Escuela Superior de Computo

Grupo: 2CV2

Bases de datos

M. en C. Euler Hernández Contreras

2° Parcial

Lunes 09 de Abril de 2018

Tarea 1: Indices

Alumno: Aaron Antonio Garcia Gonzalez

Índice

Desarrollo

Indices

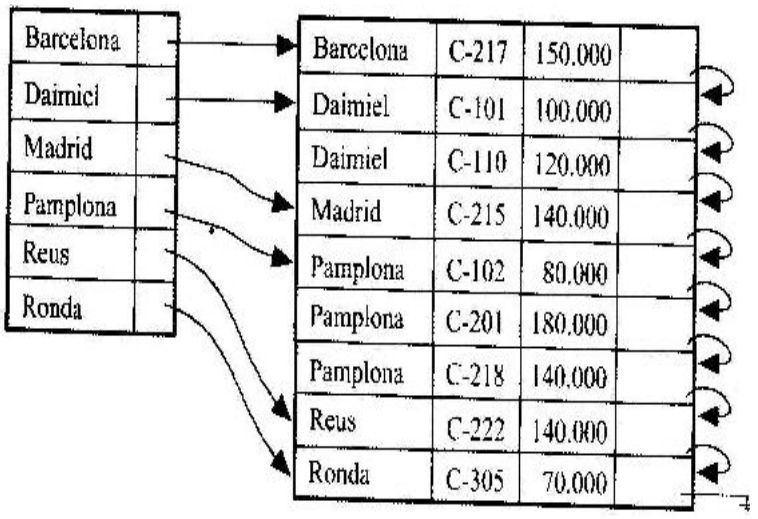
La idea tras la estructura de acceso de un índice ordenado es parecida a la que hay tras el índice de un libro, que enumera al final de la obra los términos importantes ordenados alfabéticamente, junto con las páginas en las que aparecen. Podemos buscar en un índice en busca de una lista de direcciones (números de páginas en este caso) y utilizar esas direcciones para localizar un término en el libro, buscando en las páginas especificadas.

La alternativa, de no indicarse lo contrario, sería desplazarse lentamente por todo el libro, palabra por palabra, hasta encontrar el término en el que estuviéramos interesados; sería como hacer una búsqueda lineal en un fichero. Por supuesto, la mayoría de los libros tienen información adicional, como títulos de capítulo y de sección, que nos ayuda a encontrar el término sin tener que buscar por todo el libro. Sin embargo, el índice es la única indicación exacta del lugar del libro donde se encuentra cada término.

En un fichero con una estructura de registro dada compuesta por varios campos (o atributos), normalmente se define una estructura de índice con un solo campo del fichero, que se conoce como campo de indexación (o atributo de indexación). Normalmente, el índice almacena todos los valores del campo de índice, junto con una lista de punteros a todos los bloques de disco que contienen los registros con ese valor de campo. Los valores del índice están ordenados, de modo que podemos hacer una búsqueda binaria en el índice. El fichero de índice es mucho más pequeño que el fichero de datos, por 10 que la búsqueda en el índice mediante una búsqueda binaria es razonablemente eficaz. La indexación multinivel anula la necesidad de una búsqueda binaria a expensas de crear índices al propio índice.Hay varios tipos de índices ordenados.

Indice principal o primario

Un índice principal es un fichero ordenado cuyos registros son de longitud fija con dos campos. El primer campo es del mismo tipo de datos que el campo clave de ordenación (denominado clave principal, PRIMARY KEY) del fichero de datos, y el segundo campo es un puntero a un bloque del disco (una dirección de bloque). En el fichero índice hay una entrada de índice (o registro de índice) por cada bloque del fichero de datos. Cada entrada del índice tiene dos valores: el valor del campo clave principal para el primer registro de un bloque, y un puntero a ese bloque. Nos referiremos a los dos valores de la entrada i del índice como <K(i), P(i)>



Clasificacion de indices

Los índices se pueden clasificar como densos o escasos. Un índice denso tiene una entrada de índice por cada valor de la clave de búsqueda (y, por tanto, cada registro) del fichero de datos. Un índice escaso (o no denso) sólo tiene entradas para algunos de los valores de búsqueda. Por consiguiente, un Índice principal es un índice escaso, porque incluye una entrada por cada bloque de disco del fichero de datos y las claves de su registro ancla, en lugar de una entrada por cada valor de búsqueda (es decir, por cada registro).

