**Arboles** 🡪 introduccion

Problemas con los indices?

* La busqueda binaria aun es costosa
* Mantener los indices ordenados es costoso
* Solucion 🡪 RAM
* Objetivo 🡪 persistencia de datos

**Arboles**

* Estructuras de datos que permiten localizar en forma mas rapida informacion de un archivo, tienen intrinsecamente busqueda binaria

**Arboles binarios**

Que es un arbol binario?

* Estructuras de datos donde cada nodo tiene dos sucesores, a izquierda y a derecha
* Arbol balanceado: un arbol esta balanceado cuando la altura de la trayectoria mas corta hacia una hoja no difiere de la altura de la trayectoria mas grande.
* Inconveniente de los binarios: se desbalancean facilmente.

**Arboles AVL**

* Arbol binario balanceado en altura (BA(1)) en el que las inserciones y eliminaciones se efectuan con un minimo de accesos.
* Arbol balanceado en altura:
  1. Para cada nodo existe un limite en la diferencia que se permite entre las alturas de cualquiera de los subarboles del nodo (BA(k)), donde k es el nivel de balance)

**Arboles AVL y Binarios**

Caracteristicas/Conclusiones

* Estructura que debe ser respetada
* Mantener arbol, rotaciones restringidas a un area local del arbol
  1. Binario: 🡪 Busqueda: Log2(N+1)
  2. AVL: 🡪 Busqueda: 1.44 log2(N+2)
  3. Ambas performance por el peor caso posible

**Arboles Binarios Paginados**

* Problemas de almacenamiento secundario, buffering, paginas de memoria, varios registros individuales, minimiza el numero de accesos

**Arboles multicamino**

* Generalizacion de arboles binarios, c/nodo tiene k punteros y k-1 claves (o registros), disminuye la profundidad del arbol
* Orden del arbol.

**Arboles balanceados**

* Son arboles multicamino con una construccion especial en forma ascendente que permite mantenerlo balanceado a bajo costo.

Propiedades de un arbol B de orden M:

* Ningun nodo tiene mas de M hijos
* C/nodo (menos raiz y los terminales) tienen como minimo [M/2] hijos
* La raiz tiene como minimo 2 hijos (o sino ninguno)
* Todos los nodos terminales a igual nivel
* Nodos no terminales con K hijos contienen K-1 registros. Los nodos terminales tienen:
  1. Minimo [M/2] –1 registros
  2. Maximo M – 1 registros

Busqueda de informacion:



Performance de busqueda

* Mejor caso: 1 lectura
* Pero caso: h lecturas (con h altura del arbol)
  1. Axioma: arbol balanceado de Orden M, si el numero de elementos del arbol es N 🡪 hay N+1 punteros nulos en nodos terminales.

Cota para h

* Nivel # minimo de descendientes
* 1 2
* 2 2 \* [M/2]
* 3 2 \* [M/2] \* [M/2]
* ………………………………………………….
* h 2 \* [M/2]h-1
* **Relacion entre h y # de nodos**
* N+1 >= 2 \* [M/2]h-1
* h <= [ 1 + log[M/2] ((N+1)/2) ]
* Si M = 512 y N = 1000000 🡪 h <= 3.37 (4 lecturas encuentra un registro)

Arbol balanceado

* La raiz tiene como minimo 2 descendientes
* El nodo verde tiene como minimo [M/2] lo mismo el amarillo
  + Por ende luego del nivel 2 hay 🡪 2 \* [M/2] hijos posibles
* Luego cada nodo del nivel tres tendra como minimo [M/2] hijos
  + En el nivel 3 hay 2\*[M/2] nodos
  + Por ende luego del nivel 3 hay 2 \* [M/2] \* [M/2] hijos
* Esto se repite hasta el nivel h

En el nivel h hay 2 \* [M/2] \*… [M/2] punteros nulos (hijos) y eso es 2 \* [M/2]h-1

Performance de la insercion

* Mejor caso (sin overflow)
  + - H lecturas
    - 1 escritura
* Peor caso (overflow hasta la raiz, aumenta en uno el nivel del arbol)
  + - H lecturas
    - 2h+1 escrituras (dos por nivel mas la raiz)
* Estudios realizados
  + - M = 10 25% divisiones
    - M = 100 2% divisiones

Eliminacion

* Siempre eliminar de nodos terminales (trabajamos con arboles)
* Si se va a eliminar un elemento que no esta en nodo terminal 🡪 llevarlo primero a nodo terminal
* Posibilidades ante eliminacion
  1. Mejor caso: borra un elemento del nodo y no produce underflow, solo reacomodos ( # elementos >= [M/2]-1
  2. Peor caso: se produce underflow, #elementos < [M/2] – 1
* Dos soluciones
  + - Redistribuir
    - concatenar
* Definicion: nodo adyacente hermano
  1. Dos nodos son adyacentes hermanos si tienen el mismo padre y son apuntados por punteros adyacentes en el padre.

