TEMA 2. ORGANIZACIONES DE FICHEROS Y ESTRUCTURAS DE ACCESO

1. Introducción
2. Conceptos fundamentales de organizaciones de ficheros
3. Dispositivos de almacenamiento secundario
4. Ficheros desordenados
5. Ficheros ordenados
6. Ficheros dispersos
7. Agrupamiento
8. Índices

1. Introducción

t. acceso a DISCO ≈ 30 mseg. **t. acceso a RAM ≈ 120 nseg.**

Características del disco como medio de almacenamiento:

- Lento.
- Gran capacidad a bajo coste.
- No volátil.

Fundamental: buen diseño de la estructura de los ficheros para ...

- tener acceso a toda la capacidad del disco ...
- ... sin que las aplicaciones tengan que esperar mucho tiempo por los datos.

Objetivos en el diseño de estructuras de ficheros

- Obtener la información con un solo acceso a disco.
- Si no se puede en un acceso, hacen falta estructuras que permitan encontrar la información con el mínimo número de accesos posible.
- Además la estructura del fichero debe permitir agrupar la información: que se pueda obtener **todos los datos** que se necesitan en un solo acceso.

Todo esto no es difícil si los ficheros nunca cambian, pero con ficheros que crecen y disminuyen en tamaño a medida que la información se añade o se borra, es **muy complejo**.

Un poco de historia sobre estructuras de ficheros

- Ficheros en cinta → Acceso secuencial.
- Discos → Se empiezan a utilizar los **índices**.
- Índices demasiado grandes y que cambian mucho → Difíciles de gestionar → Estructuras en forma de **árbol**.
- Los árboles crecen de modo muy **desigual** → Muchos accesos a disco para obtener la información.
- **Árboles AVL** (1963) → Muy buenos para **RAM**, pero muy malos para ficheros.
- **Árboles B** (1972) → Muy **buenas prestaciones**.
- **Árboles B+** → Prestaciones como el árbol B, pero **también** se permite el acceso secuencial.
- Dispersión → Acceso **muy rápido** a los datos.

2. Conceptos fundamentales de organizaciones de ficheros

Fichero de PLANTILLA

Enum	Apellido	Puesto	DNI	Onum
EL21	Pastor	Director	39432212E	O5
EG37	Cubedo	Supervisor	38766623X	O3
EG14	Collado	Administrativo	24391223L	O3
EA9	Renau	Supervisor	39233190F	O7
EG5	Prats	Director	25644309X	O3
EL41	Baeza	Supervisor	39552133T	O5

campo

valor del campo **Onum** en un registro

Enum	Apellido	Puesto	DNI	Onum
EL21	Pastor	Director	39432212E	O5
EG37	Cubedo	Supervisor	38766623X	O3
EG14	Collado	Administrativo	24391223L	O3
EA9	Renau	Supervisor	39233190F	O7
EG5	Prats	Director	25644309X	O3
EL41	Baeza	Supervisor	39552133T	O5

1
2
bloques / páginas

Estructuras (organizaciones) de ficheros

El **orden** en que se colocan los registros en un fichero depende de su ***estructura***.

Los principales ***tipos de estructuras*** son los siguientes:

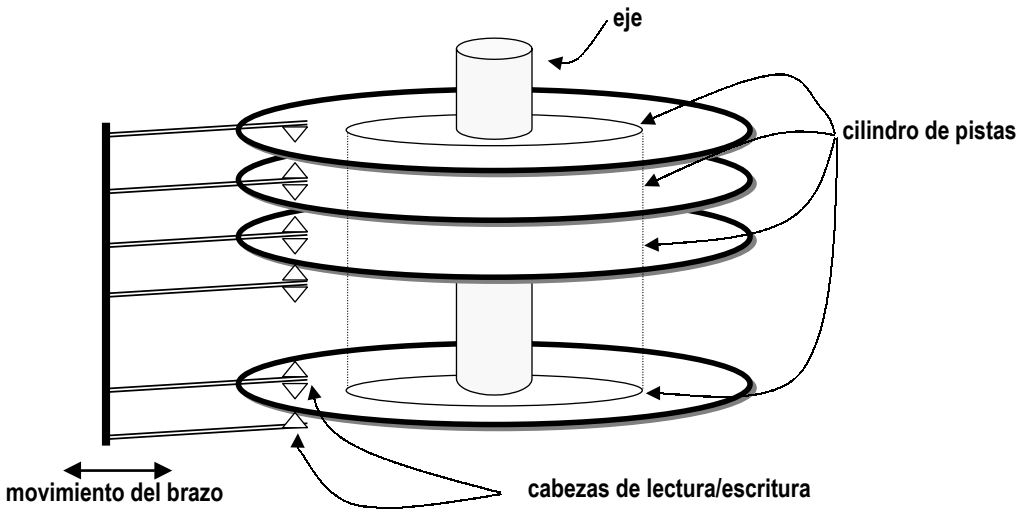
- Ficheros desordenados.
- Ficheros ordenados.
- Ficheros dispersos (hashing).
- Agrupamiento (clustering).

Además, se puede utilizar **estructuras de acceso adicionales** como los índices.

3. Dispositivos de almacenamiento secundario

Discos

La información se almacena en ***pistas*** consecutivas del disco. En un paquete de discos, las pistas que se encuentran unas sobre otras forman ***cilindros***.



Las pistas se dividen en ***sectores/bloques***:

unidades más pequeñas de espacio que se pueden direccionar.

Sectores:

- Divisiones físicas (tamaño fijo).
- Problemas de fragmentación interna.
- Información invisible al programador al principio de cada sector.

Bloques:

- Divisiones lógicas (tamaño variable, definido por el usuario).
- No hay problemas de fragmentación interna.
- Bloques: datos e información sobre los datos → tamaño en bytes y clave.
- Información invisible al programador al principio de cada bloque e información para el programador.

Cluster: unidad de espacio más pequeña que se puede asignar a un fichero (cantidad fija de sectores/bloques contiguos).

El gestor de ficheros ve un fichero como un conjunto de *clusters* (FAT).

Extent: fragmento de fichero formado por varios *clusters* contiguos.

Coste del acceso a disco

Un acceso conlleva tres **operaciones físicas**:

- Tiempo de búsqueda (*seek*).
- Tiempo de rotación (latencia).
- Tiempo de transferencia.

El disco: un cuello de botella

Disco ≈ 5 Mbytes/seg. **Red ≈ 100 Mbytes/seg.**

CPU → esperar a los datos debido a la lentitud de los discos.

Técnicas para resolver este problema:

- Multiprogramación.
- Stripping.
- Discos RAM.
- Caché de disco.

Acceso a los datos

write("texto",c,1) **c: 'P'**

1. Programa → solicita al SO que realice la operación.

2. SO → pasa el trabajo al gestor de ficheros (GF).

3. GF → comprueba que las características lógicas del fichero son consistentes la operación.

4. GF → busca en la FAT la posición física del sector/bloque.

5. GF → pide que el sector/bloque se almacene en un *buffer* de E/S en RAM y escribe sobre él.

6. GF → indica al procesador de E/S dónde está el sector/bloque en RAM y dónde va en el disco.

7. Procesador de E/S → espera a que el dispositivo esté disponible y pone los datos en el formato del disco.

8. Procesador de E/S → envía los datos al controlador del disco.

9. Controlador del disco → indica al dispositivo que mueva la cabeza a la pista, espera hasta que el sector esté bajo ella y manda los datos, que son escritos bit a bit.

4. Ficheros desordenados

Los registros se colocan en el fichero **en el orden en que se van insertando**.

Buscar:	Búsqueda lineal.
Leer ordenadamente:	Ordenación externa.
Insertar:	Añadir por el final →muy eficiente.
Eliminar:	Encontrar registro y borrarlo (o marcar como borrado). Reorganización cada cierto tiempo.
Modificar:	Si cabe: encontrar y modificar. Si no cabe: borrar e insertar.

