en Overmars (1984), Mehlhorn (1984b), Willard (1985) y Willard y Lueker (1985).

Otro problema de búsqueda frecuente es el *del vecino más próximo*. Este problema consiste en encontrar un punto en T2 que sea el más cercano a un punto de consulta x e *T ,* con respecto a alguna métrica. Este problema se plantea con frecuencia en los sistemas de bases de datos. Se ha demostrado que este problema tiene la solución dada por *Q(N)* -- 0(log *N j, P(N) -- O(N* log *N j,* y *S(N) -- O(N)* para el caso bidimensional. Los detalles pueden encontrarse en Overmars (1984), Shamos y Hoey (1975), Shamos (1978) y Kirkpatrick (1983).

Las estructuras de datos multidimensionales, como los árboles k-d, los quadtrees, los árboles de rangos, etc., son estructuras de datos basadas en la comparación. Existe otra clase de estructuras de árbol multidimensionales basadas en la búsqueda digital de nodos en el árbol, (Knuth (1974)). Los nodos de las estructuras basadas en la comparación se forman a partir del resultado de una comparación entre dos elementos. En las estructuras de árbol digitales, los nodos se forman a partir de los valores de los elementos. La clase de estructuras de árbol multidimensionales basadas en técnicas digitales incluye *los árboles encadenados dobles modificados* de Cárdenas y Sagamang (1974), *los complejos de árboles de búsqueda binaria* (BST-complex) de Lien et *al.* (1975), el *árbol B multidimensional* **(MDBT**) de Ouksel y Scheuermann (1981), *los árboles kB* de Gotlieb y Gotlieb (1978), *el árbol kB+* de Kriegel y Vaishnavi (1981a) y *los árboles de atributos múltiples* (MAT) *de* Kashyap *et al.* (1977), Gopalakrishna y Veni Madhavan (1980) y Rao *et* aJ. (1984). La principal diferencia entre las estructuras de datos de comparación y las de árbol multidimensional digital radica en la forma en que se procesan los datos al construir la estructura de datos. En los métodos basados en la comparación, el espacio multidimensional se divide recursivamente en regiones más pequeñas de la misma dimensionalidad. En las estructuras basadas en la digitalización, la dimensionalidad se reduce recursivamente en uno. Esta distinción no es aplicable en el caso unidimensional, pero es un factor vital para decidir el rendimiento de las estructuras de datos multidimensionales. En las estructuras de datos de comparación, la búsqueda se realiza recursivamente en las regiones más pequeñas de la misma dimensionalidad, mientras que en las estructuras de datos digitales la búsqueda se recursivamente en el espacio de dimensionalidad reducida. Esto significa que puede haber aplicaciones que se adapten naturalmente a uno de dos tipos.

El área de la geometría computacional es rica en problemas multidimensionales y estructuras de datos. Esta área emergente ha encontrado muchas aplicaciones en diversas disciplinas como el reconocimiento de patrones, la investigación de operaciones, la informática

gráficos, robótica, etc. La geometría computacional se ocupa principalmente de la complejidad computacional de los problemas geométricos. Las cinco áreas principales en este campo son el casco convexo, las intersecciones, la búsqueda, la búsqueda de proximidad y la optimización com- binatoria. Los problemas de búsqueda tratados en este artículo están relacionados con los problemas de búsqueda y proximidad de la geometría computacional. Se han desarrollado varios algoritmos genéricos, como divide y vencerás, construcción incremental, barrido plano, lugar geométrico, transformaciones geométricas, etc. para diversos problemas. Para más detalles sobre problemas de geometría computacional véase Mehlhorn (1984b), Edelsbrunner (1983, 1984), Overmars (1984), Preparata y Shamos (1985), y Shamos (1978).

En el ámbito de la recuperación y gestión de la información, se han estudiado varias estructuras de datos multidimensionales. *Los árboles multidimensionales* y el *directorio multidimensional* de Orenstein (1982) son algunas de las más importantes. Las estructuras como las *listas invertidas* y las *multi-listas de* Liou y Yao (1977) son ejemplos de extensiones de los conceptos de listas a múltiples dimensiones. *El archivo grid de* Nievergelt et at. (1984) es otra estructura de datos multidimensional que admite el acceso multiclave. Esta estructura ha ser superior a las estructuras convencionales como los archivos invertidos y las listas múltiples en términos de adaptación a los entornos dinámicos. A diferencia de las de datos mencionadas anteriormente, el archivo grid se ocupa del almacenamiento del directorio en la memoria secundaria. También existe un pedigrí de estructuras de datos para la gestión directorio en la memoria secundaria. La generalización de las funciones hash de Rothnie *et* at. (1974) describe una estructura de archivos hash multiclave. Pero encontrar una función hash que optimice los costes de consulta, inserción, borrado, resolución de colisiones y almacenamiento es muy difícil (véase Lunn (1970)) y dependiente del problema. Lunn (1970) describe ***índices*** *combinados* que se forman concatenando varios valores de atributos. *Los mapas de bits comprimidos* se basan en los archivos codificados por bits. La organización de *ficheros transpuestos* de Batory (1979) y Barnes y Collens (1973) también hace uso de técnicas de compresión para proporcionar acceso multiclave. La generalización de los *intentos* para soportar el acceso multiclave puede en Orenstein (1982) y Tamminen (1982). Casey (1973) describe una estructura de acceso multiclave basada en un árbol complejo, que utiliza codificación *superpuesta* para caracterizar los registros situados bajo los nodos del árbol. Este enfoque es complejo, y un enfoque más práctico basado en conceptos similares puede encontrarse en Pealtz *et al.* (1980). Los directorios multidimensionales sugeridos por Liou y Yao (1977) proporcionan una forma eficiente de acceso multiclave, utilizando el orden de prioridad de los atributos. El mantenimiento de índices basado en interpolación de Burkhard (1983) es una extensión multidimensional del hashing lineal de Litwin (1980), y está relacionado con el hashing extensible de Fagin et *al.* (1979). Estos métodos de acceso multiclave para los datos almacenados en la memoria secundaria proporcionan soluciones eficientes para muchas aplicaciones prácticas. Dado que muchos de estos enfoques se basan en la identificación de una función hash adecuada, no son muy susceptibles de un análisis teórico riguroso.

