Unidad 10

ARCHIVOS

Objetivos

- Entender el almacenamiento externo como medio para guardar grandes cantidades de datos de forma permanente.
- Distinguir las ventajas e inconvenientes de las estructuras externas (ficheros), frente a las internas.
- Conocer y comprender el concepto de archivo o fichero.
- Comprender los conceptos relacionados con la nueva estructura de datos.
- Conocer y aplicar las operaciones de manejo de archivos (creación, lectura, escritura,...).
- Conocer las diferencias entre los distintos tipos de organización de archivos.
- Saber diferenciar los distintos modos de acceso a los archivos según su organización.
- Aplicar los conocimientos adquiridos hasta ahora para el diseño y construcción de programas que usan archivos.
- Integrar el uso de archivos en la construcción de aplicaciones.

Contenidos

1. INTRODUCCIÓN

2. GENERALIDADES

3. TIPOS DE FICHEROS

3.1. Según su función

3.2. Según su organización

4. MODOS DE ACCESO

4.1. Acceso secuencial

5. OPERACIONES SOBRE FICHEROS

Creación, Recorrido, ordenación, Actualización, Partición, Fusión, Borrado.

6. OPERACIONES SOBRE REGISTROS

Consulta, Modificación, Supresión, Inserción.

7. FICHEROS DE ORGANIZACIÓN SECUENCIAL

Creación, Recorrido, Consulta, Partición, Fusión Actualización, Actualización interactiva.

8. INCONVENIENTES DE LAS ESTRUCTURAS DE DATOS ARCHIVOS

Introducción

Aplicaciones triviales

- Estructuras de datos arrays, problemas:
  - Limitación en almacenamiento de datos (memoria principal).
  - Introducción de datos cada vez que se ejecuta el programa.
  - Tiempo de vida de los datos limitado (memoria volátil).
- Solución de estos inconvenientes: **ARCHIVOS**
  - Pueden tratar grandes cantidades de datos (memoria secundaria).
  - Almacenamiento permanente (memoria no volátil).

Aplicaciones mucho más complejas

ARCHIVOS o FICHEROS

- **Los *registros* tienen formatos preestablecidos.**
- **Los archivos tienen un número finito e indeterminado de registros**

Definición:

"Es una estructura de datos externa, formada por un conjunto de elementos todos del mismo tipo, organizados en unidades de acceso, llamadas **registros**."

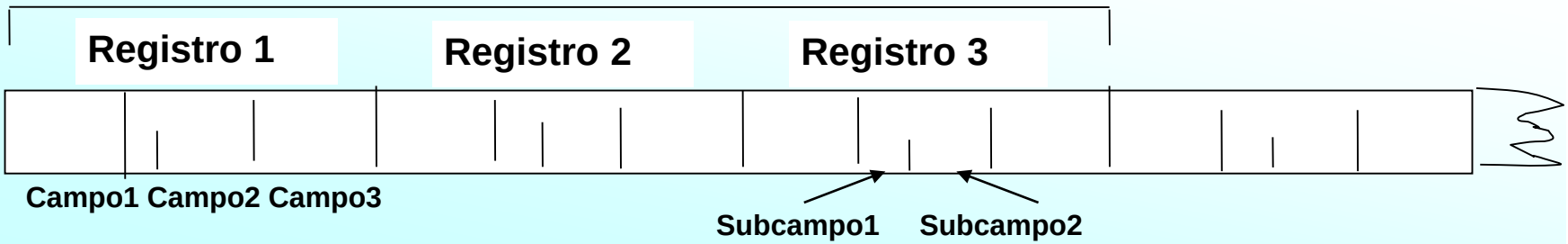
Características:

- Residen en soporte de almacenamiento externo.
- La información se almacena de forma permanente.
- Alta capacidad de almacenamiento de datos.
- **Independencia de la información que guardan respecto a los programas que los usan**.