- Se suelen utilizar con caminos de acceso adicionales (índices).
- También se utilizan para almacenar datos que se van a procesar más tarde.

5. Ficheros ordenados

Los registros se encuentran ordenados físicamente en el fichero **según el valor de un campo**, el campo de ordenación.

Buscar:	Por el campo de ordenación: búsqueda binaria Por otro campo: búsqueda lineal.
Leer ordenadamente:	Por el campo de ordenación: muy eficiente. Por otro campo: ordenación externa.
Insertar:	Encontrar posición, hacer hueco y escribir. Opciones: tener espacio vacío en los bloques para inserciones o tener un fichero de desbordamiento (*overflow*).
Eliminar:	Encontrar registro y borrarlo (o marcar como borrado). Reorganización cada cierto tiempo.
Modificar:	Si cabe: encontrar y modificar. Si no cabe: borrar, hacer hueco e insertar. Si se modifica el campo de ordenación: cambiar el registro de lugar.

- Los ficheros ordenados se suelen utilizar para hacer **índices**.

6. Ficheros dispersos (*hashing*)

Dirección de cada registro: se calcula aplicando cierta **función** sobre uno de sus campos, el ***campo de dispersión***.

Acceso a los datos: muy rápido **sólo** si se busca con la condición de **igualdad** sobre el campo de dispersión.

La dispersión se puede utilizar a nivel interno (RAM), como una estructura de datos de un programa, o bien a nivel externo para ficheros en disco.

Tipos de dispersión a nivel externo:

- Estática
- Dinámica
- Extensible
- Lineal

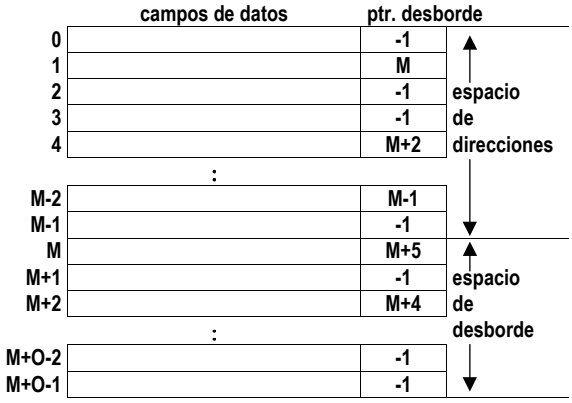
Dispersión Interna

Funciones más comunes:

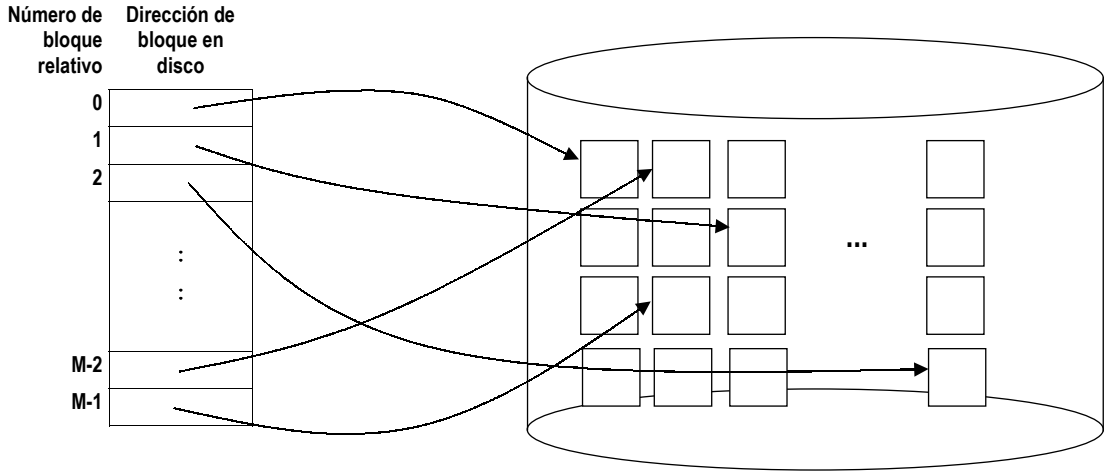
- h(K) = K *mod* M
- Partir el campo en trozos y sumarlos o aplicar función lógica.
- Extraer ciertos dígitos del campo.

