# TEMA 2

# INDEXACIÓN

‘*Una piedra angular de la teoría de estructura de datos es la distinción entre estructuras fundamentales y “avanzadas”.*’

Niklaus Wirth,
‘Algoritmos + Estructuras de Datos = Programas’

# OBJETIVOS DE ESTE CAPÍTULO:

- Comparar las ED's básicas con sus equivalentes avanzadas
- Estudiar el concepto y propiedades fundamentales de los árboles AVL
- Estudiar el concepto y propiedades fundamentales de los árboles B como solución al problema de gran número de claves
- Introducir el concepto de Indexación

# ÍNDICE TEMA 2.

# 1. INDEXACIÓN.- Justificación

- Formas de trabajo sobre Memoria Principal:
  - ➔ Tablas
  - ➔ Listas enlazadas
  - ➔ Árboles binarios de búsqueda
- Formas de trabajo sobre Memoria Secundaria:
  - ➔ Ficheros secuenciales
  - ➔ Ficheros de acceso directo
- Objetivo: Acceso directo por clave que reduzca el nº de accesos a realizar.

# Algunas definiciones

Árbol de Búsqueda:

> Las claves del subárbol izquierdo de un nodo son menores, y las del subárbol derecho, mayores, que la del nodo que se trata.

Árbol Ordenado:

> Aquel en que las ramas de cada nodo están ordenadas

Grado:

> Número de descendientes directos de un nodo interior

Grado de un Árbol:

> Máximo de los grados de todos los nodos del árbol

Árbol Binario:

> Árboles ordenados de Grado 2.

# 2. CRITERIOS DE EQUILIBRIO

■ Árbol perfectamente equilibrado

'Para cada nodo, 'el número de nodos' del subárbol izquierdo y del derecho difieren como mucho en una unidad'

➔ Las claves pueden estar ordenadas o no

➔ Criterio algo 'rígido'

■ Generación de un árbol perfectamente equilibrado:

- Hay que conocer el número de claves del árbol

- El procedimiento ha de ser recursivo

- No es necesario que el árbol esté ordenado

- Si 'N' es el número de claves del árbol, se generan dos subárboles, con:

Ni = N div 2 nodos

Nd = N - Ni - 1 nodos

- Altura máxima del árbol <= log (n) + 1

■ Árbol equilibrado

➔ 'Para cada nodo, 'la longitud' del subárbol izquierdo y del derecho difieren como mucho en una unidad'

➔ Si además el árbol es binario de búsqueda ➔ **Árboles AVL** (Adelson-Velskii y Landis, 1962)

➔ Altura mínima del árbol: log ( N + 1 )

■ **Inserción en árbol equilibrado AVL**

➔ Situación de desequilibrio. Factor de equilibrio.

➔ Rotaciones: secuencia de rotaciones de punteros que se intercambian cíclicamente:

➔ Simple derecha, DD

➔ Simple izquierda, II

➔ Doble derecha, ID(*)

➔ Doble izquierda, DI(*)

➔ Estado de equilibrio de cada nodo:

```
TYPE
tarbol = ^tnodo
tnodo = RECORD
        clave: tclave;
        izquierdo, derecho: tarbol;
        equilibrio: -1..1;
END;
```

➔ Proceso de Inserción :

1. Seguir el camino de búsqueda hasta ver que la clave no esta en el árbol.
2. Insertar el nuevo nodo, y obtener los nuevos valores de equilibrio.
3. Volver por el camino de búsqueda comprobando el factor de equilibrio de cada nodo

➔ Parámetros que se necesitan :

1. Árbol donde realizar la inserción
2. Clave que se quiere insertar
3. Información de la altura del subárbol (tipo de parámetro ?)

