

# Fundamentos de Organización de Datos

## Clase 3

# Agenda

## Viaje del byte

## Tipos de archivo

- Secuencia de Bytes
- Registros / campos longitud predecible
- Registros / campos sin longitud predecible

## Claves

- Primaria
- Candidata
- Secundaria

## Eliminación

- Recuperación de espacio
- Reg. Long Variable
- Eliminación

# Archivos → Introducción

La **memoria primaria (RAM)** es rápida y de simple acceso, pero su uso tiene algunas desventajas respecto al almacenamiento secundario:

- Capacidad limitada
- Mayor costo
- Es volátil

# Archivos → Introducción

**Almacenamiento secundario** necesita más tiempo para tener acceso a los datos que en RAM

- Su acceso es tan “lento” que es imprescindible enviar y recuperar datos con inteligencia
- Al buscar un dato, se espera encontrarlo en el primer intento (o en pocos)
- Si se buscan varios datos, se espera obtenerlos todos de una sola vez

La información está organizada en **archivos**

- **Archivo:** colección de bytes que representa información

# Archivos → Viaje de un Byte

## Archivo Físico

- Archivo que existe en almacenamiento secundario
- Es el archivo tal como lo conoce el S.O. y que aparece en su directorio de archivos

## Archivo Lógico

- Es el archivo, visto por el programa
- Permite a un programa describir las operaciones a efectuarse en un archivo,
- No se sabe cual archivo físico real se utiliza o donde esta ubicado

# Archivos – Viaje de un byte

- Viaje de un byte ➡ No es sencillo
  - Escribir un dato en un archivo
    - Write ( archivo, variable) ➡ ciclos para escribir
- Quienes están involucrados
  - Administrador de archivos
  - Buffer de E/S
  - Procesador de E/S
  - Controlador de disco

# Archivos – Viaje de un byte

**Administrador de archivos:** conjunto de programas del S.O. (capas de procedimientos) que tratan aspectos relacionados con archivos y dispositivos de E/S

- **En Capas Superiores:** aspectos lógicos de datos (tabla)
  - Establecer si las características del archivo son compatibles con la operación deseada (1)
- **En Capas Inferiores:** aspectos físicos (FAT)
  - Determinar donde se guarda el dato (cilíndro, superficie, sector) (2)
  - Si el sector está ubicado en RAM se utiliza, caso contrario debe traerse previamente. (3)

# Archivos – Viaje de un byte

**Buffers de E/S:** agilizan la E/S de datos.

- Manejar buffers implica trabajar con grandes grupos de datos en RAM , para reducir el acceso a almacenamiento secundario

**Procesador de E/S:** dispositivo utilizado para la transmisión desde o hacia almacenamiento externo. Independiente de la CPU. (3)



