Tema 4: Organización de Ficheros: **Organizaciones Base**

- **Introducción. Procesos y Parámetros.**
- **Organizaciones Consecutivas**
  - **Organización básica: Organización Serial**
  - **Organización ordenada: Secuencial**
- **Organizaciones Direccionadas**
  - **Direccionamiento Directo**
  - **Direccionamiento Disperso**
  - **Dispersión Virtual**
  - **Dispersiones Extensible y Dinámica**
- **Comparativa**

Tema 4.1: Introducción.

La elección de una determinada organización para un fichero es una decisión de diseño, que depende de ciertos parámetros:

- *tipo de los procesos* (por los que se va a ver afectado el fichero)
- *factores del fichero que se quieren optimizar*
- *parámetros del fichero*
- *características del soporte* (donde va a ser almacenado el fichero)

Estos parámetros producen una serie de condiciones, algunas flexibles y otras no. De las primeras, si aparecen contrapuestas, habrá que priorizar e irlas relajando hasta llegar a una solución de compromiso.

Tema 4.1: Procesos

Naturaleza de los procesos

- *Naturaleza de la operación: Actualización / Recuperación*
- *Tipo de Acceso (at. al orden): Ordenado / Desordenado*
- *Tasa de Actividad del proceso*
  - *tasas elevadas (>10%)*
    - tipo de acceso predominante: serial/secuencial → orgs. consecutivas
  - *tasas reducidas (<10%)*
    - tipo de acceso predominante: mixto → organizaciones invertidas
  - *selección unívoca (un elemento)*
    - tipo de acceso: directo → organizaciones direccionadas
  - procesos diversos: *organizaciones indizadas*

Tema 4.1: Parámetros

Factores del Fichero que se quieren Optimizar

- *Tiempo de respuesta: (en Actualización / en Recuperación)*
  - *disminuir los accesos (organizaciones en árbol, memorias intermedias…)*
- *Espacio de Almacenamiento*
  - Incrementar densidad, minimizar almacenamiento auxiliar, compresión de campos, buscar tamaños de cubo adecuados, etc.
  - ¡El espacio de almacenamiento influye en el Tiempo de Respuesta!
- *Tiempo de proceso:*
  - Evitar reorganizaciones en tiempo de proceso, ensamblajes ...
- *Coste de Desarrollo y Mantenimiento*
  - Reducir coste de programación, reducir coste de actualizaciones y mantenimiento, minimizar las reorganizaciones, etc.

Tema 4.1: Parámetros

Parámetros del Fichero

- *Volumen y ocupación (total y de un registro)*
- *Volatilidad (tasas de inserción, modificación, supresión, y crecimiento)*

Características del Soporte

- *Tipo de soporte (serial, direccionado, ...)*
- *Bloque: Tamaño y Tiempo de Acceso*
- *Espacio de direccionamiento (tamaño de la dirección)*
- *Memorias intermedias (caché)*

* Nota: *no hay que confundir 'Tipo de Soporte' con 'Organización del Fichero'*

Tema 4.2: Organizaciones Consecutivas
Organización básica: Organización Serial

Surge con los soportes seriales:

Soporte Serial: proporciona registros físicos en serie, esto es, que se registran uno detrás de otro, y se acceden en ese orden

Ejemplo de soporte serial: la cinta magnética

Instrucciones: leer (bloque) y reset

- procesos selectivos: tiene que buscar uno o varios registros
  - antes de buscar, se apunta al principio del fichero (reset)
  - se van leyendo todos los registros hasta identificar el buscado
  - si se quiere localizar varios, se lee todo el fichero
- procesos a la totalidad: no precisan localización ni orden→ óptimos

Tema 4.2.1: Organización Serial

Interacción con Ficheros Seriales:

Recuperación:

- consulta: se recupera el contenido

Actualización:

- inserción: se añaden registros al final del fichero
- borrado: { borrado físico: se *vacía* el registro → se desplaza el resto
  borrado lógico: se *marca* el registro → se genera un *hueco*
- modificación:
  - registros fijos: se altera el contenido
  - registros variables: se borra el antiguo y se reinserta modificado

Tema 4.2.1: Organización Serial

Cálculo del tiempo de acceso:

- *máximo:* peor caso → leer todos

$$t_{max} = n^o\ registros \cdot t_{acceso\_registro}$$

- *medio*: se calcula como el acceso a la mediana (el de en medio)

$$t_{medio} = \frac{n^o\ registros + 1}{2} \cdot t_{acceso\_registro}$$

- *También debe aplicarse la consideración de bloque, si procede*
- *En tal caso, el 'nº registros' será 'nº bloques', y '$t_{acceso\_registro}$' será 't_{acceso_bloque}'*

Tema 4.2.1: Mantenimiento Serial: Gestión de Huecos

Borrado Físico:

- en organización serial y registros fijos:
  - para borrar se lee el último registro y se escribe sobre el que se desea eliminar; el último se vacía (de tres a cinco accesos más la selección; pero casi siempre son todos sobre memoria intermedia).
- en organización serial y registros variables:
  - tamaños diversos → para borrar se desplazan todos los registros posteriores al eliminado → altamente **ineficiente**

Borrado Lógico:

- en organización serial (en general):
  - para borrar se introduce una marca de borrado lógico (1 acceso escritura), y al insertar se recorre el fichero buscando un hueco de tamaño adecuado. Si no hubiera un hueco suficientemente grande, se inserta al final.
  - *Lista de Huecos*: (tamaño+posición) si no está muy lejos, se utiliza
- **Compactación**: se van desplazando registros para eliminar huecos

Tema 4.2.2: Organización ordenada, Organización Secuencial

Surge a partir de la serial, introduciendo un orden de registros
Se propicia gracias a una clave de ordenación física

Instrucciones: las seriales más desplazar (avanzar y retroceder)

- Se acceden los bloques aleatoriamente, y por ello se precisa un mecanismo para poder localizar el comienzo del primer registro
  - a nivel físico (por bloques): comienzo de bloque
  - a nivel físico-lógico (registros consecutivos): marca de inicio/fin

Procesos:

- procesos selectivos: puede aprovechar el que vayan ordenados
  - localizar un registro: búsqueda dicotómica
  - localizar varios registros: dicotómica + acceso serial
  - claves alternativas: búsqueda serial (leer todo el fichero)
- procesos a la totalidad: gran eficiencia en procesos ordenados