Un aspecto importante de las estructuras de datos multidimensionales es la cinomiso- nización. El término "dinamización" se refiere al proceso de mantenimiento de la estructura de datos para garantizar el rendimiento de la estructura de datos estática en un entorno dinámico de inserciones y supresiones de registros. Normalmente, la dinamización de las estructuras de datos multidimensionales es mucho más complicada que la de las estructuras de datos unidimensionales. Este problema se complica aún más cuando el directorio de datos reside en la memoria secundaria. Esto se debe a que, en muchos casos, las inserciones y supresiones destruyen la agrupación de los datos en páginas de memoria secundaria. Muchas estructuras de datos son dinámicas cuando se implementan en memoria que admite acceso aleatorio. Pero las consideraciones se vuelven notablemente diferentes cuando la estructura de datos se almacena en la memoria secundaria. Una vez más, los requisitos de dinamización varían considerablemente en función de la aplicación. El marco general para la dinamización es introducido por Bentley (1979a, 1980) para una cierta clase de problemas llamados problemas *Jecomposables*. En Bentley y Saxe (1980) se discuten los conceptos generales de la transformación estática a dinámica de las estructuras de datos. Overmars (1984) analiza las nociones de reconstrucción *local, parcial* y *global* de estructuras de datos con fines de . La variada naturaleza de dinamización dificulta el tratamiento generalizado. Para más detalles sobre un método sencillo de dinamización, véase Rao, Vaishnavi y lyengar (1987).

3. Árboles equilibrados multidimensionales y ponderados

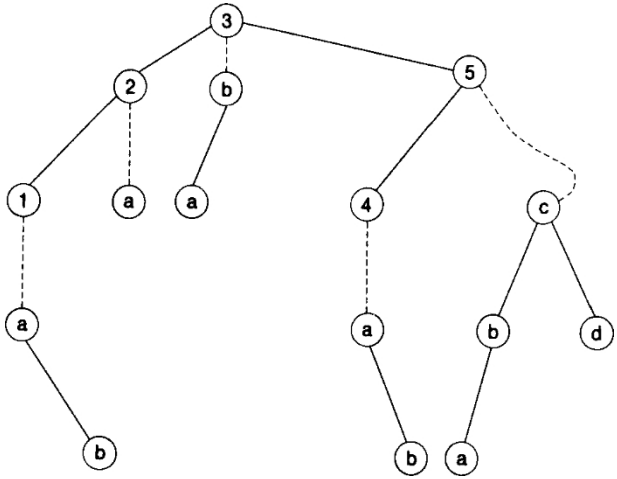
3.1 Fondo

Vaishnavi (1984, 1987) inició recientemente una nueva dirección de investigación para desarrollar árboles de búsqueda equilibrados autoorganizativos y multidimensionales que soporten diversas operaciones mediante algoritmos relativamente sencillos de entender e implementar. Esta investigación consiste en generalizar los árboles de búsqueda equilibrados utilizando un nuevo paradigma para soportar una violación de estructura mediante un subárbol EQSON.

En esta sección, discutiremos ciertos conceptos y temas básicos, así como revisaremos aplicaciones relacionadas en tiempo real para árboles multidimensionales equilibrados y ponderados. Los paradigmas descritos en esta sección proporcionan un método para generalizar las estructuras de árboles de búsqueda equilibrados a multidimensiones.

3.1.1 Árboles (de búsqueda) k-dimensionales

Un *árbol (de búsqueda) k-dimensional,* como se ilustra en la Fig. 1, es una estructura de datos natural para datos k-dimensionales. Está relacionado con los TRIE desarrollados por Fredkin (1960), los árboles de búsqueda digital de Knuth (1974) y los árboles lexicográficos de Sleator y Tarjan (1985). También puede interpretarse como una generalización *k-dimensional*



Fio. 1. Un árbol de búsqueda bidimensional para la siguiente selección de datos : ((1, a), (1, b), (2, a), (3, a), (3, b), (4, a), (4, b), (5, a), (5, b), (5, c), (5, d)}. Para el elemento de datos (1, a), el primer atributo es "a" y el segundo atributo es "1".

de un árbol de búsqueda (unidimensional). En tal árbol (ternario), como tenemos lo siguiente (Bentley y SaKe (1979): La raíz almacena x" una clave k-dimensional, el atributo k" de un elemento de datos. La estructura izquierda de la raíz es un árbol k-dimensional para aquellos elementos de datos cuyos atributos *k"* son menores que x . El subárbol derecho de la raíz es un árbol k-dimensional para aquellos elementos de datos cuyos atributos *k"* son mayores que x . El subárbol medio de la raíz, un *subárbol EQSON,* es un árbol k-dimensional.

\Árbol (k - 1)-dimensional para los elementos de datos de proyección *(k* - 1)-dimensionales omitiendo *-k* el atributo idéntico *k"* de los elementos de datos. Un árbol de hojas k-dimensional (binario) (útil para el procesamiento secuencial) y un árbol de hojas *k-*

Un árbol multi(útil para datos muy grandes) puede definirse de forma similar. En el peor de los casos, el rendimiento por operación de un árbol k-dimensional es bastante pobre a menos que su estructura esté adecuadamente restringida. *Un árbol (de búsqueda) k-dimensional equilibrado* es una estructura de árbol restringida de tal forma que su altura es como máximo logarítmica, es decir, 0(log n + *k),* lo cual es óptimo tal y como demostraron Bentley y Saxe (1979) y Hirschberg(1980), y es una estructura de árbol lo suficientemente flexible para que pueda actualizarse en el peor de los casos en tiempo logarítmico por operación.

3.1.2 Árboles ponderados

Cuando los elementos de datos (k-dimensionales) tienen *pesos* (números reales positivos) , los elementos de datos se denominan *elementos* ***ponderados***. Un árbol k-dimensional para tales ítems ponderados es un *árbol (k-dimensional) ponderado.* Una interpretación obvia del peso de un elemento de datos es que representa la

probabilidad con la que se accede al elemento. Un árbol ponderado, con pesos que representan las probabilidades de acceso, que se adapta dinámicamente al patrón de acceso (posiblemente no uniforme) es un *árbol autoorganizado.* También puede haber otras interpretaciones de los pesos, como la utilizada en la aplicación los árboles D desarrollados por Mehlhorn (1984a) para la dinamización de árboles de intervalos (véase Edelsbrunner (1980a), McCreight (1980)) y árboles de segmentos (véase Bentley (1977)).

En un árbol autoorganizado, dos operaciones adicionales, a saber, *promover* (aumentar el peso de un elemento en una cantidad determinada) y *degradar* (disminuir el peso de un elemento en una cantidad determinada), adquieren relevancia. Lo mejor que puede esperar de un árbol autoorganizado (k-dimensional) es que su tiempo de acceso en el peor de los casos sea "logarítmico", es decir, *O)og* si/w; + *k),* donde *w;* es el peso del elemento *i* en un instante de tiempo determinado e If es la suma de los pesos de todos los elementos en ese . Para un tratamiento más amplio de esta cuestión, véanse Gueting y Kriegel (1951) y Mehlhorn (1984a). El tiempo en el peor de los casos para otras operaciones puede ser, en el mejor de los casos, logarítmico. Se puede acceder a un *árbol* ponderado *equilibrado (k-dimensional*) y actualizarlo en el peor tiempo logarítmico por operación.