Contenido de los registros

Puede estar compuesto por:

- **Elementos de datos simples: Datos primitivos**
- **Elementos de datos complejos**
  - **Tradicionales:**
  - **Estructuras**
    - **Campo: Cada uno de los elementos de un registro.**
    - **Subcampo: Cada uno de los elementos de un campo.**
    - **Clave: campo que identifica unívocamente a un registro.**
  - **Objetos**
  - **Atributos**



Longitud de los registros

- **Fija: todos los campos tienen longitud fija.**
  - Por ejemplo: ficheros binarios en Java o C.
- **Variable: algún campo tienen longitud variable.**
  - Por ejemplo: ficheros de texto en Java o C.

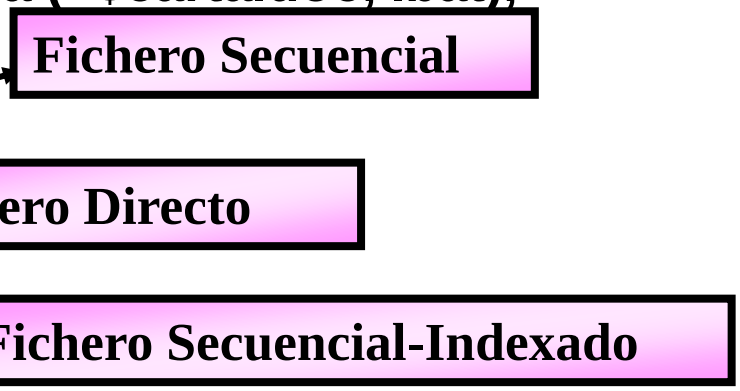
Nombre

***Externo*: nombre con el que lo reconoce el S.O.**

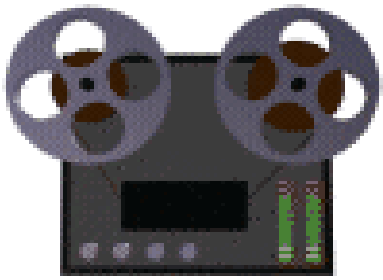
***Interno*: Variable con la que lo referencia el programa que lo usa.**

Tipos de ficheros

- **Según su función o uso que se hace de ellos:**
  - ***Permanentes*: Su información varía poco en el tiempo**
    - ***De situación*: estado actual de la información (F. maestros)**
    - ***De constantes*: Consultas. Baja o nula modificación (Códigos Postales).**
    - ***Históricos*: Nunca se modifican (Rentas de años anteriores)**
  - ***De movimientos*: información para actualizar ficheros permanentes (compra-ventas diarias).**
  - ***De trabajo*: Creados por el sistema (~$carta.doc, .bat), compilador (.bak), ...**
- **Según su organización:**
  - ***Secuencial***
  - ***Directa***
  - ***Secuencial-indexada***



Modos de accesos



- **Acceso secuencial:**
  - Para acceder a un registro en particular deben ser accedidos todos los registros anteriores.
  - Este modo de acceso se permite en soportes direccionables o no.
    - Por ejemplo: Archivo de canciones almacenadas en cinta magnética

      o almacenadas en CD.

- **Acceso directo:**

Modos de accesos

- **Acceso directo:**
  - Permite acceder a un registro directamente, sin tener que acceder antes a todos los registros que le preceden.
  - Sólo se permite este modo de acceso en soportes direccionables, discos duros, pendrive, etc.
  - El acceso puede realizarse bien por el número de registro (o por la clave de acceso si el fichero es de organización directa)..