Resolución de colisiones:

- Direccionamiento abierto.
- Encadenamiento.
- Dispersión múltiple.



Dispersión Estática

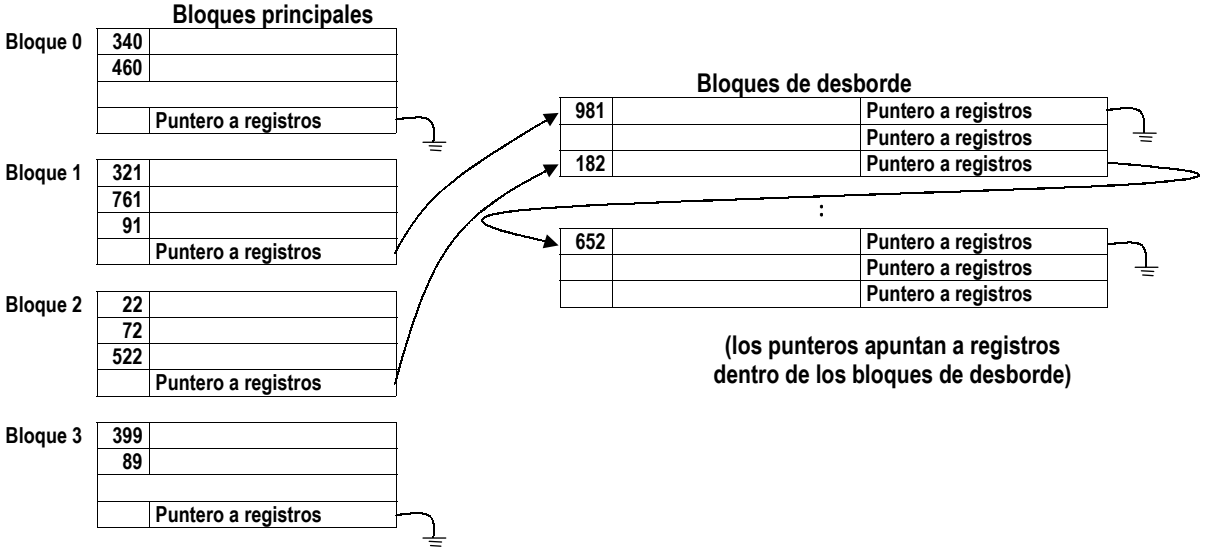


La función de dispersión produce un **número de bloque relativo**.
En una tabla que se guarda en la cabecera del fichero se convierte ese número relativo en una **dirección efectiva del disco** (bloque o *cluster*).

Dispersión Estática

Manejo de colisiones mediante **encadenamiento**: cada bloque con colisiones tiene un puntero a una lista de registros de desbordamiento (*overflow*).

Los registros de esta lista están encadenados (enlazados por punteros).



Dispersión Estática

Buscar:	Por el campo de dispersión: muy eficiente. Por otro campo: búsqueda lineal.
Leer ordenadamente:	Caro (las funciones de dispersión no suelen mantener los registros en un orden).
Insertar:	Aplicar la función de dispersión y si hay colisión aplicar el algoritmo correspondiente.
Eliminar:	Encontrar registro y borrarlo. Si hay lista de desbordamiento, mover un registro de la lista al bloque.
Modificar:	Encontrar y modificar. Si se modifica el campo de dispersión: cambiar el registro de lugar (borrar e insertar).

Gran inconveniente de la dispersión estática: el espacio de almacenamiento es **fijo**.

- Si hay menos de registros de los que caben: hay **espacio no utilizado**.
- Si hay más de registros de los que caben: habrá largas listas de desbordamiento → **acceso muy lento**.

¡cambiar a otra función de dispersión y redistribuir los registros!