3.2 Árboles multidimensionales y árboles ponderados

Los árboles ponderados y los árboles multidimensionales están relacionados entre sí. puede partir de una estructura arbórea unidimensional ponderada eficiente y utilizarla para construir estructuras arbóreas multidimensionales eficientes o incluso estructuras arbóreas multidimensionales ponderadas eficientes utilizándolas adecuadamente para implementar nodos de TRIE desarrollados por Fredkin (1960). Alternativamente, se puede partir de una estructura arbórea eficiente (k + 1)-dimensional y utilizarla para construir una estructura arbórea eficiente ponderada k-dimensional dejando que los subárboles del nivel más bajo se utilicen como subárboles virtuales que representan pesos de elementos (véase Gueting y Kriegel (1981)).

3.2.1 Comportamiento I/l/orst-Case vs. Amortizado

Muchas aplicaciones requieren el uso de una secuencia de . En tales casos, no es importante que el tiempo en el peor de los casos por operación sea pequeño; basta con que los tiempos de operación *amortizados* en una secuencia de operaciones en el peor de los casos sean pequeños (véanse Bent et at. (1980) y Huddleston y Mehlhorn (1962)). El tiempo amortizado por operación también es una buena medida del rendimiento medio de una estructura de datos, como demuestran Bentley y McGeoch (1985). Sin embargo, hay situaciones en las que es importante que el tiempo por operación en el peor de los casos sea pequeño. Una aplicación en tiempo real es un ejemplo de este tipo de situación. Además, un tiempo por operación en el peor de los casos pequeño a veces se traduce en un tiempo amortizado por operación aún menor. Además, un

El buen comportamiento en el peor de los casos de una estructura de datos que es la base de una estructura de datos más compleja a veces puede ser importante para un buen comportamiento amortizado de esta última (véase, por ejemplo, los árboles dinámicos de intervalos y segmentos basados en árboles D Mehlhorn (1984b)). Aquí nos centraremos principalmente en los árboles equilibrados multidimensionales y ponderados, que son estructuras de datos eficientes en el peor de los casos. El árbol Splay, presentado por Sleator y Tarjan (1983a), es una elegante estructura de datos que garantiza un comportamiento eficiente en el peor de los casos tanto para las operaciones de acceso como de actualización.

3.3 Aplicaciones

Los árboles equilibrados multidimensionales y ponderados tienen aplicaciones en ámbitos como (a) la organización física de bases de datos, (b) la recuperación de información, (c) las estructuras de archivos autoorganizadas y (d) la geometría computacional. Pueden utilizarse como estructuras de datos independientes (véase Bent *et al.* (1985), Gueting y Kriegel (1980) y Gueting y Kriegel (1981)), o como esquemas subyacentes para estructuras de datos complejas (Mehlhorn (1984b), Sleator y Tarjan (1983b), o pueden utilizarse en una forma convenientemente mejorada en aplicaciones complejas (Kriegel (1984), Kriegel y Vaishnavi (1951c), y Scheuermann y Ouksel (1982)). Profundicemos en estas aplicaciones.

3.3.1 Organización física de la base de datos

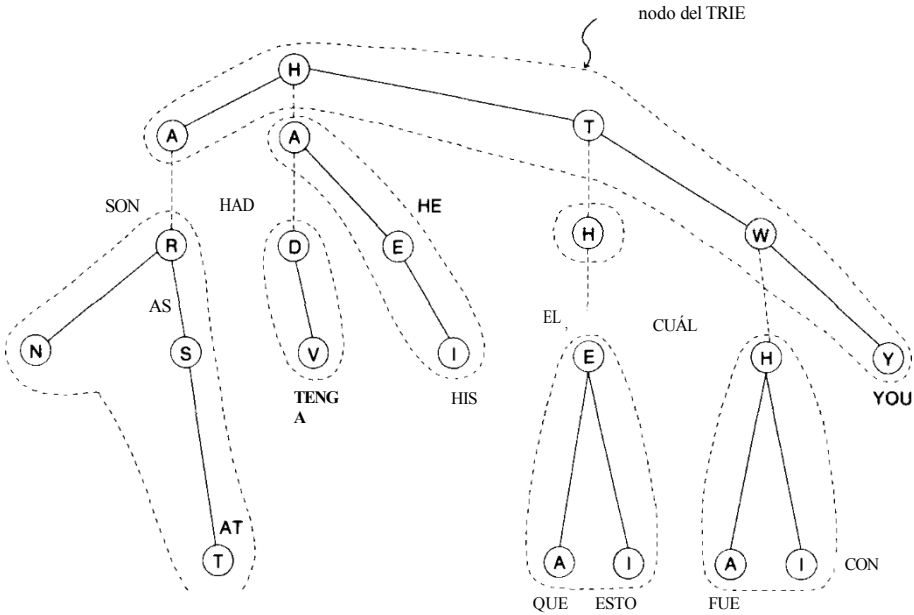
El problema de recuperar todos los registros que satisfacen una consulta que implica una multiplicidad de atributos, conocido como recuperación multiclave o recuperación asociativa, es una de las principales preocupaciones en la organización física de bases de datos (Kriegel (1984)). El uso cada vez más extendido de los sistemas de bases de datos para la toma de decisiones ejecutivas exige una organización física que pueda dar respuestas rápidas a las consultas asociativas, así como actualizaciones eficientes. Las estructuras de archivos disponibles (por ejemplo, el archivo invertido de Knuth (1974), el archivo de cuadrícula de Nievergelt *et al.* (1984), la organización M DBT de Scheuermann y Ouksel (1982), los árboles kB de Gueting y Kriegel (1980) y Kriegel (1984)) adolecen de ciertas deficiencias, especialmente si la base de datos es muy dinámica y las consultas asociativas, como las consultas de rango y las consultas de rango parcial, son importantes.

Una estructura de árbol multidimensional multidireccional equilibrada, convenientemente mejorada con punteros adicionales como en la organización M DBT de Scheuermann y Ouksel (1982), puede servir como organización dinámica de archivos para la recuperación asociativa. Se ha comparado el rendimiento de los árboles *k* B de Gueting y Kriegel (1980) y Kriegel (1984), una estructura de árbol multidimensional multidireccional equilibrada (más información en la Sección 5) (véase Kriegel (1984)) con el archivo invertido (Knuth (1974)), el archivo de cuadrícula (Nievergelt et al. (1973)) y la organización MDBT (Scheuermann y Ouksel (1982)). El archivo invertido tiene

Mientras que el archivo grid ha demostrado tener el mejor rendimiento en algunos aspectos, los árboles kB tener el mejor rendimiento en otros aspectos.