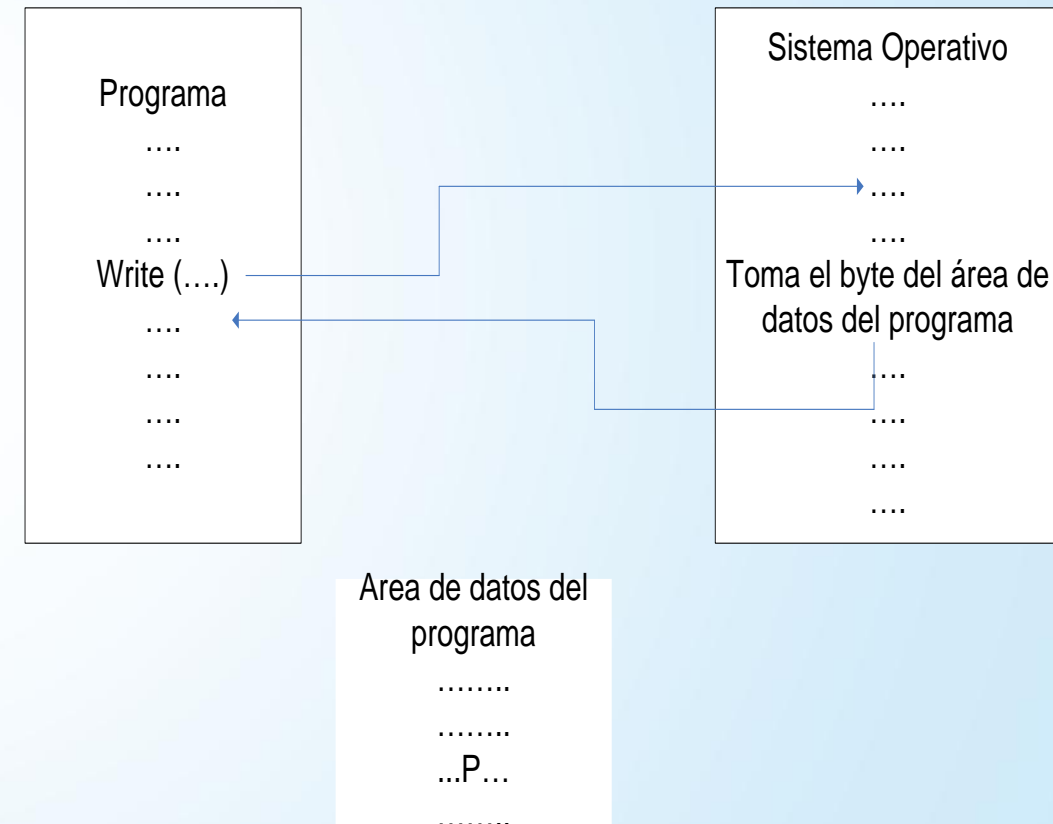
# Archivos – Viaje de un byte

**Controlador de disco:** encargado de controlar la operación de disco.

- Colocarse en la pista
- Colocarse en el sector
- Transferencia a disco

# Archivos – Viaje de un byte

- Qué sucede cuando un programa escribe un byte en disco?
  - Operación
    - Write(.....)
  - Veamos los elementos que se involucran en esta simple operación
  - Supongamos que se desea agregar un byte que representa el carácter 'P' almacenado en una variable c de tipo carácter, en un archivo denominado TEXTO que se encuentra en algún lugar del disco rígido.



# Archivos – Viaje de un byte

11

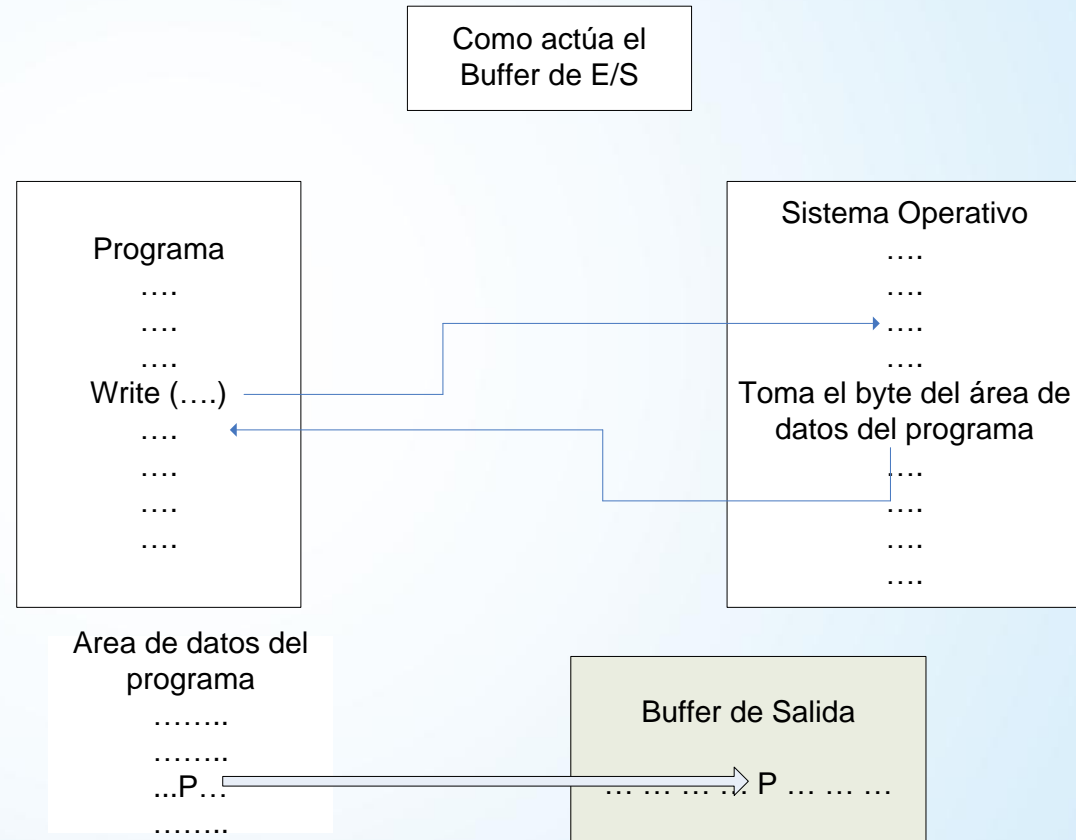
## Capas del protocolo de transmisión de un byte