➔ Casos posibles en la realización:

hi < hd $\rightarrow$ a$\uparrow$.equilibrio = +1

hi = hd $\rightarrow$ a$\uparrow$.equilibrio = 0

hi > hd $\rightarrow$ a$\uparrow$.equilibrio = -1

## ■ Borrado en árbol equilibrado AVL

➔ Proceso de Borrado:

1. Búsqueda de la clave.
2. Estudio del equilibrio del árbol.

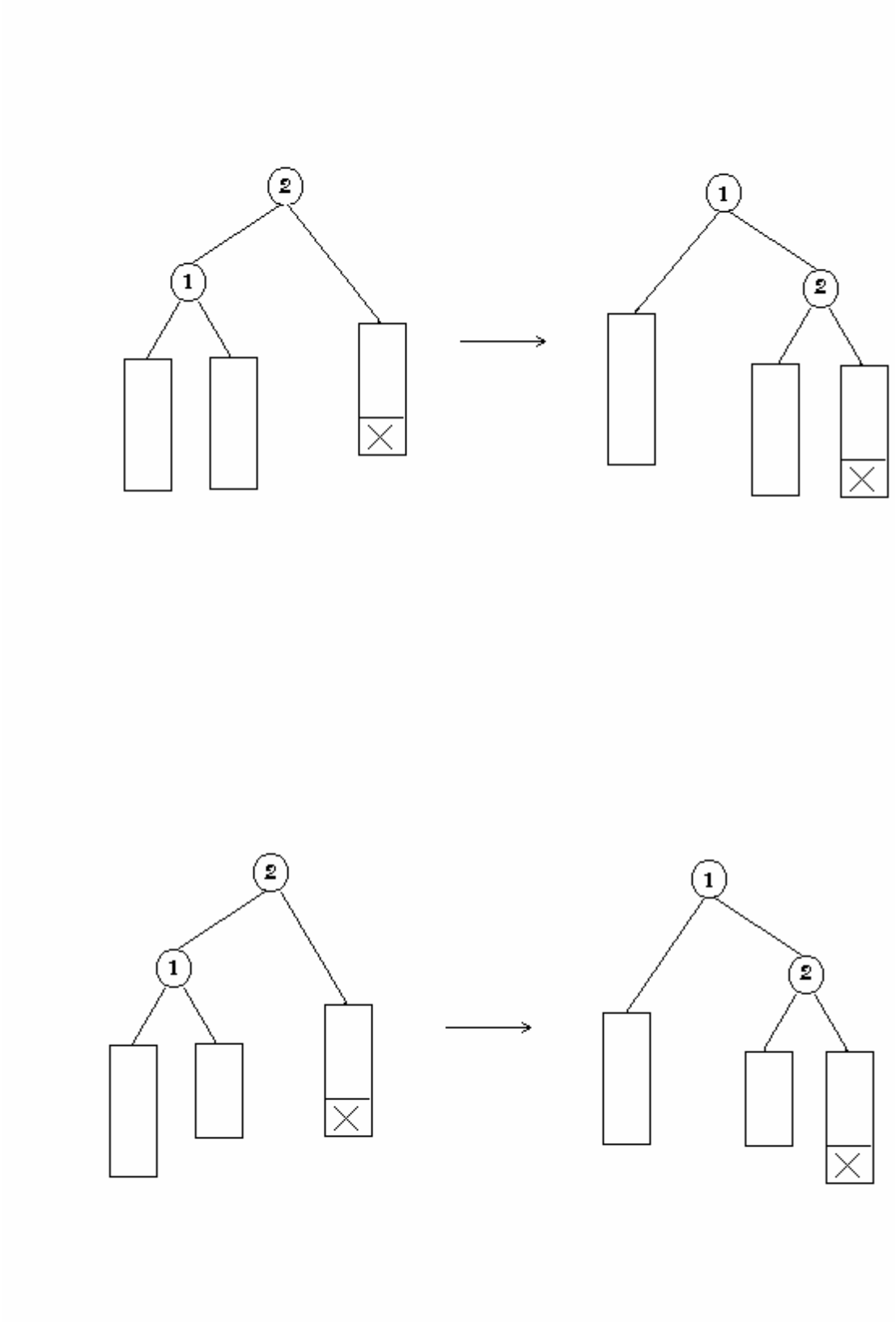
➔ Parámetros que se necesitan:

1. Árbol donde realizar la inserción
2. Clave que se quiere borrar.
3. Información de la altura del subárbol.