El fichero de Índice para un índice principal necesita menos bloques que el fichero de datos, por dos razones. En primer lugar, hay menos entradas de índice que registros en el fichero de datos. En segundo lugar, cada entrada del índice tiene normalmente un tamai'ío más pequei'io que un registro de datos, porque sólo tiene dos campos; en consecuencia, en un bloque entran más entradas de índice que registros de datos. Por tanto, una búsqueda binaria en el fichero de índice requiere menos accesos a bloques que una búsqueda binaria en el fichero de datos.

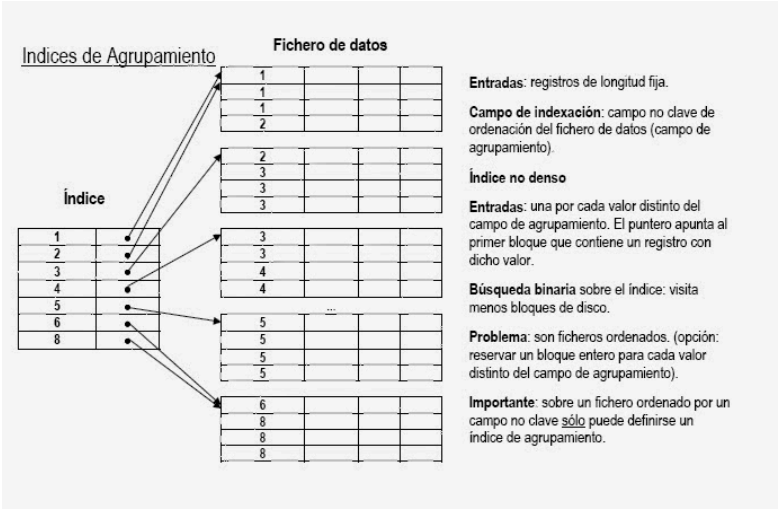
Indice agrupado o cluster

Si los registros del fichero están ordenados fisicamente por un campo no clave (es decir, un campo que no tiene un valor distinto para cada registro), este campo se denomina campo agrupado. Podemos crear un tipo de índice diferente, denominado índice agrupado, para acelerar la recuperación de los registros que tienen el mismo valor para el campo agrupado. Esto difiere de un Índice principal, que requiere que el campo de ordenación del fichero de datos tenga un valor distinto para cada registro. Un Índice agrupado también es un fichero ordenado con dos campos; el primero es del mismo tipo que el campo agrupado del fichero de datos, y el segundo es un puntero a un bloque. En el Índice agrupado hay una entrada por cada valor distinto del campo agrupado, que contiene el valor y un puntero al primer bloque del fichero de datos que tiene un registro con ese valor para su campo agrupado.

Una diferencia importante es que una búsqueda en un Índice utiliza los valores del propio campo de búsqueda, mientras que una búsqueda en un directorio disperso utiliza el valor de dispersión que se calcula aplicando la función de dispersión al campo de búsqueda.

Para poder usar este tipo de indices, se necesita

1. El archivo de datos debe de estar ordenado y pueden repetirse
2. El campo que se ordena no es primario



Indice secundario

Un índice secundario proporciona un medio secundario de acceso a un fichero para el que ya existe algún acceso principal. El Índice secundario puede ser un campo que es una clave candidata y tiene un valor único en cada registro, o puede ser una "no clave" con valores duplicados. El Índice es un fichero ordenado con dos campos. El primero es del mismo tipo de datos que algún campo no ordenado del fichero de datos que es un campo de indexación. El segundo campo es un puntero de bloque o un puntero de registro. Puede haber muchos Índices secundarios (y, por tanto, campos de indexación) para el mismo fichero.

Primero consideramos que la estructura de acceso de un Índice secundario en un campo clave tiene un valor distinto por cada registro. Dicho campo recibe a veces el nombre de clave secundaria. En este caso, hay una entrada de Índice por cada registro del fichero de datos, que contiene el valor de la clave secundaria para el registro y un puntero al bloque en el que está almacenado el registro o al propio registro. Por tanto, dicho Índice es denso.

