Tema I.- Diseño Físico

Ficheros

Índices

Fundamentos de sistemas de bases de datos. Elmasri, R; Navathe, S. Addison-Wesley [cap.13 en 5ª edic.]

Índice

- Registros / Ficheros
- Operaciones sobre ficheros
- Ficheros de registros desordenados (heap)
- Ficheros de registros ordenados
- **Técnicas de dispersión (fichero hash)**
  - **Dispersión interna (memoria)**
  - **Dispersión externa (disco)**
    - **estática**
    - **dinámica**

Técnicas de dispersión (hash)

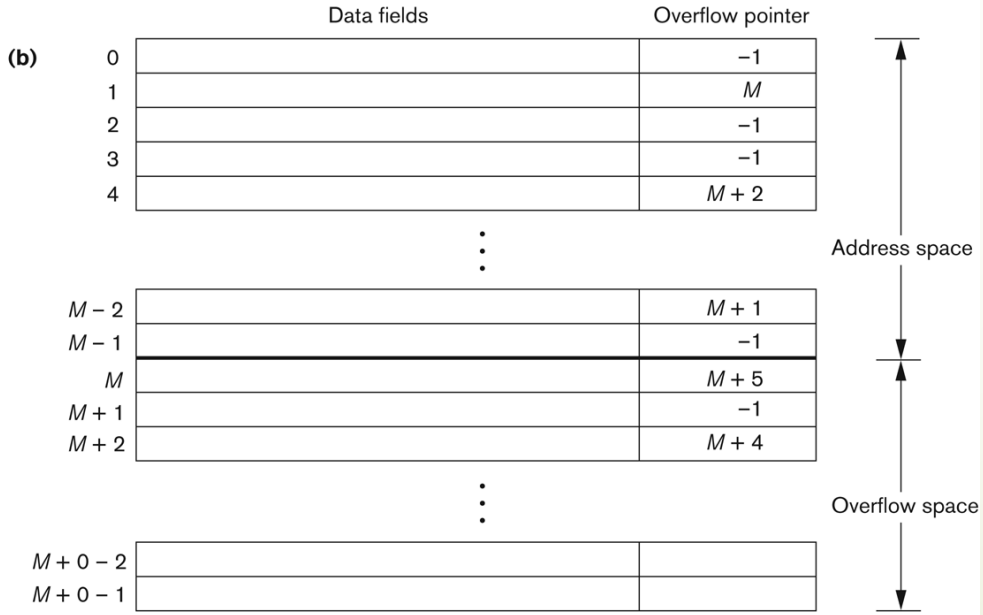
- Proporcionan un acceso muy rápido a los registros bajo una condición de igualdad sobre el **campo de dispersión** (campo hash).
  - Si el campo es clave se denomina clave de dispersión (**clave hash**)
- Un registro con campo hash de valor *K* se almacena en la posición *i*, donde **i=h(K)**, y *h* es la **función hash**.
- Tipos de dispersión:
  - Interna (en memoria)
  - Externa (en disco)
    - Estática
    - Dinámica

Dispersión interna(memoria)

- La tabla hash es un **array de registros**
  - Cada elemento del array es un registro de datos
- Resolución de colisiones:
  1. **Direccionamiento abierto**
     - A partir de la posición ocupada se recorren las siguientes posiciones hasta encontrar una vacía y se coloca.
  2. **Encadenamiento**
     - El array se extiende con posiciones de desbordamiento.
     - Se añade un puntero a cada ubicación de registro.
     - El registro se coloca en una ubicación de desbordamiento libre.
     - Se conserva una lista enlazada de registros de desbordamiento por cada dirección hash.
  3. **Dispersión múltiple**
     - Se aplica una segunda función (llamada **rehash**).
     - Si se produce otra colisión, se utiliza direccionamiento abierto u otra tercera función, y después direccionamiento abierto, si fuese necesario.

Dispersión interna(memoria)

Figure 13.8
Internal hashing data structures. (a) Array of *M* positions for use in internal hashing. (b) Collision resolution by chaining records.



a) Array de M posiciones para hashing interno

b) Resolución de colisiones por encadenamiento

Ejercicio

Registro	K	h(K)
record1	7107	3
record2	844	4
record3	1540	4
record4	4800	0
record5	826	2
record6	7110	6
record7	1821	5
record8	2376	0
record9	2002	2
record10	4981	5
record11	962	2
record12	2084	4
record13	2306	2

3 – Supongamos un fichero CLIENTES con *Numcli* como clave de direccionamiento calculado, que contiene registros con los siguientes valores de *Numcli*.

Cargar estos registros en un array de tamaño 8 en el orden dado, empleando la función de direccionamiento calculado h(K) = K mod 8.

a) Resolver las colisiones utilizando la técnica de **encadenamiento**

ESPACIO DE DIRECCIONES

Posición	Registros	puntero
0	record 4	9
1		-1
2	record 5	10
3	record 1	-1
4	record 2	8
5	record 7	11
6	record 6	-1
7		-1

ESPACIO DE DESBORDAMIENTO

Posición	Registros	puntero
8	record 3	13
9	record 8	-1
10	record 9	12
11	record 10	-1
12	record 11	14
13	record 12	-1
14	record 13	-1

b) Resolver las colisiones por **direccionamiento abierto**

ESPACIO DE DIRECCIONES

Posición	Registros
0	record4
1	record8
2	record5
3	record1
4	record2
5	record3
6	record6
7	record7

A partir del registro 9 (inclusive) no se incluyen más registros

Dispersión externa estática

- El espacio de direcciones se compone de cubos (buckets)
  - Un **cubo** puede ser 1 bloque de disco o un grupo de bloques contiguos
- La función hash mapea el valor del campo hash a un nº de cubo
- La cabecera del fichero incluye una tabla que convierte el nº de cubo en la dirección de bloque de disco

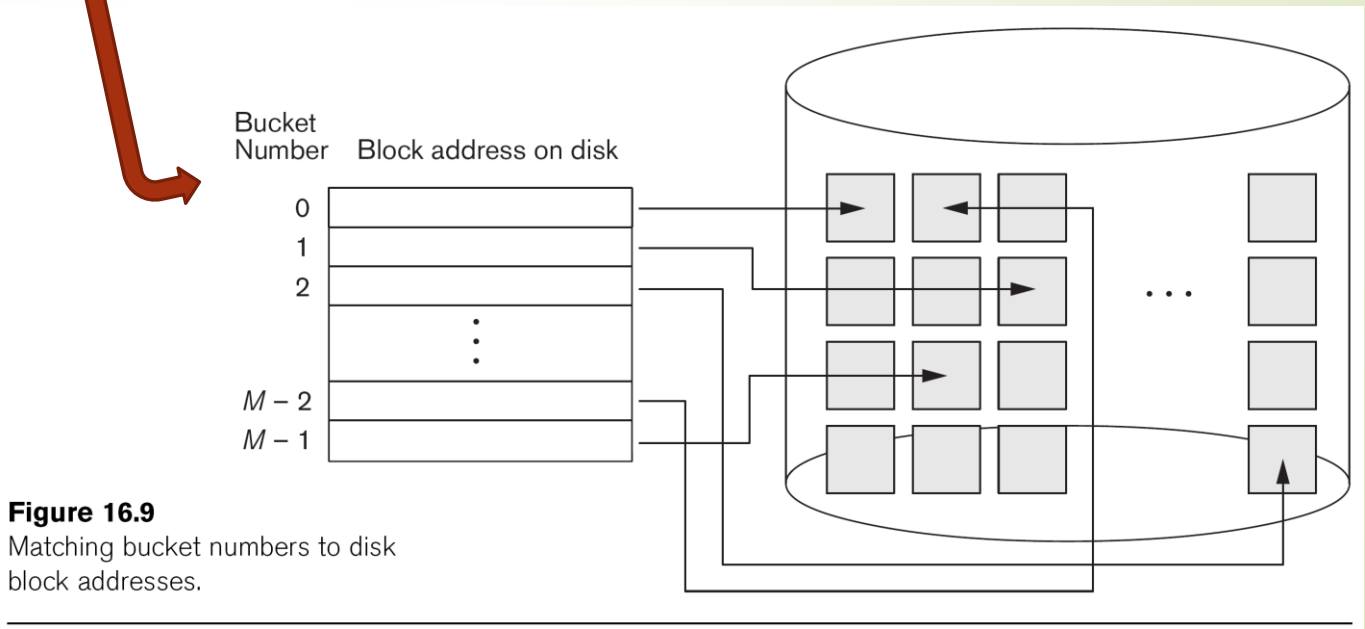


Figure 16.9
Matching bucket numbers to disk block addresses.