Tema 4.2.2: Organización Secuencial

Interacción con Ficheros Secuenciales Consecutivos:

El primer carácter de un bloque puede no corresponder al primer registro
→ se necesita una marca de inicio (o fin) de registro

Recuperación (consulta): recuperar el contenido de { un registro; varios regs. { sin orden; ordenados }

Actualización:

- inserción: se añaden registros al final del fichero → altera el orden (*)
- borrado: igual que en serial, pero es más difícil reutilizar huecos.
- modificación:
  - registros fijos: se altera el contenido si no altera el orden (si no se modifica la clave de ordenación)
  - para modificar clave de ordenación, y en registros variables: se borra el antiguo y se reinserta modificado → altera el orden (*)

Nota(*): observar que se genera un *área desordenada* al final del fichero

Tema 4.2.2: Organización Secuencial

Interacción con Ficheros Secuenciales NO Consecutivos:

El primer registro del bloque comienza en el primer carácter del bloque
→ puede prescindirse de la marca separadora de registros

Operaciones de Recuperación: mismo mecanismo que las consecutivas

Operaciones de Actualización: todos los bloques (*cubos*) tienen un hueco.

- Inserción: se inserta en su bloque si cabe (si no, en el área desordenada)
- Borrado: se agranda el hueco del cubo donde se borra
- Modificación (de campo no clave ord.): borrado y re-inserción (es decir, si el reg. modificado cabe en su sitio se mantendrá allí)

Espacio Libre Distribuido: dejar un porcentaje de espacio libre en los cubos

- Pueden diferenciarse dos porcentajes (para modificación y para inserción)
- Técnicas:
  - *rotaciones*: traspasar elementos de un cubo lleno a otro vecino con sitio
  - intercalar cubos completamente vacíos (al crear o reorganizar el fichero)

Tema 4.2.2: Organización Secuencial

Localización en Ficheros Secuenciales: *Búsqueda Dicotómica*

Búsqueda Dicotómica:

- mirar el de en medio: si coincide, fin
- si no coincide, escoger la mitad que contiene el elemento buscado
- volver a empezar (sobre la mitad escogida)

Para claves ***no unívocas***, se hará *Búsqueda Dicotómica Extendida*:

- Buscar primer elemento (por *búsqueda dicotómica*)
- Buscar hacia arriba hasta encontrar uno distinto (fallo)
- Buscar hacia abajo hasta encontrar uno distinto (fallo)
- Las entradas que coinciden, se incluyen en un *'conjunto resultado'*

Tema 4.2.2: Organización Secuencial

Tiempo de acceso Búsqueda Dicotómica:

- *consideramos sólo el máximo:*
  peor caso → encontrarlo en la última vuelta

$$nºaccesos_{max} = \lceil \log_2 (n+1) \rceil$$

$$t_{max} = nºaccesos_{max} \cdot t_{acceso\_registro}$$

- El número de elementos en la búsqueda (n) depende de la relación físico-lógica:
  - en ficheros con registros expandidos, se trata del número de registros
  - en el resto (caso habitual) se trata del número de bloques del fichero

- En ficheros secuenciales, se usa búsqueda dicotómica con una salvedad:
  - búsqueda dicotómica sobre área ordenada
  - si no se encuentra, **búsqueda serial sobre área desordenada**

$$nºaccesos_{sec} = \lceil \log_2 (n+1) \rceil + nºaccesos_{serial}$$

Tema 4.2.2: Organización Secuencial

Tiempo de acceso Búsqueda Dicotómica Extendida:

- *Se busca a través de una clave que presenta k coincidencias (de media), y que tiene v valores distintos (k * v = número total de registros).*
- *Para hallar el número de accesos, consideramos sólo el máximo:*

peor caso → - encontrar el primer elemento en la última vuelta
- el bloque anterior se leería sólo para encontrar un fallo
- se recuperarán k+1 registros (k coincidencias y un fallo)

$$n^o accesos_{max} = \lceil \log_2 (n+1) \rceil + \lceil \frac{(k+1)}{f_b} \rceil$$

- El número de elementos en la búsqueda (*n*) depende de la relación físico-lógica:
  - si los *k* registros caben en un bloque, se utilizará n=número de bloques
  - en el resto de casos, se debe utilizar el número de valores distintos (v)

- Se debe seguir considerando el área desordenada (siempre que exista).

Tema 4.2.3: Mantenimiento: Huecos y Orden

1. **Reutilización de huecos**:

   * Lista ordenada de huecos: (clave ant, clave post, tamaño)
     - al borrar actualiza la lista de huecos;
     - al insertar, comprueba si existe un hueco apropiado, lo usa y actualiza la lista; si no existe hueco, inserta el nuevo registro al final

   * Espacio libre distribuido: (org. secuencial no consecutiva)
   - el fichero se divide en 'cubos' (bloque físico-lógico de registros)
   - cada cubo tendrá un bloque (o más; habitualmente uno)
   - cada cubo reserva un porcentaje de su espacio para registros que crecen, y un porcentaje para nuevos registros
     → (casi) todos los cubos tienen hueco
   - para insertar, se mira si cabe en su cubo; si no, se inserta al final

2. **Reorganización:** reescribir todos los registros ordenados
   - *Problema: algoritmos sobre el propio fichero muy pesados (coste elevado)*
   - *Habitualmente se usan algoritmos basados en almacenamiento auxiliar.*

Tema 4.3: Organización Direccionada

Surge gracias a los soportes direccionados:

Soporte Direccionado: proporciona registros físicos localizables, es decir, que cuentan con una dirección física en el soporte.

