ARBOLES B+

Espinoza Maciel Gilberto
Paredes Padilla Jorge Xavier
Figueroa Miranda José Jesús

January 3, 2018

Historia

Los Arboles B+ fueron introducidos en 1972 por Rudolf Bayer y Edward McCreight en su articúlo *Organization and maintenance of large ordered indexes*, ellos estaban trabajando en Boeing's Mathematical and Information Sciences Laboratory en los 60, tratando de desarrollar algoritmos para el almacenamiento y recuperacion de datos de las computadoras.

Asumimos que la información es tan voluminosa que solo pequeñas partes de ella pueden ser almecendas en el procesamiento en vez en vez. Entonces el bulto de información debe ser respaldado. La clase de respaldos que se consideraron son dispositivos de acceso pseudo aleatorio que tienen un gran tiempo de espera y una gran rango de datos una vez que la transmicion de la secuencia fisica de informaciona a empezado. Dispositivos de acceso pseudo aleatorio tipicos son discos duros, celdas de datos.

Varios equipos han trabajado en los formatos de las paginas, representacion de llaves, tecnicas de comprension y encriptado, entre otros. La consolidation mas importante durante este tiempo es el arbol B+, una variante donde todas las llaves, residen en las hojas y los nodos internos son puramente estructuras de busqueda redundante, esto tiene beneficios sustanciales comparado con la version original.

El fundamental problema con el contral en los arboles B es que los patrones de acceso no son consistentes:

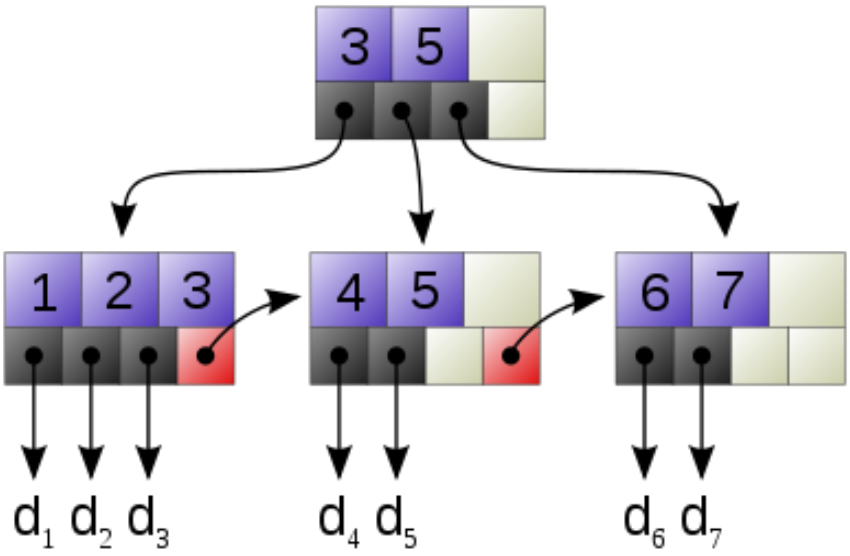
- Busqueda por llave empieza en la raíz y desciende hacia la hoja.
- Dividir una hoja (insercion o actualizacion) o juntando dos hojas (borrando o actualizacion) resulta en moverse desde arriba desde la hoja hacia la raiz.
- Recuperar entrandas en orden de la lista se hace recorriendo las hojas hacia un lado

Esto significa que los protoclos que permiten transacciones independientes para procesar un arbol B simultaneamente son complicadas

Progreso nunca se detiene, y en los centros de datos del internet del dia de hoy, distribuidos, virtualizados y en la nube estan empujando hacia y mas allá de los limites que los árboles B centralizados. Recientemente, interés se ha cambiado a una investigacion de implementación de árboles B distribuidos, como el trabajo de árboles B particionados hecho por *Microsoft* y un árbol B distribuido escalable, hecho por un equipo en Hewlett-Packard.

Como una modificacion de los arboles B, son un tipo de estructura que sirve para mantener datos ordenados de forma eficiente tal que un arbol B+ tiene una altura $O(\lg n)$, esto significa que a medida que el volumen de datos crece, la eficiencia decrece aunque no de modo lineal.

Podemos visualizar un arbol B+ como un arbol B donde cada nodo contiene llaves y las llaves hacen referencia a otras sublistas o hojas.



