Tema 6. ORGANIZACIÓN DE ARCHIVOS

Estructuras de Datos y Algoritmos II
Grado en Ingeniería Informática

M José Polo Martín
mjpolo@usal.es

Universidad de Salamanca

curso 2022-2023

VNiVERSiDAD D SALAMANCA
CAMPUS DE EXCELENCIA INTERNACIONAL

800 AÑOS 1218~2018

DEPARTAMENTO DE INFORMÁTICA Y AVTOMÁTICA

Contenido

1 Introducción

- Ventajas del almacenamiento de información en memoria principal:
  - rapidez de acceso
  - igual cantidad de tiempo para acceder a datos en diferentes posiciones
- No siempre es posible almacenar datos en memoria principal:
  - situaciones en que la cantidad de datos a procesar supera la capacidad de la memoria principal
  - situaciones en que se necesita guardar la información en almacenamiento estable
- Necesidad de utilizar dispositivos de almacenamiento externo cuyas características de acceso difieren bastante de las de la memoria principal
- Las estructuras de datos aplicadas a colecciones de datos en almacenamiento secundarios se denominan ARCHIVOS o FICHEROS

Conceptos básicos relacionados con la organización de archivos

Archivo, registro, campo

- Los lenguajes de programación suelen tener el tipo de datos archivo, destinado a representar datos en memoria secundaria
  - **Archivo**. Conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros.
  - **Registro**. Colección de diferentes entidades de nivel inferior denominadas campos
  - **Campo**. Elemento de datos caracterizado por su nombre, tamaño y tipo de datos. Unidad mínima de información de un registro
- La organización de archivos define la forma en que los registros se disponen en el medio de almacenamiento
- Tres organizaciones fundamentales
  - Organización secuencial
  - Organizaciones indexadas
  - Organización directa o aleatoria

Conceptos básicos relacionados con el acceso a archivos

- El sistema operativo divide la memoria secundaria en bloques de igual tamaño y considera un archivo como una colección de bloques
- **Operación básica en archivos:** copiar un solo bloque del archivo a un área temporal de la memoria principal (buffer), cuyo tamaño es idéntico al de un bloque, de acuerdo con el orden en que el sistema operativo maneja los bloques
- El tiempo para encontrar un bloque y copiarlo de memoria secundaria a memoria principal es **muy grande** comparado con el tiempo de proceso de los datos
- Al evaluar el **tiempo de ejecución** de los algoritmos que operan con información almacenada en memoria secundaria es importante considerar el número de veces que se lee/escribe un bloque: número de accesos a bloque o **número de operaciones de entrada /salida**

2 Organización Secuencial

- Forma básica de almacenar un conjunto de registros que constituyen un archivo
- Los registros quedan grabados consecutivamente cuando el archivo se crea y debe accederse a ellos de forma consecutiva cuando el archivo se procesa

  Búsqueda SECUENCIAL
- Método de búsqueda o direccionamiento de un registro en un archivo secuencial que consiste en leer y comprobar uno tras otro los registros del archivo hasta encontrar el registro deseado
- Método muy lento, normalmente solo se emplea en medios de almacenamiento secuencial, pero sirve como punto de partida y comparación para medir las mejoras de otros métodos
- Búsqueda secuencial o búsqueda por bloque es O(n)
  - El número de operaciones de E/S es **proporcional** al número de registros y se incrementa en proporción directa al incremento del tamaño del archivo
- **Objetivo**. Encontrar métodos mejores para acceder a registros individuales

Archivos secuenciales ORDENADOS

- En muchos casos, los registros de un archivo secuencial se clasifican según el valor de algún campo o conjunto de campos de cada registro
- El campo elegido para ordenar, clasificar e identificar a cada registro se le denomina **clave** (no tiene ninguna relación con la posición física de los registros)
- **Dirección relativa o número relativo de registro (NRR)**: posición relativa de un registro respecto al primer registro del archivo

  Búsqueda BINARIA
- Método de direccionamiento que requiere que el archivo esté clasificado según el valor de la clave que se emplea en la búsqueda
- Proceso para encontrar un registro con valor K_b para la clave:
  1. Comparar K_b con la clave del registro que ocupa la posición central del fichero
  2. Según el resultado de la comparación se repite el mismo proceso en el área del fichero que contenga el registro hasta encontrar el registro buscado, si existe

Número máximo de accesos búqueda binaria

Ejemplo: búqueda binaria en archivo secuencial ordenado. $K_b =$ "Felix"

- En promedio el número de operaciones de E/S para un archivo de n registros es proporcional a lg n:
  - Mejora considerable frente a la búsqueda secuencial
  - Incrementar el tamaño del archivo no supone aumentar directamente el número de accesos: duplicar el tamaño del archivo sólo implica una comparación más como máximo
- Búsqueda binaria más efectiva pero **solo funciona si los registros están ordenados** en términos de la clave que se está buscando