*3.3.2 **Recuperación de** información*

Un árbol (de búsqueda) multidimensional equilibrado puede servir como implementación eficiente en el espacio de un TRIE desarrollado por Fredkin (1960) o del árbol de búsqueda digital de Knuth (1974), una estructura de datos utilizada habitualmente en la recuperación de información (véase la Fig. 2). Si cada nodo de un TRIE para un conjunto de palabras en inglés se como una matriz, entonces se puede recuperar, insertar o eliminar una palabra en un tiempo proporcional a la longitud de la palabra. Pero esta implementación es bastante derrochadora en almacenamiento. Por otro lado, una implementación basada en una estructura de árbol multidimensional equilibrada ocupa un espacio de almacenamiento proporcional al número total de caracteres de las palabras y permite la recuperación y actualización en un tiempo a la longitud de la palabra más log n, donde *n* es el número total de palabras. Este rendimiento es óptimo, como demostró Hirschberg (1980), si la comparación entre caracteres es una operación unitaria.



Fiyi. 2. A TRl E para las palabras inglesas (AN, A RE, AS, AT, HAD, HAVE, HE, HIS, THAT. THE, THIS. WAS, WH, WITH, YOU}.

3.3.3 3Estructuras de archivos autoorganizadas

Si los pesos de los elementos de datos se interpretan como frecuencias con las que se accede a ellos, entonces una estructura de árbol ponderada equilibrada es una estructura de archivo dinámica autoorganizada que se adapta al uso. Se mantiene en una forma casi óptima con respecto a las frecuencias de acceso, sin necesidad de información a *priori* sobre estas frecuencias y sin utilizar contadores, etc., para registrarlas. Así, una estructura de árbol multidimensional ponderada y equilibrada puede utilizarse como estructura de archivo autoorganizada en bases de datos (véase Kriegel y Vaishnavi (198 lc)) o en la recuperación de información.

3.3.4 Geometría computacional

La geometría computacional se ocupa de los aspectos algorítmicos de los problemas geométricos. Los problemas computacionales que surgen en los gráficos por ordenador bidimensionales y tridimensionales y en el diseño VLSI han generado mucho interés en este campo.

Los árboles de intervalos (Edelsbrunner (1980a), McCreight (1980)) y los árboles de segmentos (Bentley (1977)) son algunas de las nuevas estructuras de datos que se han descubierto para desarrollar algoritmos eficientes para problemas relacionados con la geometría computacional. Sin embargo, son estructuras de datos estáticas; su estructura de datos subyacente es un árbol binario completamente equilibrado. Sin embargo, pueden hacerse dinámicas utilizando una estructura de árbol ponderada equilibrada adecuada como estructura de datos subyacente e interpretando los pesos adecuadamente. Los árboles D (Mehlhorn (1979)) (que se con más detalle en la sección 4), una estructura de árbol ponderada equilibrada que tiene un límite bastante fuerte en su tiempo de actualización, se han utilizado con éxito para la dinamización de árboles de intervalos y segmentos. Sin embargo, no es conveniente utilizar los árboles D porque requieren complejas operaciones de reestructuración en sus algoritmos de actualización.

4. Árboles equilibrados, multidimensionales y ponderados: Un análisis detallado de la bibliografía

Además de las estructuras de árbol multidimensionales y ponderadas equilibradas, incluimos en esta revisión las estructuras de árbol multidimensionales y ponderadas *semibalanceadas*: estructuras de árbol a las que se puede acceder en tiempo logarítmico en el peor de los casos, pero que sólo se pueden actualizar en tiempo logarítmico en el caso amortizado. Se han utilizado dos enfoques para descubrir estas estructuras de datos:

(a) extender una estructura de árbol equilibrada, y
(b) generalizar un concepto de equilibrio.

Las estructuras de datos desarrolladas, incluso utilizando el mismo enfoque, difieren en sus puntos fuertes básicos, su rendimiento y la simplicidad de sus algoritmos de actualización. Veamos con más detalle estos enfoques y las correspondientes estructuras de árbol que se han desarrollado.

4.1 Ampliación de una estructura de árbol equilibrada

En este enfoque, Mehlhorn (1979) amplió una estructura de árbol unidimensional equilibrada para manejar datos unidimensionales ponderados. La idea básica es que un elemento de datos con un gran peso está representado por un número de nodos hoja para mantener la restricción de equilibrio. La única estructura de árbol equilibrado que se ha ampliado con éxito utilizando este enfoque es un árbol equilibrado por peso (véase Nievergelt y Reingold (t 973)), dando lugar a los árboles D (Mehlhorn (1979)). Los árboles D admiten el acceso, la inserción, la eliminación, la promoción y la degradación en tiempo logarítmico por operación en el peor de los casos. Sin embargo, requieren un almacenamiento no lineal el uso de complicadas operaciones de reestructuración. En el lado positivo, heredan límites bastante fuertes en su coste de actualización de los árboles de peso equilibrado, lo que los hace útiles como estructuras de datos subyacentes para intervalos dinámicos y árboles de segmentos (Mehlhorn (1984b)).

4.2 Generalización de un concepto de equilibrio

Este enfoque consiste en generalizar los árboles equilibrados (unidimensionales) al caso multidimensional o de elementos ponderados. Se han generalizado las principales estructuras de árboles equilibrados, como los árboles B (véase Bayer y McCreight (1972)), los árboles equilibrados por peso (véase Nievergelt y Reingold (1973)) y los árboles AVL (Adel'son-Velskij y Landis (1962)). Sin embargo, debido a las diversas formas en que puede generalizarse un concepto de equilibrio, se han propuesto e investigado diversas estructuras de árbol (véase Bent (1982), Bent, Sleator y Tarjan (1980, 1985), Feigenbaum y Tarjan (1983), Gueting y Kriegel (1980), Kriegel y Vaishnavi (198 t a), Vaishnavi (1984, 1987)). La Fig. 3 resume estas estructuras de árbol. Los algoritmos para *k* árboles B (Gueting y Kriegel (1980), Gueting y Kriegel (198 l), Kriegel y Vaishnavi (198 l b)) y *k* árboles *B*+ (Kriegel y Vaishnavi (l98la)) implican una multiplicidad de casos y operaciones de restructuración tan complejas como "Equilibrar con un hermano indirecto" y "Colapsar con un hermano indirecto" (Gueting y Kriegel (1980)). Por otro lado, un atributo importante de los árboles AVL multidimensionales (Vaishnavi (1983b, 1984)) es que heredan una serie de propiedades importantes de la estructura de datos base, es decir, los árboles AVL. Sus algoritmos de actualización implican casi las mismas operaciones de reestructuración que los utilizados en los árboles AVL. Sus algoritmos de inserción y eliminación son relativamente sencillos. Al igual que los árboles AVL, la inserción de un dato requiere como máximo una operación de reestructuración. El sitio

