

Tema 7: Organización 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.
 - **Dirección de Máquina:** la dirección física del registro.
 - **Dirección Relativa:** Registro en el espacio de Direccionamiento.
 - **Puntero simbólico:** Identificación lógica del registro.
 - **Entrada:** Registro formado por punteros.
 - Entrada de **índice primario:** $\text{clave} * \text{puntero_externo}$
 - Entrada de **índice secundario:** $\text{clave} * \text{long_lista} * (\text{puntero_externo})^{\text{long_lista}}$.
 - El número de entradas es igual a la cardinalidad del dominio(CI).
 - Al buscar recorreremos el índice hasta encontrar una entrada, y si insertamos se pone al final de la lista de punteros. Conviene que el índice este ordenado, y no consecutivo.
 - **Directorio:** Archivo formado por entradas.
 - **Tipos de índices:**
 - **Primario:** La clave de indización es identificativa. 1 entrada - 1 clave - 1 registro. Filtrado máximo.
 - **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 índices 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 Índice:** Acceso al índice.
 - **Localización por varios índices:** Suma del acceso a cada índice.
 - **Recuperación:** $\text{acceso_índice} + \text{acceso_datos}$.
 - **Actualización:**
 - **Inserciones:** Inserción de entradas. **Coste = acc_índice + 1.**
 - **Borrados:** **Coste = acc_índice + 1**
 - **Índice primario:** suelen requerir borrado de entradas, se vacía al ser 1 solo.
 - **Índice secundario:** pueden requerir modificación de entradas, si se vacía.
 - **Modificaciones:**
 - **CI:** Suele implicar borrado + reinserción de entrada. $2 * \text{acc_ind} + 2$
 - **CD/CO:** Cambia ubicación reg., cambia puntero. $\text{acc_índice} + 1$
 - **Taxonomía de índices:**
 - Según el **carácter de la clave de indización:**
 - **Índices primarios** vs. **índices secundarios.**
 - Según la **correspondencia entre entrada y registros:**
 - **Denso:** Una entrada del índice para cada registro.
 - **No denso:** Una entrada para cada cubo de datos.
 - Según el **recubrimiento del índice:**
 - **Exhaustivo:** Todos los registros tienen un entrada.
 - **Parcial:** No se indizan todos los registros.
 - Según la **estructura:** Índices simples vs. índices multinivel.

▸ **Índice Simple Denso:**

- **Naturaleza:** serial, secuencial, o Direcccionada.
- **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:
 - **Índice ordenado:** Preferible inserción ordenada + reorganización local.
 - **Índice disperso:** Pierde eficiencia si cambia, es mas útil como índice temporal.

▸ **Índice Simple No Denso:** Una entrada por cubo de datos.

- **Restricción:** índice y organización base deben ser necesariamente secuenciales y con $\text{clave_indización} = \text{clave_ordenación}$ (CO=CI)
- **Usos:** Varias posibilidades de acceso:
 - **Procesos ordenados** (a la totalidad): Acceso serial.
 - **Procesos selectivos** (solución única): A través del índice.
 - **Mixtos:** (selección de un rango): acceso indizado + serial.
- **Ventajas:**
 - Tamaño muy reducido, tiene menor coste y mayor tasa de acierto. (Apunta a cubo)
 - Se ahorra muchas actualizaciones de índice.
 - Se pueden utilizar prefijos, el mínimo tamaño para reconocerlo, en lugar de utilizar toda la clave en la entrada.
- **Inconvenientes:**
 - Solo puede existir un índice no denso para cada archivo.
 - La inserción del registro debe ser ordenada, pero se localiza con el índice.
 - La inserción de la entrada es ordenada, y conlleva pesadas reorganizaciones.

▸ **Índice Multinivel:** Es un índice con n niveles, árbol de índices.

- El coste es un acceso por nivel, interesa definir nodos pequeños a costa de tener mas niveles. Bloquear la raíz 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 solución, pero cuando son volátiles requiere reorganización.
- **Posibles soluciones:**
 - **Árboles binarios:** Presenta problemas de vecindad y desequilibrio.
 - **Árboles AVL:** Resuelve el desequilibrio con reorganización 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.
 - **Árboles B:** La mejor solución. Incluye varias entradas por nodo y se construye en orden ascendente.

◦ **Indización en Árboles B:** Comienza por las hojas y se va construyendo hacia arriba.

- **Nodo:** Contiene entradas de índice (CI-puntero) y punteros (a hijos).
- **Orden del árbol (m):** Capacidad de los nodos, según los punteros, el número de hijos de un nodo.

▸ **Corolario:**
$$m \cdot T_{\text{ptro_interno}} + k \cdot T_{\text{entrada}} \leq T_{\text{nodo}}$$

- $k = m - 1$. 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**:

$$k_{\min} = \left\lfloor \frac{k}{2} \right\rfloor$$

extra cu

Corolario

primario: $r = e$
— = entradas

secundario: $e = \# \text{ clave}$
Dem de clave en índice.