Operaciones sobre ficheros

Operaciones que atañen al fichero completo

- **Creación:**
  - **Consiste en:**
    - **Darle nombre**
    - **Escribir registros iniciales**
  - **Normalmente se requiere establecer:**
    - **Forma en que se almacena: texto o binario.**
    - **Tipo de organización: Secuencial, Directa o Secuencial-indexada**
    - **Modo de acceso: Secuencial o Directo.**
  - **Los datos pueden venir de teclado, otro fichero o generados mediante proceso.**
  - **Si los datos provienen de teclado**

En java texto o binario

En java no se especifica, son secuenciales

Crear el fichero con programa independiente

Operaciones sobre ficheros

- **Recorrido**:
  - **Lectura secuencial de todos los registros para un determinado proceso.**
- **Ordenación**
- **Actualización**
  - **Proceso para altas, bajas o modificaciones (sincronización de ficheros).**
- **Partición**
  - **Descomponer un fichero en varios.**
- **Fusión**
  - **Obtener un fichero a partir de dos o más ficheros de idéntica estructura.**
- **Borrado**
  - **Eliminación física del fichero.**

Utilización de ficheros

Una vez creado el fichero, para usarlo:

- **Realizar apertura**
  - **Comprobar su existencia y prever caso de error**
  - **La apertura:**
    - **Establece vínculo NombreInterno⇔ NombreExterno.**
    - ***Bloquea* el fichero en exclusividad para su uso.**
    - **Crea *buffer* de memoria intermedia.**
      - **Para comunicar memoria secundaria, central y programa.**
      - **A él se pasa un registro físico en cada acceso.**
    - **Al abrir el apuntador señala al primer registro del fichero y se mueve en cada acceso (registro activo)**
    - **Hay que indicar si se abre para leer, escribir o ambas cosas. (en Java solo leer o escribir)**
- **Procesar**
- **Cerrar**

Utilización de ficheros

- **Procesar:**
  - **Realizar las operaciones previstas.**
- **Cierre:**
  - **Desvincula NombreInterno ⇔ NombreExterno.**
  - **Desbloquea el fichero.**
  - **Vacía el buffer.**
  - **Coloca marca de *fin de fichero*.**

Operaciones sobre registros

- **Consulta:**
  - **Localización de uno o varios registros.**
- **Modificación:**
  - **Cambio de contenido de uno o varios de los campos de un registro que no sean clave.**
  - **Para modificar:**
    - **Lectura previa y Búsqueda.**
    - **Escritura de nuevos datos.**
- **Supresión:**
  - **Por marca**
    - **Poner marca de obsoleto, pero el registro es accesible.**
  - **Supresión real**
    - **Registro inaccesible o eliminación física por sobrescribir otro registro.**
- **Inserción de un registro:** **añadir nuevo registro.**

Ficheros secuenciales

- **Características:**
  - **Pueden crearse en todo tipo de soporte de almacenamiento.**
  - **Los registros se almacenan como una *lista lineal*.**
    - **Ocupan posiciones consecutivas de memoria.**
    - **En orden de entrada.**
  - **El acceso se realiza según el soporte de almacenamiento.**
    - **Si el soporte es direccionable pueden accederse directamente.**

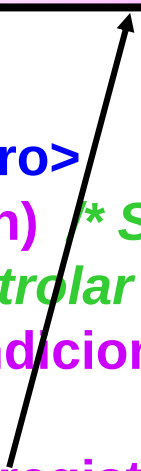
Ficheros secuenciales

- **Ventajas:**
  - **Fácil localización ⇨ Basta conocer campo de búsqueda.**
  - **Eficiente con una tasa de actividad alta.**
  - **Eficaz aprovechamiento del espacio de almacenamiento (sin huecos libres)**
- **Inconvenientes:**
  - **Consulta lenta si son grandes.**
  - **Procesar todo el archivo cualquiera que sea la operación a realizar.**
  - **Uso de ficheros auxiliares para actualizar.**

F.S. Creación

- **Directamente con un editor de texto, si son ficheros de texto.**
- **Diseñando un programa que lo cree (sea o no de texto).**