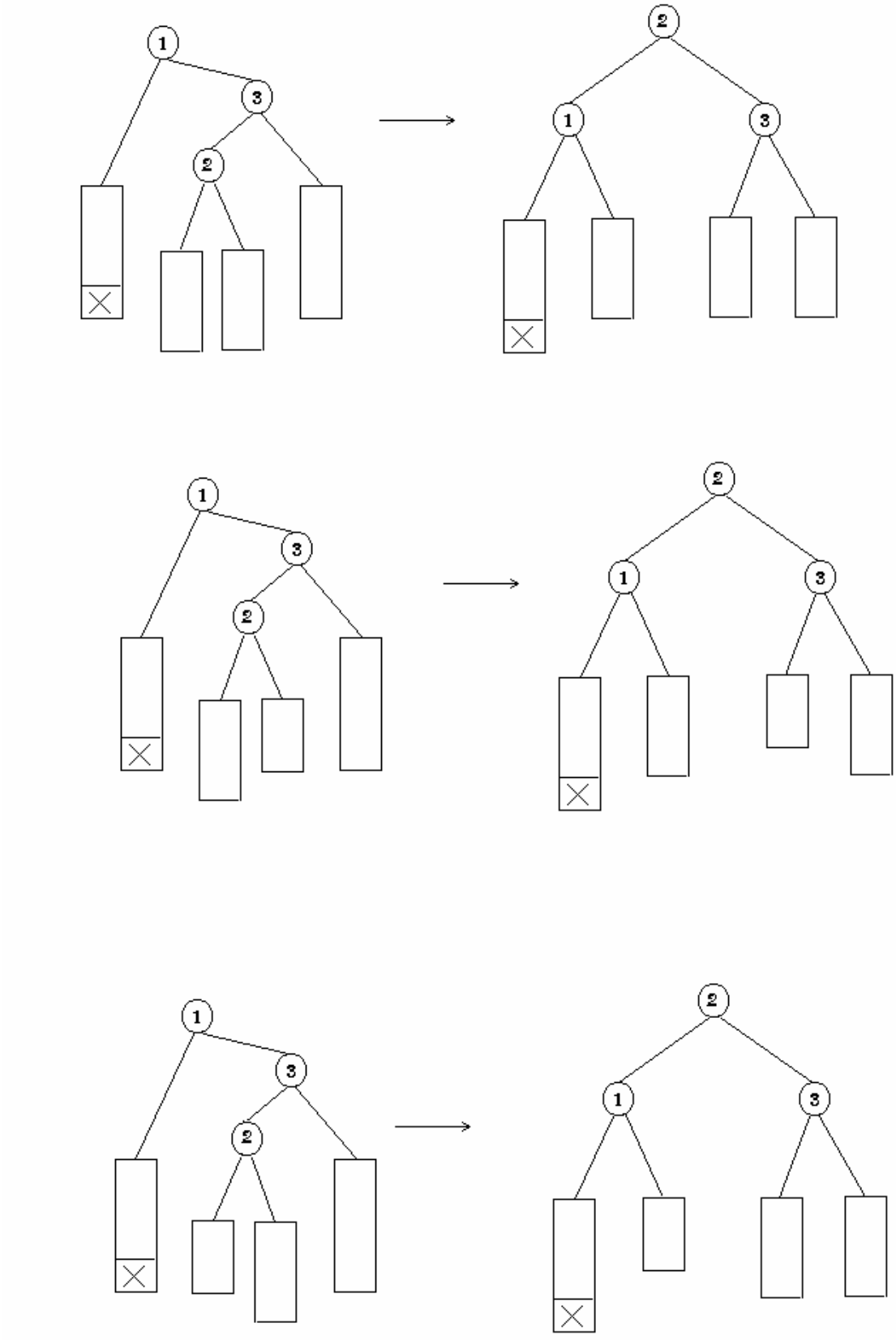
➔ Casos posibles en la implementación:

1. Borrado de una hoja sin desequilibrio
2. Borrado de una hoja con desequilibrio
3. Borrado de un nodo interno sin desequilibrio
4. Borrado de un nodo interno con desequilibrio

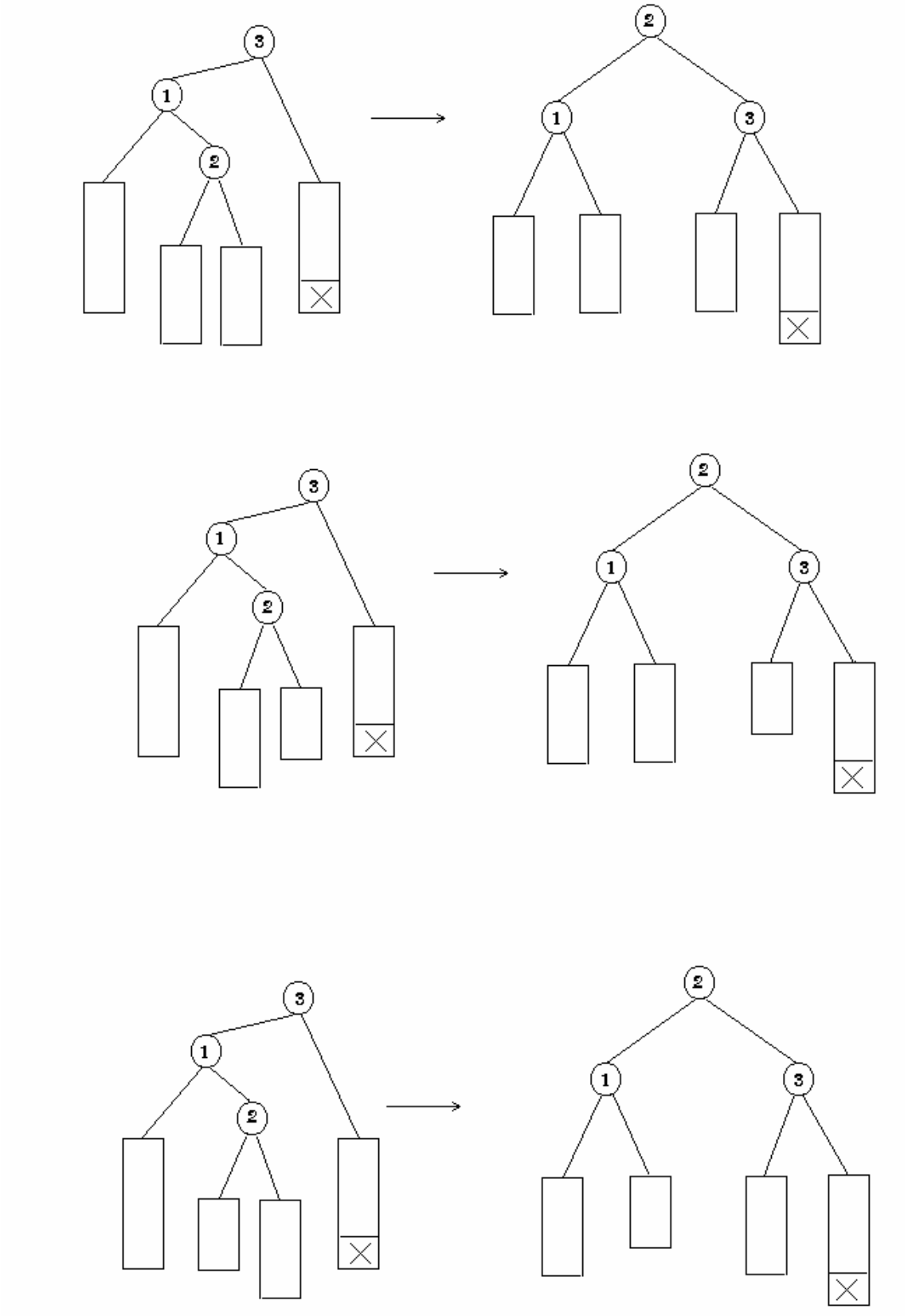
➔ Caso 2: DD, II, DI, ID,



➔ Caso 2: DD, II, DI, ID,



➔ Caso 2: DD, II, DI, ID,



# 3. ÁRBOLES B

*"Al buscar una forma de controlar el crecimiento, el árbol perfectamente equilibrado se deshecha inmediatamente debido al gran esfuerzo que requiere la operación de reequilibrado."*

N. Wirth.[1]

- Árboles **MULTICAMINO**: Cada nodo puede tener más de dos ramas.

  - ➔ Realización:

  - → Array de registros

  - → Lista lineal

  - ➔ Utilización:

    - → Construcción y mantenimiento de árboles de búsqueda a gran escala.

  - ➔ División del árbol en subárboles

    - → Páginas

[1] Niklaus Wirth, Algoritmos + Estructuras de Datos = Programas, Ed. del Castillo, Madrid 1980, pág. 261.

## ■ Árboles **B**

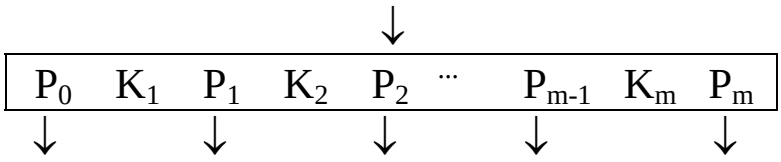
➔ El orden del árbol es n

➔ Todas las páginas, excepto la raíz, tienen entre n y 2n claves

➔ Cada página es una página-hoja (no tiene descendientes), o tiene m+1 descendientes; m → número de claves de la página

➔ Todas las páginas-hoja se encuentran al mismo nivel

Supóngase la página del árbol:




¿Qué casos pueden presentarse si una clave ‘x’ no se encuentra en esta página?

■ Inserción de un elemento en un árbol **B**

➔ Si la página donde se inserta tiene m > 2n nodos?

➔ Si " " " tiene m = 2n nodos?

➔ Declaración de la estructura de una página:

```
type pagina = record
                  m: indice;
                  p0: ref;
                  e: array [1 .. nn] of item
              end;

const  nn = 2 * n;

type   ref = ↑pagina;
       indice = 0 .. nn;
       item = record
                  clave: integer;
                  p: ref;
                  contador: integer