Estructuras de árboles	Conceptos de equilibrio de los árboles B	Conceptos de equilibrio o árboles de peso equilibrado	Conceptos de equilibrio de árboles AVL	Algoritmos directos para insertar/borrar	Algoritmos directos para unir/dividir
k Árboles B	X			sí; logarítmico en el peor de los casos, pero complejo	
kB″ -trees Sesgo	X			igual que arriba	
global 2, fi árboles	X				sí; peor caso logarítmico
Árboles binarios globalmente sesgados	X				igual que arriba
Glo bally biased n, *b* t rees	X				igual que arriba
Sesgo 2-3 árboles	X				sí; amortizado logarítmicamente
Árboles 2, b sesgados	X				igual que arriba
Árboles binarios sesgados	X				igual que igual
Árboles *a, b* localmente sesgados	X				que arriba igual
Peso sesgado - árboles equilibrados		X			que arriba
Árboles con equilibrio pseudoponderal		X			igual que arriba
Árboles AVL multidimensionales			X	sí, logarítmico en el peor de los casos	
Árboles AVL de hojas ponderadas			X	igual que arriba	

Fic. 3. El resumen de las estructuras arbóreas generaliza los conceptos de equilibrio y requiere un tiempo de acceso logarítmico en el peor de los casos.

La simplicidad de actualización de esta estructura de árbol no va en detrimento de su tiempo de acceso (Vaishnavi (1986)). Los árboles AVL de hojas ponderadas (Vaishnavi (1987)) también se caracterizan por algoritmos de actualización bastante sencillos que reflejan los algoritmos de los árboles AVL correspondientes.

5. Paradigmas para el modelado equilibrado **multidimensional y** Árboles ponderados

En esta sección, describimos un paradigma que modela las estructuras multidimensionales y ponderadas equilibradas y semilibradas disponibles. Se espera que el paradigma sirva de guía para futuras investigaciones en este ámbito.

El paradigma consiste en una estrategia utilizada para generalizar una estructura de árbol equilibrada a dimensiones superiores, junto con un conjunto de estrategias para soportar "violaciones de la estructura" en la estructura generalizada. A continuación describiremos e ilustraremos el paradigma.

5.1 El paradigma de la estrategia de generalización

En la estructura generalizada, se utiliza el mismo concepto de equilibrio que en la estructura base, salvo que se permiten "violaciones de la estructura" que estén "soportadas" (véase Vaishnavi (1984)) por determinado(s) subárbol(es) EQSON especificado(s). La última parte de la estrategia, es decir, el significado de "soporte" y la especificación de subárbol(es) EQSON que deben soportar una violación de la estructura, se concreta de tal manera que la altura en el peor de los casos de la estructura generalizada es logarítmica y se pueden diseñar algoritmos de actualización logarítmicos.

5.2 Paradigma de la estrategia de violación de la estructura de apoyo

Hemos identificado tres estrategias distintas que pueden utilizarse para apoyar las violaciones de la estructura y que modelan (junto con el paradigma general) la estructura disponible. Estas estrategias pueden denominarse "estrategia de soporte local" (véase, por ejemplo, Bent (1982), Bent et al. (1980, 1985), "estrategia de soporte global" (véase Bent et al. (1985), Feigenbaum y Tarjan (1983), Gueting y Kriegel (198 I)) y "estrategia de soporte vecino" (véase, por ejemplo, Vaishnavi (1984, 1987)). La estrategia de apoyo local conduce a algoritmos de actualización que son logarítmicos en tiempo amortizado por operación. Las otras dos estrategias conducen a algoritmos de actualización que son logarítmicos en el peor caso de tiempo por operación.

En cada una de las tres estrategias anteriores, una "violación de la estructura" debe estar "respaldada" por determinado(s) "subárbol(es) inmediatamente inferior(es)". Las estrategias
difieren en la especificación del subárbol o subárboles que deben "soportar" la violación de la estructura. En la estrategia de soporte local, la violación de la estructura debe estar "soportada" por

cada uno de los subárboles *hermanos* (de dimensión inmediatamente inferior) de los nodos causantes de la violación de la estructura. En *estrategia* de *apoyo global,* la violación de la estructura debe ser "apoyada" por *cada uno* los subárboles *adyacentes* del nodo o nodos causantes de la violación de la estructura. En la *estrategia de apoyo a los vecinos,* la violación de la estructura debe ser "apoyada" por al *menos uno de* subárboles *"colindantes"* del nodo o nodos causantes de la violación de la estructura.

A continuación describiremos brevemente la estructura de los árboles kB (véase Gueting y Kriegel (1980)), y de los árboles AVL k-dimensionales (Vaishnavi (1984)), para ilustrar la estrategia de generalización descrita anteriormente junto con las estrategias de soporte global y soporte vecino, respectivamente, para soportar violaciones de la estructura.

5.3 Paradigma de estrategia de apoyo global para árboles kB

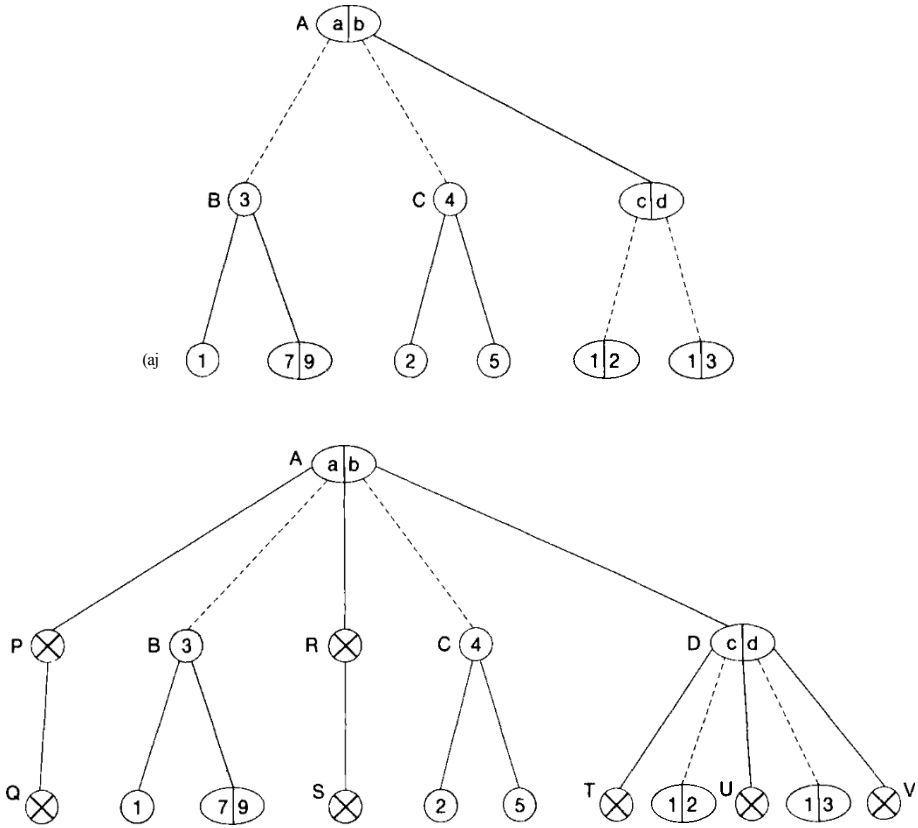
kLos árboles B de orden m, en 1, son generalizaciones de orden superior de los árboles B de orden m. Por lo tanto, los árboles lB de orden m son lo mismo que los árboles B de orden m. En un árbol B de orden in, todos los nodos excepto la raíz contienen s claves, *d* ñ *s* <*p* 2d. En un árbol kB, sin embargo, puede haber nodos que tengan menos de d claves (que constituyen violaciones de la estructura) siempre que las violaciones de la estructura estén "soportadas" por *cada uno* de los "subárboles EQSON de dimensión inmediatamente inferior que colindan" con los nodos que causan las violaciones de la estructura. Una violación de la estructura causada por un nodo *N* a la altura Ii es *soportada* por un subárbol si la altura del subárbol es igual o superior a It.

