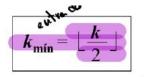
## Tema 7: Organizacion de Fichero: Organizaciones Auxiliares.

- Índice: Archivo aparte donde se almacena la ubicación física de los valores de una clave alternativa que es muy frecuente. Es un directorio cuya entrada se refiere a un solo registro.
  - El archivo auxiliar es el Índice y la clave privilegiada es la Clave de indización.
  - Tipos de punteros: De menos a mas independiza y de mas a menos velocidad.
    - Direccion de Máquina: la dirección física del registro.
    - Direccion Relativa: Registro en el espacio de Direccionamiento.
    - Puntero simbólico: Identificación logica del registro.
  - Entrada: Registro formado por punteros.
    - Entrada de indice primario: clave \* puntero\_externo
    - Entrada de indice secundario: clave \* long\_lista \* (puntero\_externo)^long\_lista.
    - El numero de entradas es igual a la cardinalidad del dominio(CI).
    - Al buscar recorremos el indice hasta encontrar una entrada, y si insertamos se pone al final de la lista de punteros. Conviene que el indice este ordenado, y no consecutivo.
  - Directorio: Archivo formado por entradas.
  - Tipos de indices:
    - Primario: La clave de indización es identificativa. 1 entrada 1 clave 1 registro.
       Filtrado maximo.
    - Secundario: La clave de indización es no identificativa. N registros 1 clave 1 entrada. Filtra menos.
  - Ventajas:
    - Acceso por clave alternativa, hasta ahora no privilegiadas.
    - Aumenta la Tasa de Acierto, al ocupar menos y ser recurrente la mantenemos en memoria.
    - Reorganización menos costosa, los indices tiene menos bloques que los propios datos.
  - Desventajas:
    - Procesos de Actualización más costosos. (Muy importante)
    - Necesita almacenamiento auxiliar.
    - Necesita mantenimiento.
  - Operaciones: Creación, Borrado, Consulta, Localización y Actualización.
  - Coste de Procesos sobre Ficheros Indizados:
    - Localización a través del Indice: Acceso al indice.
    - Localización por varios indices: Suma del acceso a cada indice.
    - Recuperacion: acceso indice + acceso datos.
    - Actualización:
      - Inserciones: Inserción de entradas. Coste = acc\_indice + 1.
      - Borrados: Coste = acc indice + 1
        - Indice primario: suelen requerir borrado de entradas, se vacía al ser 1 solo.
        - Indice secundario: pueden requerir modificación de entradas, si se vacía.
      - Modificaciones:
        - CI: Suele implicar borrado + reinserción de entrada. 2\*acc\_ind + 2
        - CD/CO: Cambia ubicación reg., cambia puntero. acc indice + 1
  - Taxonomía de indices:
    - Segun el caracter de la clave de indización:
      - Indices primarios vs. indices secundarios.
    - Según la correspondencia entre entrada y registros:
      - Denso: Una entrada del indice para cada registro.
      - No denso: Una entrada para cada cubo de datos.
    - Segun el recubrimiento del indice:
      - Exhaustivo: Todos los registros tienen un entrada.
      - Parcial: No se indizan todos los registros.
    - Segun la estructura: Indices simples vs. indices multinivel.

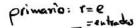
- Indice Simple Denso:
  - Naturaleza: serial, secuencial, o Direccionada.
  - Coste: Depende de la naturaleza.
  - Restricciones: Sobre claves no privilegiadas.
  - Mantenimiento: Ordenado o disperso, puede desbordar —> Reorganización.
    - Se debe evitar la degeneración de la estructura:
      - Indice ordenado: Preferible insercion ordenada + reorganización local.
      - Indice disperso: Pierde eficiencia si cambia, es mas útil como indice temporal.
- Indice Simple No Denso: Una entrada por cubo de datos.
  - Restricción: indice y organizacion base deben ser necesariamente secuenciales y con clave\_indizacion = clave\_ordenacion (CO=CI)
  - Usos: Varias posibilidades de acceso:
    - Procesos ordenados (a la totalidad): Acceso serial.
    - Procesos selectivos (solución única): A traves del indice.
    - Mixtos: (selección de un rango): acceso indizado + serial.
  - Ventajas:
    - · Tamaño muy reducido, tiene menor coste y mayor tasa de acierto. (Apunta a
    - Se ahorra muchas actualizaciones de indice.
    - Se pueden utilizar prefijos, el mínimo tamaño para reconocerlo, en lugar de utilizar toda la clave en la entrada.
  - Inconvenientes:
    - Solo puede existir un indice no denso para cada archivo.
    - La inserción del registro debe ser ordenada, pero se localiza con el indice.
    - · La insercion de la entrada es ordenada, y conlleva pesadas reorganizaciones.
- Indice Multinivel: Es un indice con n niveles, árbol de indices.
  - El coste es un acceso por nivel, interesa definir nodos pequeños a costa de tener mas niveles. Bloquear la raiz en memoria intermedia, nos ahorra un acceso.
  - El ultimo nivel, n, suele ser denso. Aunque al ser secuencial puede ser no denso.
  - Es eficiente, pero degenera.
  - Para fichero constantes es buena solucion, pero cuando son volatiles requiere reorganización.
  - Posibles soluciones:
    - Árboles binarios: Presenta problemas de vecindad y deseguilibrio.
    - Árboles AVL: Resuelve el desequilibrio con reorganizacion local.
    - Árboles Binarios Paginados: Almacena en cada nodo sus dos hijos, 2 niveles.
    - Árboles AVL- Paginados: Buen rendimiento, pero necesita muchos punteros internos, bajísima densidad y reorganizaciones frecuentes.
    - Arboles B: La mejor solucion. Incluye varias entradas por nodo y se construye en orden ascendente.
- Indizacion en Arboles B: Comienza por las hojas y se va construyendo hacia arriba.
  - Nodo: Contiene entradas de indice(CI-puntero) y punteros(a hijos).
  - Orden del árbol (m): Capacidad de los nodos, según los punteros, el numero de hijos de un nodo.  $m \cdot T_{\text{ptro interno}} + k \cdot T_{\text{entrada}} \leq T_{\text{nodo}}$
  - Corolario:
    - k = m 1. Si un nodo tiene m descendientes, tendrá m-1 entradas.
    - El nodo raíz tiene al menos un elemento y por lo tanto al menos dos hijos.
    - El tamaño de nodo es múltiplo del tamaño de bloque.
    - El tamaño de la entrada, es el de la clave mas el/los punteros internos.
  - Partición y promoción: Las entradas de un nodo están ordenadas, y cuando desborda, se