Ejemplo de soporte direccionado: el disco

Instrucciones: leer(n) / escribir(n)

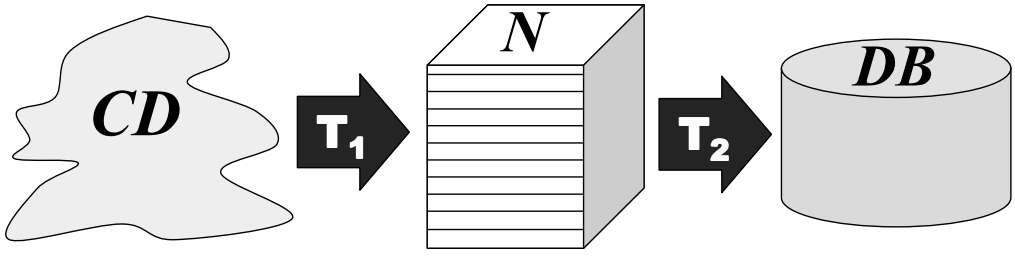
- procesos selectivos: localización inmediata → óptimos
  - la localización se hará mediante la clave de direccionamiento
  - es necesario mantener una correspondencia con la dir. física
- procesos a la totalidad: la localización no es una ventaja

Tema 4.3: Organización Direccionada (*Hashing*)

- La clave de direccionamiento no suele coincidir con la dirección física
- Es necesario transformarla

  clave direccionamiento → espacio direccionamiento

  espacio direccionamiento → dirección base (física)



Elemento:	clave	←→	dir. cubo	←→ dir. bloque
Ámbito:	lógico	←→	físico-lógico	←→ físico

Tema 4.3: Organización Direccionada

• ***Espacio de Direccionamiento (N):***
rango de direcciones relativas disponible (para un fichero)

Direccionamiento:
clave de direccionamiento: localiza sobre el dominio de la clave
dirección relativa: ordinal sobre el espacio de direccionamiento (N)
dirección base: dirección física

Algoritmo de Transformación. Pasos:
T_1: *proporciona la dirección relativa a partir de la clave de dir.*
T_2: *proporciona la dirección base a partir de la dir. relativa*

Tema 4.3: Tipos de Org.Direccionada

1. Organización Direccionada Directa:
cada registro tiene su dirección reservada
- una dirección para cada clave de direccionamiento
- la clave de direccionamiento es también clave de identificación

Ejemplo: los buzones de conserjería, cada nombre tiene un buzón

2. Organización Direccionada Dispersa *(a cubo)**:***
una dirección corresponde a varias claves de direccionamiento.
Ejemplo: las estanterías en una biblioteca

Tema 4.3.1: Direccionada Directa

- ***Tipos de Organización Direccionada Directa:***

i) Absoluta: la clave de direccionamiento es la dirección base

Ventajas: no es necesario aplicar algoritmo de transformación

Inconvenientes: - la CD es información oscura para los usuarios
Ejemplo: dirección postal = lat. 40º57'N long 4º10'O... (¿?)
- hace al fichero dependiente del dispositivo

ii) Relativa: la clave de direccionamiento necesita transformación

- a) clave de direccionamiento coincide con dirección relativa
(se trata de un número ordinal dentro del espacio de direccionamiento)
- b) función de transformación biyectiva,, f: CD $\rightarrow$ N

Tema 4.3.1: Direccionada Directa

Ventajas

- la localización mediante CD es inmediata $\rightarrow$ 1 acceso

Inconvenientes:

- difícil encontrar CD y función de transformación adecuadas
- cuando se dispone de ellas, suelen producir baja densidad
observar que cada CD tiene una posición de almacenamiento reservada
si no se encuentran todas las CD en el fichero, hay mucho espacio vacío

Ejemplo: pupitres en la biblioteca con D.N.I. (¿uno por cada D.N.I.? ¡...!)

Solución 1: *dispersar* las CD → ***Orgs. Direccionadas Dispersas***

Ejemplo: pupitres en la biblioteca con nombre de alumno (¿uno por cada alumno?)

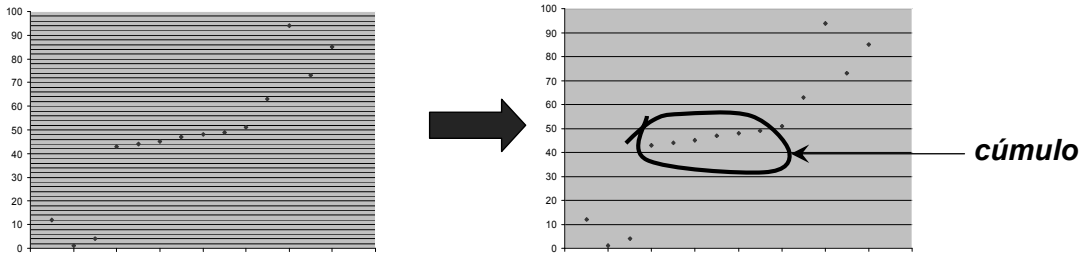
Problema 1: se desperdicia mucho espacio vacío

Tema 4.3.2: Direccionada Dispersa

Organización Direccionada Dispersa (Hashing)

- Se reduce el espacio de direccionamiento N/m,, con m ≥ 1
- Habrá varias CD con la misma dirección: *claves sinónimas*
- Al hacer que varias CD coincidan en la misma dirección relativa se disminuye el número de posiciones vacías y aumenta la densidad

Problema 2: se pueden producir colisiones y cúmulos
(muchos registros coinciden en el mismo lugar)



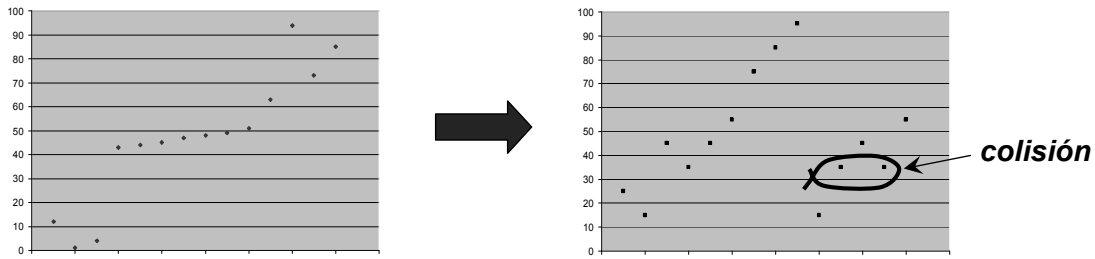
Tema 4.3.2: Direccionada Dispersa

Problema 2: se forman cúmulos mientras que quedan huecos libres

Solución 2: *mejorar la dispersión*

- cambiar la f. de transformación para que reparta mejor los registros
(así se disminuyen las coincidencias en dirección de varios registros)