Dispersión externa estática

- Las **colisiones** se pueden resolver utilizando una variación del encadenamiento:
  - Cada cubo tiene un puntero a una lista enlazada de registros de desbordamiento
  - Son punteros de registro <*dirección de bloque, posición del registro dentro del bloque*>

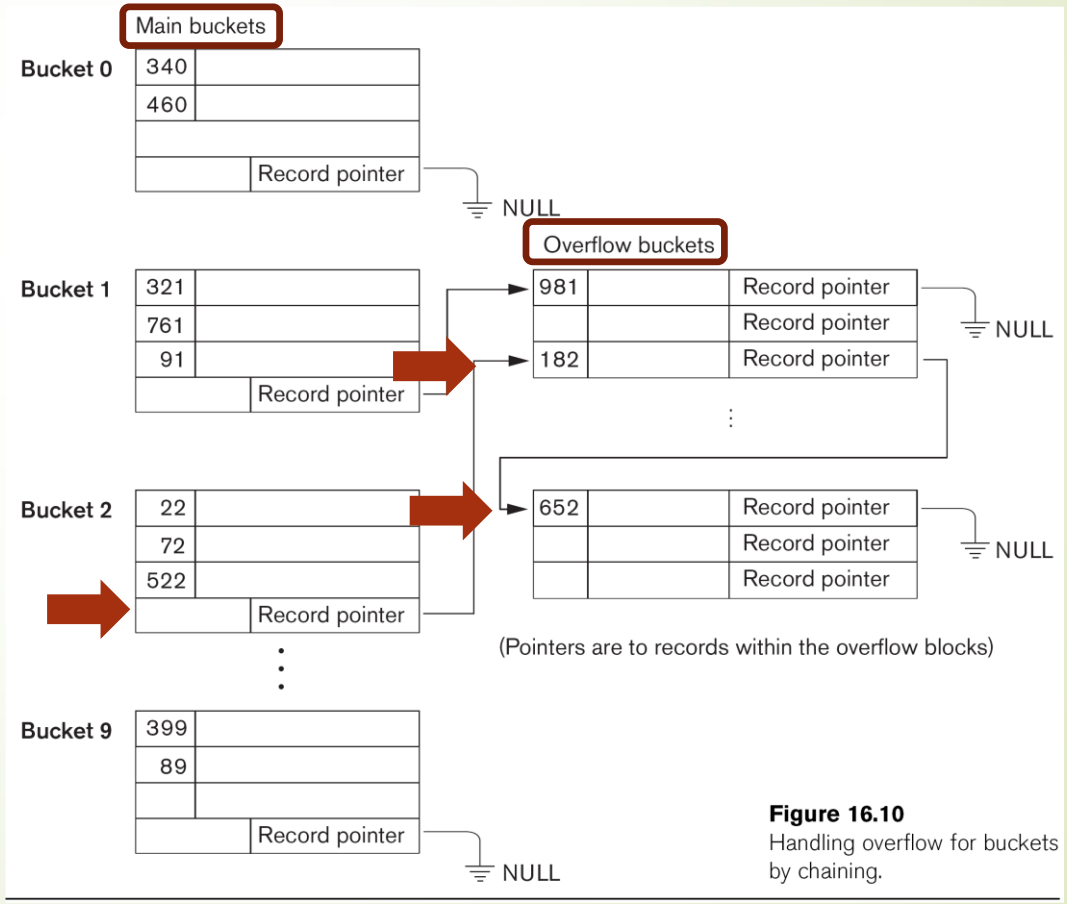
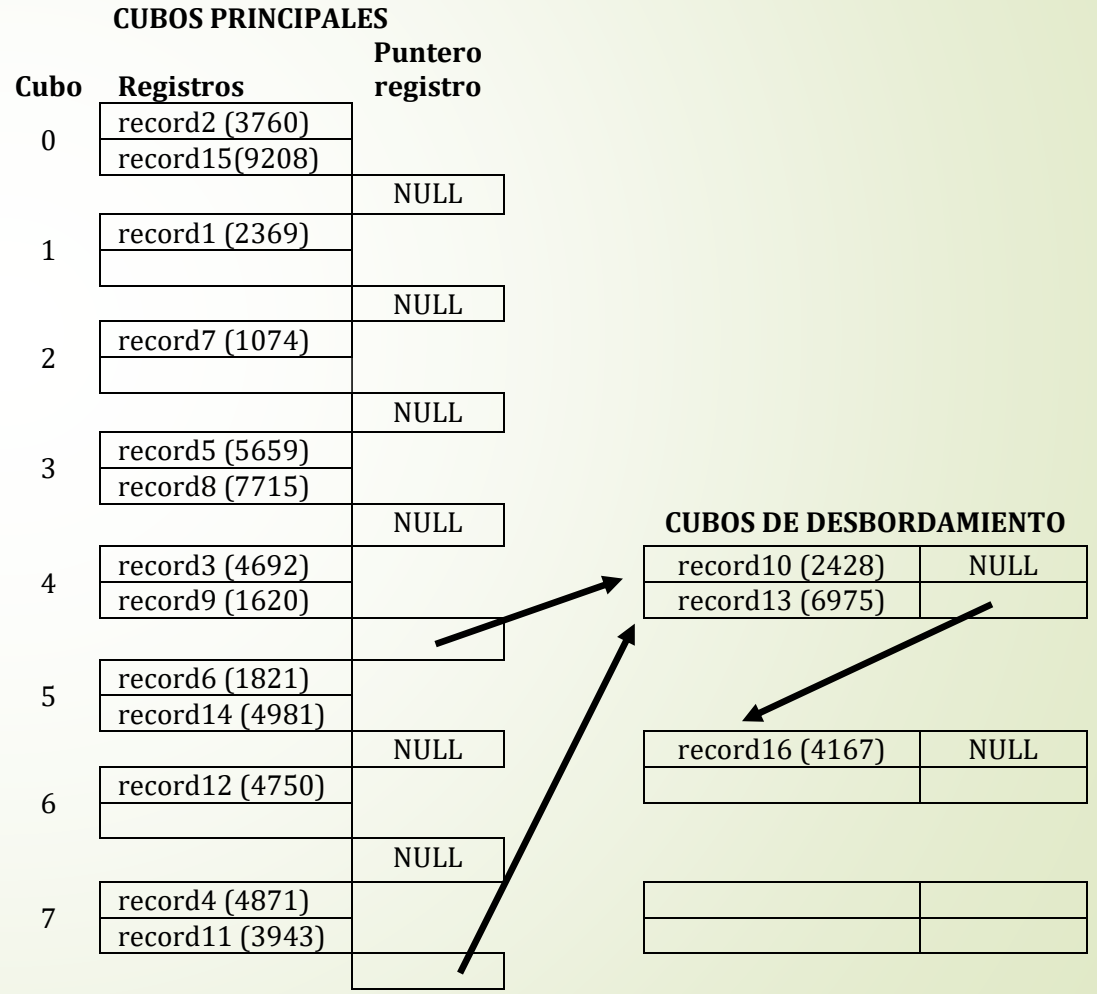


Figure 16.10
Handling overflow for buckets by chaining.

Ejercicio: Un fichero COMPONENTS con *Numcomp* como clave de direccionamiento calculado contiene registros con los siguientes valores de Numcomp. El fichero utiliza 8 cubos, numerados del 0 a 7. El tamaño de cada cubo es 1 bloque y puede contener 2 registros.
Cargar estos registros en el fichero en el orden dado, empleando la función de direccionamiento calculado h(K) = K mod 8. Para resolver las colisiones se utiliza la técnica de encadenamiento.

Registro	K	h(K)
record1	2369	1
record2	3760	0
record3	4692	4
record4	4871	7
record5	5659	3
record6	1821	5
record7	1074	2
record8	7115	3
record9	1620	4
record10	**2428**	**4**
record11	3943	7
record12	4750	6
record13	**6975**	**7**
record14	4981	5
record15	9208	0
record16	**4167**	**7**



Dispersión externa estática

- Para reducir los registros de overflow, el fichero hash habitualmente se mantiene entre un 70-80% lleno
- La función hash debe redistribuir los registros uniformemente entre los cubos
  - En caso contrario, el tiempo de búsqueda se incrementa mucho debido a la cantidad de registros de overflow existente
- **Desventajas de la dispersión externa estática**:
  - Un nº fijo de cubos es problemático si el nº de registros del fichero crece o disminuye
  - La búsqueda de un registro por un campo distinto al campo hash es tan costosa como en caso de un fichero desordenado
  - El acceso ordenado por el campo hash es ineficiente (hay que ordenar los registros)

Dispersión externa dinámica

- Permite la expansión o contracción del fichero de forma dinámica en función del nº de registros

- Tipos:
  - **Extensible**: almacena una estructura de acceso adicional (directorio)
  - **Lineal**: no requiere estructura de acceso adicional

- La estructura de acceso se fundamenta en una **representación binaria** del resultado de la función hash

Dispersión externa dinámica: 1) extensible

- Los registros se distribuyen entre cubos basándose en los **bits más significativos** del campo hash