NRR	Clave	Otros campos	
1	ADOLFO		
2	ALBERTO		
3	ALONSO		
4	ANA		
5	ANTONIO		
6	BEATRIZ		
7	BERTA		
8	BRAULIO		
9	CARLOS		
10	CARMEN		
11	CAROLINA		
12	CLARA	2ª inspección:	$K_b > K_{12} \Downarrow$
13	DANIEL		
14	DIANA		
15	DORA		
16	ELISA		
17	ELSA	3ª inspección:	$K_b > K_{17} \Downarrow$
18	EMILIIO		
19	ESTEBAN		
20	FÉLIX	4ª inspección:	$K_b = K_{12}$ **Éxito**
21	FRANCISCO		
22	GONZALO		
23	IGNACIO	1ª inspección:	$K_b < K_{23} \Uparrow$
24	JOAQUÍN		
25	LEONARDO		
26	MANUEL		
27	MARÍA		
28	MARIANO		
29	MIGUEL		
30	NATALIA		
31	OLGA		
32	PASCUAL		
33	PEDRO		
34	PILAR		
35	RAMIRO		
36	RAMÓN		
37	RAQUEL		
38	RAÚL		
39	RICARDO		
40	RUBÉN		
41	SANTIAGO		
42	SARA		
43	SUSANA		
44	TERESA		
45	VERÓNICA		
46	VICENTE		

3 Organizaciones Indexadas

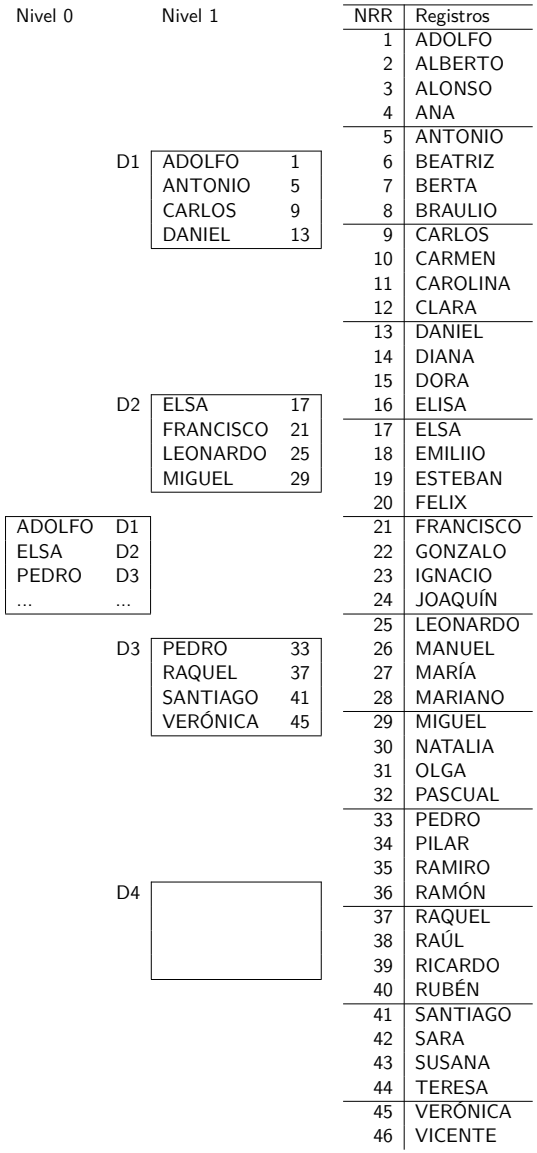
- Las organizaciones indexadas constan de dos tipo de archivos
  - Archivo de datos para almacenar los registros
  - Archivo índice para localizar los registros. Contiene la posición relativa de cada registro o grupo de registros en el archivo de datos
- **Método de direccionamiento directo**: acceso a un registro determinado sin necesidad de consultar los registros precedentes
- Archivo índice

  tabla cuya entrada es la clave del registro buscado y da como salida la dirección relativa del registro en el fichero de datos o del área del fichero que contiene al registro buscado
- Organizaciones indexadas:
  - Organización **Secuencial Indexada**: fichero de datos **ordenado** según el valor de la clave de búsqueda
  - Organización **No Secuencial Indexada**: fichero de datos **no ordenado** según la clave de búsqueda

Organización Secuencial Indexada

- El archivo está clasificado secuencialmente según el valor de la clave de entrada utilizada en el índice
- No es necesaria una entrada por cada registro: el índice incluye referencias a bloques de registros, los cuales se explorarán después para finalizar la búsqueda