Problema 3: siguen produciéndose colisiones
(varios registros coinciden en el mismo lugar)



Tema 4.3.2: Direccionada Dispersa

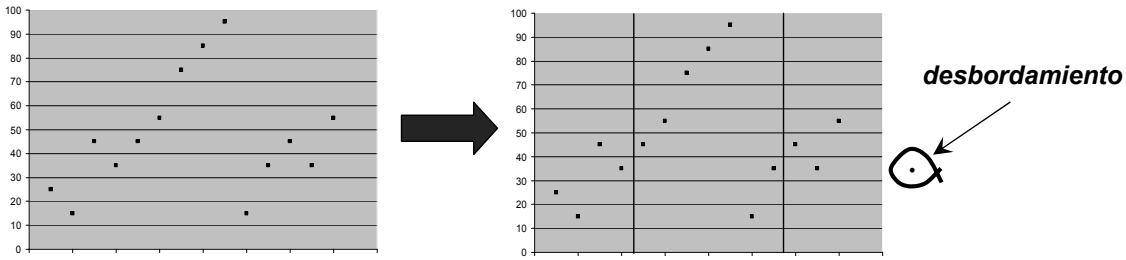
Solución 3 *en cada dir. física varios registros lógicos* → ***cubo***

- ***Organización Direccionada Dispersa a cubo.***

  Cubo de tamaño *n* = en cada dirección caben n registros

- Se asignan varias direcciones reales a cada dirección base
  (así, aún existiendo colisión, caben varios registros en cada dirección)

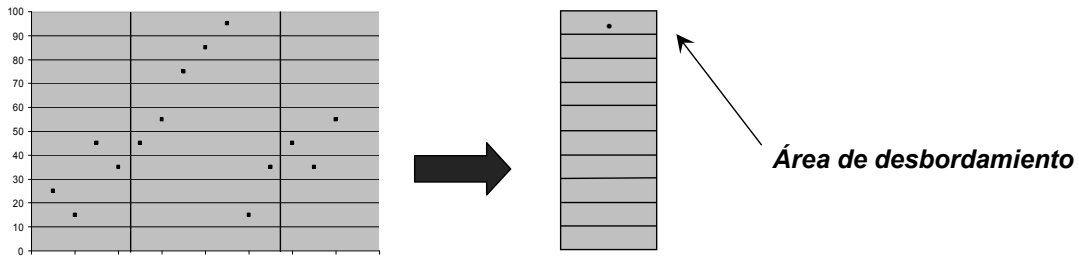
Problema 4: ¿y si hay más colisiones que espacio disponible?
(coinciden en un cubo más registros de los que caben)



Tema 4.3.2: Direccionada Dispersa

Soluciones a los desbordamientos: *consideraremos dos...*

- aumentar el tamaño del cubo (baja la densidad, y hay que preveerlo)
- aplicar políticas de gestión de desbordamientos
  - buscar otra ubicación a los registros excedentarios
  - almacenar registros excedentarios en área de desbordamiento
    (situada al final del fichero, y cuyo acceso es serial o secuencial)



Tema 4.3.3: Algoritmos de Transformación

Se pueden diferenciar tres pasos:

- Si la CD es alfanumérica, transformarla a numérica
- Aplicar una función de trasformación para dispersar
  - la función debe estar definida sobre el espacio de direccionamiento (N)
  - si no es así, aplicar otra función para ajustarla a ese espacio (N)
- Transformar la dirección relativa obtenida en dirección base

Ejemplo: nombre de autor con N=100

'Dumas' → 68 + 117 + 109 + 97 + 115 = 506 → 506 MOD 100 = 6 → E0AX1006

Tema 4.3.3: Algoritmos de Transformación

Funciones de Transformación:

- *La mejor función de transformación es la que proporcione una distribución uniforme sobre el espacio de direccionamiento (dispersión ideal)*
- *No existe una solución universal. La elección de una función de transformación depende del problema en particular.*
- *Se puede hacer un estudio estadístico de la distribución de las CD sobre un espacio de direccionamiento (concreto o genérico) para mejorar después la dispersión (para esos espacios)*
- *La función escogida puede ser el resultado de la combinación de otras funciones comunes para conseguir el resultado deseado*
- *Si la clave de direccionamiento produce pocos valores distintos, por más que se transforme no aumentará su capacidad de direccionamiento.*

Tema 4.3.3: Algoritmos de Transformación

Funciones de Transformación: (continuación I)

- **Truncamiento**

  consiste en deshechar parte de la clave (por ejemplo, los x primeros dígitos)
  Ejemplo: resto o resíduo; 115279 → 115279 MOD 1000 → 279

- **División-Resto o Resíduo:**

  consiste en dividir la clave entre un número natural y tomar el resto
  - muy útil para adaptar cualquier resultado de cualquier otra función de transformación al espacio de direccionamiento utilizado: CD MOD N
  - Observar que el truncamiento es un caso concreto de la función resíduo

- **Plegado:**

  consiste en dividir la clave en varios grupos numéricos y combinarlos
  Ejemplo: mitad y suma; 115279 → 115 | 279 → 115 + 279 = 394

Tema 4.3.3: Algoritmos de Transformación

Funciones de Transformación: (continuación II)

- **Cambio de Base**

  Tomando la CD en decimal, el resultado es la CD expresada en otra base
  Ejemplo: base 11, CD = 95

  525 MOD 11 = 8; 525 DIV 11 = 47; 47 MOD 11 = 3; 4 MOD 11 = 4 → **438**

  - Observar que pueden aparecer símbolos nuevos que habría que transformar
    Ejemplo: 527 MOD 11 = 10; ... → **4· 100 + 3·10 + 10·1 = 440**

- **Método de Lin** *(referido a p y q^n, con p y q primos)*

  se considera CD expresada en base p; se expresa en decimal y divide entre q^n
  Ejemplo: p=11, q=7, n=2, CD 95;

  $9*11^1 + 5*11^0 = 104 \rightarrow 104$ DIV $7^2 =$ **2**

Tema 4.3.4: Desbordamientos

Tratamiento de Desbordamientos:

Colisión: dos CD tienen la misma dirección base; al llegar la segunda, colisiona

Desbordamiento: un elemento, que ha producido una colisión, no cabe en su dir. base