              end;
```

➔ Conveniencia de una formulación recursiva?

## ■ Algoritmo de Búsqueda e Inserción en un árbol B

```
Buscar ( x: integer;                    (* Nodo a insertar/buscar *)
         a: tArbol;                              (* Árbol B *)
         var cambio: boolean;            (* El árbol ha crecido *)
         var u: tItem;)                  (* Elemento que sube de nivel *)


    if a = nil
            - copiar 'x' en 'u'
            - cambio := true

    else
            - buscar x en el array
            - si está → operaciones a realizar con la clave
            - si no está →
                llamada recursiva (x, página siguiente, cambio, u)

                if cambio   (* se sube un elemento *)
                    - si m < 2n
                            - se inserta en esa página
                            - m := m + 1
                            - cambio := false

                    - si m = 2n

                            - divide página
                            - copiar en u el elemento en el medio
    end
```

■ Borrado de un elemento en un árbol **B**

➔ Casos:

- la clave a borrar está en una hoja

- la clave a borrar no está en una hoja

↓

Se sustituye por elementos adyacentes

## ■ Número de claves de la página en un árbol B

1.- La información se almacena dentro de la página asociada a las claves

- Estructura para las claves: .....2n * tamaño clave
- Estructura para la información: .....2n * tamaño info
- Estructura para los descendientes:..... (2n + 1) * tamaño descendientes

➔ Tamaño de la página ≥

$$1 + (2n * Tclave) + (2n * Tinfo) + (2n + 1) * Tdescen =$$

$$= (1 + Tdescen) + 2n * (Tclave + Tinfo + Tdescen)$$

$$\rightarrow n \leq \frac{Tpagina - (1 + Tdescen)}{Tclave + Tinfo * Tdescen}$$

2.- La información se almacena en un TAD independiente (p.ej. un fichero).

