Introducción a la Organización de Archivos

Definición de Archivo

Un *archivo* se puede definir en términos generales como una unidad lógica de almacenamiento permanente de datos, administrada por un Sistema Operativo.

> Es una *unidad lógica de almacenamiento* porque pertenece a la esfera del software. En general, todo lo relativo o concerniente al software se califica como *lógico*, en oposición a lo relativo o concerniente al hardware, que se califica como *físico*. Las unidades físicas de almacenamiento varían según el dispositivo: en discos magnéticos las unidades son discos, cilindros, pistas y sectores, y en discos compactos, bloques.
>
> Es una *unidad de almacenamiento permanente* porque se implementa en dispositivos de almacenamiento persistente, en general en discos magnéticos o compactos, y por lo tanto subsiste independientemente de que el dispositivo donde se encuentre esté en línea (accesible por una computadora sin necesidad de intervención humana) u operando.
>
> Es una *unidad administrada por un Sistema Operativo* porque sólo un Sistema Operativo, mediante su *Sistema de Archivo*, permite que usuarios y programas creen o eliminen archivos, y que programas accedan y actualicen la información contenida en ellos. Los sistemas de archivo permiten también crear y administrar unidades lógicas para el almacenamiento de archivos: las *carpetas* (o indistintamente, *directorios*), y para dispositivos con gran capacidad de almacenamiento, como los discos rígidos, permiten crear y administrar también unidades lógicas de un nivel superior: las *particiones de disco*.

Clasificación de Archivos

Los archivos se pueden clasificar según la clase de datos que contengan. En general pueden ser

- *De Datos Maestros:* datos de un sistema de información que representan entidades de existencia real o ideal, por ejemplo productos o servicios, o valores de referencia para determinar características o atributos de otros datos (dominios de atributos definidos por extensión).
- *De Datos Transaccionales:* registros de hechos o eventos relacionados con datos maestros, por ejemplo de ventas de productos o de prestaciones de servicios.
- *De Reporte:* información editada para su presentación al usuario (en general en formatos pdf, html o de texto).
- *De Trabajo:* resultados parciales o intermedios de procesamiento, o datos de intercambio entre programas.
- *De Control de Datos:* para almacenar metadatos (definiciones de datos), administrar espacios libres, registrar identificadores de registro vacantes o acceder al contenido de otro archivo (índices y tablas de acceso).
- *De Intercambio de Datos:* para representar datos en formatos estándar de manera que puedan ser procesados libremente conociendo el estándar. Generalmente son archivos de texto, con alguna convención para rotular o delimitar datos, que pueden incluir o no definiciones sobre la estructura de la información contenida (un estándar actual es el XML: eXtended Markup Language).
- *De Programa:* módulos fuente, objeto o de carga, y librerías de enlace dinámico (instrucciones de máquina en vez de datos).
- *De Recursos de Programa o Unidades Grandes de Información:* imágenes, audio, vídeo.
- *De Productos de Programas:* archivos con tipo asociado a un programa o aplicación (.doc, .xls, etc.).
- *De Empaquetado de Archivos:* para agrupar, normalmente en forma comprimida, archivos y directorios, con propósitos de trasmisión o resguardo (.zip, .rar, etc.).

Composición de Archivos de Datos

El almacenamiento de datos dentro de un archivo se efectúa mediante unidades de almacenamiento también lógicas o de aplicación, llamadas *registros lógicos* (en general, registros, a secas), que a su vez se componen de elementos individuales de información llamados *campos* o *atributos*.

> Según la RAE (Real Academia Española www.rae.es), un *dato* es un antecedente necesario para llegar al conocimiento exacto de algo o para deducir las consecuencias legítimas de un hecho. Para que un dato pueda ser procesado por una computadora, debe estar "registrado" de manera adecuada, en términos de atributos o características identificables por su posición en el registro o por rotulación.

Los campos o atributos pueden ser:

- *Simples*, cuando admiten valores atómicos o indivisibles, o *Compuestos*, cuando admiten valores estructurados, a su vez, con atributos propios (por ejemplo, un domicilio se puede componer de una calle, una ubicación en esa calle y una localidad de la ubicación en la calle).
- *Optativos*, cuando pueden faltar o tener valores desconocidos y por consiguiente no se registran, u *Obligatorios*, cuando se impone registrarles valor (cardinalidad mínima)
- *Monovalentes*, cuando admiten a lo sumo un valor, o *Polivalentes*, cuando pueden tener varios valores del mismo tipo o estructura (cardinalidad máxima)

La composición de un archivo se puede especificar en términos de los componentes de sus registros, que son las mínimas unidades lógicas de información; así, para especificar la estructura lógica de un archivo se debe tener en cuenta la identidad, estructura y cardinalidad de los campos o atributos de sus registros.