Los elementos que desbordan han de ser almacenados localizables, según una política.
Las Políticas de Gestión de Desbordamientos se pueden clasificar de dos modos:

- **Según la zona donde se ubique el registro desbordado**
  - **a)** Saturación: otra dirección dentro del espacio de direccionamiento
  - **b)** Área Desbordamiento: fuera del área de datos (en otro fichero)
- **Según el mecanismo de ubicación:**
  1. Direccionamiento abierto
  2. Encadenamiento
  3. Otros (organización independiente)

m \ z	a	b
1	✓	×
2	✓	✓
3	×	✓

Tema 4.3.4: Desbordamientos

a.1.- *Saturación con Direccionamiento Abierto:*

- La nueva dirección se averigua a partir de la dirección antigua (desbordada).
- Si esta estuviera ocupada se produce un *choque.*
- Si además no cupiera, sería un *rebote*
- *Si se produce un rebote se buscará otra dirección nueva hasta* encontrar una posición libre o hasta haber recorrido todo el espacio (área de datos saturada).
- Si el área de datos está saturada (completa) esto provoca automáticamente una reorganización del direccionamiento sobre un espacio N' mayor que N.

  → ***extensión del espacio de direccionamiento***

**** Técnicas de Direccionamiento abierto:***

- Saturación progresiva (Sondeo Lineal)
- *Rehashing* (Sondeo Aleatorio)
- Sondeo Cuadrático
- Doble Transformación

Tema 4.3.4: Desbordamientos

a.1.- *(cont.)* ***Técnicas de Direccionamiento Abierto:***

- **Saturación progresiva (Sondeo Lineal):**

  Al desbordar, la nueva dirección es la siguiente a la dirección base: $D'=(D+1) \text{ MOD } N$

- **Rehashing (Sondeo Aleatorio):**

  En vez de la siguiente, se suma otra cifra: $D'=(D+k) \text{ MOD } N$
  - N y k deben ser primos relativos para que se recorra todo el espacio de direcc. (N)
  - Observar que la 'saturación progresiva' es un caso particular de Rehashing (k=1)

- **Sondeo Cuadrático:**

  El sondeo sigue una progresión $D'=(D+k^2) \text{ MOD } N$,, tomando valores D+1, D+4, ... $D+n^2$

- **Doble Transformación:**

  En este caso, en lugar de sumar, se aplica cualquier función de transformación. Queda $D'= f(D) \text{ MOD } N$,, observar que el rehashing es un caso particular de este

Tema 4.3.4: Desbordamientos

ENCADENAMIENTO:

En todos los cubos se reserva espacio para un puntero.

Si el cubo desborda, se usará este puntero para registrar la nueva dirección de los elementos desbordados.

- Los registros desbordados permanecen localizables en una nueva dirección

- Se pueden encadenar **cubos** enteros o **registros** individuales
  - si se encadenan cubos, cada uno tendrá un puntero al siguiente cubo
  - para encadenar registros, el cubo desbordado apunta al primero, y luego cada registro (individualmente) apunta al siguiente.

- Normalmente, cada elemento encadenado tiene un puntero al **siguiente** elemento encadenado, pero también se pueden almacenar varios punteros a posiciones encadenadas (*lista de encadenamiento*, con o sin orden)

Tema 4.3.4: Desbordamientos

a.2.- *Saturación Progresiva Encadenada:*

- Esta técnica consiste en buscar una nueva posición dentro del espacio de direccionamiento, almacenar allí el registro desbordado, y apuntar la dirección en el puntero.
- Este encadenamiento es siempre 'a registro' (1er ptro. en cubo desb.)
- El apuntamiento incluirá el identificador del cubo destino, y la posición dentro del mismo (partes alta y baja, respectivamente).
- Como es una dirección independiente de la dirección anterior (CD), puede utilizarse cualquier cubo como destino. Se suele usar el cubo menos utilizado (el primero vacío).
- Reduce el número de choques y rebotes (pero no los elimina).

Tema 4.3.4: Desbordamientos

ÁREA DE DESBORDAMIENTO:

Consiste en almacenar los registros desbordados en un área especial (área de desbordamiento o de saturación) fuera del área de datos.

Ventaja: se eliminarán los choques (y los rebotes).

Desventaja: es preciso usar más espacio (**auxiliar**).

- El uso de espacio auxiliar no es tan malo por el gasto de espacio, sino porque implica aumentar el espacio de búsqueda en procesos basados en claves de búsqueda alternativas (no privilegiadas).

Tema 4.3.4: Desbordamientos

b.2.- *Área de Desbordamiento Encadenada:*

- Es un encadenamiento fuera del área de datos
- Encadenamiento de Cubos:
  - El puntero de encadenamiento sólo requiere parte alta.
  - Cada cubo encadenado se reserva para esa dirección
  - Todos los cubos encadenados se procesarán de modo serial (equivale a un área serial independiente para cada dirección de N, aunque sus cubos no sean necesariamente contiguos físicamente)
- Encadenamiento de Registros:
  - Es menos eficiente para el proceso 'get all' por la CD, pero reduce notablemente el tamaño del área de desbordamiento.
  - Dependiendo de los procesos frecuentes, se escogerá uno u otro.

Tema 4.3.4: Desbordamientos

b.3.- *Archivo de Desbordamiento (Org. Independiente):*

El área de desbordamiento no se encadena, sino que se trata y maneja como un fichero independiente con su propia organización

- Su organización generalmente es serial o secuencial, pero también se pueden organizar mediante un *direccionamiento secundario:*
  - se definen sobre un espacio de direccionamiento bastante menor que el otro, que será el direccionamiento principal.
  - pueden basarse en una CD distinta a la del direccionamiento ppal. (de menor potencia de direccionamiento pero buena dispersión), o con la misma CD y distinta función de transformación.
  - pueden desbordar, lo que produce una reorganización automática del direccionamiento principal sobre un espacio de dir. mayor.

  → ***extensión del espacio de direccionamiento***

Tema 4.3.5: Dispersión Extensible

- **Concepto:** el espacio de direccionamiento no es fijo.