- Estructura para las claves: .....2n * tamaño clave

- Estructura para las posiciones: .....2n * tamaño posición

- Estructura para los descendientes: ..... (2n + 1) * tamaño descendientes

➔ Tamaño de la página ≥

$$1 + (2n * Tclave) + (2n * Tposición) + (2n + 1) * Tdescen =$$

$$= (1 + Tdescen) + 2n * (Tclave + Tposición + Tdescen)$$

➔

$$n \leq \frac{Tpagina - (1 + Tdescen)}{Tclave + Tposición * Tdescen}$$

■ Altura de la página de un árbol B

➔ En el peor caso:

- La raíz sólo tiene una clave

- Todas las demás páginas tienen n claves

Nivel 1 ➢ 1 sola clave ➢ 2 descendientes

Nivel 2 ➢ 2 pág. con n claves ➢ $2 * (n + 1)$ descen.

Nivel 3 ➢ $2 * (n + 1)$ pág. con n claves ➢ $2 * (n + 1)^2$ descen.

Nivel 4 ➢ $2 * (n + 1)^2$ pág. con n claves ➢ $2 * (n + 1)^3$ descen.

.
.
.

Nivel d ➢ $2 * (n + 1)^{d-2}$ pág. con n claves ➢ $2 * (n + 1)^{d-1}$ descen.

➔ Un razonamiento iterativo:

- Una página tiene 'n' claves ➔ 'n + 1' descendientes

- La raíz sólo tiene 1 clave

- Un árbol con 'x' claves tiene 'x + 1' descendientes en el nivel de las hojas

➔ La altura será aquella que cumpla:

$x + 1 \geq$ Nº de descendientes al nivel de las hojas $= 2 * (n + 1)^{d - 1}$

de donde:

$$d \leq 1 + \log_{n*1} ((x + 1)/2)$$

■ Árboles B*:

➔ Concepto de 'Redistribución'

➔ Árboles B*:

Árbol B especial, en el que cada nodo está lleno por lo menos en dos terceras partes (66%). Proporcionan mejor utilización del almacenamiento que los árboles B[(6)]. La situación se genera cuando se produce una inserción al dividir, de dos páginas a tres en lugar de una a dos.

→ Cada página tiene un máximo de m descendientes.

→ Todas las páginas, excepto la raíz y las hojas, tiene al menos (2m - 1) / 3) descendientes.

→ La raíz tiene al menos dos descendientes, al menos que sea 'hoja'.

→ Todas las páginas-hoja se encuentran al mismo nivel.

→ Una página que no sea hoja, con k descendientes, contiene k - 1 claves.

→ Una página-hoja contiene al menos [(2m - 1) / 3] claves, y no más de (m - 1)

[6] Folk & Zoellick, Estructuras de Archivos, un conjunto de Herramientas Conceptuales, pág. 375.

- Árboles B+: El índice del árbol se almacena en un árbol B separado, que permite encontrar la información, que se encuentra en los registros de un fichero secuencial indexado.

# 5. INDEXACIÓN

- **Índice** ➔ Establece un 'Orden' en un fichero
  - ➔ Puede ser múltiple
  - ➔ Primera aproximación : tabla de registros

- **Operaciones básicas de un fichero Indexado :**
  - ➔ Creación
  - ➔ Lectura de Índice
  - ➔ Actualización de Índice
  - ➔ Inserción de registros
  - ➔ Eliminación de registros
  - ➔ Modificación de registros