Similar a ... *escribir (datos)*



```
<Procesar Fichero>
   leer (condicion)  * Se lee la VCB
          para controlar la entrada de datos */
    mientras (condicion sea cierta)
       leer(datos)
       escribir en registro de FICHERO_N (datos)
       leer (condicion);
    fin_mientras
FinPF
```

Pseudocódigo Generalizado

```
<CREAR FICHERO_N>
inicio
   abrir FICHERO_N para escribir
   <Procesar Fichero>
   cerrar FICHERO_N
finCF
```

F.S. Recorrido

- **Acceso secuencial desde el primero al último registro para procesar: Escribir, pintar en pantalla, actualizar, etc.**
- **Para el recorrido se usará un bucle controlado por fin de fichero.**

```
<RECORRIDO>
inicio
    abrir Fichero_N  para.../* ...leer, escribir, según
                               proceso a realizar */

    <ProcesarFichero_N>
    Cerrar Fichero_N
FinR
```

Similar a ... *leer (registro)*

```
<ProcesarFichero_N>
Inicio
   Leer registro de Fichero_N (registro)
   Mientras (no FF)  //FF, fin de fichero
            <Procesar registro>
            Leer registro de Fichero_N (registro)
   Fin mientras
FinPR
```

F.S. Consulta

- **Buscar secuencialmente en el archivo información referente a un registro, según el valor de alguno de los campos.**
- **Si existen varios registros con el mismo valor del campo de búsqueda, se obtendrá el primero de ellos, salvo que se especifique como precondición buscarlos todos.**
- **El archivo puede estar**
  - **Desordenado ➔ hay que llegar al final buscando para afirmar que no existe el registro buscado.**
  - **Ordenado ➔ no es necesario recorrerlo hasta el final para asegurar que el registro buscado no existe.**

F.S. Ordenación

- **HÍBRIDA**
  - **Proceso:**
    - **cargar la información del archivo en un array**
    - **ordenar el array**
    - **volver la información del array ya ordenado al archivo.**
  - **Método aplicable sólo si la totalidad de la información a ordenar cabe íntegramente en la memoria central.**
  - **El array tendrá:**
    - **la misma estructura que los registros del archivo**
    - **y tantas casillas como registros tenga éste.**
  - **Pueden usarse arrays "dinámicos".**
- **EXTERNA**

  Veremos el algoritmo cuando pueda ser implementado en java.
  - **Ordenación por Mezcla directa**

F.S. Ordenación Híbrida

```
<OrdenHibrida FICHERO> >
inicio
    <Calcular NumRegistros del FICHERO>
    pedir memoria para el ARRAY
    <Volcar FICHERO en ARRAY>
    <Ordenar ARRAY>
    <Volcar ARRAY en FICHERO>
    Devolver memoria del ARRAY
Fin Ordenar
```

En Java no se necesita devolver la memoria

```
<Volcar FICHERO en ARRAY>
Inicio
abrir FICHERO para leer
Si no hay errores de apertura
    leer reg de FICHERO (Aux)
    Para (I = 0, mientras (no FF), I = I+1)
        ARRAY [ I ] = Aux
        leer reg de FICHERO (Aux)
    fin_para
    cerrar FICHERO
Finsi
Fin volcar
```

F.S. Partición

- **La operación no afecta al fichero de origen.**
- **Distintos algoritmos de partir ficheros en razón de diferentes especificaciones:**
  - **Partición por contenido.**
    - **Objetivo: "dividir un archivo según una condición de partición que determina que registros se almacenan en un fichero y cuáles en otro".**
    - **Por ejemplo:**
  - **Partición en secuencias.**
    - **Objetivo: "distribuir los registros de un fichero en otros ficheros, obedeciendo a secuencias alternativas de igual o diferentes longitudes"**
    - **Por ejemplo:**

Ver diapositiva sig.

