Práctica 1: Ficheros

Semanas		Temas
29/09/2008	SESIÓN 1	Diseño físico y lógico inicial, y optimizados
06/10/2008	SESIÓN 2	Organización serial no consecutiva
13/10/2008	SESIÓN 3	Consultas sobre organización serial no consecutiva
20/10/2008	SESIÓN 4	Direccionamiento
27/10/2008	SESIÓN 5	Desbordamiento
03/11/2008	SESIÓN 6	Consulta sobre organización direccionada
10/11/2008	SESIÓN 7	Índices
17/11/2008	SESIÓN 8	Consulta sobre organización indizada + direccionada

Práctica 1:
Optimización en Tiempo de Acceso

- De sesiones anteriores ya se tiene:
  - Optimización físico-lógica del fichero inicial
  - Organización serial no consecutiva
  - Búsquedas seriales sobre el fichero solución
- Ahora, se organizará el fichero de forma direccionada para optimizar el número necesario de accesos de un proceso de consulta.

Práctica 1: 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 (duro)

Instrucciones: acceso aleatorio a bloques

leer(*n*) / **escribir**(*n*)

- procesos selectivos: localización inmediata → **óptimos**
  - la localización se hará mediante la clave de direccionamiento
  - es necesario hacer corresponder la CD con la dirección física
- procesos a la totalidad: la localización no es una ventaja (al contrario)

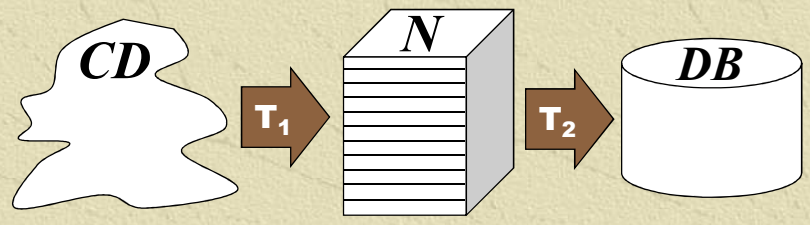
Práctica 1:
Organización Física de los Registros

La clave de direccionamiento no suele coincidir con la dirección física

Es necesario transformarla

T1: clave direccionamiento → espacio direccionamiento

T2: espacio direccionamiento → dirección base (física)



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

Práctica 1: Organización Direccionada

- La manera más eficiente de encontrar un registro a través de una clave es que la posición física de este registro (con respecto al total de registros) pueda deducirse a partir de esa clave
- Se trata de aplicar una función que asigne a cada CD una dirección relativa (función de transformación)
- Esta función debe 'repartir' lo mejor posible los registros en todo el espacio de direccionamiento.
  - no debe formar cúmulos
  - no debe dejar cubos vacíos
  - debe repartir 'uniformemente'...

Práctica 1: Organización Direccionada Dispersa

- Cuando cada elemento tiene su dirección 'reservada', se trata de un direccionamiento directo.

Problemas:

- es difícil encontrar funciones de transformación adecuadas
- si algún valor de CD no ocurre, deja huecos → baja densidad (ejemplo: CD = DNI, fichero de alumnos, ¿cuántos huecos?...)
- pésima eficiencia en acceso por claves alternativas (¡¡serial...!!)

- Cuando distribuimos sobre un área registros, permitiendo colisiones (varios registros en la misma dirección) se trata de un direccionamiento disperso.
  (ejemplo: tomar los dos últimos dígitos del DNI)

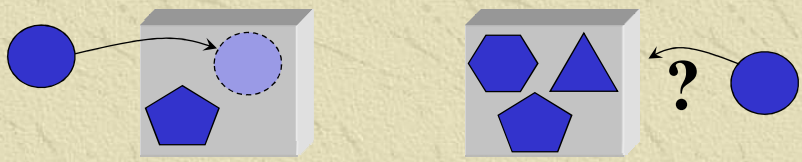
Ventajas:

- espacio direccionamiento reducido → densidad mayor
- Reducción drástica de los cubos vacíos

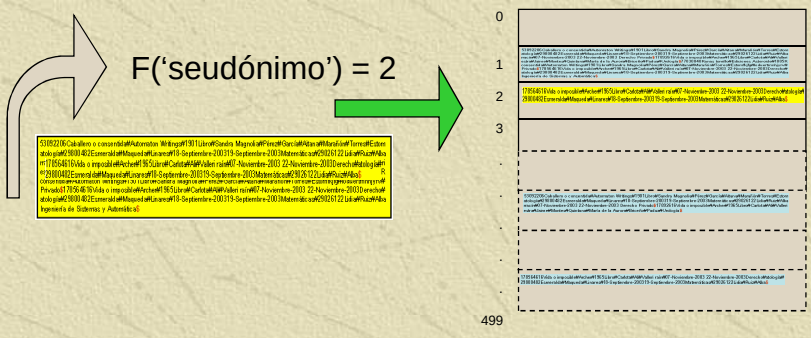
Práctica 1:
Parámetros de la Organización

- Para la consulta se utilizará una organización **direccionada dispersa**
- La clave de direccionamiento será el '**seudónimo**'
- La densidad de ocupación deberá superar el 50%

- **Colisión**: al almacenar un registro, el cubo está ocupado
- **Desbordamiento**: un registro colisiona, **y no cabe**



Práctica 1:
Almacenamiento Direccionado



Práctica 1: Funciones de Transformación

Funciones de Transformación

1. Truncamiento:
115279 → 279

2. División-Resto o Residuo:
115279 MOD N / N área del direccionamiento

3. Plegado:
115279 → 115 | 279 → 115 + 279 = 394

4. Cambio de Base:
525 MOD 11 = 8; 525 DIV 11 = 47; 47 MOD 11 = 3; 4 MOD 11 = 4 → 438

5. Método de Lin:
p=11, q=7, n=2, CD 95; → $9*11^1 + 5*11^0 = 104$ → 104 DIV $7^2 = 2$

Práctica 1: Colisiones

- Para saber rápidamente si un elemento colisiona:
  - se almacena en cada cubo una marca de *primera posición libre* (2 bytes)
  - si la *ppl* = 0 → el cubo está vacío
    - almacenar registro desde el principio del cubo
    - actualizar *ppl* := *<tamaño_registro>*
  - else, si *ppl* + *<tamaño_registro>* menor o igual que 1022
    - almacenar registro desde el byte de posición *ppl*
    - actualizar *ppl* := *ppl* + *<tamaño_registro>*
  - else, el **registro desborda** (de momento, no hacemos nada)