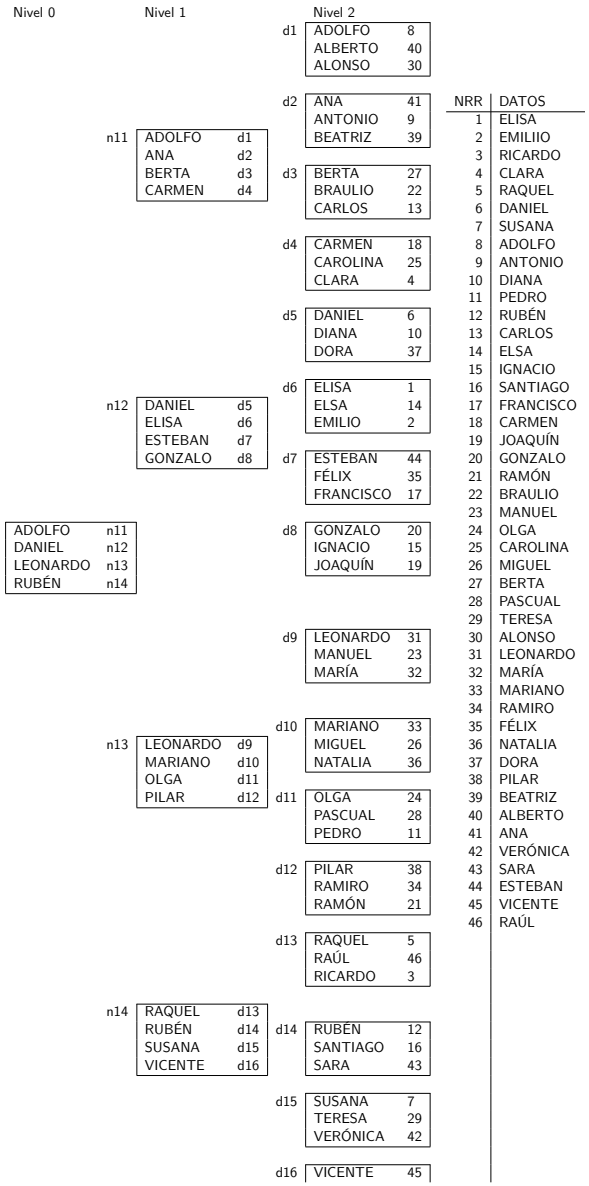
Ejemplo con índice de dos niveles →



Organización NO secuencial indexada

- El archivo no está clasificado según el valor de la clave de entrada utilizada en el índice
- Se necesita una entrada por cada registro, ocupan más espacio y aumenta el tiempo para explorarlos
- Normalmente dan una **dirección simbólica** que se corresponde con el valor de la clave primaria

Ejemplo con índice de tres niveles →



Índices primarios y secundarios

- En la mayoría de los ficheros se buscan registros según el valor de más de una clave. El fichero solo puede estar ordenado respecto a una de las claves
  - Se ordena según la clave de búsqueda más frecuente y se le asocia una organización secuencial indexada (índice primario)
  - Para el resto de las claves (no secuenciales) se asocia una organización no secuencial indexada (índices secundarios)
- Para evitar índices muy grandes, a veces, se recurre a índices de varios niveles:
  - El índice de nivel más alto da la posición del índice del siguiente nivel
  - El índice del nivel más bajo da la posición del bloque de registros que contiene al registro buscado
  - El bloque se explora secuencialmente o mediante búsqueda binaria para encontrar el registro

Inserción de registros en Organización Secuencial Indexada

- La organización secuencial indexada ofrece como ventajas:
  - admite procesamiento secuencial y directo
  - el índice es de menor extensión
- Desventaja
  - la distribución secuencial del fichero de datos es más difícil de mantener al insertar nuevos registros
- Técnicas para tratar las inserciones:
  - **Área de desborde**. Se almacenan los nuevos registros en otras posiciones dejando punteros en el lugar del fichero donde debería almacenarse el registro $\Rightarrow$ un acceso adicional cada vez que se busque uno de los nuevos registros
  - **Espacio libre distribuido**. Se dejan espacios libres en el fichero al cargarlo inicialmente
- En ambos casos serán necesarias operaciones básicas de mantenimiento

Ejemplo de inserción de registros con claves: SAMUEL, JUAN, JOSÉ y MARTA

Área de Desborde

Nivel 0

ADOLFO	D1
ELSA	D2
PEDRO	D3
...	...

Nivel 1

D1	ADOLFO	1
	ANTONIO	5
	CARLOS	9
	DANIEL	13
D2	ELSA	17
	FRANCISCO	21
	LEONARDO	25
	MIGUEL	29
D3	PEDRO	33
	RAQUEL	37
	SANTIAGO	41
	VERÓNICA	45
D4		

NRR	Registros	
1	ADOLFO	
2	ALBERTO	
3	ALONSO	
4	ANA	
5	ANTONIO	
6	BEATRIZ	
7	BERTA	
8	BRAULIO	
9	CARLOS	
10	CARMEN	
11	CAROLINA	
12	CLARA	
13	DANIEL	
14	DIANA	
15	DORA	
16	ELISA	
17	ELSA	
18	EMILIIO	
19	ESTEBAN	
20	FÉLIX	
21	FRANCISCO	
22	GONZALO	
23	IGNACIO	
24	JOAQUÍN ...	102
25	LEONARDO	
26	MANUEL	
27	MARÍA	
28	MARIANO ...	103
29	MIGUEL	
30	NATALIA	
31	OLGA	
32	PASCUAL	
33	PEDRO	
34	PILAR	
35	RAMIRO	
36	RAMÓN	
37	RAQUEL	
38	RAÚL	
39	RICARDO	
40	RUBÉN ...	100
41	SANTIAGO	
42	SARA	
43	SUSANA	
44	TERESA	
45	VERÓNICA	
46	VICENTE	