F.S. Ejemplos para partir

- ***Contenido***: **Partir un fichero de nombre *Fichero* en otros dos, *Fichero1* y *Fichero2*, pasando a *Fichero1* los registros de *Fichero* cuyo valor del *CampoRegistro* sea igual al valor prefijado de *Campo* y al *Fichero2* los registros que no cumplan esta condición.**
- ***Secuencias***: **Partir *Fichero* en *Fichero1* y *Fichero2*, con longitud de secuencias de distribución de registros igual a 1, es decir, que pasen a *Fichero1* los registros que ocupan las posiciones pares y a *Fichero2* los que ocupan posiciones impares.**

Ver ejemplos apuntes

F.S. Fusión

- **Objetivo: agrupar en un sólo fichero los registros de otros archivos.**
- **La operación no afecta a los ficheros de origen.**
- **El orden de origen puede o no ser mantenido y el destino puede quedar ordenado o no:**
  - **Archivos de origen desordenado.**
    - **Objetivo: "almacenar secuencias de registros, de longitud determinada, de cada archivo origen en el destino (queda ordenado o no)".**
    - **Por ejemplo: Sean *Fichero1* y *Fichero2* archivos desordenados a fusionar, *Fichero* el destino, también desordenado y *Longitud* el tamaño de las secuencias.**
  - **Archivos de origen ordenado.**
    - **Objetivo: "Fusionar conservando el orden"**

Ver ejemplos apuntes

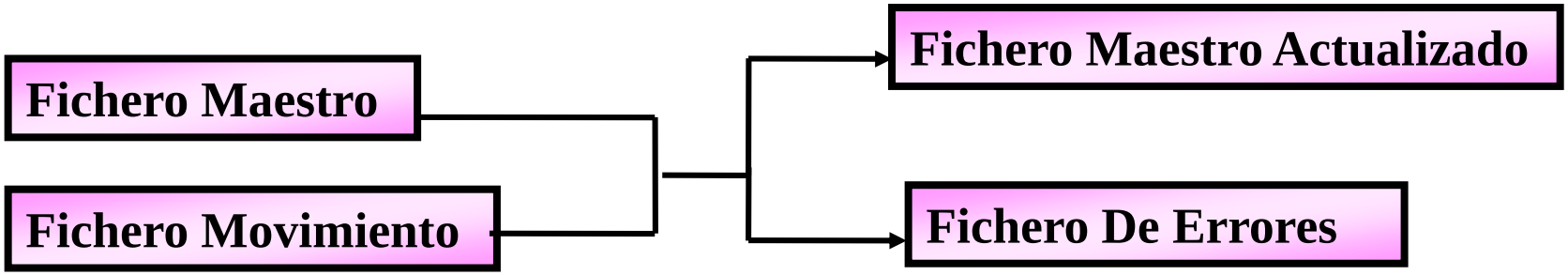
Mezcla controlada por FF

F.S. Actualización

- **Objetivo: Poner al día los ficheros de situación (maestros) para que reflejen fielmente el nuevo estado actual de los datos.**
- **La actualización pueden ser:**
  - **Interactiva:**
    - **Transacciones suministradas en línea directamente al fichero de situación.**
    - **Método poco eficiente, debido a la propia organización del fichero secuencial.**
  - **A través de ficheros de movimientos:**
    - **Transacciones almacenadas en ficheros durante un periodo que fija la política de la empresa.**

F.S. Actualización

- **Operaciones que se pueden realizar**
  - **Altas**
    - **Incluir nuevos registros**
  - **Bajas**
    - **Suprimir algún registro.**
  - **Modificaciones**
    - **Cambiar el contenido de algún registro.**
- **Ficheros que intervienen**
  - **Todos los ficheros deben estar ordenados con el mismo criterio.**