Por caso, para especificar la estructura lógica de un archivo de personas, se puede usar como convención:

```
Persona(((apellido)+, (nombre)+, fecha de nacimiento(año, mes, día))i,
      (DNI)i, (domicilio(tipo('real' | 'laboral' | 'legal'), calle,
      ubicación, (teléfono)*, localidad, provincia, (CPA)?))*3)
```

Donde para cada atributo se indica la identidad, mediante un nombre, la estructura, listando los atributos componentes entre paréntesis, y la cardinalidad, mediante los calificadores

? : opcionalidad (0 ó 1 valor)

* : ninguno o varios valores (*3 implica ninguno o hasta 3 valores)

+ : uno o varios valores (+ seguido de un número implicaría uno hasta ese número de valores)

También procede especificar valores por extensión, cuando fuera pertinente, como en el caso del tipo de domicilio

```
tipo('real' | 'laboral' | 'legal')
```

donde la barra vertical (|) indica alternativa excluyente.

Y también se impone indicar cualesquiera combinaciones de atributos que puedan identificar a una persona:

```
((apellido)+, (nombre)+, fecha de nacimiento(año, mes, día))i
(DNI)i
```

La cuestión de la identificación de registros es sumamente importante debido a que los archivos representan conjuntos de registros, en los que por cuestiones de integridad cada elemento o registro puede aparecer sólo una vez. Así, por imposición de la Teoría de Conjuntos, todo archivo debe tener al menos una combinación de atributos (o un atributo) cuyos valores identifiquen unívocamente a cada registro. Y aunque baste con determinar una única combinación de atributos que funcione como identificador de los registros para cumplir con la Teoría de Conjuntos, no es ocioso determinar todas las combinaciones posibles, ya que cada una de ellas implica una "guarda" o "garantía" adicional de identificación que además aporta más información sobre el Sistema real al que refiere el archivo.

También es necesario indicar si algún atributo o conjunto de ellos es identificador externo, es decir, si es un identificador de otro archivo que sirve para conectar o relacionar registros. Para este caso la convención es encerrar al atributo o conjunto de atributos entre paréntesis y ponerle como calificador `ie`.

Estructura de Archivos de Datos (Organización de Registros)

Cuando se debe implementar efectivamente un archivo de datos, se debe pasar de las consideraciones a nivel lógico o de aplicación, es decir, de la *definición o especificación lógica* de un archivo, a consideraciones de nivel físico o de implementación, es decir, a la *definición o especificación física* del archivo, que implica la determinación de cómo habrán de almacenarse concretamente los registros en disco para su procesamiento automático.

Esta división en fases o etapas de la definición de un archivo no es más que la aplicación de la metodología de refinamientos sucesivos (divide y vencerás) de la algoritmia, ya que en primera instancia se define con precisión las entidades y eventos del sistema de información que habrán de registrarse para su tratamiento automático, y finalmente se pasa a considerar la forma más adecuada de almacenarlos en disco con el mismo propósito. Cabe señalar que la optimalidad del almacenamiento no pasa precisamente por la minimización del espacio en disco sino por la inmediatez (mínima cantidad de accesos a disco) en la recuperación y, eventualmente, en la actualización; esto lleva a consideraciones respecto a la localidad de los datos (proximidad física en el disco) y a las técnicas de acceso, que implican la organización del archivo más que la de los registros.

Una de las primeras cuestiones a dirimir para la definición física de un archivo es si se emplearán registros de longitud fija o de longitud variable. La decisión debe tomarse analizando la estructura lógica del archivo y los patrones de acceso que se verificarán sobre los registros, fundamentalmente para actualizaciones y recuperaciones.