Área de DESBORDE		
100	SAMUEL	
101	JUAN	
102	JOSÉ ...	101
103	MARTA	

Espacio Libre Distribuido

Nivel 0

ADOLFO	D1
DANIEL	D2
LEONARDO	D3
RAQUEL	D4

Nivel 1

D1	ADOLFO	1
	ANA	5
	BERTA	9
	CARMEN	13
D2	DANIEL	17
	ELISA	21
	ESTEBAN	25
	GONZALO	29
D3	LONARDO	23
	MARIANO	37
	OLGA	41
	PILAR	45
D4	RAQUEL	49
	RUBÉN	53
	SUSANA	57
	VICENTE	61

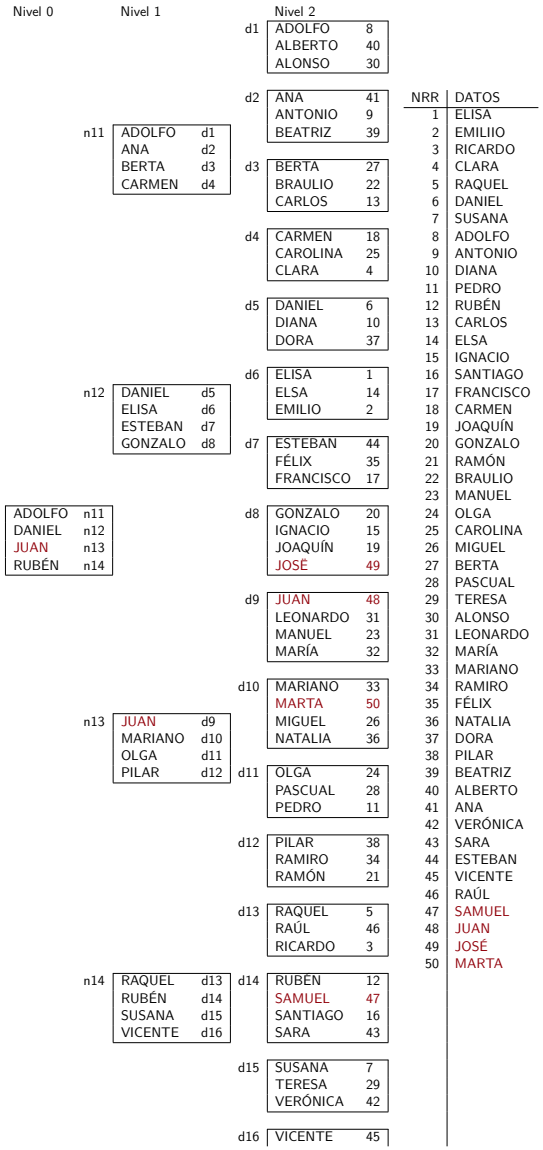
NRR	Registros	
1	ADOLFO	
2	ALBERTO	
3	ALONSO	
4		
5	ANA	
6	ANTONIO	
7	BEATRIZ	
8		
9	BERTA	
10	BRAULIO	
11	CARLOS	
12		
13	CARMEN	
14	CAROLINA	
15	CLARA	
16		
17	DANIEL	
18	DIANA	
19	DORA	
20		
21	ELISA	
22	ELSA	
23	EMILIIO	
24		
25	ESTEBAN	
26	FÉLIX	
27	FRANCISCO	
28		
29	GONZALO	
30	IGNACIO	
31	JOAQUÍN	
32	JUAN	JOSÉ!!!
33	LEONARDO	
34	MANUEL	
35	MARÍA	
36		
37	MARIANO	
38	MIGUEL	
39	NATALIA	
40		
41	OLGA	
42	PASCUAL	
43	PEDRO	
44	6	
45	PILAR	
46	RAMIRO	
47	RAMÓN	
48		
49	RAQUEL	
50	RAÚL	
51	RICARDO	
52		
53	RUBÉN	
54	SAMUEL	
55	SANTIAGO	
56	SARA	
57	SUSANA	
58	TERESA	
59	VERÓNICA	
60		
61	VICENTE	

Inserción de registros en Organización No Secuencial Indexada

- La inserción de nuevos registros no plantea problemas en el archivo de datos: simplemente se añaden al final del fichero
- La organización del índice, sin embargo, se complica

  en las organizaciones indexadas. Realmente lo que se organiza es el archivo índice (organizaciones B y B+ que estudiaremos en el tema 7)

Ejemplo →



4 Organización Directa. Dispersión

- Método de organización que opera basándose en una función de cálculo de dirección o función HASH que convierte la clave de un registro en una dirección dentro del fichero: $h : K \rightarrow D$
- Ventajas
  - Rapidez de acceso: permite el acceso directo a cualquier registro sin consultar índices ni leer registros anteriores
  - Aumenta la velocidad de búsqueda sin necesidad de tener los registros ordenados
  - El tiempo de búsqueda es independiente del número de registros del fichero
- Características de la función hash:
  - Simple de calcular
  - Transformación de claves en direcciones apropiadas equilibrando el espacio de almacenamiento y el tiempo de búsqueda
  - Distribución uniforme generando posiciones diferentes para claves diferentes

Método de DISPERSIÓN

- Consiste en dividir el área de almacenamiento en grupos denominados CUBOS que permiten almacenar un determinado número de registros cada uno
- Si el archivo se divide en N cubos, éstos se numeran de 0 a N-1 y se utiliza una función de dispersión de forma que a cada valor de la clave le hace corresponder algún entero en el rango 0..N-1
- Algunos cubos pueden estar llenos mientras que otros pueden estar vacíos o poco ocupados⇒ colisiones o desbordes