- Emplea una estructura (directorio) adicional = array de $\mathbf{2^d}$ direcciones de cubos de registros
  - **d** = **profundidad global:** nº máx. de bits necesario para determinar el contenido de algún cubo
  - El directorio se expande o contrae dinámicamente
  - Varias entradas de directorio pueden apuntar al mismo cubo
  - Cada cubo tiene una **profundidad local d´:** nº bits en que se basa su contenido

- Los cubos no se expanden hasta que se produce **overflow:**
  - Los registros se redistribuyen entre 2 cubos, utilizando el siguiente bit (de + a - significativo)
  - El directorio se actualiza
    - si d´= d y el cubo desborda

      entonces el directorio se duplica

Dispersión externa dinámica: 1) extensible

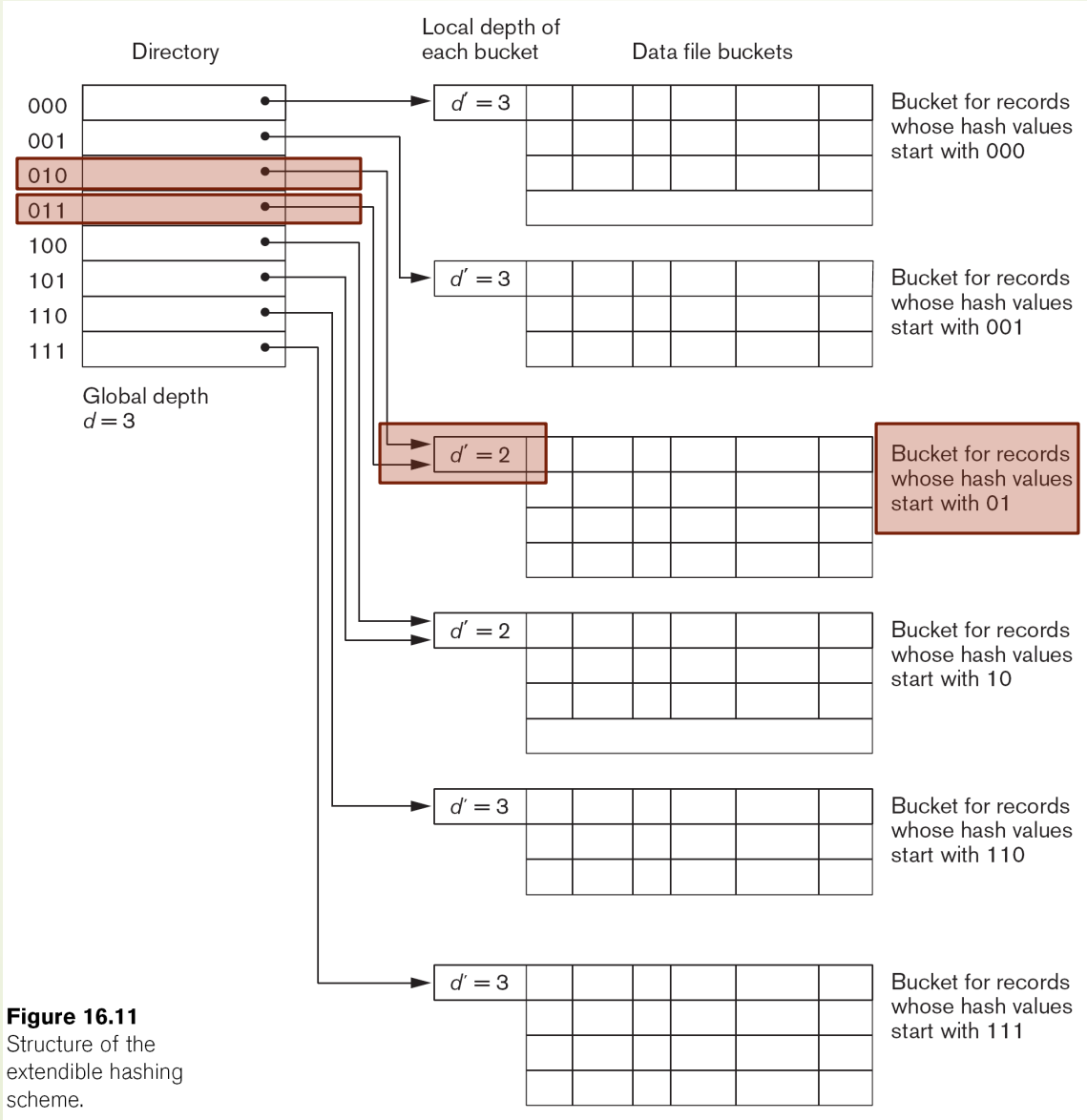


Figure 16.11
Structure of the extendible hashing scheme.

Ejercicio (2 registros por cubo):

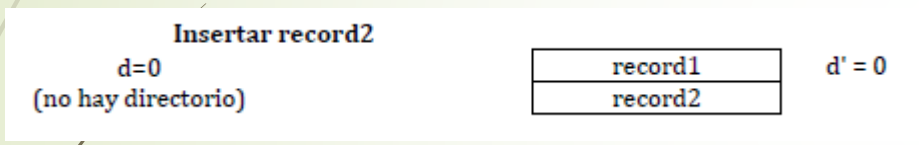
record 1: h(record1) = **0** 0 0 1

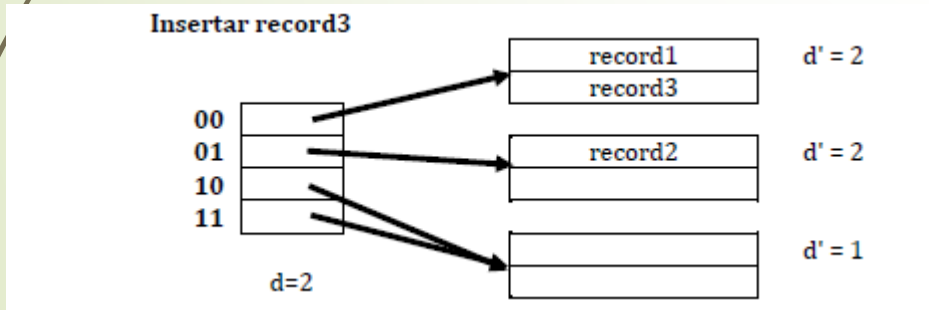
record 2: h(record2) = **0** 1 0 0

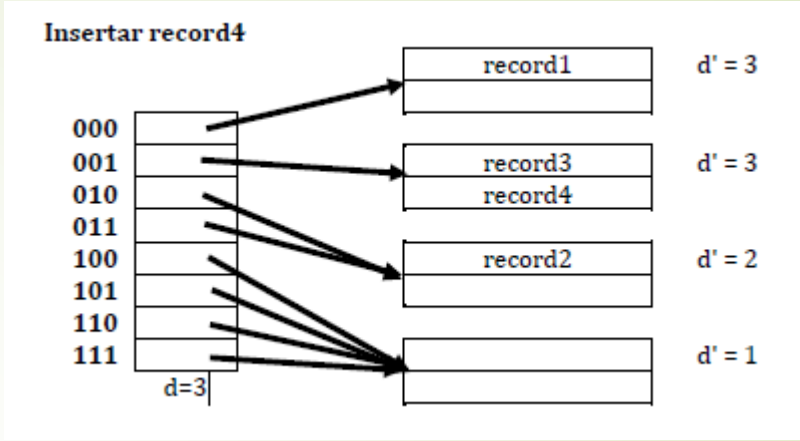
record 3: h(record3) = **0 0** 1 0

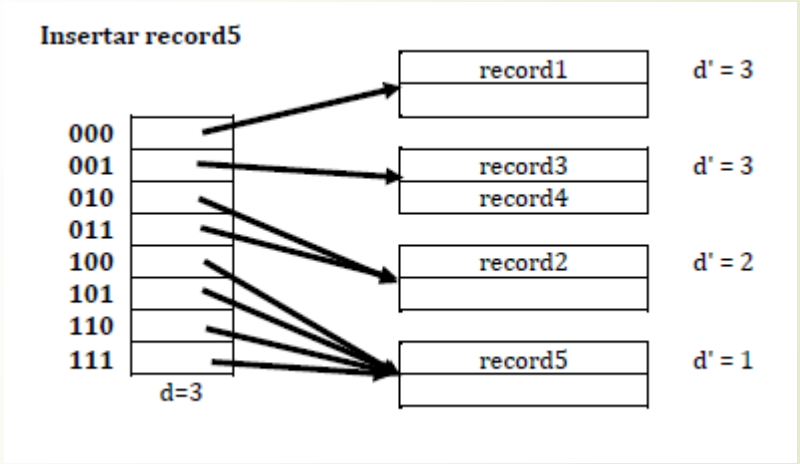
record 4: h(record4) = **0 0 1** 1

record 5: h(record5) = **1** 1 1 0









Dispersión externa dinámica: 1) extensible

- Ventajas:
  - El rendimiento del fichero no se degrada al crecer
  - No se asigna espacio para un crecimiento futuro
    - Se asignan cubos adicionales a medida que se necesitan
  - La sobrecarga de espacio para el directorio es pequeña
  - La división de un cubo provoca una reorganización pequeña
    - Solo se redistribuyen en 2 cubos los registros del cubo desbordado
- Inconvenientes:
  - Siempre hay que hacer 2 accesos (directorio + cubo)