- El Programa pide al **S.O.** escribir el contenido de una variable en un archivo
- El **S.O.** transfiere el trabajo al **Administrador de archivos**
- El **Adm.** busca el archivo en su tabla de archivos y verifica las características
- El **Adm.** obtiene de la FAT la ubicación física del sector del archivo donde se guardará el byte.
- El **Adm** se asegura que el sector del archivo está en un **buffer** y graba el dato donde va dentro del sector en el **buffer**
- El **Adm.** de archivos da instrucciones al **procesador de E/S** (donde está el byte en RAM y en que parte del disco deberá almacenarse)
- El **procesador de E/S** encuentra el momento para transmitir el dato a disco, la CPU se libera
- El **procesador de E/S** envía el dato al **controlador de disco** (con la dirección de escritura)
- El **controlador** prepara la escritura y transfiere el dato bit por bit en la superficie del disco.

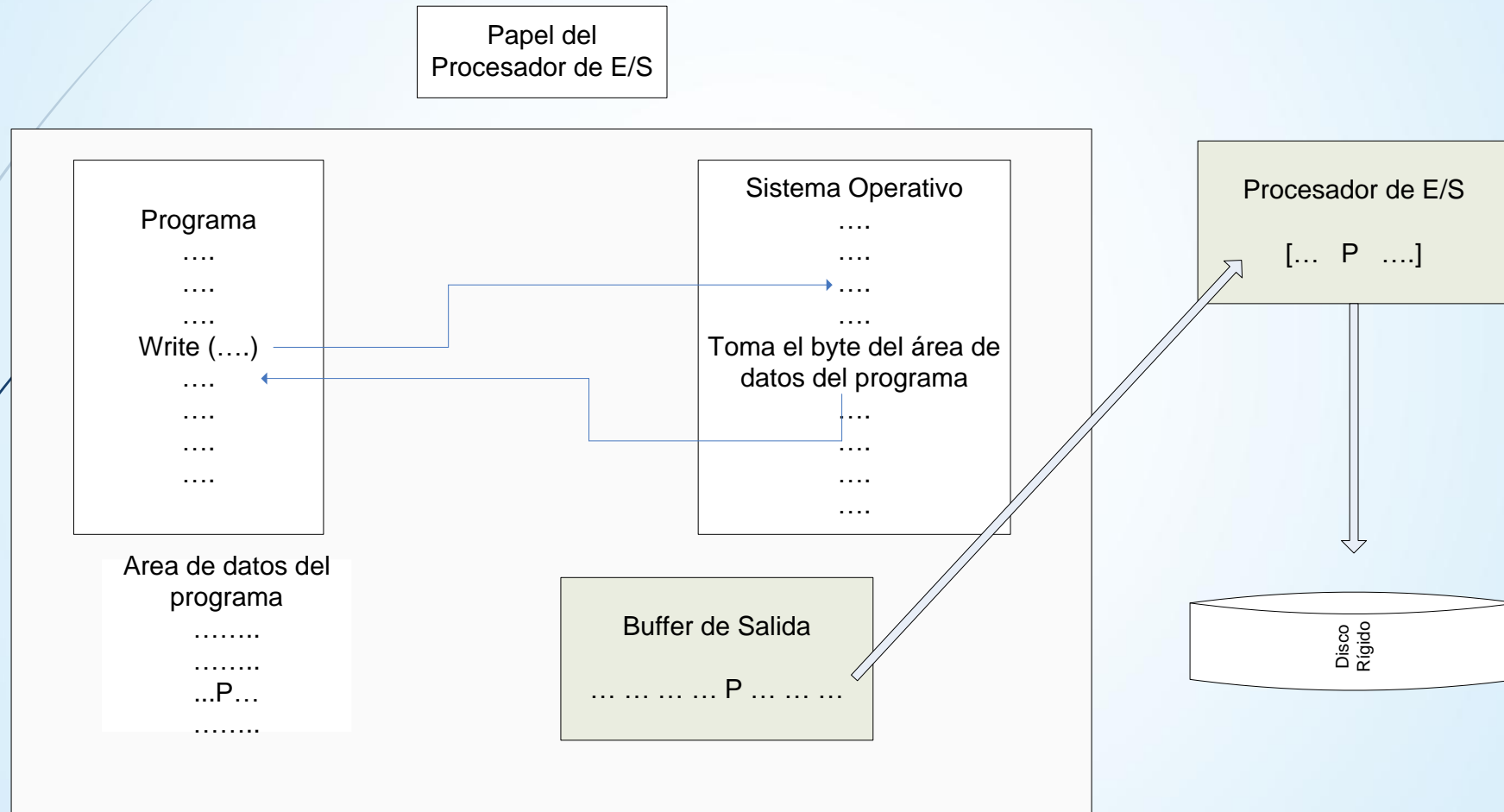
# Archivos – El viaje de un Byte

## Tabla

Nombre	Abrió	Acceso	Propietario	Protección
Archivo a	Perez	L/E	Gomez	prop:L/E otro: L/E
Archivo b	García	L	García	prop:L/E otro: L
Archivo c	Gomez	E	omez	prop:L/E otro: E