Búsqueda directa

Si k es el valor de la clave, $h(k)$ proporciona el número de cubo que contiene al registro con clave k, si ese registro existe

Colisión o desborde

Si la función de cálculo de dirección asigna un registro a un cubo que está lleno

Método de Dispersión. Ejemplo

El método de dispersión requiere la elección previa de:

- **Función de cálculo de dirección o función hash**: simple de calcular y que asigne direcciones de la forma más uniforme posible (módulo, cuadrado dígitos centrales, elegir dítitos, ...)
- **Método para resolver los desbordes** que genere posiciones alternativas para almacenar los registros que sean asignados a cubos llenos

CLAVE	conversión numérica de la clave	clave%29
ELISA	53921	10
EMILIIO	5493997	5
RICARDO	1931147	8
CLARA	33111	22
RAQUEL	119453	2
DANIEL	415953	6
SUSANA	242151	1
ADOLFO	147367	18
ANTONIO	1537597	17
DIANA	49151	25
PEDRO	85417	12
RUBÉN	14255	16
CARLOS	311372	28
ELSA	5321	14
IGNACIO	9751397	2
SANTIAGO	21539177	7
FRANCISCO	611539237	26
CARMEN	311455	24
JOAQUÍN	1719495	27
GONZALO	7759137	13
RAMÓN	11475	20
BRAULIO	2114397	7
MANUEL	415453	28
OLGA	7371	5
CAROLINA	31173951	24
MIGUEL	497453	16
BERTA	25131	17
PASCUAL	8123413	20
TERESA	351521	12
ALONSO	137527	9
LEONARDO	35751147	5
MARÍA	41191	11
MARIANO	4119157	26
RAMIRO	114917	19
FÉLIX	65397	2
NATALIA	5131391	15
DORA	4711	13
PILAR	89311	20
BEATRIZ	2513199	1
ALBERTO	1325137	11
ANA	151	6
VERÓNICA	55175931	9
SARA	2111	23
ESTEBAN	5235215	19
VICENTE	5935535	18
RAÚL	1143	12

Espacio inicial de Almacenamiento
29 cubos de capacidad 2

CUBO			¡desbordes!
0			
1	SUSANA	BEATRIZ	
2	RAQUEL	IGNACIO	FÉLIX
3			
4			
5	EMILIO	OLGA	LEONARDO
6	DANIEL	ANA	
7	SANTIAGO	BRAULIO	
8	RICARDO		
9	ALONSO	VERÓNICA	
10	ELISA		
11	MARÍA	ALBERTO	
12	PEDRO	TERESA	RAÚL
13	GONZALO	DORA	
14	ELSA		
15	NATALIA		
16	RUBÉN	MIGUEL	
17	ANTONIO	BERTA	
18	ADOLFO	VICENTE	
19	RAMIRO	ESTEBAN	
20	RAMÓN	PASCUAL	PILAR
21			
22	CLARA		
23	SARA		
24	CARMEN	CAROLINA	
25	DIANA		
26	FRANCISCO	MARIANO	
27	JOAQUÍN		
28	CARLOS	MANUEL	

Desbordes y Nuevas Inserciones

- Algunos métodos utilizan un área especial, **área de Desborde**, para almacenar los registros desbordados, registrando la dirección del cubo de desborde en el cubo inicialmente asignado: **encadenamiento de desbordes**
- ¡¡No se necesita ninguna técnica adicional para tratar las nuevas inserciones!!!

CLAVE	conversión numérica de la clave	clave%29
ELISA	53921	10
EMILIIO	5493997	5
RICARDO	1931147	8
CLARA	33111	22
RAQUEL	119453	2
DANIEL	415953	6
cline5-6 SUSANA	242151	1
ADOLFO	147367	18
ANTONIO	1537597	17
DIANA	49151	25
PEDRO	85417	12
RUBÉN	14255	16
CARLOS	311372	28
ELSA	5321	14
IGNACIO	9751397	2
SANTIAGO	21539177	7
FRANCISCO	611539237	26
CARMEN	311455	24
JOAQUÍN	1719495	27
GONZALO	7759137	13
RAMÓN	11475	20
BRAULIO	2114397	7
MANUEL	415453	28
OLGA	7371	5
CAROLINA	31173951	24
MIGUEL	497453	16
BERTA	25131	17
PASCUAL	8123413	20
TERESA	351521	12
ALONSO	137527	9
LEONARDO	35751147	5
MARÍA	41191	11
MARIANO	4119157	26
RAMIRO	114917	19
FÉLIX	65397	2
NATALIA	5131391	15
DORA	4711	13
PILAR	89311	20
BEATRIZ	2513199	1
ALBERTO	1325137	11
ANA	151	6
VERÓNICA	55175931	9
SARA	2111	23
ESTEBAN	5235215	19
VICENTE	5935535	18
RAÚL	1143	12
SAMUEL	214453	27
JUAN	1415	23
JOSÉ	1725	14
MARTA	41131	9