Técnicas que permiten la expansión dinámica del fichero

- Dispersión dinámica
- Dispersión extensible

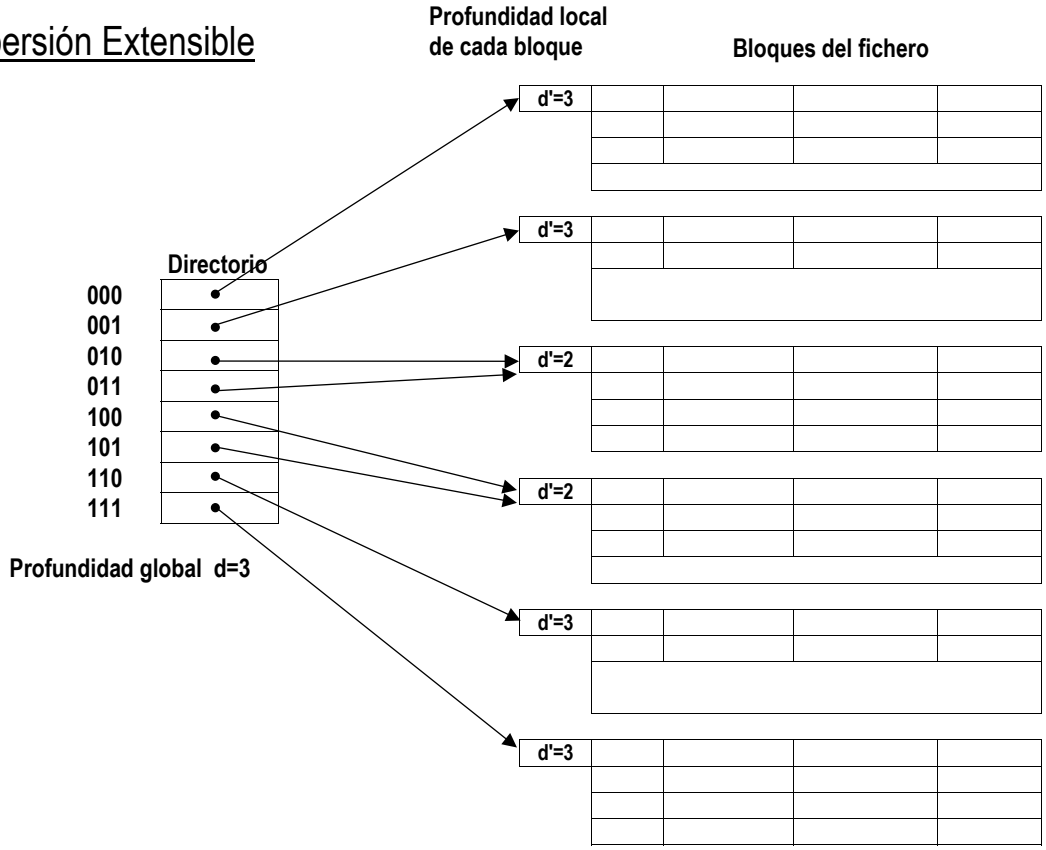
} necesitan almacenar una estructura de acceso (directorio)

