

## ORDENACIÓN

### TEMA 4

*...o como perder menos tiempo al ordenar un archivo haciendo menos accesos a memoria.*

### OBJETIVOS DE ESTE CAPITULO:

- Mostrar distintos procesos de ordenación sobre estructuras de datos conocidas.
- Simplificar la tarea de búsqueda de un elemento dentro de un conjunto *ordenado*.
- Comprobar cómo la estructura de datos de base influye en los algoritmos que llevan a cabo una tarea dada.

---

### -INDICE-

4.1 Justificación

4.2 Ordenación por mezcla

4.3. Ordenación externa

4.3.1. Mezcla directa

4.3.2. Mezcla natural

4.4. Intercalación múltiple

4.4.1. Definición

4.4.2. Intercalación múltiple para Clasificación de Archivos Grandes

## 4.1. JUSTIFICACIÓN

- Existencia de volúmenes de datos amplios
- Tamaño de memoria limitado
- Objetivo: minimizar el número de accesos a los ficheros

## 2. ORDENACIÓN POR MEZCLA.

- N ficheros ‘ordenados’ de unen para formar un único archivo ordenado:

A	B	Operación C
Vacío	Vacío	nada
Vacío	No vacío	copiar todos los registros de B en C
No vacío	Vacío	copiar todos los registros de A en C
No vacío	No vacío	bucle a repetir hasta que A o B estén vacíos (*)

(\*) Operaciones a realizar en este caso:

Leer ( A, ElemA )

Leer ( B, ElemB )

Bucle hasta llegar al final de A o B

Comparar ( ElemA, ElemB )

if ElemA > ElemB

then Escribir ElemB en C

Leer B

else Escribir ElemA en C

Leer A

Introducir el resto de A o B en C

### 3. ORDENACIÓN EXTERNA.

#### ■ MEZCLA DIRECTA

Se ordenan las secuencias de un archivo en grupos de tamaño creciente =  $2^n$  (1, 2, 4, 8, ...):

Sea la secuencia:

22 6 25 48 32 5 12 20 31 35

- Para grupos de tamaño = 1, tenemos:

22 25 32 12 31  
6 48 5 20 35

- Ordenando estas secuencias:

6, 22 25, 48 5, 32 12, 20 31, 35

- Agrupando en grupos de tamaño = 2:

6, 22 5, 32 31, 35  
25, 48 12, 20

- Ordenando estos grupos:

6, 22, 25, 48 5, 12, 30, 32, 31, 35

- Agrupando en grupos de tamaño = 4:

6, 22, 25, 48 31, 35  
5, 12, 20, 32

- Ordenando:

5, 6, 12, 20, 22, 25, 32, 48 31, 35

- La última agrupación ( tamaño = 8 ):

5, 6, 12, 20, 22, 25, 32, 48  
31, 35

- Ordenando:

5 , 6 , 12 , 20 , 22 , 25 , 31 , 32 , 35 , 48

- El procedimiento a seguir es el siguiente:

- 1- Dividir la secuencia a ordenar en 2 subsecuencias de menor tamaño, cada una la mitad de la secuencia original.
- 2- Mezclar las dos secuencias de forma ordenada, para generar otra mayor, formada por conjuntos ordenados de valores con  $2^0$ ,  $2^1$ ,  $2^2$ , ... elementos.
- 3- Iterar los pasos 1 y 2 hasta que el tamaño del conjunto ordenado ( $2^n$ ) sea mayor que el número de elementos a ordenar.

## ■ MEZCLA NATURAL

Las secuencias intermedias no tienen tamaño prefijado ni longitud constante. Estas se generan con sus elementos ordenados, separando un elemento nuevo a otra secuencia si no se respeta esta condición.

Se incluyen separadores de secuencia.

- Sea la cadena de números:

**F1:** 3 15 7 17 9 8 14 2 41 24 36 1 13 42 26 35 5 33

Aplicando el método de mezcla natural con tres ficheros, uno original y dos auxiliares (frecuentemente llamados 'cintas'), el resultado es:

**F2:**        3 15     9        2 41     1 13 42     5 33  
               7 17     8 14    24 36     36 35

## 4.4. Intercalación múltiple

- Definición

Problema: Dadas 'k' listas de claves, crear a partir de ellas, una sola lista ordenada secuencialmente

Proceso: Comparar dos valores de la clave, seleccionar la menor, e insertarlo en la lista final.