La Fig. 4(a) muestra un árbol 2B de orden 1, es decir, un árbol 2-3 bidimensional. El árbol almacena los elementos de datos:

(a,1), (a,3), (a,7), (a,9), (b,2), (b,4), (b,5), (c,1), (c,2), (d, l) y (d,3).

Como en un árbol 2-3, un nodo, *N,* que almacena *r* claves tiene r + I subárboles, todos los cuales almacenan claves de la misma dimensión que las almacenadas en *N.* Además, cada clave bidimensional de un nodo tiene un subárbol EQSON unidimensional. Así (en la Fig. 4a) los subárboles enraizados en B y C son subárboles EQSON de las claves *a* y *b,* respectivamente. Obsérvese que los subárboles izquierdo y medio del nodo A están vacíos.

La Fig. 4(b) muestra el mismo árbol 2-3 bidimensional de la Fig. 4a, salvo que cada subárbol vacío izquierdo, central o derecho de un nodo *N* de altura /i está representado por una cadena de nodos unarios #-1 que almacenan claves cero (mostrados en la figura como nodos cruzados). Observe en la Fig. 4b que el subárbol enraizado en B es un "subárbol adyacente de la dimensión inmediatamente inferior" de los nodos P, Q, R y S. De forma similar, el subárbol enraizado en C es un "subárbol adyacente de la dimensión inmediatamente inferior" de los nodos R, S, D y T. La violación de la estructura causada por los nodos P, Q, R, S, T, U y V está permitida porque cada una de estas violaciones de la estructura está soportada por cada uno de los correspondientes subárboles EQSON contiguos de la dimensión inmediatamente inferior.



Fic. 4. ta) muestra un árbol 2B de orden 1, decir, árbol 2-3 bidimensional; b) muestra el mismo árbol 2-3 bidimensional de la figura 4ta) excepto que cada subárbol vacío izquierdo, medio o derecho de un nodo N de altura h está representado por una cadena de h-1 nodos unarios que almacenan claves cero (mostrados en la figura como nodos cruzados).

5.4 Paradigma de la estrategia de apoyo a vecinos para árboles AVL k-dimensionales

Los árboles AVL k-dimensionales (árboles kAVL) son generalizaciones de orden superior de los árboles AVL. La figura 5a muestra un árbol 2AVL. En aras de la explicación, supongamos que si un nodo, M, de altura *h* t tiene un hijo izquierdo o derecho de altura, N, de altura *h*
h< *h;* - 1, entonces hay una cadena de /i, - i2 - I nodos unarios, cada uno almacenando claves cero, entre M y N. En particular, supongamos que cada subárbol vacío izquierdo o derecho de un nodo a la altura h está representado por una cadena de /i-1 nodos unarios. La Fig. 5b muestra el mismo árbol que en la Fig. 5a excepto por el cambio en su
representación.

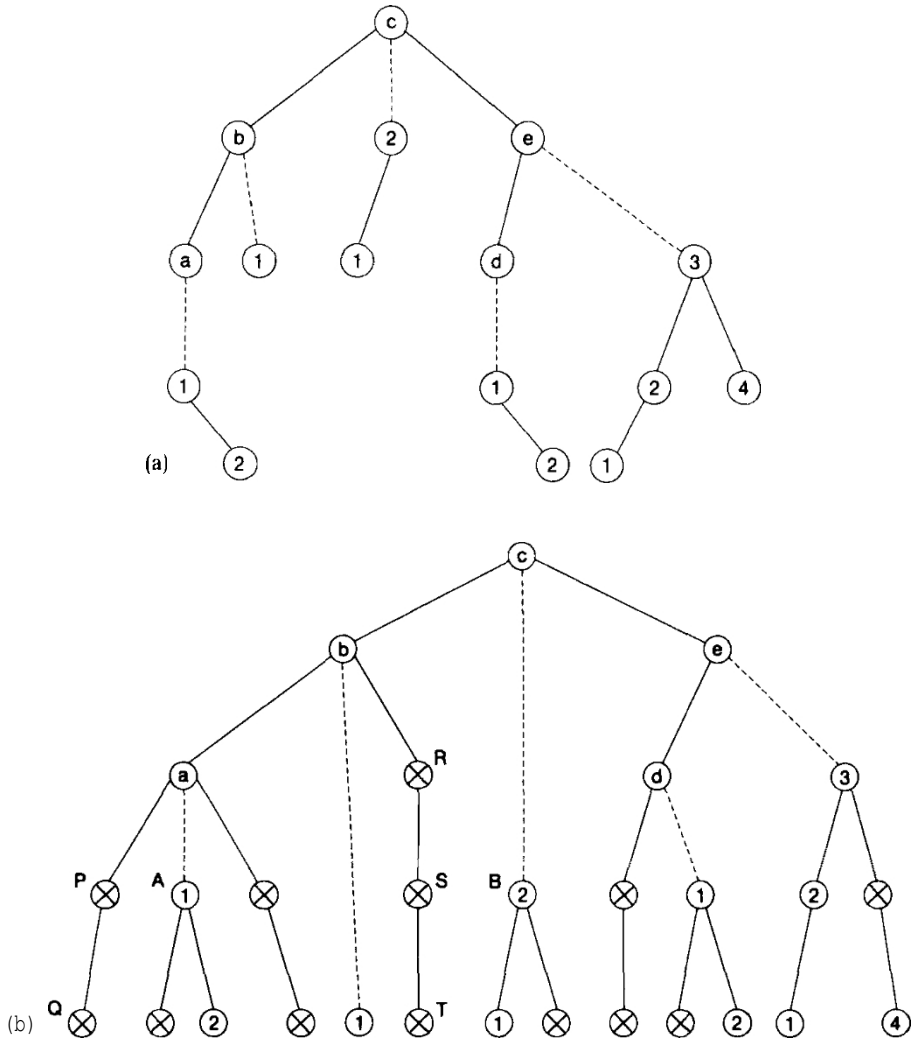


FiG. 5. (a) muestra un 2AYL-lree; (b) muestra el mismo árbol que en (a) excepto por el cambio en su represenlación.