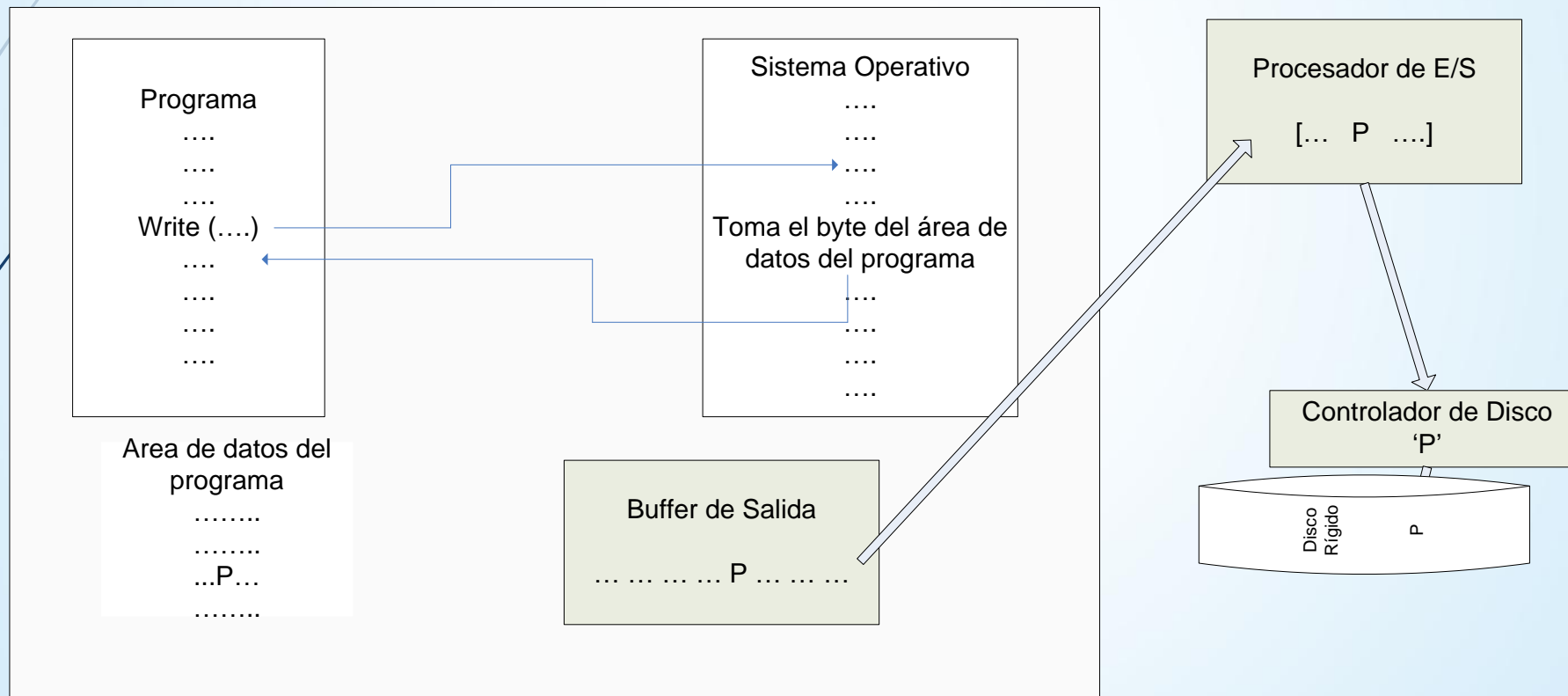


# Archivos – El viaje de un Byte



# Archivos – El viaje de un Byte

## Controlador de disco



# Archivos → Tipos de Archivo

## Archivos como Secuencia de bytes

- No se puede determinar fácilmente comienzo y fin de cada dato.
- Ejemplo: archivos de texto

## Archivos estructurados

- **Registros**
  - Longitud fija o variable
- **Campos**
  - Longitud fija o variable

# Archivos → Tipos de Archivo

## Campos

- Unidad lógicamente significativa más pequeña de un archivo. *Permite separar la información*
- Identidad de campos: variantes, pro y contras.
  - **Longitud predecible** (long. Fija), desperdicio de espacio, si el tamaño es pequeño al agrandarlo se podría desperdiciar más espacio)
  - **Indicador de longitud** (al ppio de cada campo)
  - **Delimitador al final de cada campo** (carácter especial no usado como dato)



# Archivos → Tipos de Archivo

## Registros

- Organización de registros
- **Longitud predecible** (en cant. de bytes o cant. de campos)
  - Campos fijos o variables
- **Longitud variable**
  - **Indicador de longitud** (al comienzo, indica la cant. de bytes que contiene)
  - **Segundo archivo** (mantiene la info de la dirección del byte de inicio de cada registro)
  - **Delimitador** (carácter especial no usado como dato)
- **Estudio de casos:** ventajas y desventajas

# Archivos → Claves

## Clave

- Se concibe al Registro como la cantidad de información que se lee o escribe
- **Objetivo:** acceder sólo un registro específico
