CDB2

Los arboles surgieron para almacenar archivos de indice de forma más eficiente y ordenada.

Permite localizar de forma rapida la información del archivo.

Ante cambio, solo debe reorganizarce localmente.

Existen distintos tipos de arboles:

- Binarios: Maximo 2 hijos por cada nodo.
- AVL: Binario de busqueda auto-balanceado.
- Multicamino: tiene N caminos, con N>2.
- Balanceados: la altura de los subárboles de cualquier nodo no difiere en más de una unidad.

Binario

Es un arbol de grado 2, se utiliza para representar expresion arimeticas y de decisión. Tambien puede ser utilizados como arboles binarios de busqueda.

Las estructuras de datos utilizadas para almecenarlos deben ser registros de long fija, la informacion del archivo no está ordenada, cada nodo es un registro. Existe un desperdicio de espacio ya que hay muchos campos vacios.

Balanceado

Un arbol donde la altura de la trayectoria más corta hacia una hoja no difiere de la altura hacia dicha hoja. Estos se desordenan facilmente asi que se buscó una solucion y esta fue los llamados AVL.

AVL: No esta balanceado completamente.

Sin embargo todas estas opciones sigue teniendo 2 problemas ppales la busqueda requiere demasiados desplazamientos y mantener un indice de orden es muy costoso. Los arboles **balanceados** ofrecen una solucion admisible al 2do problema.

Binarios paginados

Los desplazamientos en mem sec tiene un costo de tiempo relativamente alto pero una vez en poscion los conjuntos de bits de interes estan contiguos, entonces el desplazamiento es lento pero la transferencia es rápida.

Por eso se llego a la idea de dividir a los arboles en paginas almacendas cada una en un bloque de direcciones contiguas en disco, esto reduce el nro de desplazamientos considerablemnete.

Tiene un pequeño incoveniente que es que requiere transferir muchos datos que a veces no son usados, aún así sigue siendo más rápido.

Está idea era buena pero nuca se encontro una forma de aplicarla eficientemente.

Multicamino

Un arbol con N hijos, N>2. Estos resuelven el problema que tenian los arboles binarios que era la cantidad de niveles miuy elevada, sin embargo se debe mantener balancead para asegurar eficiencia. Para ello estosa rboles se contruyen desde la base hasta la raíz.

Para sus contruccion surgiero varias operaciones:

- Insercion: Busca hasta que llega al nivel hoja, se inserta en un nodo terminal. Hay 2 casos, uno cuando hay espacios que solo se realizan desplazmientos internos y otra donde el lugar esta lleno y se pruduce la promocion
- Eliminacion: Busca por cada indice hasta encontrar el nodo a eliminar, y de nuevo hay 2 casos, uno en donde sea una hoja y dicho nodo no quede vacio luego, y el otro en donde se produce underflow y debes o bien reordenar con el hermano o concatenarlo, perdiendo uno de los nodos.

Arboles B

Son arboles multicamino, se construyen de forma ascendete. La insercion de un nuevo registro puede producir overflow, en dichos casos puedes tenes que redistribuir o contanter con el hermano de dicho nodo. Cuando el hermano tiene elementos suficientes se puede realizar la resdist y cuando se debe concatener donde es posible perder un nivel de arbol.

Para hacer más eficiente en terminos de utilizacion de espacios se hizo una mejora y se la llamo B+, donde cada nodo esta lleno por lo menos 2/3 partes, mismo manejo interno que un arbol B pero su manejo de nodos se realiza en

buffers lo que permite minimizar los acceso a disco. En la ram se almacena la raiz para ahorra en acceso a mem sec.

Conclusion

Los arboles B y B+ permiten el acceso aleatorio por clave de forma eficiente ademas los B+ exigen que sus nodos se mantegan más llenos que en el otro. Los B+ son los unicso que permiten un recorrido ordeneado secuecial rapido. Este ultimo consta de 2 partes: *Indices* en la raiz y nodos interiores donde las claves son duplicadas en la *secuencia* que es donde se encuetran todas la claves colocadas en las hojas de arbol.

Los B+ ocupan más que el B ya que tiene q alamcenar los separadores.