Preguntas sobre hashing extensible

- ¿Por qué son necesarias las profundidades local y global?
  - Si d' = d ⇒ el bucket está siendo apuntado por 1 E del directorio
    - El directorio debe doblarse para poder tener 1 E que apunte al nuevo bucket
  - Si d' < d ⇒ el bucket está siendo apuntado por 2 o + entradas del directorio
    - Crear 1 nuevo bucket, redistribuir los registros y recolocar los punteros
- Si una inserción provoca que el tamaño del directorio se doble, ¿cuántos buckets tendrán exactamente 1 E de directorio apuntando a ellas?
  - 2 (el bucket donde se quería realizar la inserción, y el nuevo bucket)
- Si se borra 1 E de uno de esos buckets, ¿qué le sucede al tamaño del directorio?
  - Depende del algoritmo de borrado
    - si los buckets se combinan siempre que es posible ⇒ el tamaño del directorio se decrementa

Dispersión externa dinámica: 2) lineal

- NO existe directorio, pero sí un área de overflow

- Utiliza 2 funciones hash:
  1. **Función h_i de inserción** de nuevos registros (p.ej: $h_0 = K \ mod \ 2^0$)
  2. **Función h_{i+1} de expansión** para resolver las colisiones (p.ej: $h_1 = K \ mod \ 2^1$)

- Cuando se produce la **1ª colisión** en cualquier cubo del fichero, se divide el **PRIMER** cubo en 2:
  1. El cubo 0 original
  2. Un nuevo cubo M al final del fichero, siendo M el nº de cubos del fichero

  - Los registros del cubo 0 se distribuyen entre los 2 cubos (0 y M) mediante la función de expansion h_{i+1}

Dispersión externa dinámica: 2) lineal

- Las funciones $\boldsymbol{h_i}$ y $\boldsymbol{h_{i+1}}$ deben garantizar que cualquier registro que se ubique en el cubo 0 basándose en $\boldsymbol{h_i}$ se dispersará al cubo 0 o al cubo M basándose en $\boldsymbol{h_{i+1}}$.
- Viene garantizado por la aritmética modular:

Ej: $h_0 = k \; mod \; 2^0 = k \; mod \; 1 = 0$ en este caso, todos los registros se asignan al cubo 0

$h_1 = k \; mod \; 2^1 = k \; mod \; 2 = \{0, 1\} = \{XXX\mathbf{0}, XXX\mathbf{1}\}$ usa el bit menos significativo, los registros pares irán al cubo 0 y los impares al cubo 1

$h_2 = k \; mod \; 2^2 = k \; mod \; 4 = \{0, 1, 2, 3\} = \{XX\mathbf{00}, XX\mathbf{01}, XX\mathbf{10}, XX\mathbf{11}\}$ usa los 2 bits menos significativos, los registros pares irán a los cubos 00 y 10, y los impares a los cubos 01 y 11

$h_3 = k \; mod \; 2^3 = k \; mod \; 8 = \{0, 1, 2, 3, 4, 5, 6, 7\} = \{X\mathbf{000}, X\mathbf{001}, X\mathbf{010}, X\mathbf{011}, X\mathbf{100}, X\mathbf{101}, X\mathbf{110}, X\mathbf{111}\}$ en este caso, se usan los 3 bits menos significativos, …

Dispersión externa dinámica: 2) lineal

- Cuando se produce la **1ª colisión** en cualquier cubo del fichero, se divide el **PRIMER** cubo en 2:
  1. El cubo 0 original
  2. Un nuevo cubo M al final del fichero, siendo M el nº de cubos del fichero
  - Los registros del cubo 0 se distribuyen entre los 2 cubos (0 y M) mediante la función de expansion $\boldsymbol{h_{i+1}}$

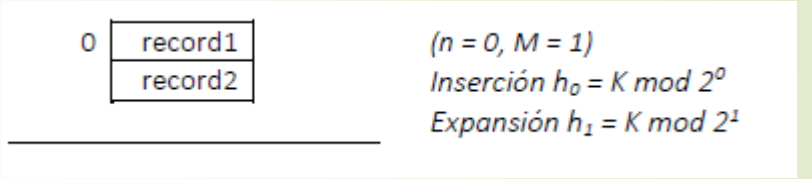
- Cuando se produce la **2ª colisión** en cualquier cubo del fichero, se divide el **SEGUNDO** cubo, y así sucesivamente
  - Se utiliza una variable *n* para apuntar al siguiente bloque a expandir
  - Los registros del cubo 1 se distribuyen entre los 2 cubos (1 y M+1) mediante la función de expansion $\boldsymbol{h_{i+1}}$

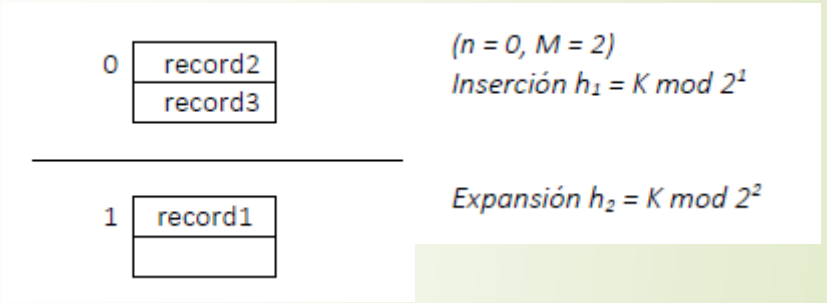
- Cuando se expanda el último cubo (cubo M-1), la función $\boldsymbol{h_{i+1}}$ se habrá aplicado a todos los registros del fichero. A partir de este momento:
  1. La función de expansión $\boldsymbol{h_{i+1}}$ pasa a ser la **función de inserción** de registros (p.ej. $h_1 = K \; mod \; 2^1$)
  2. Se emplea una nueva **función $\boldsymbol{h_{i+2}}$ de expansión** para resolver las colisiones (p.ej. $h_2 = K \; mod \; 2^2$)

Dados los registros con los siguientes valores de clave hash: **2369, 3760, 4692**, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981 y 9208.
Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el nº de registros por bloque es 2. Empezar con 1 solo bloque.
Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.

Inicialmente:

- Tamaño del fichero: **M = 1** bloque
- Siguiente bloque a expandir: **n = 0**
- Función de inserción: h_0 = K mod 2^0
- Función de expansión: h_1 = K mod 2^1



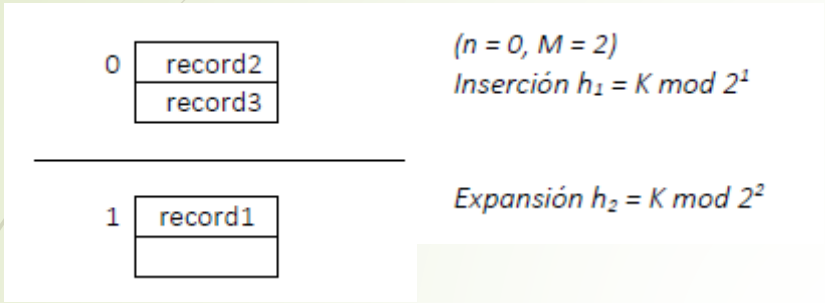


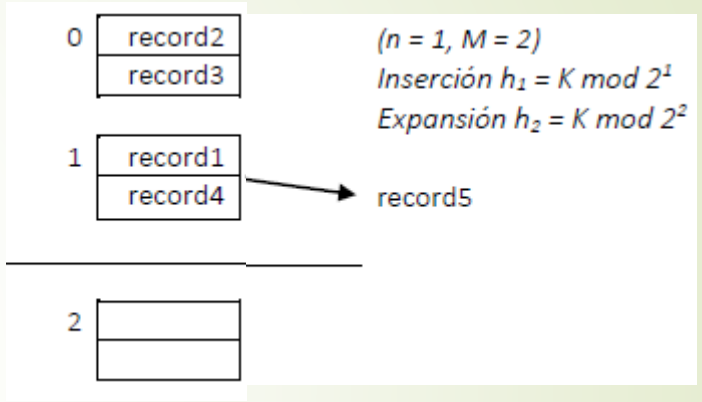
h_0 (record1) = 2369 mod 2^0 = 0
h_0 (record2) = 3760 mod 2^0 = 0
h_0 (record3) = 4692 mod 2^0 = **0** OVERFLOW ⇒ desdoblar el cubo señalado por n (es decir, el cubo 0) mediante **h_1(K)** creando el cubo 1 ⇒ n++ ⇒ **n=1**

h_1 (record1) = 2369 mod 2^1 = 1
h_1 (record2) = 3760 mod 2^1 = 0
h_1 (record3) = 4692 mod 2^1 = 0
Como n = M ⇒ Se ha aplicado h_1 a todo el fichero (formado por 1 único bloque, en este caso) ⇒
EXPANSIÓN COMPLETA ⇒

- Nuevo tamaño del fichero: M = 2 bloques
- Siguiente bloque a expandir: n= 0
- Nueva función de inserción: h_1 = K mod 2^1
- Nueva función de expansión: h_2 = K mod 2^2

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, **4871, 5659**, 1821, 1074, 7115, 1620 , 2428, 3943, 4750, 6975, 4981 y 9208.
Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el nº de registros por bloque es 2. Empezar con 1 solo bloque.
Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.