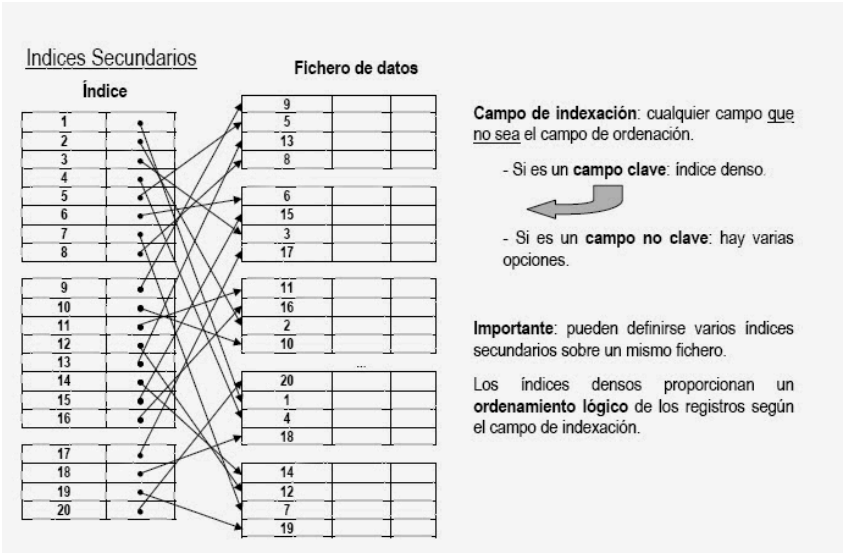
De nuevo nos referimos a los dos valores de la entrada de Índice i como <K(i), P(i». Las entradas están ordenadas por el valor de K(i), por lo que podemos efectuar una búsqueda binaria. Como los registros del fichero de datos no están físicamente ordenados por los valores del campo clave secundario, no podemos utilizar las anclas de bloque. Por eso creamos una entrada de Índice por cada registro del fichero de datos, en lugar de hacerlo por cada bloque, como en el caso de un Índice principal.

Un Índice secundario normalmente necesita más espacio de almacenamiento y un tiempo de búsqueda mayor que un Índice principal, debido a su mayor cantidad de entradas. Sin embargo, la mejora en el tiempo de búsqueda de un registro arbitrario es mucho mayor para un Índice secundario que para un Índice principal, puesto que tendríamos que hacer una búsqueda lineal en el fichero de datos si no existiera el Índice secundario.

En el caso de un Índice principal, todavía podríamos utilizar una búsqueda binaria en el fichero principal, aun cuando no existiera el Índice.

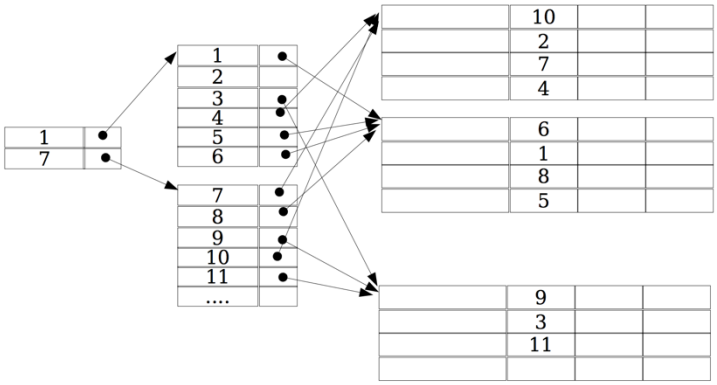
Se utiliza cuando el archivo de datos no esta ordenado por el campo que interesa indexar, el campo que se quiere indexar puede o no ser llava candidata.

El indice de dicho tipo indice ya estara ordenado pero, es denso, por lo que hay una entrada en el registro de indice por cada registro del archivo e datos.



Indices multinivel

Permiten reducir la parte del índice que se requiere acceder en un valor equivalente al factor de bloqueo del índice. La principal desventaja es su naturaleza estática, es preferible usar un índice multinivel dinámico (árbol B/B+).



Arboles B, B+ (tipo de indice multinivel)

Los árboles B y B+ son casos especiales de estructuras de datos en forma de árbol bien conocidas, de cuya terminología hacemos una introducción. Un árbol está formado por nodos. Cada nodo del árbol, excepto el nodo especial denominado raíz, tiene un nodo padre y varios (ninguno o más) nodos hijo. El nodo raíz no tiene padre. Un nodo que no tiene ningún nodo hijo se llama nodo hoja; un nodo que no es hoja es un nodo interno. El nivel de un nodo siempre es uno más que el nivel de su padre, siendo cero el nivel del nodo raíz.

Un subárbol de un nodo consta de ese nodo y de todos sus nodos descendientes (sus nodos hijo, los nodos hijo de sus nodos hijo, etcétera). Una definición recursiva precisa de subárbol es que consta de un nodo n y de los subárboles de todos los nodos hijo de n.