- Es conveniente identificar una registro con una llave o clave que se base en el contenido del mismo

# Archivos → Claves

## Clave

- Permite la identificación del registro
- Deben permitir generar orden en el archivo por ese criterio

## Únivoca / Primaria:

- Identifican un elemento particular dentro de un archivo

## Secundaria

- Generalmente no identifican un único registro

# Archivos → Claves

**Forma canónica:**  
forma  
estándar  
para una  
llave, puede  
derivarse a  
partir de  
reglas bien  
definidas.

- Representación única para la llave, ajustada a la regla
  - Ej: Clave sólo con letras mayúsculas y sin espacios al final.
- Al introducir un registro nuevo:
  - 1ro se forma una llave canónica para ese registro
  - 2do se la busca en el archivo. Si ya existe, y es unívoca → no se puede ingresar

# Archivos → Claves (performance)

## Estudio de performance

- Punto de partida para futuras evaluaciones
- Costo: acceso a disco, N° de comparaciones
- Caso promedio

# Archivos → Claves (performance)

## En el caso secuencial

- Mejor caso: leer **1** reg. , peor caso leer **n** registros
- Supongamos que tenemos 1000 registros, buscar uno en particular mejor caso 1, peor caso 1000, promedio 500, en realidad el mejor caso es 0, el buffer puede estar en memoria.
- Promedio:  $n/2$  comparaciones
- Es de  $O(n)$ , porque depende de la cantidad de registros
- **Lectura de Bloques de registros**
  - mejora el acceso a disco,
  - no varían las comparaciones.