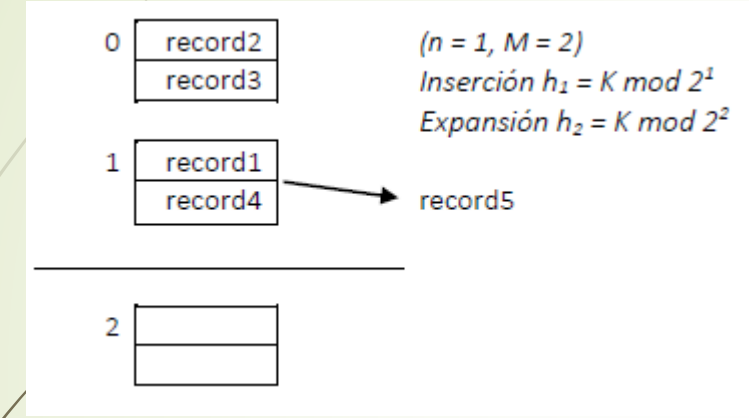


h_1 (record4) = 4871 mod 2^1 = 1
h_1 (record5) = 5659 mod 2^1 = **1** OVERFLOW ⇒ desdoblar el cubo 0 mediante $\mathbf{h_2(K)}$
creando el cubo2 ⇒ n++ ⇒ **n=1**
h_2 (record2) = 3760 mod 2^2 = 0
h_2 (record3) = 4692 mod 2^2 = 0

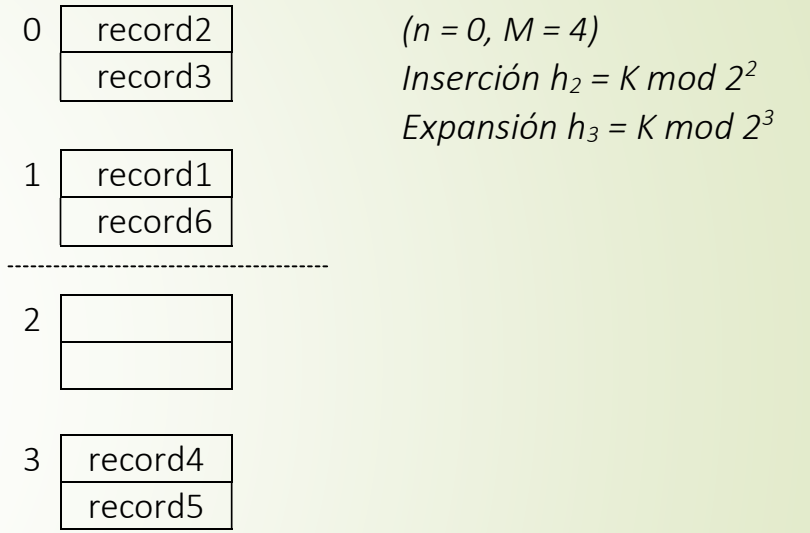
Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, **1821,** 1074, 7115, 1620 , 2428, 3943, 4750, 6975, 4981 y 9208.

Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el nº de registros por bloque es 2. Empezar con 1 solo bloque.

Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.



INSERCIÓN RECORD6



h_1 (record6) = 1821 mod 2^1 = **1** OVERFLOW ⇒ desdoblar el cubo 1 mediante **h_2(K)** creando el cubo 3 ⇒ n++ ⇒ **n=2**

h_2 (record1) = 2369 mod 2^2 = 1

h_2 (record4) = 4871 mod 2^2 = 3

h_2 (record5) = 5659 mod 2^2 = 3

h_2 (record6) = 1821 mod 2^2 = 1

Como n = M ⇒ Se ha aplicado h_2 a todo el fichero (formado por 2 bloques) ⇒ **EXPANSIÓN COMPLETA** ⇒

- Nuevo tamaño del fichero: M = 4 bloques
- Siguiente bloque a expandir es n= 0
- Nueva función de inserción: h_2 = K mod 2^2
- Nueva función de expansión: h_3 = K mod 2^3

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, **1074, 7115**, 1620 , 2428, 3943, 4750, 6975, 4981 y 9208.
Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque.
Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.

0	record2
	record3
1	record1
	record6
2	
3	record4
	record5

(n = 0, M = 4)
Inserción $h_2 = K \ mod \ 2^2$
Expansión $h_3 = K \ mod \ 2^3$

INSERCIÓN RECORD7 Y RECORD8

0	record2
1	record1
	record6
2	record7
3	record4
	record5
4	record3

record8

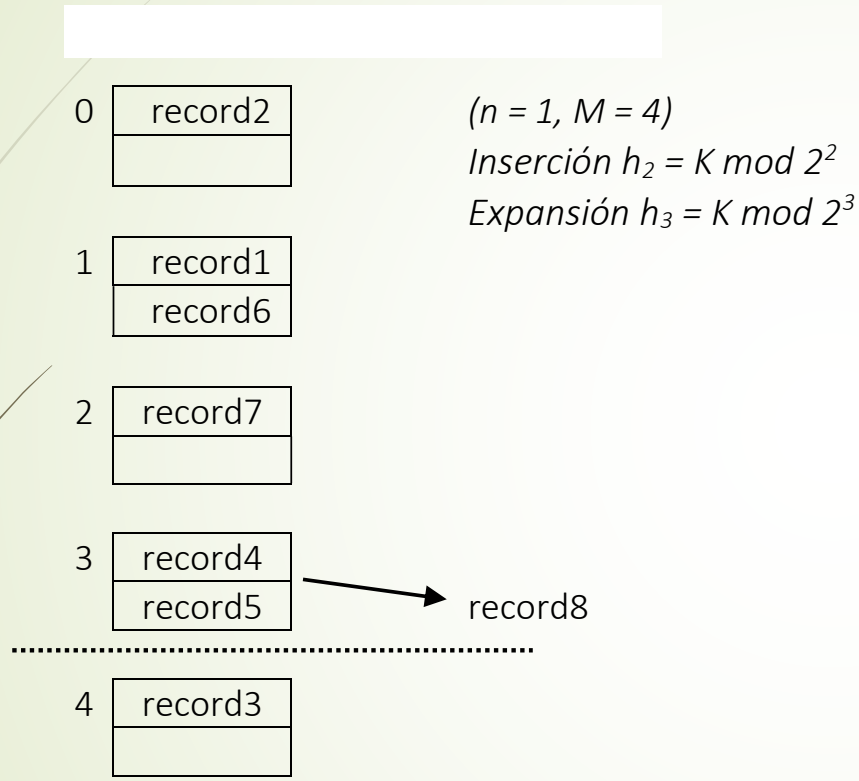
(n = 1, M = 4)
Inserción $h_2 = K \ mod \ 2^2$
Expansión $h_3 = K \ mod \ 2^3$

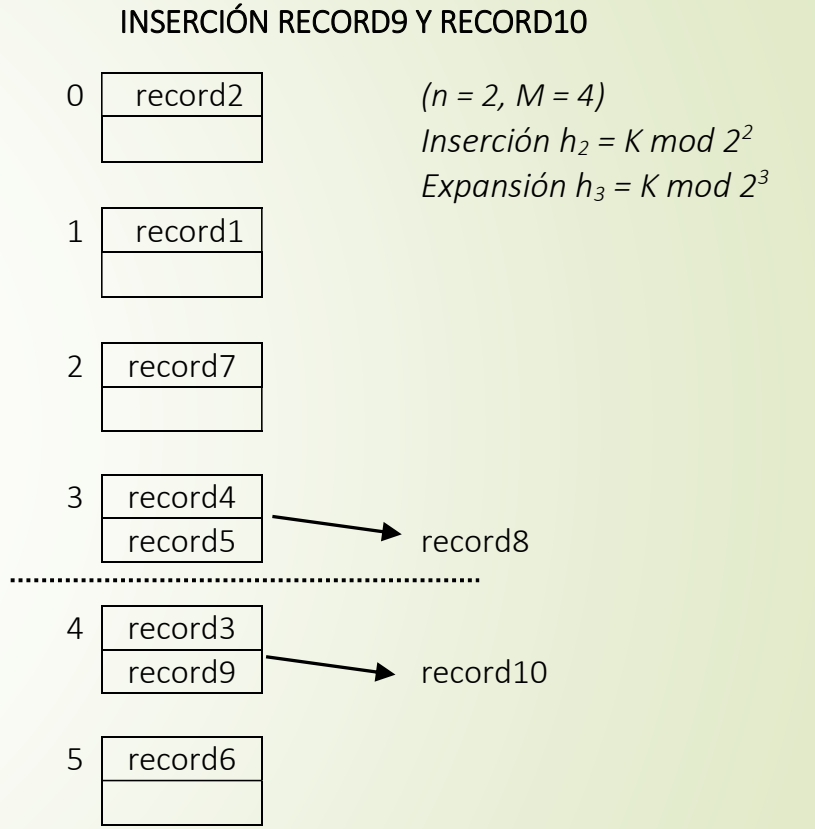
h_2 (record7) = 1074 mod 2^2 = 2
h_2 (record8) = 7115 mod 2^2 = **3** OVERFLOW ⇒ desdoblar el cubo 0 mediante $\mathbf{h_3(K)}$ creando el cubo4 ⇒ n++ ⇒ **n=1**
h_3 (record2) = 3760 mod 2^3 = 0
h_3 (record3) = 4692 mod 2^3 = 4

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, **1620**, **2428**, 3943, 4750, 6975, 4981 y 9208.

Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque.

Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.





h_2 (record9) = 1620 mod 2^2 = 0 ⇒ **como 0 < n=1 ⇒ el cubo 0 ya está desdoblado ⇒ debe aplicarse la función h_3(K)**

h_3 (record9) = 1620 mod 2^3 = 4

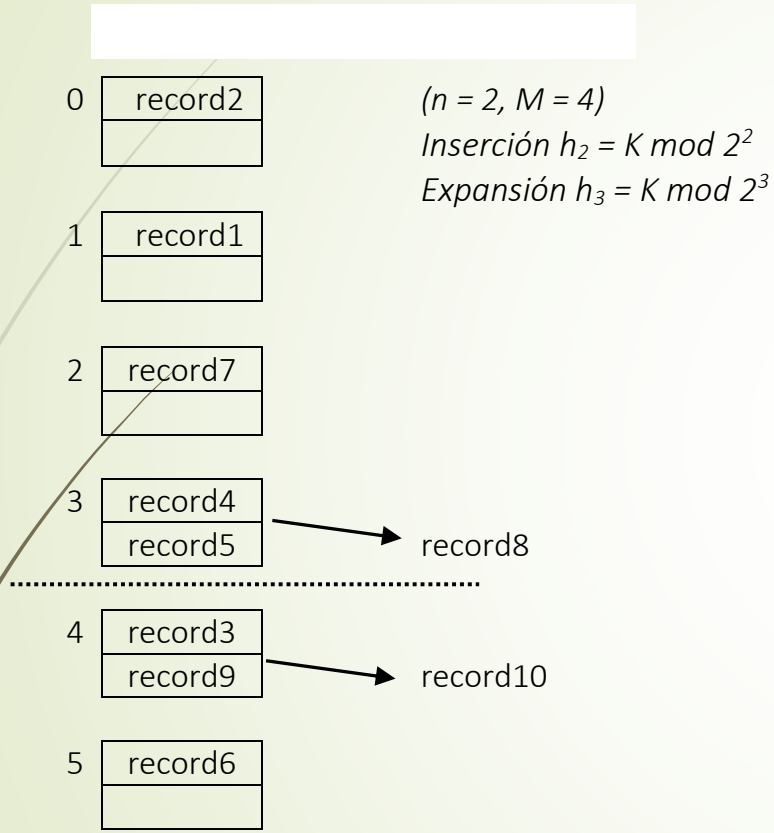
h_2 (record10) = 2428 mod 2^2 = 0 ⇒**como 0 < n=1 ⇒ el cubo 0 ya está desdoblado ⇒ debe aplicarse la función h_3(K)**

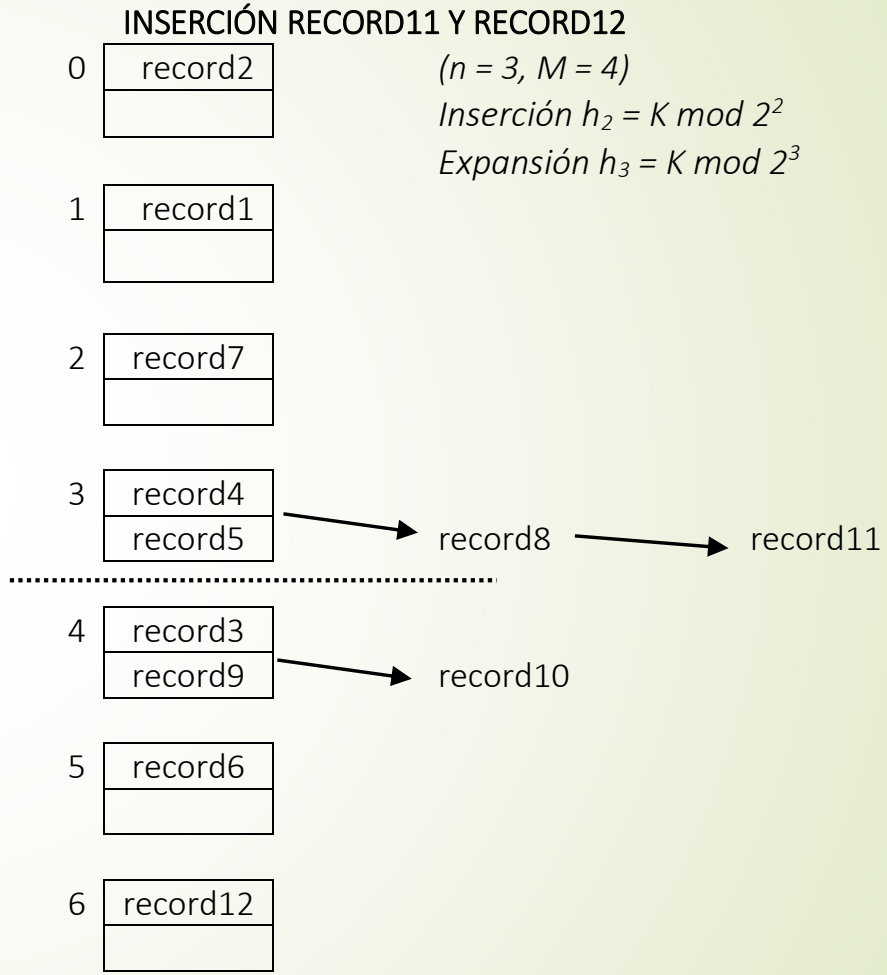
h_3 (record10) = 2428 mod 2^3 = **4** OVERFLOW ⇒ dividir el cubo 1 mediante **h_3(K)** creando el cubo 5 ⇒ n++ ⇒ **n=2**

h_3 (record1) = 2369 mod 2^3 = 1

h_3 (record6) = 1821 mod 2^3 = 5

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, **3943**, **4750**, 6975, 4981 y 9208. Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque. Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.





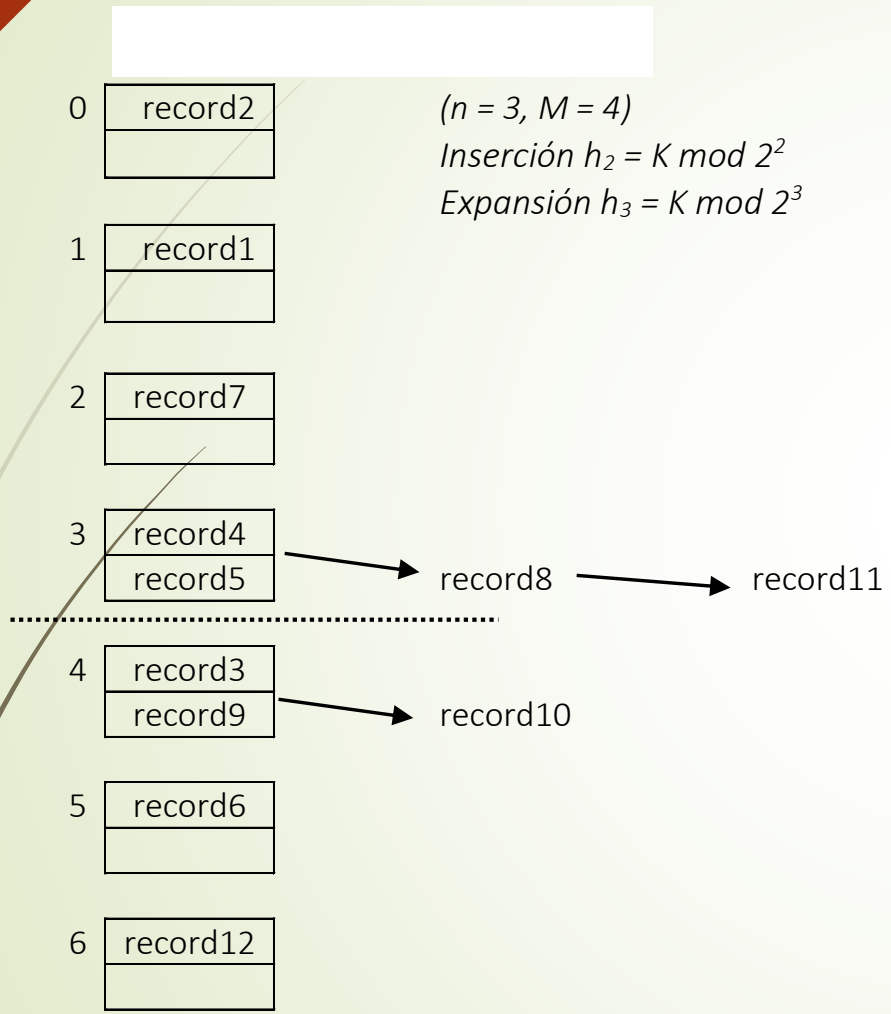
h_2 (record11) = 3943 mod 2^2 = **3** OVERFLOW ⇒ desdoblar el cubo 2 mediante **h_3(K)** creando el cubo 6 ⇒ n++ ⇒ **n=3**