- ♦ Si las claves pueden estar repetidas en varias listas, el problema es más complejo, pues habría que recorrerlas todas para cada clave, incrementándose exponencialmente el número de comparaciones necesarias.
- ♦ Si cada clave sólo se encuentra en una lista, el problema se simplifica bastante. Un algoritmo sencillo en este caso sería:

**Para**  $i = 1$  **hasta**  $k$

Llamar a Entrada ( ) para leer Clave (  $i$  ) de la Lista (  $i$  )

**Siguiente**  $i$

{ Intercalación de  $k$  claves }

**Mientras** ( Existan más claves )

{ Encontrar una clave como el menor valor de secuencia  
entre las claves disponibles en las  $k$  claves }

MENOR = 1

**Para**  $i = 2$  **hasta**  $k$

**Si** nombre [  $i$  ] < clave [ MENOR ]

MENOR =  $i$

**Siguiente**  $i$

Escribir clave [MENOR ] en ARCHIVO\_SALIDA

{ Se reemplaza la clave que se escribió }

Llamar a Entrada ( ) para tomar clave [ MENOR ] de lista [ MENOR ]

**Fin Mientras**

- ♦ Si el número de listas a intercalar es mayor que 8, el n° de comparaciones a realizar es considerablemente elevado, y éste método no resulta eficiente.

### ● Intercalación múltiple para Clasificación de Archivos Grandes

Procedimiento: 'División del archivo en múltiples *particiones*'

Método:

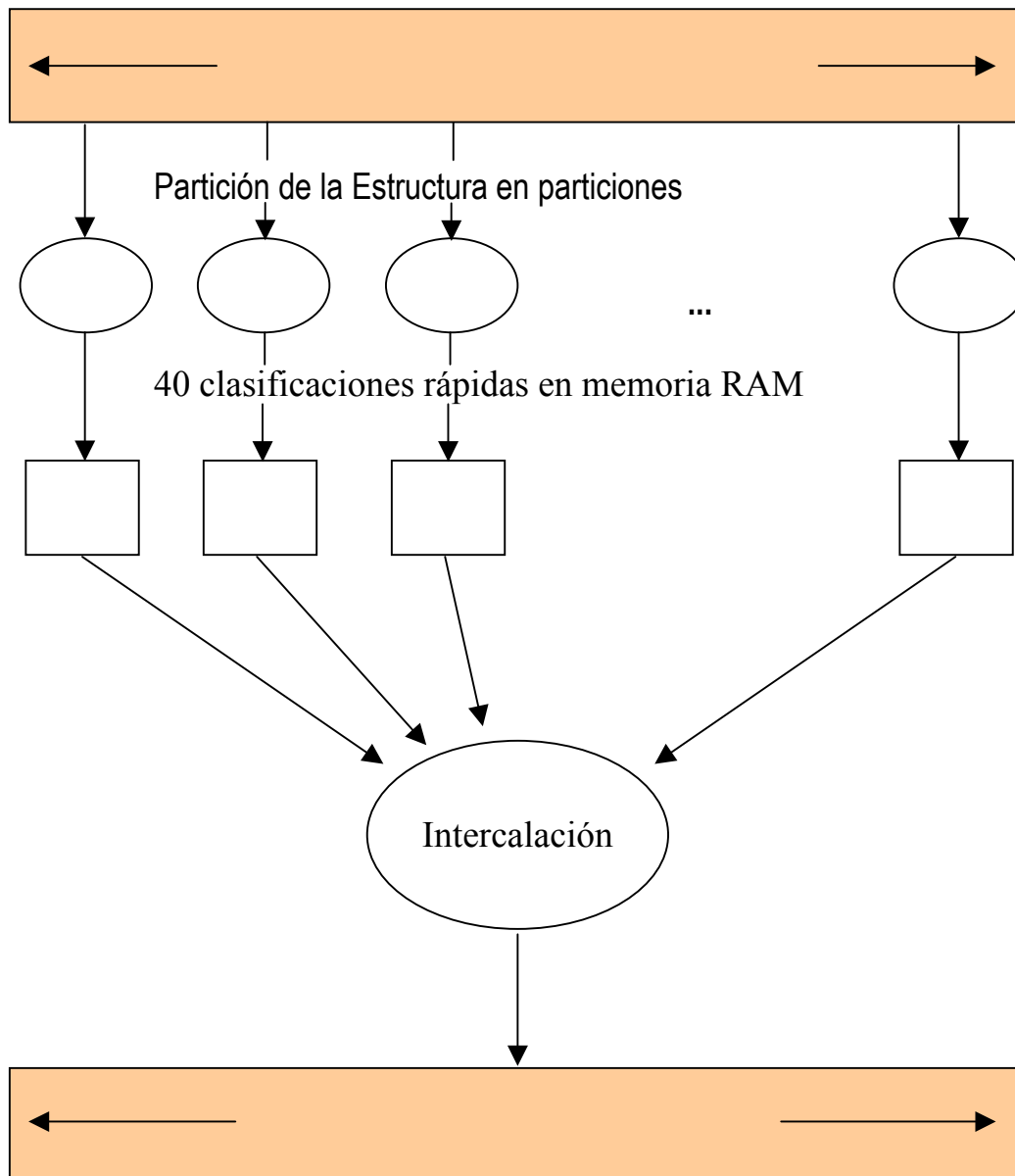
- Los algoritmos de clasificación en memoria RAM trabajan en su propio espacio, junto con una pequeña cantidad para la pila y algunas variables temporales.
- Por ello, puede transferirse un subconjunto temporal del archivo a memoria RAM, ordenar allí sus registros, y luego devolver de nuevo al disco el resultado como un ‘subarchivo clasificado’.
- Por último, aplicando el modelo de intercalación múltiple desarrollado en el punto anterior. Un ejemplo ilustra este proceso.