Diferencias

La principal diferencia con un Arbol B es que estos almacenan los valores en las hojas del arbol y los nodos intermedio solo almacenan referencias a otros nodos (internos u hojas). Estos nodos intermedios solo son usados para guiar la busqueda en el arbol. Mientras que en un arbol B clasico los valores son almacenados en ambos (hojas y nodos intermedios). Otra diferencia es con el proceso de inserción en árboles B consiste en que cuando se inserta una nueva clave en una página llena, ésta se divide también en otras dos y lo que subirá a la página antecesora será una copia. La diferencia consiste en que las llaves se almacenan en los nodos del último nivel del árbol y esos nodos se enlazan unos con otros formando una lista ligada que puede ser ligada en forma sencilla o doblemente ligada. Las llaves se repiten en los nodos intermedios y en el nodo raíz para facilitar la búsqueda. Puede haber llaves en los nodos intermedios que no existan en los nodos hoja porque alguna vez se insertaron y luego se borraron pero se conservan para propósitos de localizar algunas llaves.

Caracteristicas

- El número máximo de claves en un registro es llamado el orden del árbol B+.
- El mínimo número de claves por registro es la mitad del máximo número de claves. Por ejemplo, si el orden de un árbol B+ es n, cada nodo (exceptuando la raíz) debe tener entre n/2 y n claves.
- El número de claves que pueden ser indexadas usando un árbol B+ está en función del orden del árbol y su altura.

Es de notar que los arboles-B+ ocupan un poco más de espacio que los arboles-B, y esto ocurre al existir duplicidad en algunas claves. Sin embargo, esto es aceptable si el archivo se modifica frecuentemente, puesto que se evita la operación de reorganización del árbol que es tan costosa en los arboles-B.

Operaciones básicas de un arbol

Búsqueda

La operación de búsqueda en árboles-B+ es similar a la operación de búsqueda en árboles-B. El proceso es simple,Se comienza buscando por el nodo raíz y se compara la llave y con las llaves k_i que se encuentran en ese nodo. Si Y es igual a algún k_i termina la búsqueda satisfactoriamente. Sin embargo puede suceder que al buscar una determinada clave la misma se encuentre en un nodo raíz o interior, en dicho caso no debe detenerse el proceso, sino que debe continuarse la búsqueda con el nodo apuntado por la rama derecha de dicha clave. Un ejemplo de la implementacion de la busqueda es:

```
int Arbol::buscar(int a){
    Nodo *p=raiz;
    CajaValor *val;
    CajaDireccion *q;
    int n=0;
    while(true){//Busca la hoja del valor
        if(p->esHoja==true){//Busca si esta agregado el elemento en esa hoja
            val=p->llaves.dondePrincipio();
            while(val!=NULL){
                if(val->valor==a){
                    encontrado=true;
                    donde=p;
                    return 0;
                }
                val=val->siguiente;
            }
            encontrado = false;
            donde=p;
```

```
            return 1;
        }else{//Busca en que direccion del nodo del arbol debe bajar
            n=0;
            val=p->llaves.dondePrincipio();
            q=p->direccion.dondePrincipio();
            while(q!=NULL && val!=NULL){
              if(a>=val->valor){
               n++;
               val=val->siguiente;
               q=q->siguiente;
              }else{
               val=NULL;
              }
            }
            p=q->direccion;
        }
    }
```

Inserción

El proceso de inserción en árboles-B+ es relativamente simple. La dificultad se presenta cuando desea insertarse una clave en un nodo que se encuentra lleno. En este caso, el nodo afectado se divide en 2, distribuyéndose las claves de la siguiente forma:

"Las $m/2$ primeras claves en el nodo de la izquierda y las $m/2+1$ restantes claves en el nodo de la derecha".

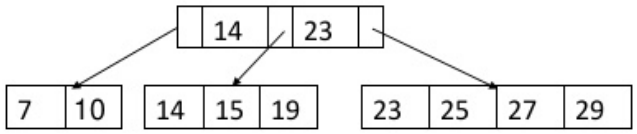
Una copia de la clave del medio sube al nodo padre.