divide en dos nodos y se promociona el elemento intermedio hacia el nivel superior.

- Propiedades:
  - Todos los nodos menos el raíz garantizan una ocupación mínima:

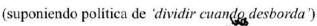


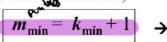
<u>Corolario</u>

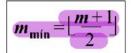


Secundario: e = # clave Dom do Clave in dia

• ¿Cuántos descendientes como mínimo tienen los nodos intermedios?





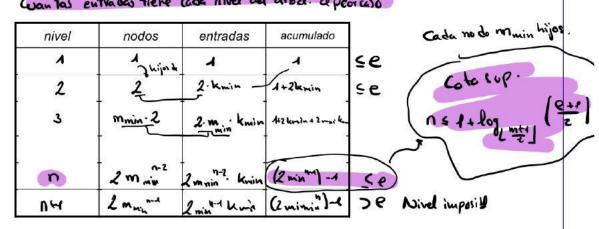


· Tamaño del fichero de índices

Se puede obtener una cota superior del fichero de índices

$$N_{m\acute{a}x}^{o}$$
 nodos fichero =  $n_{m\acute{a}x}^{o}$  nodos fichero ·  $T_{nodo}$  =  $n_{m\acute{a}x}^{o}$  nodos fichero ·  $T_{nodo}$  =  $n_{m\acute{a}x}^{o}$  nodos fichero ·  $T_{nodo}$ 

El nº de niveles (n) para un árbol de orden m y e entradas tiene cota superior



- Recuperar una entrada: #accesos = #niveles.
- Recuperar un registro aleatorio, se recuperan la entrada y tantos cubos de datos como punteros tenga la entrada. Coste= (n-1)\*Tnodo + c\*Ecubo.
- El coste de cualquier actualización sobre el indice, es el coste de localizacion mas un acceso de escritura: (n-1) +1 = n
- El coste extra de una partición es de dos accesos de escritura.
- Aspectos positivos:
  - Es bueno para indices primarios, pero no tanto para los secundarios.
  - Las reestructuraciones son locales.
  - La paginas estan ocupadas a la mitad, hay espacio para cambios.
- Aspectos a mejorar:
  - La densidad de los nodos es baja.
  - Los punteros de las hojas no son necesarios.
- Indización en Árboles B\*: Pretende aumentar la densidad.
  - En lugar de dividir un nodo en dos, se dividen dos nodos en tres. Pasa del 50% al 66%.
  - Cuando un nodo desborda, lo primero que se intenta es hacer una rotación al nodos vecinos, pasando a ser el discriminante.
  - Si tambien se llena el nodo vencido es cuando se pasa de dos llenos a tres nodos.
  - Todo lo demás funciona como en los arboles B.
  - Ventajas:

- Aumento de la densidad.
- Un desbordamiento, no siempre supone partición.
- Desventajas:
  - Aumenta la probabilidad de desbordamiento, al estar de media mas llenos.
- Propiedades:
  - Todos los nodos menos el raiz garantizan una ocupación minima: kmin = floor(2k/3)
  - Los nodos intermedios cuentan con (2k/3) +1 descendientes: mmin = floor((2m+1)/3)
- Calculo de costes: Saberlo como teoría, pero no como práctica.
  - La localizacion es idéntica al árbol B.
  - Coste extra de hacer Rotación: 3 accesos.
  - Coste extra de la Partición: 4 accesos.
- Indización en Árboles B^+: Coste proporcional a la profundidad, crecer en amplitud, aumenta el numero de hijos por nodo, aumenta el orden.
  - Los nodos con hijos suprimen los punteros externos (apuntan a registros), por lo que caben mas punteros interno, es decir mas hijos. Son solo punteros a otros nodos.
  - En los nodos hoja no hay punteros a nodo hijo, pero sí habrá punteros externos, para apuntar a los datos.
  - En las hojas se usa un puntero interno adicional para apuntar al siguiente nodo hoja. Es un mecanismo de acceso alternativo. El la partición se hace:
    - El puntero de encadenamiento nuevo apunta al nodo viejo, y el viejo apunta a al direccion del nodo nuevo.
  - Especialmente eficiente con punteros externos grandes, indice secundario.
  - La partición/promocion es igual que en nodos de árbol B.
  - Propiedades:
    - Orden del arbol (m): Se calcula para nodos no hoja, como en arboles B, pero teniendo en cuenta que las entradas no contienen punteros externos.



Ocupación maxima (k) de los nodos hoja: Si los tamaños de los punteros internos y externos son distintos.



Sino alcanza a tenor
una entrada se aumenta
el tamaño de nodo a
Zo mas blogres.

- Ocupación minima de las hojas sera: kmin = floor(k+1/2)
- Ocupación minima de los nodos intermedios sera: mmin = floor(m+1/2)
- · Calculo del numero de niveles:
  - numero de hojas= floor(e/kmin) tal que e=numero total de entradas.
  - numero de nodos(n-1) = floor(nodosNiveln/mmin)
  - Cuando llega un nivel con un solo nodo, es la raíz nivel 1.
  - Tamaño maximo del fichero indice: Es la suma de los nodos necesarios para cada nivel multiplicado por el tamaño de un nodo.
- Las mejores logradas con arboles B+ y B\* son combinables.
- Estructuras especiales:
  - Indice intermedio: indice primario, denso y exhaustivo, cuyos punteros apuntan a los datos, y el resto de los indices apuntan a este. Al cambiar los registros de ubicación, solo es necesario actualizar punteros.
    - Es muy eficiente, esta bloqueado en memoria privilegiada, es de tamaño reducido y casi constante.
  - Indice agrupado o Cluster: Dos o mas indices sobre distintos archivos con la misma CI y valores validados pueden combinarse. La entrada tendrá una clave de indizacion y uno o mas esquemas de punteros.
  - Indice Multiclave: el arbol R. Admite la creacion de indices especiales que no estén basados en una clave. Un arbol R, es una evolucion del arbol B+ para d dimensiones.

- Esquemas de bits(BITMAP): Para dominios con cardinalidad pequeña. Un esquema de bits para un campo es un vector de valores booleanos. A cada valor del dominio se le hace corresponder una posición.
  - Puede ser simple o multiclave, concatenando esquemas de varios campos.
- Mascaras sobre BITMAP:
  - Máscaras para condiciones de igualdad: Un bit para cada posible valor, 0 o 1, bivaluada. Filtra los que tienen exactamente un 1 o 0. S and Q = Q
  - Máscaras con bits que admitan cualquier valor: Un bit para posible valor, 0, 1 o q, trivaluada. Permite filtrar valores exactos y otros que no tener en cuenta, q.
    - ∘ S XOR Q = 1
- Esquema de bits Simple vs. Multiclave: Es conveniente que el diseño de los esquemas de bits se realice atendiendo a las necesidades de procesamiento.
- Acceso Invertido: Es un tipo de acceso indicado multiclave orientado a optimizar el coste de acceso en procesos muy concretos. Trata de averiguar informacion delimitada de ciertos archivos con condiciones muy concretas.
  - Procura averiguar toda esta informacion accediendo solo a los indices, sin llegar a acceder al dato. Deben localizarse unívocamente cada registro.
    - Se ejecutaran primero las condiciones y después se busca en los indices objetivo, pero observando los indices en vez de las entradas y lo que buscamos es la entrada par un determinado indice.
  - Es eficiente si requiere acceder a pocos indices. Los indices que soporta este acceso se denominan indices inversos. Los secundarios tambien se denominan listas invertidas.
  - Un fichero invertido es el que soporta este tipo de acceso, y se llama totalmente invertido si todos los campos invertidos.
  - Costes: n es el numero de bloques del indice. r el numero de resultados.
    - Selección:
      - Listas invertidas no ordenadas: El maximo es n, y el medio (n+1)/2.
      - Listas invertidas ordenadas: log2(n+1)
      - Esquema de bits: n.
      - Otro tipo de indices: Depende de la estructura.
    - Proyección:
      - Esquema de bits con puntero implícito: min(n, r)
      - Cualquier otro caso: n.