- La idea viene de: *si desborda, reorganizo sobre un espacio mayor*
- El problema es que para hacerlo, tendría que reescribir todo el área de datos (cuando realmente ha desbordado un cubo).

- **Aproximación:** la dirección relativa se obtiene al concatenar un *prefijo* con el resultado de la función de transformación (módulo).
  - módulo**:** definido en un espacio potencia diez ($N=10^n$)
  - prefijo**:** se calcula aparte, según las necesidades de almacenamiento

- **En particular:** la extensibilidad se suele enfocar con el truncamiento:
  - la transformada tiene *p* dígitos, y en cada momento se usan *k* (con $k \leq p$)
  - si desborda y k<p, usaré un dígito más (k+1)

  → necesito reorganizar cada cubo sobre otros 10 nuevos cubos

Tema 4.3.5: Dispersión Extensible

Problema: exige frecuentes reorganizaciones

- **Solución 0: Demorar reorganizaciones**
  - Se utilizarán técnicas de gestión de desbordamientos para evitar reorganizar
  - Sin embargo, esto implica aumentar el núm.accesos para localizar elementos

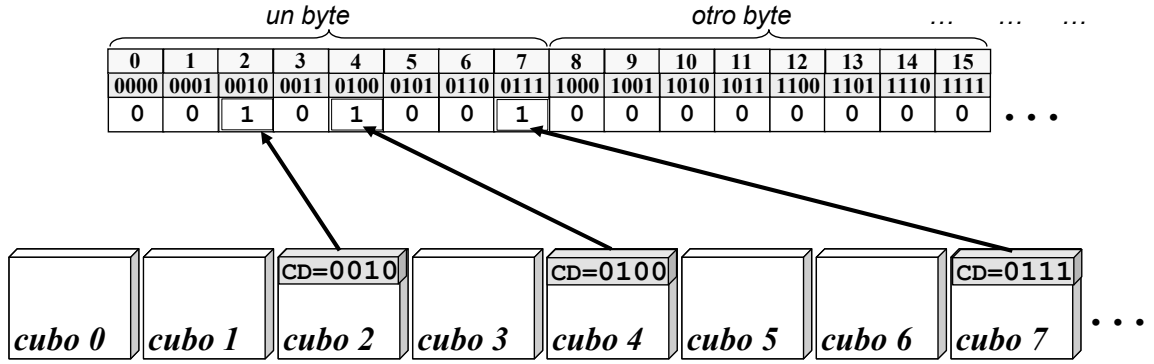
- **Solución 1: minimizar reorganizaciones:**
  - * escogiendo bien los valores de N: utilizar base 2 para direccionar → **$N=2^n$**
    - en cada extensión, se duplica el espacio de direccionamiento
    - en cada compresión, se divide por la mitad
  - * flexibilizando adecuadamente el criterio de extensión/compresión

- **Solución 2: Aplicar Dispersión Virtual**

© 2007 LABDA – Univ. Carlos III de Madrid

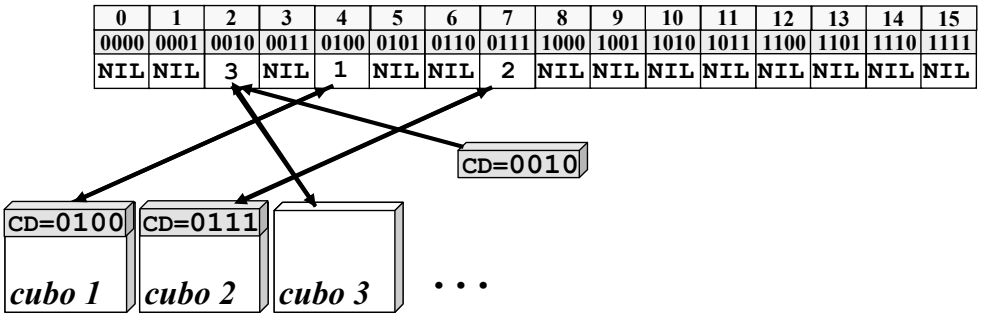
Tema 4.3.6: Dispersión Virtual (I)

- *Consiste en utilizar almacenamiento auxiliar (un directorio) para diferenciar los cubos ocupados de los que están vacíos*
- **Aproximación 1: <u>Directorio de Ocurrencias</u>**
  - Almacena un bit por cada cubo, distinguiendo los vacíos (0) y llenos (1)
  - Ahorra accesos (evita recuperar cubos vacíos)
  - El directorio tiene un tamaño muy reducido



Tema 4.3.6: Dispersión Virtual (II)

- **Aproximación 2: <u>Directorio Virtual</u> (de punteros)**
  - La trasformada de la CD no será la dirección real, sino dirección virtual
  - En cada posición del directorio (dirección virtual), en lugar de un bit, se va a almacenar un puntero con la dirección real del cubo
  - Además de accesos, ahorra espacio; pero el directorio es más grande…
- La dispersión virtual puede ser utilizada para cualquier organización direccionada, aumentando notablemente la densidad de la misma.
  - Posibilita la dispersión extensible y la dispersión dinámica



Tema 4.3.7: Disp. Virtual Extensible

- Consiste en aplicar la extensibilidad a una organización direccionada **¡basada en directorio virtual!**
- Observar que en cada desdoblamiento lo que duplicamos no es el área de datos, sino solamente el directorio (bastante eficiente)
- El directorio debe ser virtual, porque si fuera de ocurrencias se debería duplicar también el área de datos: sería como la extensible normal, pero con directorio de ocurrencias para reducir los accesos → no es aplicable, el coste de los desdoblamientos sería inabordable

Tema 4.3.7: Disp. Virtual Extensible
(basada en directorio – I)

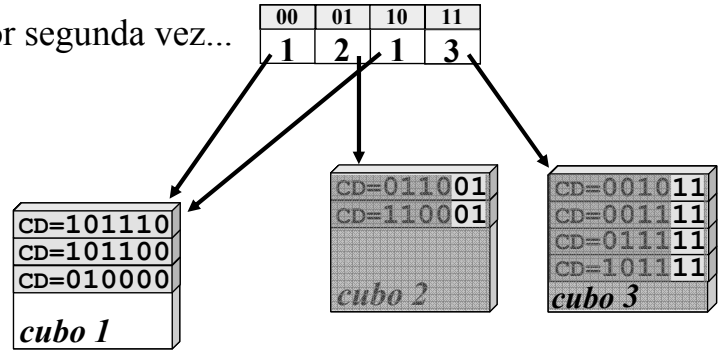
Cuando sólo hay un cubo, sólo hay un puntero que le apunte... 