Espacio inicial de Almacenamiento
29 cubos de capacidad 2

CUBO			
0			
1	SUSANA	BEATRIZ	
2	RAQUEL	IGNACIO	29
3			
4			
5	EMILIO	OLGA	29
6	DANIEL	ANA	
7	SANTIAGO	BRAULIO	
8	RICARDO		
9	ALONSO	VERÓNICA	31
10	ELISA		
11	MARÍA	ALBERTO	
12	PEDRO	TERESA	30
13	GONZALO	DORA	
14	ELSA	JOSÉ	
15	NATALIA		
16	RUBÉN	MIGUEL	
17	ANTONIO	BERTA	
18	ADOLFO	VICENTE	
19	RAMIRO	ESTEBAN	
20	RAMÓN	PASCUAL	30
21			
22	CLARA		
23	SARA	JUAN	
24	CARMEN	CAROLINA	
25	DIANA		
26	FRANCISCO	MARIANO	
27	JOAQUÍN	SAMUEL	
28	CARLOS	MANUEL	
29	LEONARDO	FÉLIX	
30	PILAR	RAÚL	Cubos de desborde
31	MARTA		
32			

Dispersión Dinámica

- Modificación del método de organización directa que permite variar el número de cubos asignados en función de su densidad de ocupación
- El número de cubos asignados inicialmente aumenta o disminuye a medida que estos se van llenando por nuevas inserciones o se van desocupando por eliminaciones
- Gran flexibilidad pues se incrementa o disminuye el espacio de almacenamiento según se necesita
- Las operaciones de expansión y reducción originan reasignación de registros, esto supone un alto coste por las lecturas y escrituras que deben realizarse
- La asignación dinámica de cubos se realiza mediante expansiones

Expansiones

- Expansión total: cuando se supera la densidad de ocupación el número de cubos se duplica

  Inicialmente $\Rightarrow$ N cubos
  Primera expansión $\Rightarrow$ $2N$ cubos
  Segunda expansión $\Rightarrow$ $4N$ cubos
  ...

- Expansión parcial: se incrementa el número de cubos en un 50%, de forma que dos expansiones parciales equivalen a un expansión total

  Inicialmente $\Rightarrow$ N cubos
  Primera expansión $\Rightarrow$ $N + N/2 = 3N/2$ cubos
  Segunda expansión $\Rightarrow$ $3N/2 + N/2 = 2N$ cubos
  Tercera expansión $\Rightarrow$ $2N + N = 3N$ cubos
  Cuarta expansión $\Rightarrow$ $3N + N = 4N$ cubos
  ...

Ejemplo

N=2 C=2

Densidad máxima de ocupación 80%

K	KmodN (densidad)	KmodN (densidad)	KmodN (densidad)
42	0 (25%)	2	2
24	0 (50%)	0	0
15	1 (75%)	3	7
53	1 (100%)	1	5
21		1 (63%)	5
12		0 (75%)	4
14		2 (88%)	6
18			2 (50%)
49			1 (56%)
128			0 (63%)
22			6 (69%)
23			7 (75%)
67			3 (81%)

N=2

0	42	24
1	15	53

1ª expansión

N=4

0	24	12
1	53	21
2	42	14
3	15	

2ª expansión⇒

N=8

0	24	128
1	49	
2	42	18
3	67	
4	12	
5	53	21
6	14	22
7	15	23

Densidad mínima de ocupación 40%

Eliminar

K	densidad
53	75%
24	69%
12	63%
18	56%
22	50%
128	44%
23	37,5%

N=8

0	24	128
1	49	
2	42	18
3	67	
4	12	
5	53	21
6	14	22
7	15	23

K	K mod N
49	1
42	2
67	3
21	1
14	2
15	3

Reducción->

N=4

0		
1	49	21
2	42	14
3	67	15

Observaciones

- La idea básica de la dispersión consiste en trasformar las claves en direcciones de memoria mediante una función de cálculo de dirección
- Generalmente se aprovecha su potencial para la implementación en memoria secundaria pero . . .
  - Estas direcciones pueden hacer referencia a memoria primaria, en cuyo caso tendríamos un vector o array
  - Todos los conceptos pueden utilizarse para construir el TAD conocido como tablas de dispersión (hashing) o tablas asociativas
  - Prácticamente todos los compiladores utilizan tablas asociativas para implementar la tabla de símbolos (la búsqueda de cualquier símbolo se realiza en un tiempo constante)

5 Clasificación externa

- Clasificación de datos almacenados en memoria secundaria, es decir, de datos organizados en archivos
- Métodos:
  1. Clasificación por intercalación
  2. Clasificación por intercalación múltiple
  3. Clasificación polifase

5.1 Clasificación por Intercalación

Consiste en organizar un archivo en fragmentos o secuencias ordenadas cada vez más grandes

Fragmento de longitud k

Secuencia de k registros clasificados según el valor de su clave
$r_1, r_2, ..., r_k \quad / \; clave(r_i) \leq clave(r_{i+1})\; para\; 1 < i < k$

Archivo organizado en fragmentos

Un archivo de n registros está organizado en fragmentos de longitud k si para todo $i \geq 0$ tal que $k * i \leq N$ la secuencia $r_{k*(i-1)+1}, r_{k*(i-2)+1}, ..., r_{k*i}$ es un fragmento de longitud k