- Si los registros lógicos de un archivo no tuvieren atributos polivalentes, o los tuvieren pero con cantidades de valores previsibles y acotadas, lo preferible sería emplear *registros de longitud fija en secuencia.* Por caso, el archivo de personas del ejemplo anterior:

  ```
  Persona(((apellido)+, (nombre)+, fecha de nacimiento(año, mes, día))i,
       (DNI)i, (domicilio(tipo(‘real’ | ‘laboral’ | ‘legal’), calle,
       ubicación, (teléfono)*, localidad, provincia, (CPA)?))*3)
  ```

  Podría definirse físicamente:

  ```
  Persona(apellidos: C(64), nombres: C(64), fecha de nacimiento(año: E, mes:
       E, día: E), DNI: EL, (domicilio(tipo(‘real   ’ | ‘laboral’ |
       ‘legal’): C(7), calle: C(32), ubicación: C(32), teléfonos: C(32),
       localidad: C(32), provincia: C(16), CPA: C(8)))3)
  ```

  Donde como se puede ver, los atributos polivalentes como nombre, apellido y teléfono se implementan en una única cadena de caracteres, para los tipos de domicilio se emplean cadenas de igual longitud, para el Código Postal Argentino se usan indefectiblemente ocho caracteres (si no se conoce el valor se registrarán ocho caracteres en blanco) y se reservará espacio para exactamente tres domicilios, aunque eventualmente no se registre ninguno o menos de tres, como lo admite la definición lógica.

  Como es evidente en este ejemplo, el costado negativo de los registros de longitud fija es el desperdicio de espacio de almacenamiento cuando hay valores nulos o por rellenado con espacios en blanco de los campos de caracteres.

- Si los registros lógicos de un archivo tuvieren atributos polivalentes con cantidades de valores imprevisibles y no tuvieren actualizaciones, entonces lo preferible sería usar *registros de longitud variable en secuencia.*

  Por ejemplo, un archivo de facturas con la siguiente estructura lógica:

  ```
  Factura((número)i, fecha(año, mes, día), forma de pago(‘CO’ | ‘CH’ | ‘TD’
       | ‘TC’ | ‘CC’), (referencia pago)?, descuento, (componente(código
       producto, cantidad, precio de venta unitario))+)
  ```

  podría tener una definición física:

  ```
  Factura(número: EL, fecha(año: E, mes: EC, día: EC), forma de pago(‘CO’ |
       ‘CH’ | ‘TD’ | ‘TC’ | ‘CC’): C(2), (longitud referencia pago: EC,
       referencia pago: CV), descuento: F, (cantidad componentes: EC,
       (componente(código producto: EL, cantidad: EC, precio de venta
       unitario: F))+))
  ```

  Donde como se puede ver, aparecen nuevos atributos, llamados *atributos de control*, necesarios para poder recuperar secuencias de longitud variable de caracteres o de valores de atributos polivalentes. Nótese que este archivo de facturas es transaccional, y, como es característico en la mayoría de los archivos de esta clase, los registros no se actualizan (como representan eventos, en este caso ventas, una vez registrados no sufren ninguna actualización sino sólo recuperaciones para consultas); entonces, los registros de longitud variable se pueden almacenar *en secuencia* y desaparece el problema del desperdicio de espacio por rellenado a longitud fija y valores inexistentes.

Para los campos de caracteres, en lugar de asociarles prefijos de longitud para su control, como en este caso, se les puede asociar como posfijo un carácter de control que indique su fin (como es estándar en el lenguaje C, o en Pascal para el tipo PChar).

La convención de encerrar entre paréntesis a los campos de control con los campos que controlan se emplea para mayor claridad, aunque no resulta imprescindible.

- Si los registros lógicos de un archivo no tuvieren atributos polivalentes, o los tuvieren pero con cantidades de valores previsibles y acotadas, y además se quisieren agrupar por afinidad, entonces lo preferible sería usar *registros de longitud fija en bloques*.