→ el directorio sólo tiene un puntero y la dirección no usa ningún byte (2^0=1)

PERO si se llena, desdobla y hay que utilizar un bit más en la dirección... (0+1 = 1)

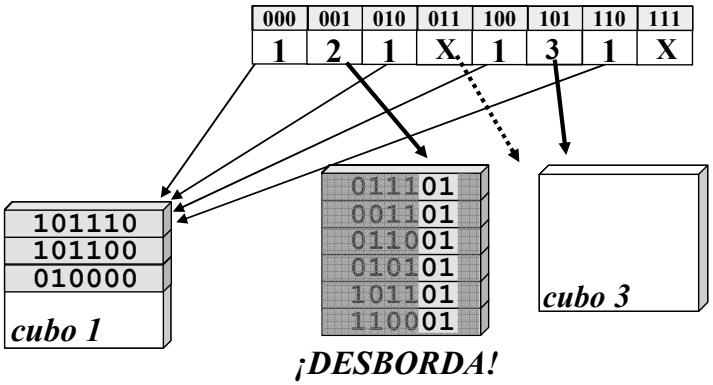
Desborda por primera vez...

Desborda por segunda vez...



Tema 4.3.7: Disp. Virtual Extensible
(basada en directorio – II)

¿Qué habría que hacer si, al repartir los registros del cubo desbordado, **fuesen todos al mismo cubo?** (coinciden en la nueva terminación)



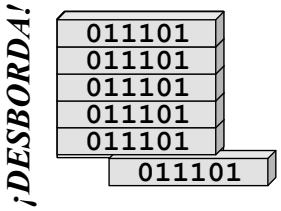
PERO...
¡¡NO SE REPARTEN!!

Pasos:

- **Anular el puntero** de la dirección virtual sin elementos (asignar a NIL)
- **Repetir el proceso** provocado por el desbordamiento (copio, desdoblo, ...)

Tema 4.3.7: Disp. Virtual Extensible
(basada en directorio – II)

¿Qué ocurre si los elementos de un cubo no se pueden repartir nunca?



Claves de Direccionamiento Homónimas:

- **Producen la misma dirección**
- Por más desdoblamientos que se practiquen, ¡ esos elementos nunca se repartirán !

SOLUCIÓN:

- **Política secundaria de Gestión de Desbordamientos**
- Antes de realizar una extensión se comprobará que es útil (que distribuye)
- Si no es así, se aplicará la política secundaria (encadenamiento de un cubo de desbordamiento, aumento dinámico del espacio de cubo, etc.)
- Esta técnica también puede utilizarse para retrasar desdoblamientos

Tema 4.3.7: Disp. Virtual Extensible
(basada en directorio – III)

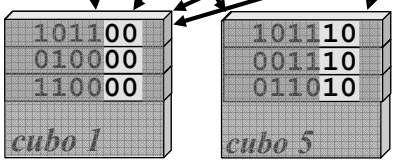
Desborda por tercera vez...

000	001	010	011	100	101	110	111
1	2	1	3	1	4	1	3

Pero si ahora desborda el cubo real 1... **¡no necesito duplicar el directorio!**

Sólo necesito desdoblar el cubo 1...

000	001	010	011	100	101	110	111
1	2	5	3	1	4	5	3



Y si vuelve a desbordar el cubo 1... tampoco hay que duplicar el directorio

Sólo hay que desdoblar el cubo 1...

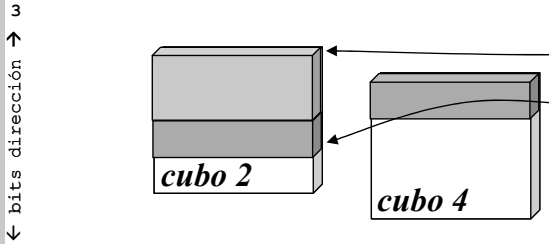
000	001	010	011	100	101	110	111
1	2	5	3	6	4	5	3

Tema 4.3.7: Disp. Virtual Extensible
(basada en directorio – y IV)

• **¿Qué hacer si, al operar el archivo, sucede que 2+4 caben en un cubo?**

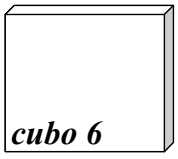
Combinación: dos cubos pasan a ser uno:

000	001	010	011	100	101	110	111
1	2	5	3	6	2	5	3



→ **puedo liberar el cubo 4**
(espacio libre para reutilizar)

• **¿Y qué deberá ocurrir si se vacía el cubo 6? …**



→ En primer lugar, **puedo liberar ese cubo (6)**
→ Después, **se elimina el apuntamiento (NIL)**
→ Finalmente, **ambas mitades del directorio son idénticas → *plegado***

2 **Plegado**: prescindir de un bit de dirección y de la mitad del directorio (copia)

00	01	10	11
1	2	5	3

Tema 4.3.7: Disp. Virtual Extensible

- Se va a considerar un espacio de direccionamiento máximo, y un subespacio actual (menor que el máximo) de 2^j direcciones (de j bits)
- Se creará un directorio de 2^j punteros a bloque, cada uno en su dirección (vector de punteros). Cada puntero señalará al bloque que contiene el cubo (correspondiente a la dirección donde se hallaba el puntero).
- Se van utilizando sólo los *j* últimos bits de la dirección de un dato.
- Algoritmo de localización:
  - 0 - Se toma la dirección, y se extraen los j últimos bits.
  - 1 - Se mira en el directorio el puntero correspondiente a esa dirección.
  - 2 - FIN (se recupera el bloque apuntado)

Tema 4.3.7: Disp. Virtual Extensible

- ¿Qué ocurre si un cubo desborda?