- Dispersión lineal

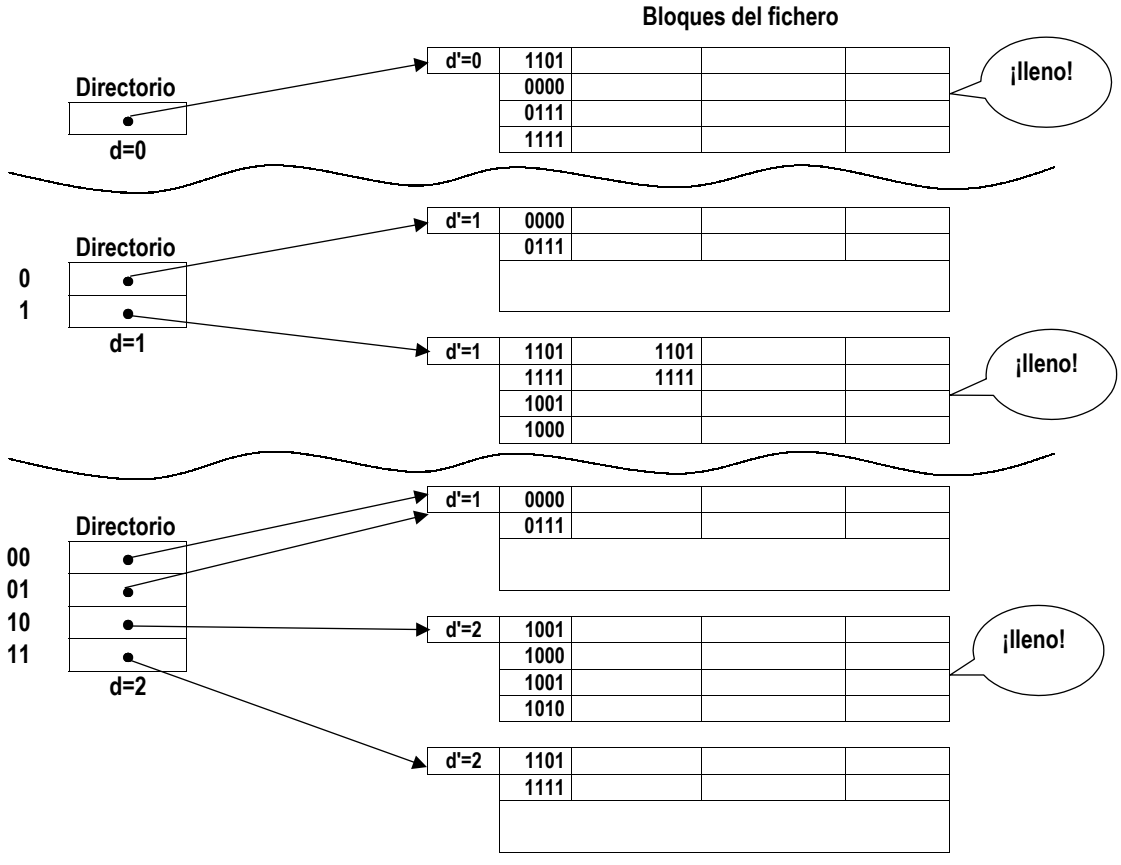
Resultado de la función de dispersión → número entero no negativo → se puede representar en **binario**.

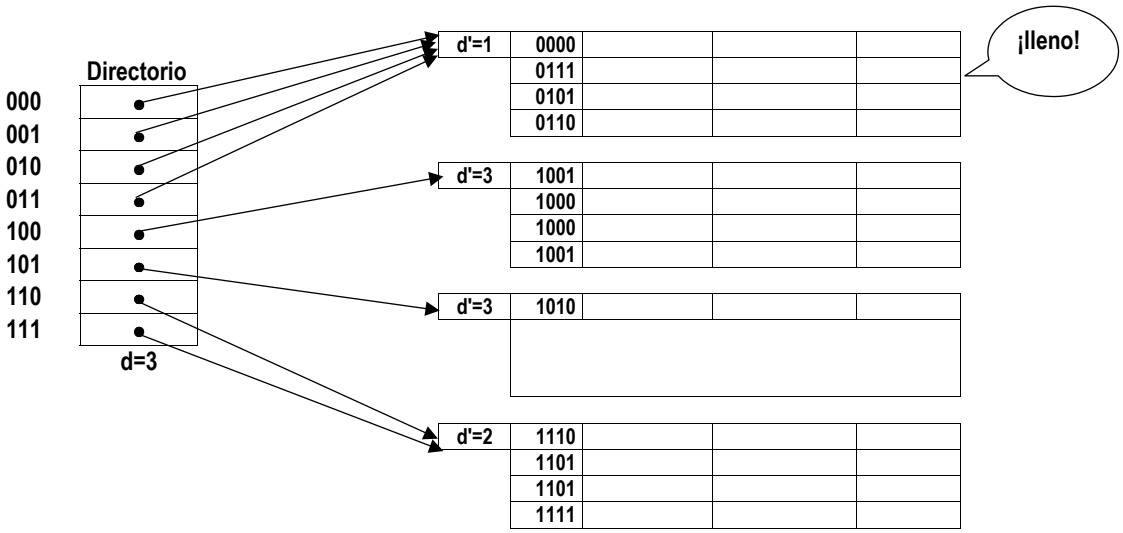
Los registros se distribuyen teniendo en cuenta los primeros bits de este número, al que se denomina ***valor de dispersión***.