h_3 (record7) = 1074 mod 2^3 = 2

h_2 (record12) = 4750 mod 2^2 = 2 ⇒ **como 2 < n=3 ⇒ el cubo 2 ya está desdoblado ⇒ debe aplicarse la función h_3(K)**

h_3 (record12) = 4750 mod 2^3 = 6

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, **6975**, 4981 y 9208. Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque. Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.



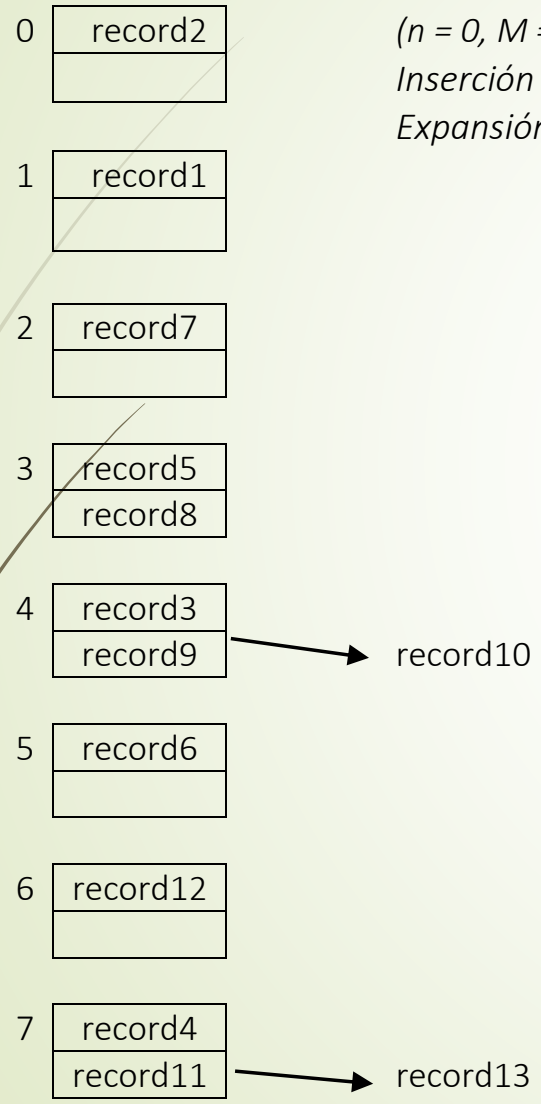
(n = 3, M = 4)
Inserción $h_2 = K \ mod \ 2^2$
Expansión $h_3 = K \ mod \ 2^3$

h_2 (record13) = 6975 mod 2^2 = 3 OVERFLOW ⇒ desdoblar el cubo 3 mediante **h_3(K)** creando el cubo 7 ⇒ n++ ⇒ **n=4**

h_3 (record4) = 4871 mod 2^3 = 7
h_3 (record5) = 5659 mod 2^3 = 3
h_3 (record8) = 7115 mod 2^3 = 3
h_3 (record11) = 3943 mod 2^3 = 7
h_3 (record13) = 6975 mod 2^3 = 7 OVERFLOW (en este caso **NO se desdobla porque se encuentra en el propio proceso de expansión**)

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, **6975**, 4981 y 9208. Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque. Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.

INSERCIÓN RECORD13



(n = 0, M = 8)
Inserción $h_3 = K \ mod \ 2^3$
Expansión $h_4 = K \ mod \ 2^4$

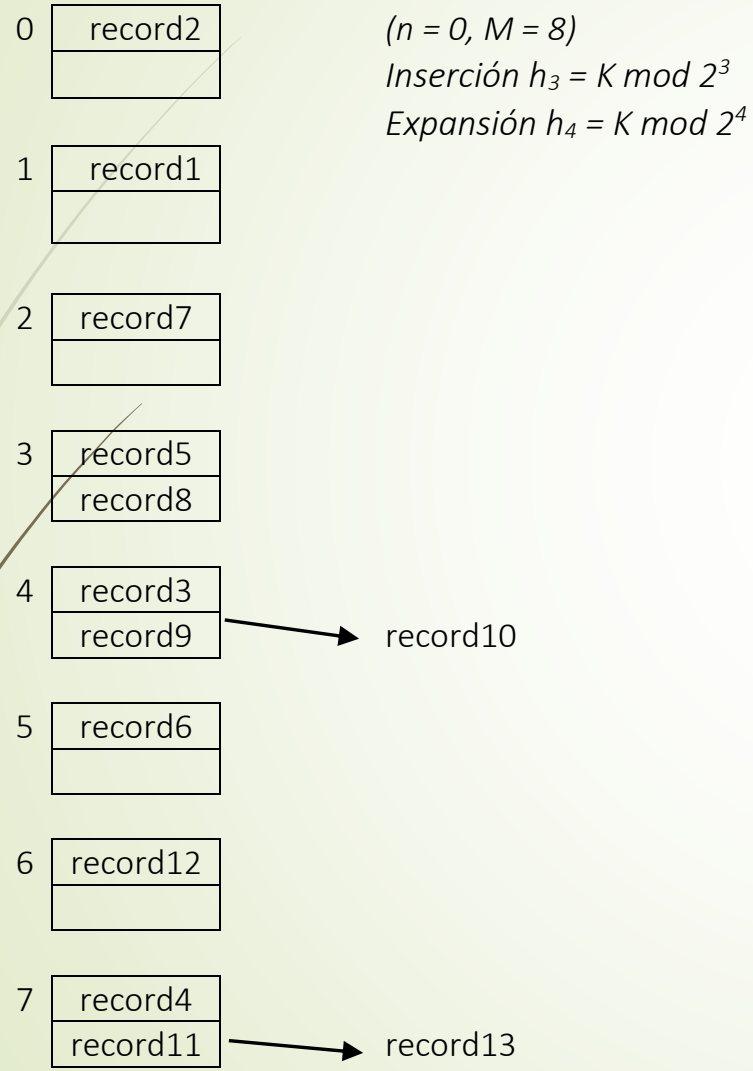
h_2 (record13) = 6975 mod 2^2 = 3 OVERFLOW ⇒ desdoblar el cubo 3 mediante **h_3(K)** creando el cubo 7 ⇒ n++ ⇒ **n=4**

h_3 (record4) = 4871 mod 2^3 = 7
h_3 (record5) = 5659 mod 2^3 = 3
h_3 (record8) = 7115 mod 2^3 = 3
h_3 (record11) = 3943 mod 2^3 = 7
h_3 (record13) = 6975 mod 2^3 = 7 OVERFLOW (en este caso **NO se desdobla porque se encuentra en el propio proceso de expansión**)

Como n = M ⇒ Se ha aplicado h_3 a todo el fichero (formado por 4 bloques) ⇒ **EXPANSIÓN COMPLETA** ⇒