  I - Se comprueba cuántas direcciones virtuales coinciden en ese cubo
  - 0 *Cont* inicializado a cero. *K* inicializado a j (tamaño dirección virtual)
  - 1 Se cambia el bit *K*-ésimo, y si el puntero es distinto FIN. Se tendrían 2^{cont} direcciones apuntando al mismo bloque.
  - 2 Se incrementa *cont* (1 unidad), se decrementa *K* (1 unidad)
  - 3 Si $K > 0$, volver al paso 1

  II - Si hay varias direcciones para un cubo $(K < j)$, este se separa en dos en base al bit (K+1)-ésimo (el de menor peso que aún coincidía) El bloque apuntado (cubo de datos) habrá de ser duplicado

  III - Si todo el cubo lo ocupa una sola dirección, habrá que desdoblar:

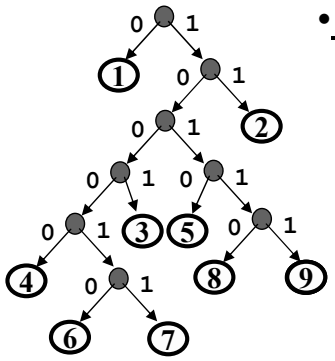
  - Se duplica todo el directorio, y se incrementa *j* en uno (siendo el bit j+1 el que discrimina entre ambas copias del directorio)

Tema 4.3.8: Disp. Virtual Dinámica

- Se va a considerar un espacio de direccionamiento máximo, cuyas direcciones expresadas en binario tendrán n bits.
- Se van utilizando sólo los *k* primeros bits de esa dirección. A medida que se van necesitando, se usan más bits o menos:
  - Inicialmente se tiene un cubo de datos, y no se necesita ningún bit.
  - Cuando este desborda, se tendrán dos cubos, y se usará un bit (el primero de la dirección) para dispersar los datos en esos cubos
- Esta organización se apoya en un directorio estructurado en forma de árbol binario, con dos tipos de nodos:
  - hojas: tienen el puntero al bloque que contiene ese cubo
  - internos: tienen dos punteros, uno para bit 0 y otro para bit 1

Tema 4.3.8: Disp. Virtual Dinámica

- Algoritmo de Localización:
  0 - Se toma el primer nodo y el primer bit de la dirección
  1 - Si es hoja, FIN (tenemos el puntero externo, la dirección del cubo)
  2 - Si no es hoja, si el bit es 0 tomar el nodo izquierdo, y si es 1 el derecho
  3 - Tomar el siguiente bit de la dirección y volver al paso 1.



- **Ejemplos**: *Sobre el directorio de la izquierda...*
  - ¿En qué cubo está el dato cuya dirección es: **0 0 1 1 0 1 0 1 0 0 1 1** ?
  - ¿En qué cubo está el 1 0 0 1 1 0 1 0 1 0 0 1 ?
  - ¿Y dónde debo almacenar un elemento cuya dirección es 1 0 0 0 1 1 0 1 1 1 0 1 ?
    - ¿Y qué pasa si este último desborda?

Tema 4.5: Comparativa

Organizaciones Seriales: (más utilizadas de lo que pueda parecer)

Adecuadas para: la inserción y procesos de consulta a la totalidad

Factores que se optimizan:

- espacio de almacenamiento (y tiempo en un *full-scan*)
- programación simple

Ventajas	*Inconvenientes*
Sencillez (diseño & implementación)	Rigidez
Permite fich. variables y heterogéneos	Tiempos de respuesta elevados
Permite búsquedas complejas	Algunos procesos precisan orgs. auxiliares
Válida en cualquier soporte	Reorganización pesada (compactación)

Tema 4.5: Comparativa

Organizaciones Secuenciales:

Tipo de proceso: - los mismos que para orgs. seriales, y alguno más:

- mejora la localización de registros (pero sólo para una clave)
- los procesos ordenados (por esa clave) también mejoran

Factores que se optimizan:

- espacio de almacenamiento y sencillez de programación
- tiempo de acceso (con respecto al acceso serial, y para pocos procesos)

Ventajas	*Inconvenientes*
Casi todas las de la org. serial	Casi todos los de la org. serial
El ordenamiento físico	Precisa soporte secuencial (o direccionado)
Mejora algunos procesos	Mejora pocos procesos (en núm.)
	Degenera (necesita reordenaciones)

Tema 4.5: Comparativa

Consecutividad vs. No Consecutividad

- Las organizaciones consecutivas aprovechan mejor el espacio, y serán más convenientes si proliferan los procesos a la totalidad
- Por otro lado, un registro cuya localización se conoce (a través de un índice o por un vínculo entre archivos) puede estar partido entre dos bloques y requerir dos accesos.
- Para organizaciones fuertemente indizadas → no consecutividad
- La no consecutividad conlleva localización del primer registro completo almacenado y espacio libre distribuido, lo que la hace habitual de las organizaciones secuenciales.

Proceso Predominante	*Interesa …*
Consulta a la Totalidad	Consecutiva
Acceso por la dirección (Indizado)	No consecutiva

Tema 4.5: Comparativa

Organizaciones Direccionadas

Tipo de proceso: de selección a través de una clave

Rasgos: clave de direccionamiento, algoritmo de transformación, tamaño del espacio de direccionamiento, tamaño del cubo, gestión desbordamientos

Factores que se optimizan: tiempo de respuesta (aprox. 1 acceso a disco)
(pero sólo para acceder mediante la CD)

Ventajas	*Inconvenientes*
Poco almacenamiento auxiliar	Mala ocupación (baja densidad)
Acceso inmediato (óptimo)	... ¡pero sólo para un proceso!!
Excelente si no sufre actualizaciones	Alta degeneración: desbordamientos,... Mantenimiento muy costoso
	Mala localización por cl. alternativas

Tema 4.5: Comparativa

Organizaciones Direccionadas Avanzadas

Tipo de proceso: de selección a través de una clave

Rasgos: los de los direccionamientos

Factores que se mejoran (con respecto al resto de direccionamientos):

- densidad (disp. virtual) y consulta por cl.alternativas
- tiempo de acceso en consultas

Ventajas	*Inconvenientes*
Mejor densidad (disp. virtual)	Precisa alm. auxiliar (en M_{int})
Mejora acceso por cl. alternativas	Acceso por cl. alternativas
Localización por CD en 1 acceso	Desdoblamientos y Plegados
Mejora acceso por cl. alternativas	Cl. alternativas precisan org. auxiliar
Minimiza impacto mala dispersión (disp.dinámica)