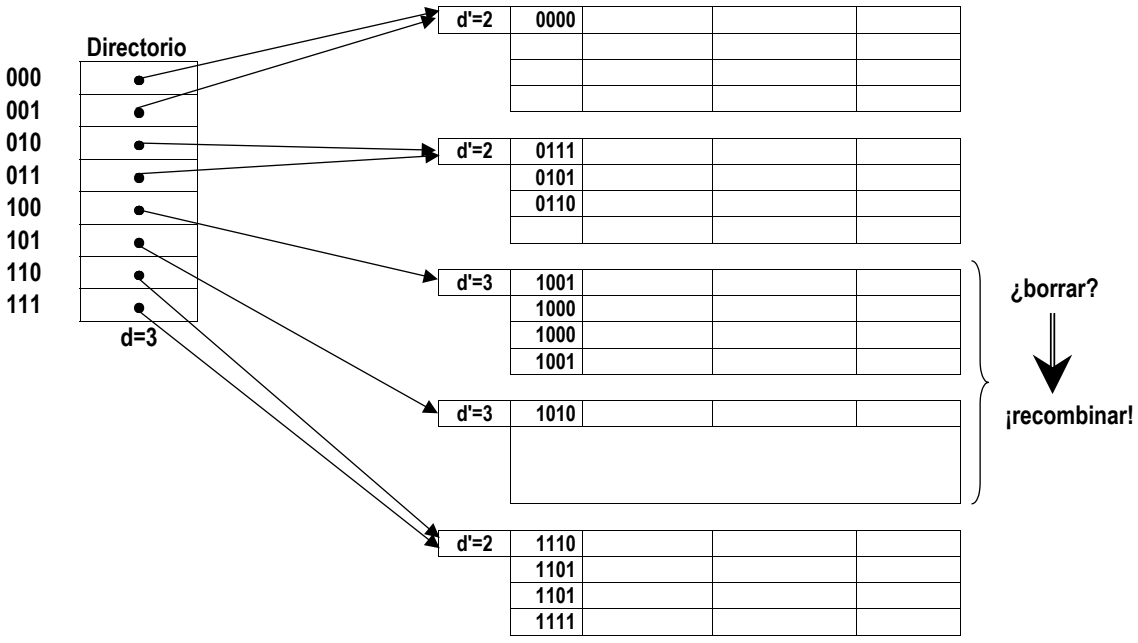
Dispersión Extensible

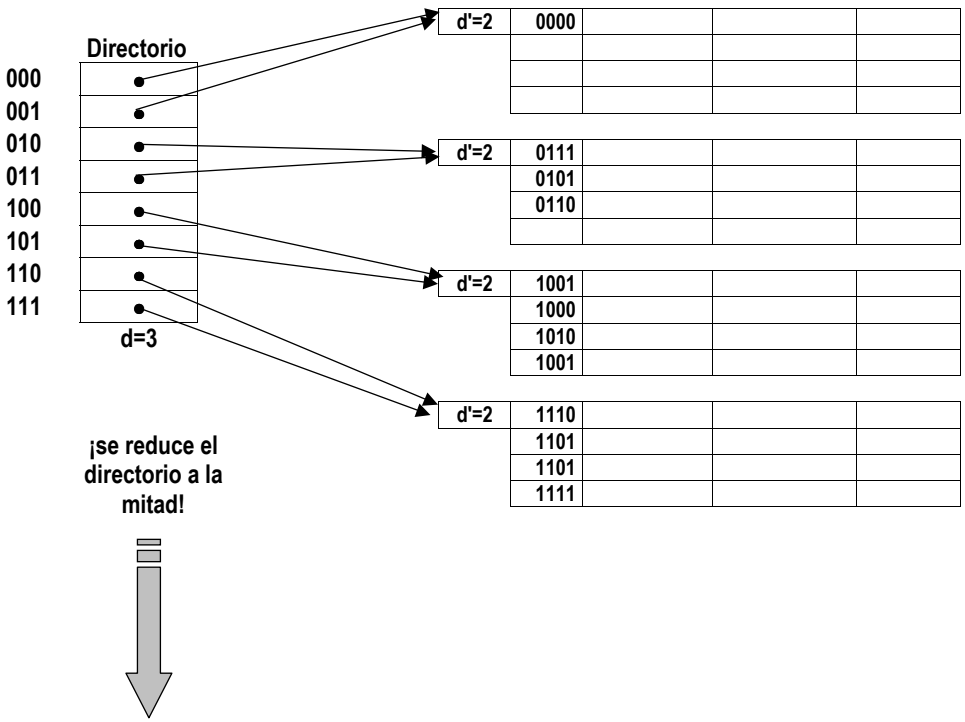


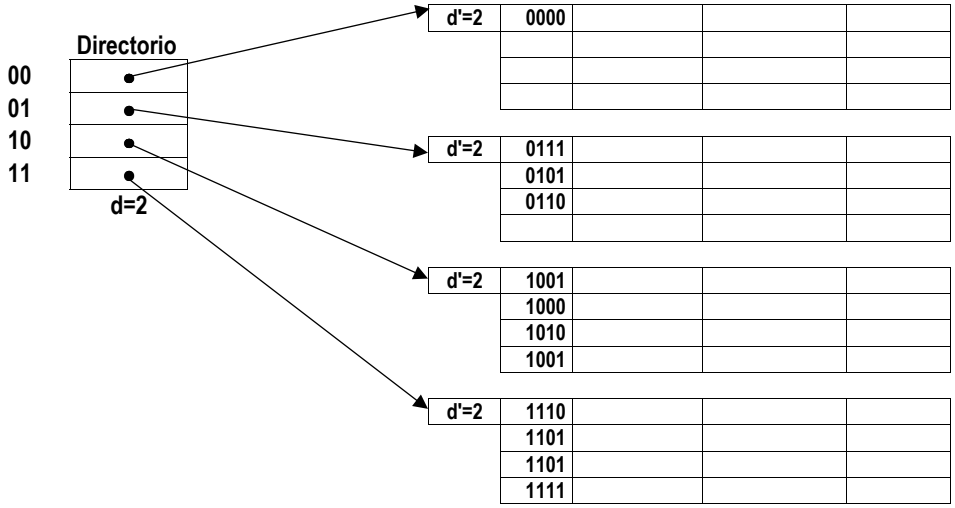
Ejemplo




Directorio
000
001
010
011
100
101
110
111
d=3
d'=1
0000
0111
0101
0110
¡lleno!
d'=3
1001
1000
1000
1001
d'=3
1010
d'=2
1110
1101
1101
1111


Directorio
000
001
010
011
100
101
110
111
d=3
d'=2
0000
d'=2
0111
0101
0110
d'=3
1001
1000
1000
1001
d'=3
1010
d'=2
1110
1101
1101
1111
¿borrar?
¡recombinar!


Directorio
000
001
010
011
100
101
110
111
d=3
d'=2
0000
d'=2
0111
0101
0110
d'=2
1001
1000
1010
1001
d'=2
1110
1101
1101
1111
¡se reduce el directorio a la mitad!


Directorio
00
01
10
11
d=2
d'=2
0000
d'=2
0111
0101
0110
d'=2
1001
1000
1010
1001
d'=2
1110
1101
1101
1111