Si n no es divisible por $k (n = p * k + q)$ habrá una secuencia final de q registros $(q \leq k)$ llamada cola, que será un fragmento de longitud q

Método de clasificación por intercalación

Clasifica los N registros utilizando cuatro archivos de la siguiente forma:

- Se dividen los N registros del archivo inicial en dos archivos F_1 y F_2, lo más uniformemente. Inicialmente F_1 y F_2 serán archivos organizados en ***fragmentos de longitud 1***
- Se combinan los fragmentos de F_1 y F_2 y se distribuyen en otros dos archivos G_1 y G_2 organizados en ***fragmentos de longitud 2***
- Se combinan los fragmentos de G_1 y G_2 y se distribuyen en los archivos F_1 y F_2 organizados en ***fragmentos de longitud 4***
- Se continua el proceso hasta que el archivo quede clasificado: todo los registros en un archivo con un único ***fragmento de longitud N***

Ejemplo

Archivo Inicial	F1 (k=1)	F2 (K=1)	G1 (k=2)	G2 (K=2)	F1 (k=4)	F2 (K=4)	G1 (k=8)	G2 (K=8)	F1 (k=16)	F2 (K=16)	G1 (k=32)	G2 (K=32)
93	93	31	31	3	3	10	3	9	3	8	3	
3	3	5	93	5	5	28	5	13	5	8	5	
28	28	96	28	10	31	40	10	30	9	10	8	
10	10	40	96	40	93	96	28	39	10	10	8	
54	54	85	54	9	9	13	31	54	13	22	9	
65	65	9	85	65	54	30	40	65	28	69	10	
30	30	39	30	13	65	39	93	85	30	77	10	
90	90	13	39	90	85	90	96	90	31		10	
10	10	8	8	69	8	8	8		39		13	
69	69	77	10	77	10	10	8		40		22	
8	8	10	8	22	69	22	10		54		28	
22	22		10		77		10		65		30	
31							22		85		31	
5							69		90		39	
96							77		93		40	
40									96		54	
85											65	
9											69	
39											77	
13											85	
8											90	
77											93	
10											96	

Algoritmo de clasificación

Algorithm clasificación(f:tipoArchivo)

```
 1: f1,f2,g1,g2: tipoArchivo
 2: k: tipoEntero                                   ->longitud fragmento
 3: cambiaEntrada:tipoLógico
 4: k ← 1
 5: dividir(f, f1,f2)
 6: cambiaEntrada ← FALSO
 7: mientras  k < númeroRegistros hacer
 8:     si  NO (cambiaEntrada) entonces
 9:         intercalación(k,f1,f2,g1,g2)
10:     si no
11:         intercalación(k,g1,g2,f1,f2)
12:     fin si
13:     k ← k*2
14:     cambiaEntrada ← NO(cambiaEntrada)
15: fin mientras
16: si cambiaEntrada entonces
17:     copiar(f1,f)
18: si no
19:     copiar(g1,f)
20: fin si
```

Algorithm leerRegistro(f: tipoArchivo, k: tipoEntero,ref finFrag: tipoLógico, ref registro: tipoRegistro, ref númeroRegistros: tipoEntero)

```
1: si  númeroRegistros = k O finFichero(f) entonces
2:     finFrag ← VERDADERO
3: si no
4:     leer(f,registro)
5:     númeroRegistros ← númeroRegistros + 1
6: fin si
```

Algoritmo de intercalación

Algorithm intercalación(k: tipoEntero,e1, e2,s1,s2: tipoArchivo)

```
 1: mientras NO(finFichero(e1)) O NO(finFichero(e2)) hacer
 2:     nReg1 ← 0
 3:     nReg2 ← 0
 4:     finFrag1 ← FASLO
 5:     finFrag2 ← FASLO
 6:     leerRegistro(e1,k,finFrag1, actual1, nReg1)
 7:     leerRegistro(e2,k,finFrag2, actual2, nReg2)
 8:     mientras NO(finFrag1) O NO(finFrag2) hacer
 9:         si  finFrag1 entonces
10:             actual ← actual2
11:             leerRegistro(e2,k,finFrag2, actual2, nReg2)
12:         si no, si  finFrag2 entonces
13:             actual ← actual1
14:             leerRegistro(e1,k,finFrag1, actual1, nReg1)
15:         si no, si actual1.clave < actual2.clave entonces
16:             actual ← actual1
17:             leerRegistro(e1,k,finFrag1, actual1, nReg1)
18:         si no
19:             actual ← actual2
20:             leerRegistro(e2,k,finFrag2, actual2, nReg2)
21:         fin si
22:         si cambiaSalida entonces
23:             escribe(actual, s1)
24:         si no
25:             escribe(actual, s2)
26:         fin si
27:     fin mientras
28:     cambiaSalida ← NO(cambiaSalida)
29: fin mientras
```

