Unidad Didáctica 0:

# EVOLUCIÓN DE LOS SISTEMAS DE ALMACENAMIENTO DE LA INFORMACION

# 1 Introducción.

Debemos entender la evolución de los sistemas de almacenamiento como un proceso continuo de modernización que nos ha llevado desde el papel hasta el estado actual.

La mayor parte de las empresas de hace treinta años utilizaban archivos de papel, donde anotaban los datos de sus clientes en forma de fichas escritas con una máquina de escribir de la época. Se empezaron a utilizar los ordenadores para controlar los procesos básicos de una empresa.

La primera informatización empezaba por pasar del sistema de fichas de papel o de cartulina a un sistema informatizado. Así los problemas se podían solucionar con una decena de archivos, ¿Quién me quiere comprar algo? Mis clientes. Por tanto necesito un archivo de clientes. ¿Quién me suministra la materia prima? Mis proveedores. Por tanto necesitamos un archivo de proveedores. Y así sucesivamente.

La principal ventaja del sistema informatizado estaba en evitar tiempo para encontrar una ficha de papel dentro de un enorme cajón de fichas. Pasábamos, pues, de un sistema en que el soporte era papel, a un nuevo sistema en el que el soporte era digital.

Toda la historia de los archivos ha ido ligada al avance de la tecnología; principalmente del hardware y en menor medida del software.

# 2 Representación de la información

Un sistema informático maneja información de todo tipo (números, texto, imágenes, sonidos, video, etc.), dándole entrada, salida o procesándola.

Bajo nuestra perspectiva humana es fácil diferenciar rápidamente lo que son números de lo que es texto, imagen, sonido o video, utilizando para ello los órganos sensoriales y el cerebro.

El ordenador en su funcionamiento trata de emular el comportamiento humano, pero al ser una máquina digital, cuyo soporte es la electrónica, sólo es capaz de representar información binaria por lo que los ordenadores necesitan codificar la información del mundo real al equivalente binario y utilizar mecanismos para su presentación. Para ello utilizar mecanismos de representación, almacenamiento y presentación como veremos a continuación.

Desde los inicios de la informática la codificación ha sido problemática entre otras cosas por la falta de acuerdo en la representación de esa información. Hoy día existen numerosos estándares para tal fin.

En un sistema informático los datos los podemos clasificar en:

- Datos numéricos, dígitos del 0 al 9.
- Datos alfabéticos, son letras mayúsculas y minúsculas de la “a” a la “Z”.
- Datos alfanuméricos, son una combinación de las anteriores.

Los sistemas de codificación o sistemas de representación de la información los clasificaremos en dos grandes grupos en función del tipo de elemento o dato a representar:

1. En el caso de los números no se almacena el dígito/carácter sino su valor binario. Así se almacenan dependiendo del tipo de valor que sea (natural, entero o real), utilizando diferentes sistemas binarios de representación numérica. Un número entero simple de 16 bits (de 0 a 65535) emplea 2 bytes. Si se trata de un número entero de 32 bits (de 0 hasta 4.000 millones aproximadamente) ocupa 4 bytes. Si se trata de un número en coma flotante (normalmente se ajustan a la norma IEE488 como Excel) usa 8 bytes para almacenar mantisa y exponente

2. Sistemas de codificación alfanumérica, donde se representan el alfabeto, números y otros caracteres especiales. Para el caso del texto, lo que se hace es codificar cada carácter de la cadena

a almacenar empleando una serie de valores binarios con los que se corresponde de acuerdo a un determinado código. Utilizamos los códigos ASCII, EBCDIC, Unicode, etc.

### 2.1 Medidas de la información

La mínima cantidad de información es aquella que puede adoptar únicamente dos estados posibles; es un “si” o un “no”, es decir, dos posiciones contrarias, una positiva y la otra negativa. En los ordenadores esta mínima expresión de la información se llama BIT (Binary digiT) y convencionalmente se representan sus posibles estados como 0 y 1.

Una unidad superior es el NIBBLE, formado por cuatro BITs .

La siguiente unidad es el BYTE, formado por ocho BITs, que por lo tanto es capaz de representar 256 estados diferentes. Tomando el BYTE como origen se define el resto de unidades de capacidad, que son múltiplos de ´este:
KiloByte (KB) 1024 bytes.
MegaByte (MB) 1024 KB.
GigaByte (GB) 1024 MB.
TeraByte (TB) 1024 GB.
PetaByte (PB) 1024 TB.
ExaByte (EB) 1024 PB.

Como puede observarse, los múltiplos no son 1000 veces mayores que los anteriores sino 1024 (210) veces; esto se debe a que en informática se utiliza el álgebra binaria cuya base es 2.

## 3 Organización de la información

Todas las aplicaciones necesitan almacenar y recuperar información, superando las limitaciones del almacenamiento real, trascendiendo a la duración de los procesos que las utilizan o generan e independizando a la información de los procesos permitiendo el acceso a la misma a través de varios procesos.