Una cadena de *dos o más* nodos unarios constituye una ***violación de*** *estructura* en un árbol kAVL. Se permite una violación de la estructura en un árbol kAVL siempre que esté "soportada" por al menos uno de los "subárboles EQSON de dimensión inmediatamente inferior que sean contiguos" a los nodos de la cadena que constituye la violación de la estructura.

Sea *h* la altura del nodo más alto de una cadena de nodos unarios que constituyen una violación de estructura. La violación de la estructura *está soportada* por un subárbol si la altura del subárbol es igual o superior a Ii-1.

En la Fig. 5b, la violación de estructura causada por la cadena de nodos P y Q está soportada por el subárbol EQSON enraizado en A, un subárbol EQSON unidimensional adyacente a los nodos P y Q. De forma similar, la violación de estructura causada por la cadena de nodos R, S y T está soportada por el subárbol EQSON enraizado en B. Otras violaciones de estructura en el árbol están soportadas de forma similar por subárboles EQSON apropiados.

Obsérvese que no puede haber una violación de la estructura en un árbol 1AVL, es decir, un árbol AVL, porque no es posible soportarlo adecuadamente ya que las alturas de todos los subárboles EQSON de dimensión inmediatamente inferior son cero.

6. Árboles de atributos múltiples (X/IAT)

6.1 Preliminares

La estructura de datos abstracta básica que subyace al MAT se utiliza en diferentes aplicaciones de diferentes formas. Un esfuerzo temprano se debe a Cárdenas y Sagamang (I 974) que han extendido el concepto del árbol doblemente encadenado (DCT) a datos multidimensionales. Kashyap, Subas y Yao (1977) y Subas y Kashyap (1975) utilizan una noción similar, pero su estructura de datos difiere de la estructura de datos abstracta en la clasificación de los atributos según el orden decreciente de sus probabilidades de aparición en una consulta. Existen otras estructuras, como los árboles B multidimensionales (MDBT) de Ouksel y Scheuermann (1981), los árboles kB de Gueting y Kriegal (1980), y los árboles *kB+* y (k + I)B' de Kriegel y Vaishnavi (198 la) y Kriegel (1984)-todas estas estructuras de datos se basan en la misma estructura abstracta de datos. Presentamos una variación de esta estructura de datos abstracta, que posteriormente llamaremos *Árbol de Atributos Múltiples* (MAT). La estructura de datos, MAT, tal como se utiliza en este documento, es la misma que la estructura de datos abstracta antes mencionada, con la propiedad adicional de que cada conjunto filial está ordenado según los valores de los atributos. Es la misma que la MAT de Kashyap, Subas y Yao (1977) y Subas y Kashyap (1975), sin la restricción de ordenación de atributos. Mantenemos el porque en ambas estructuras se conserva la propiedad más importante, a saber, la agrupación de los nodos del árbol que tienen valores de atributo "cercanos".

Se demostrará que el MAT es tan eficiente como otras variantes, pero es especialmente adecuado para grandes estructuras de datos que se mantienen en almacenamiento secundario. Los detalles de estas estructuras se aplazan hasta la sección 6.2. Formalmente

definen la MAT del siguiente modo:

Definición de MAT. **Un** MAT #-dimensional sobre k-atributos, *A ,* A2. *Az* para un conjunto (fichero) de registros se define como un árbol de profundidad k, con lo siguiente
propiedades:

(i) Tiene una raíz en el nivel 0.

(ii) Cada hijo de la raíz es un MAT (k-1)-dimensional, sobre (k-1) atributos, *A*2- *A , ... , Ap,* para el subconjunto de los registros que tienen el mismo valor de *A*. Este valor de *A* es el valor de la raíz del correspondiente (k-1)-dimensional MAT.

(iii) Los nodos hijos de la raíz están en orden ascendente de sus valores. Este conjunto de nodos hijos se denomina *conjunto Strat.* ■

La Fig. 6a muestra un conjunto de registros, y la MAT correspondiente se muestra en la Fig. 6c. Observamos que hay un nodo raíz en el nivel 0, que no tiene ningún valor. Cada nodo de nivel i, i = 1, 2, ... , *k* corresponde a un valor del atributo *A;* y este valor es el valor del nodo. Así, los atributos *A , A*2. . , *A* forman la jerarquía de los niveles 1 a k. La propiedad importante de la MAT es que todo conjunto hlial está ordenado. Esto es el resultado de la propiedad de que la ordenación de un conjunto de registros en todos los atributos induce naturalmente una estructura de árbol, como se ilustra en la Fig. 6b. Por lo tanto, se puede imaginar que el MAT abstracto se ha construido de la siguiente manera:

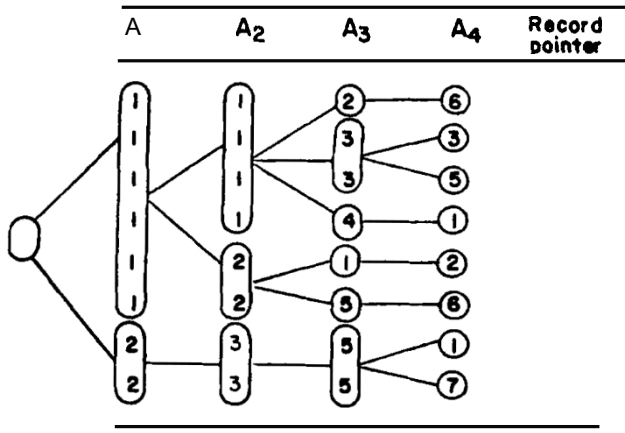
a) Los registros se clasifican en orden ascendente en todos los atributos.

b) Empezando por el primer atributo, todos los elementos de un atributo que tengan el mismo valor de atributo se combinan en un nodo. Este proceso se lleva a cabo de forma recursiva para cada subconjunto de registros que tengan el mismo primer valor de atributo.