Redistribuir

* Cuando un nodo tiene underflow puede trasladarse llaves de un nodo adyacente hermano (en caso que este tenga suficientes elementos)

Concatenacion:

* Si un nodo adyacente hermano esta al minimo (no le sobra ningun elemento) no se puede redistribuir, se concatena con un nodo adyacente disminuyendo el # de nodos (y en algunos casos la altura del arbol)

Performance de la eliminacion

* Mejor caso (borra de un nodo Terminal)
  + - H lecturas
    - 1 escritura
* Peor caso (concatenacion lleva a decrementar el nivel del arbol en 1)
  + - 2h – 1 lecturas
    - H + 1 escrituras

**Arboles Balanceados 🡪 B\***

* La redistribucion podria posponer la creacion de paginas nuevas
* Se pueden generar arboles B mas eficientes en terminos de utilizacion de espacio

Arbol B especial en que cada nodo esta lleno por lo menos en 2/3 partes

Propiedades (orden M)

* Cada pagina tiene maximo M descendientes
* Cada pagina, menos la raiz y las hojas, tienen al menos [(2M – 1) / 3] descendientes
* La raiz tiene al menos dos descendientes (o ninguno)
* Todas las hojas aparecen en igual nivel
* Una pagina que no sea hoja si tiene K descendientes contiene K-1 llaves
* Una pagina hoja contiene por lo menos [(2M – 1) / 3] –1 llaves, y no mas de M-1.

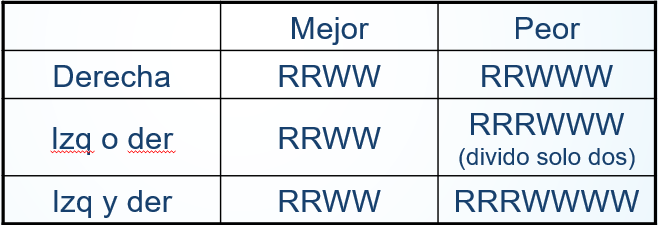
Operaciones de Busqueda

* Igual que el arbol B comun

Operaciones de Insercion

* Cuatro casos posible
  + - **Derecha**: redistribuir con nodo adyacente hermano de la derecha (o izq. si es el ultimo)
    - **Izquierda**: redistribuir con nodo adyacente hermano de la izquierda (o der. si es el ultimo)
    - **Izquierda o derecha**: si el nodo de la derecha esta lleno y no se puede redistribuir, se busca el de la izquierda.
    - **Izquierda y derecha**: busca llenar los tres nodos, estos tendran un ¾ parte llena.

Costo de la redistribucion



Tecnicas de paginado

* estrategias de reemplazo: LRU (last recently used)

Archivos secuenciales indizados

Permiten una mejor recorrida por algun tipo de orden

* Indizado (ordenado por una llave)
* Secuencial (acceder por orden fisico, devolviendo el registro en orden de llave)

Hasta ahora metodos disjuntos, se opta:

* Rapida recuperacion (Arbol)
* Recuperacion ordenada (secuencial)

Debemos encontrar una solucion que agrupe ambos casos

Conjunto de secuencias

* Conjunto de registros que mantienen un orden fisico por llave mientras que se agregan o quitan datos, si podemos mantenerlo podemos indizarlos

Posible solucion

* Mantener bloques de datos
* Cada bloque con registros y puntero al siguiente

Costo

* Aumenta el tamaño del archivo (fragmentacion interna)
* No hay orden fisico salvo dentro del un bloque.
* Tamaño del bloque
  1. Debe permitir almacenar varios bloques en RAM (redistribucioon)
  2. Las E/S deben ser rapidas y sin necesidad de desplazamientos

**Arboles Balanceados 🡪 B+**

* Consiste en un conjunto de grupos de registros ordenados por clave en forma secuencial, junto con un conjunto de indices, que proporciona acceso rapido a los registros.
* Propiedades
  + Cada pagina tiene maximo M descendientes
  + Cada pagina, menos la raiz y las hojas, tienen entre [M/2] y M hijos
  + La raiz tiene al menos dos descendientes (o ninguno)
  + Todas las hojas aparecen en igual nivel
  + Una pagina que no sea hoja si tiene K descendientes contiene K-1 llaves
  + Los nodos terminales representan un conjunto de datos y son linkeados juntos.
* Los nodos no terminales no tienen datos sino punteros a los datos.
* Separadores
  + Derivados de las llaves de los registros que limitan un bloque en el conjunto de secuencia
  + Separadores mas cortos, ocupan espacio minimo
* Arbol B+ de prefijos simples
  + Arbol B+ en el cual el conjunto indice esta constituido por separadores mas cortos

Arboles Balanceados 🡪 conclusiones