Ejemplo:

- Supongamos un archivo de **400.000 registros**, cada uno de ellos de **100 bytes**, con una clave de **10 bytes**.  
→ ¿Cuál es el tamaño de este archivo?
- Supongamos que sólo disponemos de **1Mb** de memoria libre, lo que impide ordenar directamente el archivo.
- Para aplicar Intercalación Múltiple, partimos el archivo inicial en particiones que entren en memoria, para almacenar cada parte en un fichero aparte cuando esté ordenada, e intercalar posteriormente los ficheros ordenados.

La **figura-I** explica el proceso.

→ El algoritmo de intercalación múltiple puede ser una solución adecuada a este tipo de archivos.



**Figura-I.** *Clasificación e Intercalación de un archivo por medio de particiones.*

- **¿Tamaño de las particiones?**

$$\frac{\text{Tamaño de la Memoria}}{\text{Tamaño del Registro}} = \text{N}^\circ \text{ de registros de la partición}$$



1.000.000 bytes de Memoria

---


$$\frac{\text{10 bytes por registro}}{\text{10 bytes por registro}} = 10.000 \text{ registros}$$


---

Características del mecanismo de clasificación:

- Permite la ordenación de grandes ficheros
  - Las lecturas dentro de las ‘particiones’ son secuenciales; sólo se producen saltos al cambiar de partición.
  - Al ser la E/S secuencial, pueden utilizarse cintas para las operaciones de E/S.
  - Al utilizar una cinta para E/S, el tamaño del archivo a ordenar puede ser grande.
  - Puede utilizarse un árbol de selección para reducir el tiempo y el espacio necesarios para implementar el proceso. La profundidad del árbol se mantiene en unos límites tolerables y es igual a  $\log_2 K$ , siendo ‘K’ el número de listas a intercalar<sup>1</sup>.
- 

Problemas que se presentan:

- ♦ **Número de desplazamientos** (salto de cabezas de disco de un archivo a otro) elevado.