Un ejemplo de la implementacion de la insercion es:

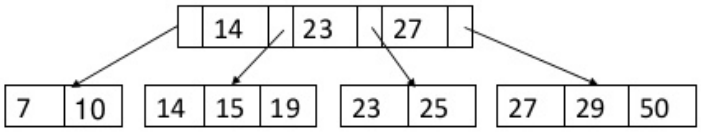
```
void Arbol::insercion(int a){
    Nodo *p;
    if(raiz==NULL){
        p= new Nodo;
        p->llaves.nuevo();
        p->direccion.nuevo();
        p->esHoja = true;
        p->padre=NULL;

        p->llaves.agregar(a);
        raiz = p;
        return;
    }else{
        buscar(a);
        if(encontrado==true){
            return;
        }
```

```
        p=donde;
        if(p->llaves.CuantosVal()<orden-1){
            p->llaves.agregar(a);
        }else{
            dividirHojas(a, p);
        }
        return;
    }
}
```



Insercion de clave : 50



Eliminación

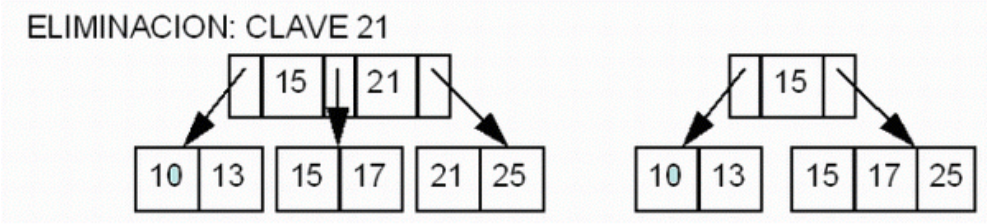
La operación de eliminación en árboles-B+ es mas simple que en árboles-B. Puesto que las claves a eliminar siempre se encuentran en las páginas hojas. En general deben distinguirse los siguientes casos:

- Si al eliminar una clave, la cantidad de llaves queda mayor o igual que $m/2$ entonces termina la operación. Las claves de los nodos raíz o internos no se modifican por más que sean una copia de la clave eliminada en las hojas.
- Si al eliminar una clave, la cantidad de llaves queda menor que $m/2$ entonces debe realizarse una redistribución de claves, tanto en el índice como en las paginas hojas.

Un ejemplo de la implementacion de la eliminacion es:

```
int Arbol::eliminar(int a){
    Nodo *p;
    buscar(a);
    if(encontrado==false) return 0;
    p=donde;
    if(p==raiz){
        if(p->llaves.CuantosVal()==1){
            p->direccion.limpiar();
```

```
            p->llaves.limpiar();
            p->esHoja=false;
            p->padre=NULL;
            delete p;
            raiz=NULL;
            return 1;
        }else{
            p->llaves.borrar(a);
            return 1;
        }
    }else{
        if(p->llaves.CuantosVal()>(int)orden/2){
            if(a!=p->llaves.dondePrincipio()->valor ||
             (p->padre)->direccion.dondePrincipio()->direccion==p){
                p->llaves.borrar(a);
            }else{
                p->llaves.borrar(a);
                (p->padre)->llaves.borrar(a);
                (p->padre)->llaves.agregar(p->llaves.dondePrincipio()->valor);
            }
            return 1;
        }else{
         modificarHojas(a, p);
        }
    }
    return 1;
}
```



a) b)

Ventajas

- Los arboles B+ es mas facil y con mayor rendimiento hacer una busqueda completa debido a que las hojas estan conectadas como si fuera una lista ligada
- El tiempo de busqueda es muy corto en comparacion a otros arbolesface

Sectores de uso

Los sistemas de archivos mas conocidos vease: NTFS, EXT4. O los sistemas de administración de bases de datos relacionales mas populares como (Oracle, Postgresql, Mysql, Microsoft SQL Server). Incluso el sistema de archivos en el nucleo de linux utiliza arboles b+ (ReiserFS, etc). Por otro lado en el lado empresarioal, hay una compañia que vende soluciones a base de arboles B para el manejo de informacion en servidores personalizados para sus clientes. O mas cotidianamente el uso masivo de nuestros dispositivos moviles Android implementan Arboles B+ para mejorar la eficiecia de la memoria y la bateria.

Referencias

https://www.inf.fu-berlin.de/lehre/SS10/DBS-Intro/Reader/BayerBTree-72.pdf
https://www.cs.helsinki.fi/u/mluukkai/tirak2010/B-tree.pdf
https://www.perforce.com/blog/short-history-btree
https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/419/1/ZACTE23.pdf
https://en.wikipedia.org/wiki/NTFS