Clases auxiliares de indexación

CC42A/CC55A - Bases de datos – Primavera de 2005

Profesor: Claudio Gutiérrez
Auxiliar: Mauricio Monsalve

1 Indexación en árboles

Se vio en clases varios tipos de árboles que sirven para indexar (o indizar) las tablas de una base de datos. Se dio especial énfasis a los árboles B+ o B+Tree. Los árboles B son una estructura de datos similar al árbol 2-3. En realidad se trata de un árbol con nodos de "tamaño impar" en los cuales hay k claves de búsqueda y $k+1$ punteros a nodos inferiores. Cuando se realiza una búsqueda sobre una clave T, en el nodo se entran a comparar las claves hasta que se encuentran dos claves contiguas C_{j-1} y C_j tales que $C_{j-1}<T<C_j$. Entonces se usa el puntero P_j para ir al siguiente nodo y continuar la búsqueda. En los casos de borde se hace la comparación con la clave de búsqueda más cercana. La siguiente figura ilustra el árbol B+:

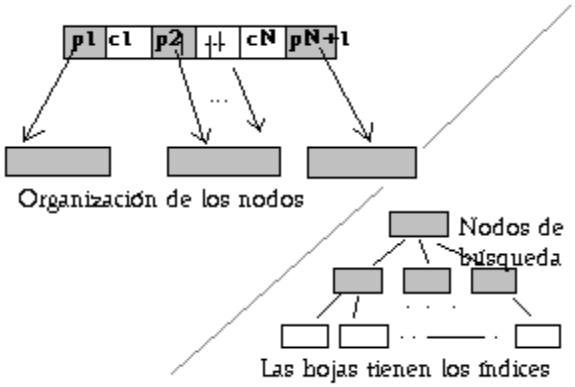

Figura: Organización de los árboles B+.

También se vio el caso de los árboles R, útiles para varias dimensiones. Estos árboles tienen el mismo costo asintótico que los árboles B+ (costo logarítmico de búsqueda). Tanto árboles B+ como árboles R son soportados en la base de datos PostgreSQL.

Otros árboles vistos son los árboles kd o kd-trees, con nodos alternantes (distinta naturaleza de claves de búsqueda), y los quad-trees, que son muy similares a los árboles R.

A continuación se muestra un ejemplo clásico sobre la búsqueda en árboles B+:

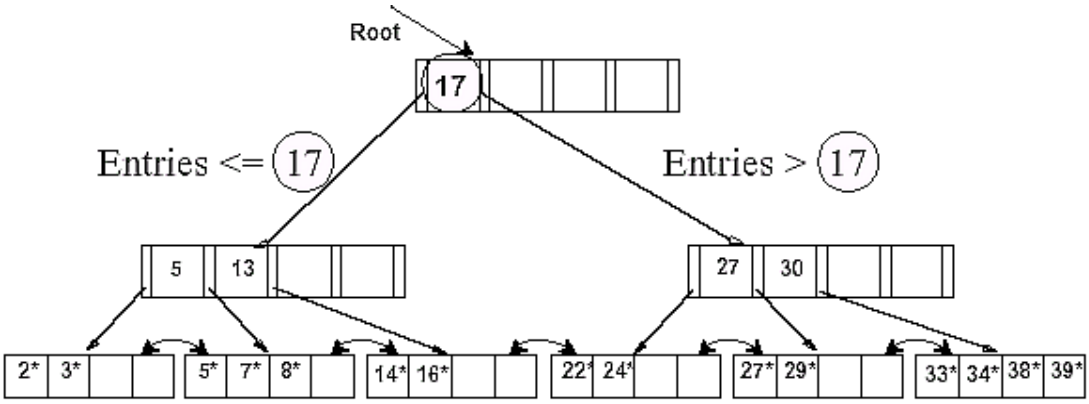

Figura: Búsqueda en árboles B+.

2 Indexación en tablas de hashing

Se utiliza una función de hashing para agrupar los índices de manera que las claves de búsqueda resulten en "clases" de claves de búsqueda. La función de hashing arroja un número y según ese número se agrupan los índices en una tabla. Esto facilita mucho la búsqueda en igualdad pero no apoya para nada la búsqueda en rango (range-search).