F.S. Act. Descripción del proceso

- **Especificaciones:**
  - El valor del campo clave de ordenación es único.
  - Cada registro tendrá como mucho un movimiento o ninguno.
- **Proceso:**
  - Recorrer simultáneamente maestro y movimiento.
  - Leer inicialmente de ambos.
  - Comparar campos clave
  - Analizar situación
  - Aplicar tratamiento particular según **situación**.
  - Leer del que corresponda.
- **Situaciones:**
  - 2^n-1 (*n* número de ficheros a emparejar)
  - Para dos ficheros
    - $(2^2-1) = 3$ situaciones

- Registro de Movimiento con registro en Maestro
- Registro de Movimiento sin registro en Maestro
- Registro de Maestro sin registro en Movimiento

Situaciones de emparejamiento

- **Registro de Movimiento con registro de maestro**
  - **Si es modificación (caso normal)**
    - **Escribir en maestro actualizado modificando los campos correspondientes.**
  - **Si es Baja (si estuviera señalado como tal)**
    - **No se escribe en maestro actualizado y podría escribirse en fichero Baja.**
  - **Si es alta (si estuviera señalado como tal)**
    - **Es un error pues la Clave ya existe.**
      - **Escribir de movimiento en errores si procede.**
      - **Escribir maestro en maestro actualizado.**
  - **En cualquier caso se lee de nuevo de maestro y de movimiento.**
- **Registro de maestro sin registro movimiento**
- **Movimiento sin registro maestro**

Ver Ejemplo pag.21

Situaciones de emparejamiento

- **Registro de maestro sin registro movimiento**
  - **Guardar del maestro en maestro actualizado.**
  - **Leer de maestro**
- **Movimiento sin registro maestro**
  - **Si es alta**
    - **Escribir de movimiento en maestro actualizado**
  - **Si baja o modificación**
    - **Es un error (Clave no existe)**
  - **En cualquier caso se leerá de fichero de movimientos**

Ver Pseud. en pag. 21-22

Ficheros Directos

- **Características:**
  - **Deben crearse en soporte de almacenamiento direccionable.**
  - **Los registros se almacenan y acceden indicando el lugar que ocupan.**
    - **No ocupan posiciones consecutivas de memoria.**
    - **Debe haber una relación entre el valor del campo clave y la dirección de almacenamiento de los registros.**
  - **El programador debe implementarlos si el lenguaje no incorpora utilidades al respecto.**

Ficheros Directos

- **Ventajas**:
  - Rapidez en el acceso a los registros
    - Adecuados para aplicaciones en tiempo real y de tiempo de acceso crítico.
  - El acceso a registro es directo mediante su propia clave, sin que deba pasarse por los registros que le preceden.
  - La actualización y recuperación de un registro es inmediata, sin necesidad de usar ficheros auxiliares.
  - Las supresiones son posibles por marcas y físicamente.
- **Inconvenientes**:

Ficheros Directos

- **Inconvenientes**:
  - El programador debe generar el software para su gestión o buscar utilidades en el mercado.
  - Poco aprovechamiento del soporte de almacenamiento (Hay que dejar semivacío el soporte para que no se produzcan muchos sinónimos).
  - No es posible el acceso secuencial, puesto que el fichero no está ordenado respecto a ningún campo.

Inconvenientes de la estructura Archivo

- **Redundancia de datos .**
- **Conflictos serios en el mantenimiento de los archivos .**
- **Conflictos en la seguridad y confidencialidad de los datos.**
- **Acoplamiento entre los programas y los archivos.**

Solución

- **BASES DE DATOS**
  - colección de datos
  - integrados en una sola estructura global
  - independientes de los programas que los usan
  - exento de redundancias perjudiciales (FN)
  - acceso inmediato
  - Se puede acceder mediante distintas formas y por múltiples usuarios al mismo tiempo.
  - Gestionados por el **Sistema Gestor de la Base de Datos** (**SGBD**)

"Lo maravilloso de aprender es que nadie puede arrebatarnos lo aprendido"

B.B. King, Guitarrista, cantante, compositor,

(Itta Bena, Misisipi, Estados Unidos, 1925-?)