Tabla de comparacion

	B	B+
Objetos	nodos	nodos terminales
Tiempo de busq	equivalente	equivalente
Procesamiento sec ord	lente	rapido(pteros)
Ins/Elim	equivalente	equivalente

Dispersion

Para los casos en que se necesita un mecanismo de acceso a regiustros dcon una sola lectura se usa **hash.**Esta tecnica genera una direccion base unica para un clave dada, convirtiendo la clave del reegistro en un numero aleatorio que luego se transforma en una direccion de mem donde debe ser alamcenado el reg. Si esa direccion ya se está ocupada se realizan un tratado especial.

Tiene los sig beneficios

- No requiere almacenamiento adicional
- Facilita la insercion y eliminacion de reg rapida
- Encuentra reg con pocos accesos a disco

Costos

- No es posible el uso de reg de long variable

- No existe orden fisico de datos
- No permite claves duplicadas

Hay 2 tipos de dispersion. Con espacio de dir estatico o con uno dinamico

Parametros que afectan a su eficiencia

1. Funcion de dispersion
2. Tamaño de los compartimientos
3. Densidad de empaquetamiento
4. Tratamiento de saturacion

Caja negra: apartir de una clave se obtiene la direccion donde debe estar el registro, como no hay realacion entre el indice del reg y la dir puede haber 2 reg que se quieran alamacenar en la misma direcc esto se llama colision.

Funciones de dispersion:

Centros cuadrados: la clave se multiplica por si misma y se toman los digitos centrales.

Division: la clave se divide por un nro aprox igual a la cantidad de dir, luego se obtiene el residuo

Desplazamiento: se realiza el desp del numero, se promociona y se suman las partes resultantes.

Plegado: se realiza un plegaod del nro, se suman partes resultantes.

Transformado de base: la base del numero se modifica tomado su modulo

1. **Cuadrados Centrales**: Supongamos que la clave es 1234. Al cuadrarla obtenemos 1522756. Tomando los dígitos centrales, por ejemplo, los cuatro dígitos centrales, obtenemos 2275 como valor de dispersión.
2. **División**: Si la clave es 1234 y usamos un número de división de 100 (que es aproximadamente igual a la cantidad de direcciones), el residuo es 34, que sería el valor de dispersión.
3. **Desplazamiento**: Si la clave es 123456 y realizamos un desplazamiento a la izquierda de 2 dígitos, obtenemos 3456. Luego, sumamos las partes resultantes (34 y 56) para obtener 90 como valor de dispersión.
4. **Plegado**: Si la clave es 123456789 y decidimos plegarla en grupos de 3 dígitos, obtenemos los grupos 123, 456 y 789. Al sumar estos grupos, obtenemos 1368 como valor de dispersión.

5. **Transformación de base**: Si la clave es 1234 y decidimos transformarla a base 8, obtenemos 2322. Luego, podemos tomar el módulo de este número con respecto a un número de direcciones para obtener el valor de dispersión.

Division es la mejor de todas papa

Las colisiones son inherentes a este tipo de sistemas por ende se busca minimizar los macimos posible. Para ello se elegin los metedos que las distribuyan de forma más aleatorea, se utilizan grandes espacio de dir para almecanar pocos reg. Un aspecto que mejora esto considerablemente es que cada dir puede almacenar más de un reg, a estos espacios de dir se les llama cubetas

El tamaño de dichas cubetas si es muy grande hay menor riesgo de saturacion pero mayor fragmentacion y la busque es más lenta dentro del mismo, además de que sies grande la recuperacion de un reg es lenta. El tamaño va depeder del sistema teniendo en uanto las psibilidades de trans en operacions E/S a buffer del SO.

Lo que se tiene en cuenta a la hora de eficiencia es la **densidad de empaquetamineto DE** y con ella es posible hacer una estimacion de la saturacion.

En gral si hay N direcciones el nro esperado de I registros asignados es igual N*P(i), donde $P(i) = (K/N)^i * e^{K-I}/I!$

Otro gran cambio que disminuye la saturacion es que los esp de dir contegan más de un reg.