A continuación se muestra un diagrama con la estructura de un índice de hashing:

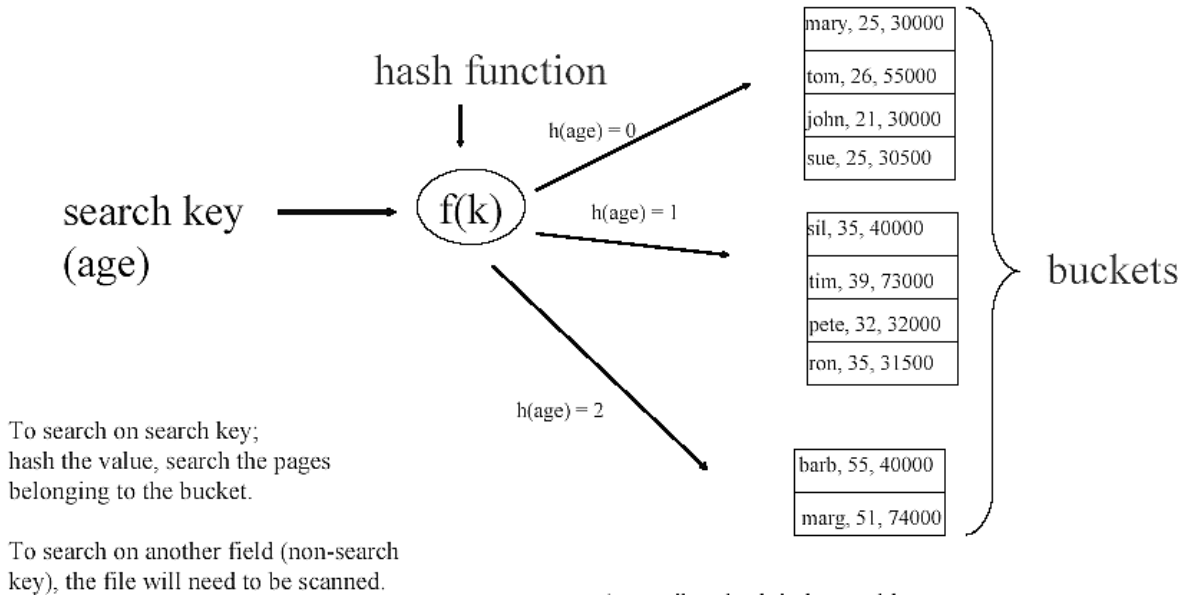

Figura: Organización de un índice de hashing.

3 Otros índices

En clase se vieron los *bitmaps,* índices en mapa de bits con la particularidad de ser extremadamente rápidos para consulta, pero extremadamente caros en construcción.

4 Costos de estos índices

La siguiente tabla explicita una estimación de los costos de operación en una tabla de una vase de datos, a base de ordenes asintóticos y varias constantes despejadas estadísticamente o parametrizadas:

	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) Delete
(1) Heap	BD	0.5BD	BD	2D	Search +D
(2) Sorted	BD	Dlog 2B	Dlog 2 B + # matches	Search + BD	Search +BD
(3) Clustered	1.5BD	Dlog F 1.5B	Dlog F 1.5B +# matches	Search + D	Search +D
(4) Unclustered Tree index	BD(R+0.15)	D(1 + log F 0.15B)	Dlog F 0.15B +# matches	D(3 + log F 0.15B)	Search + 2D
(5) Unclustered Hash index	BD(R+0.125)	2D	BD	4D	Search + 2D

Figura: Costos en cosulta y modificación de un tabla.

Heap es el archivo o tabla desordenado. Sorted es el caso ordenado. Clustered es el caso "agrupado". Es muy similar al ordenado, sólo que es un archivo ordenado por partes, con páginas que tienen espacio libre que evitan modificar todo el archivo cuando hay modificación.

El caso de los índices no usa agrupación para mostrar los costos inherentes de estas organizaciones.

En cuanto a las operaciones, Scan es la revisión completa de la tabla. Equality es la búsqueda por igualdad. Range es la búsqueda en rango (conjuntos). Insert es la inserción y Delete es el borrado, ambos sobre un registro o tupla.

En cuanto a los costes, son estimaciones gruesas. Pero son una aproximación relativamente buena. D es el tiempo de acceder a un registro en la base de datos (un ponderador de costo). B es el número de registros en el archivo o tabla. F es el tamaño del bloque o página.

5 Planes de índices

Un punto importante, sin duda, en el tema de bases de datos. Un plan de índices es una medida que indica qué índices deben ser aplicados a qué tablas sobre qué campos, atributos o columnas.

Tipo de índices: Hashing es muy bueno con búsqueda de igualdad, pero no sirve

con rangos. La búsqueda con rangos se da, por lo general, sobre campos con significación. Los árboles B+ son buenos en cuanto a búsqueda por rangos.

Frecuencia: Es necesario analizar las consultas más frecuentes para saber si aplicar índices o no. ¿Qué campos son los más consultados? ¿Qué campos son los más actualizados?

Claves de búsqueda compuesta: Pueden ser muy beneficiosas, pero requieren más actualización. Ojo con el orden: <k,d> no siempre da los mismos resultados que <d,k>.

Ordenamiento: ¿Cuándo ordenar? Aunque generalmente el ordenamiento en las bases de datos no es perfecto, tiene sus beneficios. Pero, por lo general, es mejor usar árboles.