# Archivos → Claves (performance)

## Acceso directo

- Permite acceder a un registro preciso
- Requiere una sola lectura para traer el dato [  $O(1)$  ].
- Debe necesariamente conocerse el lugar donde comienza el registro requerido

## Número relativo de registro (NRR):

- Indica la posición relativa con respecto al principio del archivo
- Solo aplicable con registros de longitud fija)
  - Ej. NRR 546 y longitud de cada registro 128 bytes →  
distancia en bytes=  $546 * 128 = 69.888$

# Archivos → Claves (performance)

El acceso directo es preferible sólo cuando se necesitan pocos registros específicos, pero este método NO siempre es el más apropiado para la extracción de información.

- Ej. generar cheques de pago a partir de un archivo de registros de empleados.
- Como todos los reg. se deben procesar → es más rápido y sencillo leer registro a registro desde el ppio. hasta el final, y NO calcular la posición en cada caso para acceder directamente.



# Archivos → diferentes visiones

Forma de  
acceso

Cantidad  
de cambios

## Forma de acceso

- **Serie:** cada registro es accesible solo luego de procesar su antecesor, simples de acceder
- **Secuencial:** los registros son accesibles en orden de alguna clave
- **Directo:** se accede al registro deseado

# Archivos → Tipos

## # de Cambios

- **Estáticos** -> pocos cambios
  - Puede actualizarse en procesamiento por lotes
  - No necesita de estructuras adicionales para agilizar los cambios
- **Volátiles** -> sometido a operaciones frecuentes:
  - Agregar / Borrar / Actualizar
  - Su organización debe facilitar cambios rápidos
  - Necesita estructuras adicionales para mejorar los tiempos de acceso

# Archivos → Operaciones

Altas

Bajas

Modificaciones

Consultas

Como influye  
registros de  
long. Fija y  
variable

# Archivos → eliminación

## Eliminar registros de un archivo

- **Baja Lógica**
- **Baja Física**
  - Cuales son las diferencias?
  - Cuales las ventajas y desventajas?

# Archivos → eliminación

30

- **Registro de longitud fija:** agregar o modificar, sin inconvenientes
- **Registros de longitud variable:** problemas
  - Ej: Intentar modificar un registro, tal que el modificado quede de mayor tamaño
  - Soluciones posibles:
    - Agregar los datos adicionales al final del archivo (con un vínculo al registro original) → complica el procesamiento del registro.
    - Reescribir el registro completo al final del archivo → queda un espacio vacío (desperdiciado) en el lugar origen
  - La operación agregar no genera inconvenientes.
- **Nos centralizaremos en la eliminación**

# Archivos → eliminación

## Baja Lógica

- Cualquier estrategia de eliminación de registros debe proveer alguna forma para reconocerlos una vez eliminados (**ejemplo: colocar una marca especial en el reg. eliminado**).
- Con este criterio se puede anular la eliminación fácilmente.
- Cómo reutilizar el espacio de registros eliminados ?
- Los programas que usan archivos deben incluir cierta lógica para ignorar los registros eliminados

# Archivos → eliminación

## Baja Física → Compactación

- Recuperar el espacio
- La forma más simple es copiar todo en un nuevo archivo a excepción de los registros eliminados → **Baja Física**
- Frecuencia
  - Tiempo (depende del dominio de aplicación)
  - Ante la necesidad de espacio
- Veremos el análisis de recuperación dinámica del almacenamiento



# Archivos → eliminación

## Aprovechamiento de espacio

- **Reg. longitud fija → es necesario garantizar:**
  - Marca especiales en los reg. borrados → **Baja Lógica**
- **Reg. longitud variable → los nuevos elementos deben “caber” en el lugar**

# Archivos → eliminación

Recuperación del espacio para su reutilización cuando se agreguen registros

- **Búsqueda secuencial** -> usa las marcas de borrado.
  - Para agregar, se busca el 1º reg. eliminado. Si no existe se llega al final del archivo y se agrega allí.
  - Es muy lento para operaciones frecuentes.
- **Es necesario**
  - Una forma de saber de inmediato si hay lugares vacíos en el archivo
  - Una forma de saltar directamente a unos de esos lugares, en caso de existir