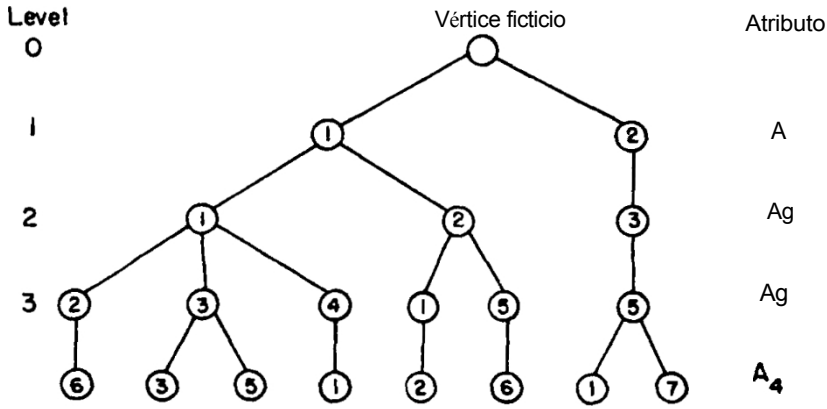
(a)

A,	Ag	Ay	A4	Record po1nter
1	2	1	2	2
1	1	2	6	4
2	3	5	7	5
1	2	5	6	7
2	3	5	I	8

Fic. 6. (a) Fichero de datos de entrada; (b) Fichero ordenado en put, e inducción de una estructura de árbol natural;
(c) Representación MAT de los datos de (a).



(b)



(c)

Flci. 6. *[Continue'l)*

La construcción del MAT puede visualizarse utilizando nociones geométricas. El sub-MAT en *A* 2. *A , .. . Ak,* generada como en ii) de la definición anterior, corresponde al conjunto de puntos que se encuentran en un hiperplano del espacio de datos. El
ecuación de este hiperplano es *A= a* es el valor de *A,* correspondiente a la raíz del sub MAT. Dado que un hiperplano en un espacio de dimensionalidad k corresponde a un espacio de dimensionalidad (L - 1), cada generación recursiva de un sub MAT se sobre un espacio de menor dimensionalidad. El mismo efecto se encontrará en los algoritmos de búsqueda, que buscan recursivamente sobre espacios de menor dimensionalidad.

Otra propiedad importante es que cada registro o punto está representado por una ruta *única* desde la raíz hasta el nodo terminal correspondiente. Debido a esta propiedad, el perfil del MAT, para un conjunto de registros, depende completamente de la ocurrencia de varios valores de atributo en los registros. En general, los tamaños de varios conjuntos de filiales dependen del conjunto exacto de registros de entrada y de los de atributo correspondientes. Como consecuencia de este , los tamaños de los conjuntos filiales son aleatorios. Por lo tanto, caracterizamos los tamaños de los conjuntos filiales del MAT mediante medidas medias. Este tipo de análisis es realizado para el MAT por Rao y lyengar (1986) y por Ouksel y Scheuermann (1981) para los árboles B multidimensionales. Estas medidas pueden corresponder a los valores esperados de los tamaños de los conjuntos filiales, si se conoce la distribución de probabilidad de los distintos atributos. La disparidad entre el tamaño real del conjunto filial y este valor medio puede tenerse en cuenta utilizando medidas de dispersión como la varianza, el tercer momento, etc.

continuación, utilizaremos la siguiente notación:

N: número de registros

M: número de nodos del MAT k:

número de atributos

s;: tamaño medio (esperado) del conjunto filial en el nivel j, j= 1, 2, ... , *k*

;: desviación típica (raíz cuadrada de la varianza) del conjunto filial en el nivel j= 1, 2, ... , *k*

dj: tamaño del conjunto del que pueden elegirse los valores del atributo *A j*, j = 1, 2, ... , *k*

Las propiedades de la MAT pueden resumirse como sigue:

=1

M ñ (1+ *kN)*

*N -- s sz ' *k*

$$N = \prod_{i} s_i \tag{6.1}$$

$$M - Z \sum_{I \quad t'=i}^{J} st \tag{6.2}$$

Cuando la distribución de los datos es simétrica sobre los atributos, caracterizamos el MAT como la versión simétrica, donde los tamaños medios de los conjuntos filiales de todos los niveles son iguales, es decir, s,= *sz* ----- -- *sz -- s, N -- s* . Esta caracterización es más apropiada cuando la distribución de los valores de cada atributo es independiente de

la distribución de otros valores, y uniforme con la misma media. Por ejemplo, dejemos que cada *A;* corresponda a una distribución independiente y uniforme con probabilidad de ocurrencia de cada valor de ser *p.* Entonces para un MAT simétrico tenemos *s -- pd,* y d, = *dz* - - *d -- d.* Para un MAT simétrico tenemos

$$sj -- 0 \backslash N'') \tag{6.3}$$

$$\prod^{k-1} s_j = N/s_k = O(N^{1-1/k}) \tag{6.4}$$

En las secciones siguientes, utilizamos la MAT simétrica para nuestro análisis. Sin pérdida de generalidad, las expresiones (6.3) y (6.4) se utilizan para estimar la complejidad de varios algoritmos. Se puede imaginar que un MAT para cualquier dato es equivalente a una versión simétrica con las estimaciones de orden que satisfacen (6.3) y (6.4). Para muchas situaciones prácticas, esta aproximación da medidas analíticas razonablemente buenas, como se informa en Veni Madhavan (1984), y Rao y lyengar (1986). Sin embargo, hay que tener en cuenta que puede haber ciertas aplicaciones en las que esta aproximación no sea completamente exacta. Pero esta es la aproximación que más se utiliza en la literatura, como en Kashyap et at. (1977) y Gopalakrishna y Veni Madhavan (1980). Un análisis completamente generalizado parece muy difícil en este momento.

6.2 **Estructuras de datos** multidimensionales **relacionadas con MAT**

Existen muchas variantes de la estructura de datos MAT. La mayoría de estas estructuras de datos difieren entre sí en la organización de los punteros o de los conjuntos filiales.

La estructura de datos propuesta por Cardinas y Sagamang (1974) es un perfeccionamiento del árbol doblemente encadenado (EDC) de Sussenguth (1963). En el modificado, cada nivel corresponde a un atributo, y la Fig. 7 muestra la idea básica del árbol doblemente encadenado para los datos de la Fig. 6a. En esta estructura de datos se utilizan más punteros para facilitar la búsqueda y otras operaciones. Una desventaja del SES es que la longitud del camino correspondiente a un único registro puede ser como máximo *N* + k, frente a *k* para las demás estructuras de datos como el MAT, etc. En el peor de los , el camino de búsqueda para una consulta de coincidencia completa puede contener hasta *N* + *k* nodos. Kashyap et al (1977) han demostrado la superioridad de la MAT sobre la DCT modificada.

Las otras estructuras como el M DBT, kB-tree, etc. son análogas al MAT en la básica de los datos. Pero difieren en la organización de los conjuntos filiales. La organización de los conjuntos filiales desempeña un papel importante en el rendimiento de la estructura de datos. El complejo de árbol de búsqueda binario (complejo BST) de Lien *et al.* (1975) es una de las primeras estructuras de este . El