Árboles B El árbol B tiene restricciones adicionales que garantizan que el árbol siempre estará equilibrado y que el espacio desperdiciado por la eliminación, si lo hay, nunca será excesivo. Los algoritmos para insertar y eliminar son más complejos para poder mantener estas restricciones. No obstante, la mayor parte de las inserciones y eliminaciones son procesos simples que se complican sólo en circunstancias especiales (por ejemplo, siempre que intentamos una inserción en un nodo que ya está lleno o una eliminación de un nodo que está a menos de la mitad de su capacidad). De manera más formal, un árbol B de orden p, utilizado como estructura de acceso según un campo clave para buscar registros en un fichero de datos.

Arboles B+: La mayoría de las implementaciones de un índice multinivel dinámico utilizan una variante de la estructura de datos en árbol B denominada árbol B+. En un árbol B, cada valor del campo de búsqueda aparece una vez en algún nivel del árbol, junto con un puntero de datos. En un árbol B + los punteros a datos se almacenan sólo en los nodos hoja del árbol, por lo cual, la estructura de los nodos hoja difiere de la de los nodos internos. Los nodos hoja tienen una entrada por cada valor del campo de indexación, junto con un puntero al registro (o al bloque que contiene ese registro) si el campo de búsqueda es un campo clave. En el caso de un campo de búsqueda que no es clave, el puntero apunta a un bloque que contiene punteros a los registros del fichero de datos, creándose un nivel extra de indirección.

La diferencia del Árbol B+ con respecto al árbol B es que no hay datos (puntero a) en los nodos internos. Los datos sólo están en las hojas. En cada nodo interno, el puntero de la izquierda de la clave apunta ahora a los menores o iguales, y el de la derecha a los mayores que la clave. Por otro lado, como los datos están en las hojas, entonces se puede organizar otra estructura sobre ellos que facilita la recorrida ordenada.

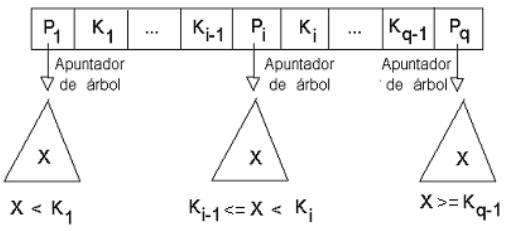


Fig 4. Nodo interno de un Árbol B+ con q-1 valores de búsqueda

Instrucciones para realizar una indice

Crear un índice,

CREATE INDEX nombre_indice ON nombre_relacion

(lista_atributos)

Utilizar create unique index para especificar e imponer indirectamente la condición de que la clave de búsqueda es una clave candidata. No se necesita si el DBMS soporta la restricción de integridad SQL unique. Eliminar un índice:

DROP INDEX nombre_indice

La mayoría de los sistemas de BD permiten especificar el tipo de índice

Conclusiones

Un árbol B es un árbol binario, es decir, de cada nodo sólo pueden partir como máximo dos nodos. En enlace de los nodos se hace a través de la estructura jerárquica, es decir, cada nodo sólo sabe quien es su padre y sus hijos.

Sin embargo un árbol B+ es un árbol B (se cumple todo lo que he dicho anteriormente) pero además los nodos tienen un enlace a los elementos de su mismo nivel. Es decir, cada nodo sabe quien es su padre, su hijo izquierdo, su hijo derecho, si hermano izquierdo y su hermano derecho.

Para poder hacer búsquedas tanto secuenciales como binarias en un árbol.

La mayoría de las bases de datos comerciales (Oracle, SQLServer...) utilizan BTree+ en su estructura interna.

Referencias

R. Elmasri and S. Navathe, Fundamentos de sistemas de bases de datos. Pearson Educací on, 2007. [Online]. Available: https://books.google.com.mx/books?id=NT3uPQAACAAJ

C. Ricardo, Bases de datos. McGraw-Hill Interamericana, 2000. [Online]. Available: https://books.google.com.mx/books?id=BmVHAQAACAAJ

(2017) Mysql :: Mysql documentation. [Online]. Available: https://dev.mysql.com/doc/

https://www.anerbarrena.com/tipos-dato-mysql-5024/
https://advenis.wordpress.com/2010/04/21/tipos-de-datos-en-mysql/
https://es.wikipedia.org/wiki/MySQL
https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-son-los-metadatos-y-cual-es-su-utilidad
https://www.lucidchart.com/pages/es/diagrama-entidad-relaci%C3%B3n-extendido