# Archivos → eliminación

## Aprovechamiento de espacio (reg. long. fija)

- **Recuperación de espacio con Lista o pilas (header)**
  - Lista encadenada de reg. disponibles.
  - Al insertar un reg. nuevo en un archivo de reg. con long. fija, cualquier registro disponible es bueno.
  - La lista NO necesita tener un orden particular, ya que todos los reg. son de long. fija y todos los espacios libres son iguales

# Archivos → eliminación

## Aprovechamiento de espacio (reg. long. fija)

- **Recuperación de espacio con Lista o pilas (header)**
  - Ej : en el encabezado estará NRR 4, el archivo tendrá
    - **alfa    beta    delta    \* 6    gamma    \* -1    epsilon**
  - Se borra beta, como inicial quedará 2
    - **alfa    \* 4    delta    \* 6    gamma    \* -1    epsilon**
  - Si se quiere agregar un elemento el programa solo debe chequear el header y desde ahí obtiene la dirección del primero. Agrego omega , como ppio queda 4 nuevamente
    - **alfa    omega    delta    \* 6    gamma    \* -1    epsilon**

# Archivos - Eliminación

37

- **Aprovechamiento de espacio**
  - **Recuperación de espacio con reg. de longitud variable**
    - Marca de borrado al igual que en reg. de long. fija (ej:\*)
    - El problema de los registros de longitud variable está en que no se puede colocar en cualquier lugar, para poder ponerlo debe caber, necesariamente.
    - Lista . No se puede usar NRR como enlace. Se utiliza un campo binario que explícitamente indica en enlace (conviene que indique el tamaño).
    - Cada registro indica en su inicio la cant. de bytes.

# Archivos - Eliminación

38

## ➤ Aprovechamiento de espacio

### ➤ Recuperación de espacio con reg. de Longitud variable

- Reutilización: buscar el registro borrado de tamaño adecuado (lo suficientemente grande).
- Como se necesita buscar, no se puede organizar la lista de disponibles como una pila.
- El tamaño "adecuado" del primer registro borrado a reutilizar ->origina Fragmentación

# Archivos - Eliminación

39

## ➡ Aprovechamiento de espacio → **Fragmentación**

➡ **Interna:** ocurre cuando se desperdicia espacio en un registro, se le asigna el lugar pero no lo ocupa totalmente.

➡ Generalmente ocurre con **Reg. longitud fija**

➡ Reg. longitud variable evitan el problema

➡ **Espacio asignado → No ocupado**

➡ **Externa:** ocurre cuando se desperdicia espacio entre registros

➡ Generalmente ocurre con **Reg. longitud variable**

➡ **Espacio no asignado → No ocupado**

# Archivos - Eliminación

- Estrategias de colocación en registros de longitud variable:
  - Primer ajuste
  - Mejor ajuste
  - Peor ajuste



# Archivos - Eliminación

- **Primer ajuste:** se selecciona la primer entrada de la lista de disponibles, que pueda almacenar al registro, y se le asigna al mismo. Asigna el espacio completo (por definición)
  - Minimiza la búsqueda
  - No se preocupa por la exactitud del ajuste
- **Mejor ajuste:** elige la entrada que más se aproxime al tamaño del registro y se le asigna completa.
  - Exige búsqueda
- **Peor ajuste:** selecciona la entrada más grande para el registro, y se le asigna solo el espacio necesario, el resto queda libre para otro registro

# Archivos - Eliminación

## ➔ Conclusiones

- ➔ Las estrategias de colocación tienen sentido con reg. de longitud variable
- ➔ Primer ajuste: más rápido. Puede generar fragmentación interna (porque se le asigna todo el espacio, por definición)
- ➔ Mejor ajuste: Puede generar fragmentación interna (porque se le asigna todo el espacio, por definición)
- ➔ Peor ajuste: Puede generar fragmentación externa. No genera Fragmentación interna por definición

# Archivos - Operaciones

## ➤ Modificaciones

### ➤ Consideraciones iniciales

#### ➤ Registro de long. Variable, se altera el tamaño

- Menor, puede no importar (aunque genere fragmentación interna o externa)
- Mayor, no cabe en el espacio

### ➤ Otros problemas

- Agregar claves duplicadas, y luego se modifica
- Cambiar la clave del registro (que pasa con el orden)