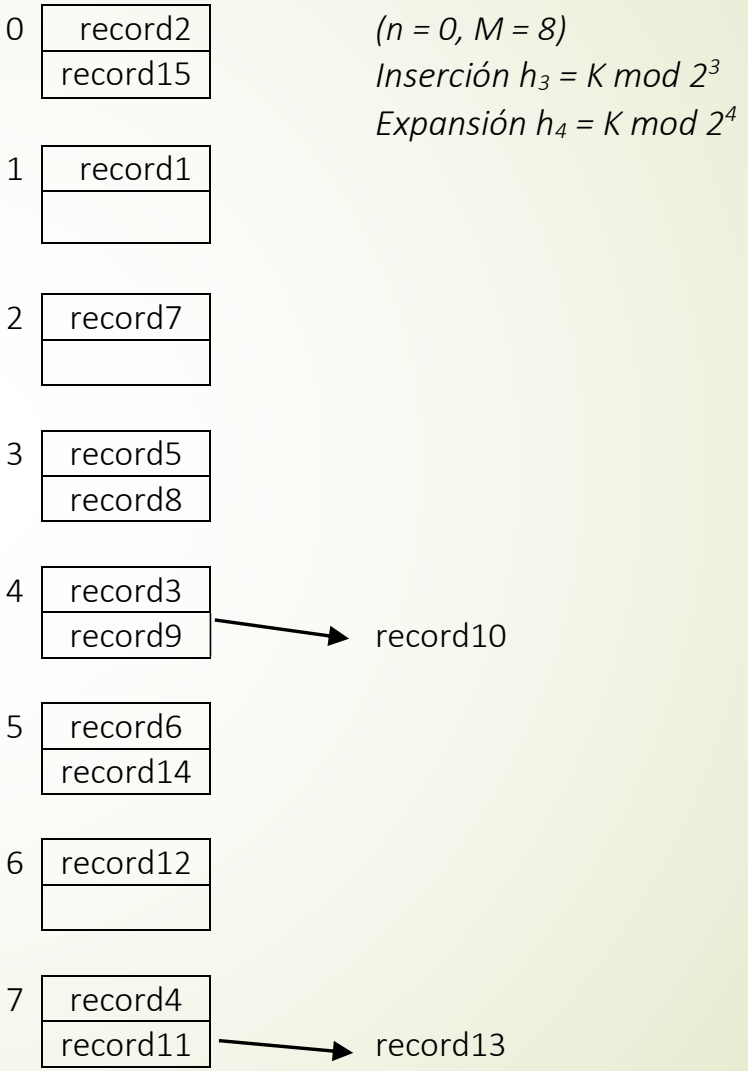
Nuevo tamaño del fichero M = 8 bloques
Siguiente bloque a expandir es n= 0
Nueva función de inserción h_3 = K mod 2^3
Nueva función de expansión h_4 = K mod 2^4

Dados los registros con los siguientes valores de clave hash: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, **4981** y **9208**. Cargarlos en un fichero con direccionamiento calculado lineal, mediante la función hash $h_i = k \ mod \ 2^i$. Los buckets están formados por 1 único bloque, y el factor de bloqueo (nº de registros por bloque) es 2. Empezar con 1 solo bloque. Mostrar cómo crece el fichero y cómo cambian las funciones hash. Los bloques se dividen cuando se produce un desbordamiento.



(n = 0, M = 8)
Inserción h_3 = K mod 2^3
Expansión h_4 = K mod 2^4

INSERCIÓN RECORD14 Y RECORD15



(n = 0, M = 8)
Inserción h_3 = K mod 2^3
Expansión h_4 = K mod 2^4

h_3 (record14) = 4981 mod 2^3 = 5
h_3 (record15) = 9208 mod 2^3 = 0

Dispersión externa dinámica: 2) lineal

- Algoritmo para realizar búsquedas

if ***n = 0***

then $v = h_j(K)$ */* aplicar la función de inserción */*

else {

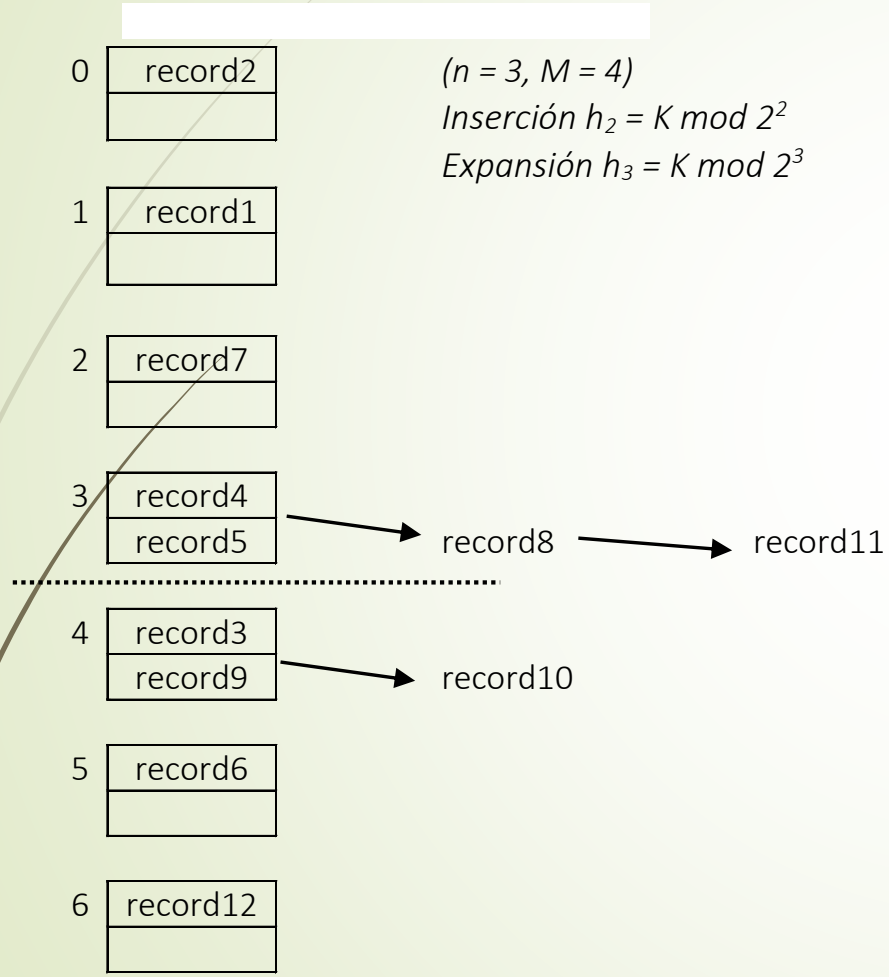
$v = h_j(K)$ */* aplicar la función de inserción */*

if ***v < n*** *then* $v = h_{j+1}(K)$ */* aplicar la función de expansión */*

}

buscar en el cubo con valor de dispersión v (y su desbordamiento, si lo hay)

Dispersión externa dinámica: 2) lineal



(n = 3, M = 4)

Inserción $h_2 = K \bmod 2^2$

Expansión $h_3 = K \bmod 2^3$

- Ejemplo de búsquedas

*Búsqueda de **record8** (7715)*

$n <> 0 \Rightarrow v = h2(record8) = 7115 \bmod 2^2 = 3$

3 no es menor que n $\Rightarrow$ buscar en el cubo 3 (y su desbordamiento, si lo hay)

*Búsqueda de **record12** (4750)*

$n <> 0 \Rightarrow v = h2\ (record12) = 4750 \bmod 2^2 = 2$

2 es menor que n $\Rightarrow$ $v = h3(record12) = 4750 \bmod 2^3 = 6$

buscar en cubo 6 (y su desbordamiento, si lo hay)

*Búsqueda de **record13** (7715)*

$n <> 0 \Rightarrow v = h_2\ (record13) = 6975 \bmod 2^2 = 3$

3 no es menor que n $\Rightarrow$ buscar en el cubo 3 (y su desbordamiento, si lo hay)

Dispersión externa dinámica: 2) lineal

- Alternativa: División en base al **factor de carga del fichero**
  - Factor de carga **l** = r / (bfr * N)
    - r: nº actual de registros del fichero
    - bfr: factor de bloqueo (nº máximo de registros que pueden encajar en un cubo)
    - N: nº actual de cubos (incluidos los de expansión)

    *Ejemplo: l = 10 reg /(5 reg. por cubo * 4 cubos) = 50 % de carga*

  - Inserción de registros:
    - Las divisiones se llevan a cabo cuando ***l*** alcance un umbral (p.ej, 0.9)

    Ejemplo: *Cuando N=1, l= r/(bfr * N) ≥ 0,9 ⇒ r ≥ 1,8 ⇒ desdoblar tras incluir el registro 2*

    *Cuando N=2, l= r/(bfr * N) ≥ 0,9 ⇒ r ≥ 3,6 ⇒ desdoblar tras incluir el registro 4*

    *Cuando N=3, l= r/(bfr * N) ≥ 0,9 ⇒ r ≥ 5,4 ⇒ desdoblar tras incluir el registro 6*

    *Cuando N=4, l= r/(bfr * N) ≥ 0,9 ⇒ r ≥ 7,2 ⇒ desdoblar tras incluir el registro 8*

    …

Dispersión externa dinámica: 2) lineal

- Alternativa: División en base al **factor de carga del fichero**
  - Factor de carga **l** = r / (bfr * N)
    - r: nº actual de registros del fichero
    - bfr: factor de bloqueo (nº máximo de registros que pueden encajar en un cubo)
    - N: nº actual de cubos (incluidos los de expansión)
    - P.ej: l = 10 reg /(5 reg. por cubo * 4 cubos actuales) = 50 % de carga
  - Inserción de registros:
    - Las divisiones se llevan a cabo cuando ***l*** alcance un umbral (p.ej, 0.9)
  - Eliminación de registros:
    - Las combinaciones se llevan a cabo cuando ***l*** cae por debajo de un umbral (p. ej, 0.7)