- Como la memoria RAM está totalmente ocupada, cada una de las particiones utiliza un buffer de tamaño:

$$T = \frac{\text{Tamaño Memoria}}{\text{Nº de particiones}}$$

En el ejemplo, esta cantidad es:  $T = \frac{400.000}{40} = 10.000$

- Para leer una partición completa, para completar la operación de intercalación habrá que desplazarse un n° de veces:

---

<sup>1</sup> M.J.Folk & B.Zoellick, Estructuras de Archivos, un Conjunto de Herramientas Conceptuales, Addison Wesley Ed., 1992, pág. 287.

$$N = N^{\circ} \text{ de particiones} * N^{\circ} \text{ de desplazamientos/partición}$$

$$\text{En el ejemplo: } N = 40 * 40 = 1.600 \text{ desplazamientos}^2$$

### Posibles soluciones:

#### 1. Realizar intercalación en más de un paso.

Para ahorrar tiempo de desplazamiento, se propone el siguiente mecanismo: Intercalar particiones de mayor tamaño, seguido de una segunda intercalación de particiones de menor tamaño, utilizando un árbol de selección para las comparaciones. El número de comparaciones necesarias para intercalar X ficheros de N registros cada uno sería ahora  $N * \log_2 X$ . La figura-II explica el proceso.

#### 2. Incrementar el tamaño de las particiones

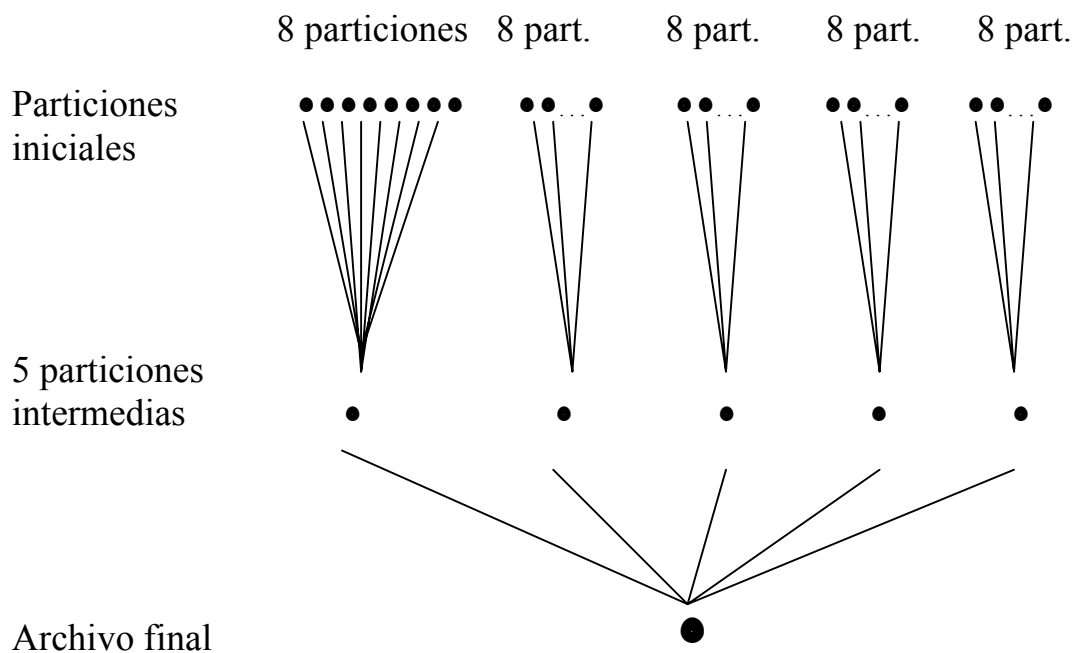
¿Cómo incrementar el tamaño de las particiones más allá de los límites impuestos por la memoria RAM disponible?.



Mecanismo de Selección por reemplazo: Seleccionar de la memoria la clave con valor más bajo, enviarla a la salida y luego reemplazarla con una nueva clave de la lista de entrada.

---

<sup>2</sup> La clasificación e intercalación media es del orden de  $O(N^2)$ , lo que indica que el proceso se deteriora rápidamente a medida que N aumenta.



**Figura-II.** *Intercalación de más de un paso.*

• **Intercalación de varios pasos:** Patrones de Intercalación **x : y : z : 1**

Indica el número de porciones a tratar en cada fase.

P.ej., con 100 porciones, un patrón de 10:5:2:1 indica lo siguiente:

Fase 1: 100 porciones/10 → quedan 10 porciones para la siguiente fase

Fase 2: 10 porciones/5 → quedan 2 porciones para la siguiente fase

Fase 3: 2 porciones/2 → queda 1 porción

$$\text{Tamaño del buffer} = \frac{\text{Tamaño RAM}}{\text{Nº de porciones de cada fase}}$$

**Patrón válido → Tamaño del buffer >= Tamaño del registro**