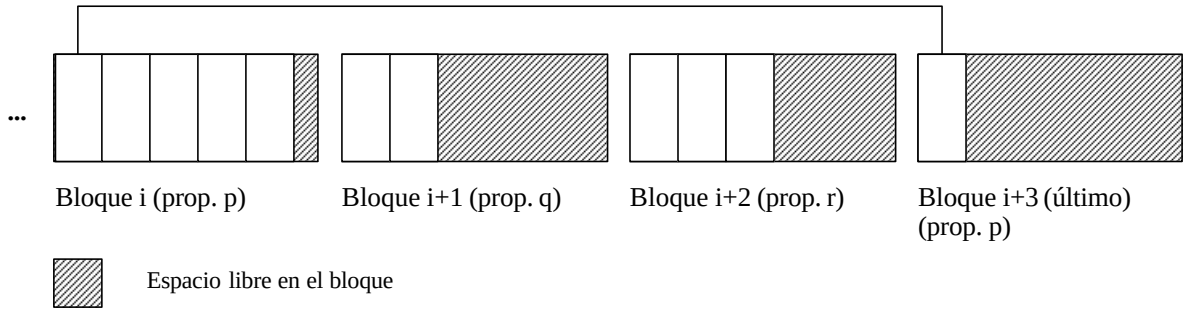
  Por caso, considérese un archivo para registrar liquidaciones y pagos de expensas en un sistema de una empresa de administración de consorcios, con la siguiente estructura lógica

  ```
  Expensa(((propiedad)ie, fecha(año, mes, día), movimiento('L' | 'C'))i,
        monto expensas, saldo expensas, monto intereses, saldo intereses)
  ```

  Si se quisieran agrupar los registros por propiedad (en bloques exclusivos para movimientos de una misma propiedad) y mantenerlos ordenados dentro de cada bloque por fecha, la implementación del archivo debería respetar la siguiente definición física:

  ```
  Expensa(cantidad de registros: EC, próximo bloque: E, (propiedad: EL,
  fecha(año: E, mes: EC, día: EC), movimiento('L' | 'C'): C(1), monto
  expensas: F, saldo expensas: F, monto intereses: F, saldo intereses: F)+)
  ```

  En este archivo se supone que hay al menos un bloque por cada propiedad, y dentro de cada uno los registros de ('L')iquidación o ('C')obro se van incorporando al final de la lista según aparezcan, quedando ordenados cronológicamente. Si un bloque no tiene capacidad para agregar un movimiento correspondiente a una propiedad, se hace crecer el archivo un bloque, el cual se refiere desde el que resulta desbordado.

- Si los registros lógicos de un archivo tuvieren atributos polivalentes con cantidades de valores imprevisibles pero tuvieren actualizaciones o se quisieren agrupar por afinidad, entonces lo preferible sería usar *registros de longitud variable en bloques*.

  Por ejemplo, considérese un archivo para registrar artículos en un negocio con la siguiente estructura lógica:

  ```
  Artículo((código)i, ((nombre)ie, (marca)ie)i, presentación(descripción,
        existencia, precio de venta unitario)*)
  ```

  Como puede observarse, tanto los nombres de artículos (de un mismo artículo puede haber muchas marcas) como las marcas (de una misma marca puede haber muchos artículos) están tabulados en un archivo independiente, de ahí que en este archivo aparezcan nombre y marca como códigos identificadores externos.

  En este caso, un artículo de una determinada marca puede tener una cantidad imprevisible de presentaciones distintas, así que una definición física posible sería:

  ```
  Artículo(cantidad de registros: EC, (código: EL, nombre: EL, marca: EL,
        (cantidad de presentaciones: EC, (presentación((longitud de
        descripción: EC, descripción: CV), existencia: E, precio de venta
        unitario: F))*))+)
  ```

La convención para especificar tipos de valores es:

EC: Enteros Cortos (complemento a dos en un byte)

E: Enteros (complemento a dos en dos bytes)

EL: Enteros Largos (complemento a dos en cuatro bytes)

F: Fraccionarios (punto flotante IEEE)

C: Caracteres (con longitud exacta entre paréntesis)

CV: Caracteres Variables (hasta 255)

T: Texto (cantidad ilimitada de caracteres, incluyendo caracteres de control como salto de línea, retorno de carro, tabulación, fin de texto)

L: Lógicos (0: Falso o No, 1: Verdadero o Sí)

En cuanto al tamaño de bloques para organizar registros, siempre debe elegirse un tamaño en bytes que sea submúltiplo o múltiplo del tamaño de los registros físicos del dispositivo de almacenamiento masivo (512 bytes) o de las unidades de asignación de espacio para archivos que use el sistema operativo para los mismos dispositivos (tamaño de las unidades de asignación o clusters en Microsoft, o de bloques físicos en Unix). Este cuidado se debe a que los sistemas operativos implementan políticas de buffering o caché de disco análogas al caché de RAM pero con registros físicos de disco en vez de celdas de memoria, y silos bloques de organización de registros no estuvieren contenidos íntegramente en estas unidades o no estuvieren conformados por un número entero de ellas se perdería parte de la ventaja del caché de disco del sistema operativo.