- ¿Cuántos descendientes como mínimo tienen los nodos intermedios?
(suponiendo política de 'dividir cuando desborda')

$$m_{\min} = k_{\min} + 1$$

→

$$m_{\min} = \left\lfloor \frac{m+1}{2} \right\rfloor$$

- Tamaño del fichero de índices**

Se puede obtener una **cota superior** del fichero de índices

$$N_{\max}^{\text{nodos fichero}} = \frac{\text{nº entradas fichero}}{k_{\min}} = 1 + (\text{nº entrd. fich}) / k_{\min}$$

$$T_{\max}^{\text{fichero}} = n_{\max}^{\text{nodos fichero}} \cdot T_{\text{nodo}}$$

Esta vale.

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

Cuántas entradas tiene cada nivel del árbol. El proceso

| nivel | nodos | entradas | acumulado |
|-------|--------------------|-----------------------------------|--|
| 1 | 1 <i>hijos</i> | 1 | 1 |
| 2 | 2 | 2 · k_{\min} | 1 + 2 k_{\min} |
| 3 | $m_{\min} \cdot 2$ | 2 · $m_{\min} \cdot k_{\min}$ | 1 + 2 $k_{\min} + 2 m_{\min} k_{\min}$ |
| n | $2 m_{\min}^{n-2}$ | $2 m_{\min}^{n-2} \cdot k_{\min}$ | $(2 m_{\min}^{n-1}) - 1$ |
| $n+1$ | $2 m_{\min}^{n-1}$ | $2 m_{\min}^{n-1} \cdot k_{\min}$ | $(2 m_{\min}^n) - 1$ |

Cada nodo m_{\min} hijos.

Cota sup.

$$n \leq 1 + \log_{\left\lfloor \frac{m+1}{2} \right\rfloor} \left(\frac{e+1}{k_{\min}} \right)$$

Nivel imposible

- Recuperar una entrada: $\# \text{ accesos} = \# \text{ niveles}$.
- Recuperar un registro aleatorio, se recuperan la entrada y tantos cubos de datos como punteros tenga la entrada. $\text{Coste} = (n-1) \cdot T_{\text{nodo}} + c \cdot E_{\text{cubo}}$.
- El coste de cualquier actualización sobre el índice, es el coste de localización 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 índices 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 raíz garantizan una ocupación minima: $k_{min} = \text{floor}(2k/3)$
 - Los nodos intermedios cuentan con $(2k/3) + 1$ descendientes: $m_{min} = \text{floor}((2m+1)/3)$
- **Calculo de costes:** Saberlo como teoría, pero no como práctica.
 - La localización 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 dirección del nodo nuevo.
 - Especialmente eficiente con punteros externos grandes, indice secundario.
 - La partición/promoción 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.

$$m \cdot T_{\text{puntero_interno}} + (m-1) \cdot T_{\text{clave}} \leq T_{\text{nodo}}$$

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

interno y externo son distintos, conveniria calcularla por separado

$$k \cdot (T_{\text{clave}} + T_{\text{puntero(s) externo}}) + T_{\text{puntero interno}} \leq T_{\text{nodo}}$$

encadmtio. bidireccional req. dos punteros internos

debe contener al menos una entrada comp.

Sino alcanza a tener una entrada se aumenta el tamaño de nodo a 2 o mas bloques.

- **Ocupación minima de las hojas sera:** $k_{min} = \text{floor}(k+1/2)$
- **Ocupación minima de los nodos intermedios sera:** $m_{min} = \text{floor}(m+1/2)$
- **Calculo del numero de niveles:**
 - numero de hojas= $\text{floor}(e/k_{min})$ tal que e=numero total de entradas.
 - numero de nodos(n-1) = $\text{floor}(\text{nodosNiveln}/m_{min})$
 - 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 logradadas 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 indización y uno o mas esquemas de punteros.
 - **Indice Multiclave: el arbol R.** Admite la creación de indices especiales que no estén basados en una clave. Un arbol R, es una evolución 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. $S \text{ and } Q = Q$
 - **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.
 - $S \text{ XOR } Q = 1$
- **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 información delimitada de ciertos archivos con condiciones muy concretas.
 - Procura averiguar toda esta información accediendo solo a los índices, sin llegar a acceder al dato. Deben localizarse unívocamente cada registro.
 - Se ejecutaran primero las condiciones y después se busca en los índices objetivo, pero observando los índices en vez de las entradas y lo que buscamos es la entrada por un determinado índice.
 - Es eficiente si requiere acceder a pocos índices. Los índices que soporta este acceso se denominan **índices inversos**. Los secundarios también 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 número de bloques del índice. r el número de resultados.
 - **Selección:**
 - **Listas invertidas no ordenadas:** El máximo es n , y el medio $(n+1)/2$.
 - **Listas invertidas ordenadas:** $\log_2(n+1)$
 - **Esquema de bits:** n .
 - **Otro tipo de índices:** Depende de la estructura.
 - **Proyección:**
 - **Esquema de bits con puntero implícito:** $\min(n, r)$
 - **Cualquier otro caso:** n .