Análisis clasificación por Intercalación

Incialmente:		
F_1	fragmentos $k = 1$	$\frac{n}{2}$ registros
F_2	fragmentos $k = 1$	$\frac{n}{2}$ registros
Intercalación:		
	ENTRADA	SALIDA
1ª pasada	$F_1(k = 1)$ $F_2(k = 1)$	$G_1(k = 2)$ $G_2(k = 2)$
2ª pasada	$G_1(k = 2)$ $G_2(k = 2)$	$F_1(k = 4)$ $F_2(k = 4)$
3ª pasada	$F_1(k = 4)$ $F_2(k = 4)$	$G_1(k = 8)$ $G_2(k = 8)$
	...	...
pasada "i"	dos archivos (F's o G's) organizados organizados en fragmentos de longitud 2^i	

Archivo CLASIFICADO $\Rightarrow 2^i \geq n$

Comportamiento del algoritmo

- Archivo clasificado $\Rightarrow 2^i \geq n \Rightarrow i \geq \log n$
- Se necesitan $\log n$ pasadas de intercalación para clasificar un archivo con n registros:
  - Cada pasada lee/escribe n registros
  - El número de operaciones de entrada salida del proceso completo es $(n \log n/b)$, siendo b el número de registros por bloque
- Clasificación por intercalación es $O(n \log n)$
- Mejora del proceso: Partir de archivos organizados en fragmentos de longitud mayor que 1
  - Paso previo que lea grupos de k registros, los ordene en memoria principal según algún método de clasificación interna (quicksort, ...) y los devuelva a memoria secundaria como un fragmento de longitud k

Observaciones

- El tiempo empleado en leer/escribir un bloque entre M.P. Y M.S. es mayor que el tiempo de proceso de datos de M.P.
- Si solo hay un canal dedicado a la transferencia de datos llegará un momento en que los archivos estén organizados en fragmentos de longitud mayor que el tamaño de un bloque $\Rightarrow$ intercalar dos fragmentos supone leer muchos bloques de cada archivo

5.2 Clasificación por Intercalación Múltiple

- Modificación del método de clasificación por intercalación que puede aplicarse cuando se dispone de más de un canal de transferencia de datos
- Suponiendo $2m$ unidades de disco, cada una con su propio canal de transferencia de datos:
  - Se consideran m archivos de entrada $F_1, F_2, ..., F_m$ organizados en ***fragmentos de longitud*** k y m archivos de salida $G_1, G_2, ..., G_m$
  - Se leen m fragmentos (uno de cada archivo), se combinan en un fragmento de longitud mk y se van copiando en los archivos de salida
  - Se intercambian los fichero de entrada y de salida y se repite el proceso hasta que la longitud de los fragmentos sea mayor o igual que el número de registros

Clasificación por Intercalación Múltiple

Inicialmente	Pasada 1	Pasada 2	...	Pasada i (i par|i impar)
$F_1(k)$	$G_1(mk)$	$F_1(m^2k)$		$F_1(m^ik) \| G_1(m^ik)$
$F_2(k)$	$G_2(mk)$	$F_2(m^2k)$		$F_2(m^ik) \| G_2(m^ik)$
...	...	...		
$F_m(k)$	$G_m(mk)$	$F_m(m^2k)$		$F_m(m^ik) \| G_m(m^ik)$

Archivo clasificado $\Rightarrow m^ik \geq n$

- Número de pasadas proporcional a $log_m n \Rightarrow O(nlog_m n)$
- Aunque se utilizan m ficheros de entrada y m de salida, el proceso de intercalación múltiple en cada momento solo escribe en un fichero de salida
- ¡En cada pasada se desaprovechan (m-1) canales!

5.3 Clasificación Polifase

- Alternativa a la intercalación múltiple utilizando en cada paso **todos** los canales transferencia de datos
- Considerando $m+1$ canales E/S: m archivos de entrada y un solo archivo de salida
- Proceso para $m = 2$ (dos archivos de entrada y uno de salida): F_1, F_2 y F_3
  1. Inicialmente se dispone de dos archivos de entrada (F_1 y F_2) organizados en ***fragmentos de longitud 1***
  2. Se intercalan los registros de F_1 y F_2 en ***fragmentos de longitud 2*** en el archivo de salida F_3, hasta que uno de los archivos de entrada quede vacío
  3. El archivo que quede vacío en el primer paso se utiliza como salida en el siguiente y en él se intercalan los fragmentos de longitud 2 de F_3 y los restantes de longitud 1 (F_1 y F_2) quedando organizado en ***fragmentos de longitud 3***
  4. Se repite el paso 3 aumentando la longitud de los fragmentos hasta que el archivo quede clasificado

Ejemplo

F_1	F_2	F_3
Número de fragmentos (Longitud de fragmentos)	Número de fragmentos (Longitud de fragmentos)	Número de fragmentos (Longitud de fragmentos)
13(1)	21(1)	vacío
vacío	8(1)	13(2)
8(3)	vacío	5(2)
3(3)	5(5)	vacío
vacío	2(5)	3(8)
2(13)	vacío	1(8)
1(13)	1(21)	vacío
vacío	vacío	1(34)

- Las longitudes de los fragmentos que se obtienen en cada paso siguen la secuencia de Fibonacci 1, 1, 2, 3, 5, 8, 13, 21, ...
- Número de fragmentos en archivos iniciales dos números consecutivos de Fibonacci