Las condiciones esenciales para el almacenamiento de la información a largo plazo son:

- Debe ser posible almacenar una cantidad muy grande de información.
- La información debe sobrevivir a la conclusión del proceso que la utiliza.
- Debe ser posible que varios procesos tengan acceso concurrente a la información.

*La solución es la organización de la información mediante ficheros de datos, o ficheros de metadatos (tecnología XML), o bases de datos, almacenando siempre la información en discos y otros medios externos en unidades llamadas archivos o ficheros.*

Los **ficheros o archivos** son mecanismos de abstracción para almacenar la información y posteriormente poder leerla sin que el usuario tenga que preocuparse de la forma y lugar físico de almacenamiento de la información. Así, un fichero es un tipo abstracto de datos definido e implementado por el sistema operativo. Sin embargo, para el usuario, un fichero es una sucesión de registros lógicos. Un registro lógico puede ser un byte, una línea o un conjunto de campos. Los ficheros pueden implementarse, por parte del sistema operativo, sobre dispositivos diferentes (discos, CDs, DVDs, cintas, ...). Los sistemas operativos ofrecen un conjunto de primitivas muy potentes para gestionar el concepto fichero (abrir, leer, avanzar puntero, cerrar, ...). Aunque quizá, el problema principal del sistema operativo consiste en volcar el concepto fichero que tiene el usuario (fichero lógico) sobre los dispositivos físicos de almacenamiento, ya que el tamaño del registro físico del dispositivo no tiene por qué coincidir con el tamaño del registro lógico.

Veamos su evolución

# 4 Ficheros Secuenciales

Los primeros sistemas de almacenamiento físico eran tambores de cinta magnética (de un diámetro comparable al de los antiguos discos de vinilo conocidos como LP), donde se almacenaban datos digitales: ceros y unos en orden secuencial. Para el usuario estos ceros y unos formaban un fichero o archivo. A este tipo de archivos se les denomino ***ficheros secuenciales***. ¿Por que secuenciales? Debido a que van ordenados en posiciones sucesivas, y que si queremos llegar a la posición número 100, debemos pasar antes por la 1, la 2 y así sucesivamente; secuencialmente.

El sistema más normal para crear un archivo de clientes era escribir su nombre, separar con un retorno de carro, escribir su dirección seguida de otro retorno de carro, población y así sucesivamente.

Entre cliente y cliente se colocaba una marca de final de bloque que indicaba el inicio de los datos de un nuevo cliente o bien el final del archivo de clientes. Cada uno de estos bloques con los datos de un cliente recibe el nombre de ***registro*** y cada una de estas informaciones (nombre, dirección, población, etc.) recibe el nombre de ***campo***.

Por tanto, un registro se compone de campos.

Veamos un ejemplo:

```
Juan Martínez
Calle del Pez, 5
Madrid
<FIN>
Comercial Martínez
Calle de la Cuesta, 10
Sevilla
<FIN>
<OEF>
```

Todos los sistemas operativos soportan operaciones de lectura y escritura sobre archivos secuenciales.

El contenido de un archivo secuencial es legible en un procesador de textos.

Todos los lenguajes de programación están dotados de funciones para acceder a archivos secuenciales.

Los programas encargados de las altas, bajas, modificaciones, consultas, listados y otras operaciones que afectan exclusivamente a un archivo (por ejemplo, clientes) reciben el nombre de ***mantenimientos***. Podemos, pues, realizar mantenimientos de archivos secuenciales mediante programas escritos en lenguajes de todo tipo: antiguos y modernos, como BASIC, Pascal, Fortran, COBOL, C/C++, Java, Visual Basic, C#, Prolog, LISP, etc.

Las principales características de los archivos secuenciales son:

- **Lectura ordenada obligatoria** Para leer un registro situado en medio del archivo debemos pasar por todos los registros anteriores.

- **No permite el retroceso (forward only)**. En los archivos secuenciales se realiza la lectura sólo hacia delante. Por ejemplo: si nosotros leemos el primer registro del cliente, el segundo, el tercero, el cuarto y después deseamos volver al segundo, la única forma de hacerlo consiste en cerrar el archivo y volver a abrirlo, ya que no podemos retroceder.

- **Los archivos secuenciales son *monousuarios*** Los archivos secuenciales no permiten el acceso simultáneo (*concurrencia*) de varios usuarios. Si por desventura se accediera simultáneamente en modo escritura, los resultados son impredecibles: puede producirse corrupción de datos por escrituras entremezcladas o desaparición inadvertida de datos. El último que escribe tiene mayores

posibilidades de mantener su información. Evitemos la concurrencia en archivos secuenciales.

- **Estructura rígida de campos**. Todos los registros deben aparecer en orden. Esto quiere decir que si nosotros escogemos el orden Nombre, Dirección, Población en el primer registro, no podemos cambiarlo por Dirección, Población, Nombre en el siguiente registro. El programa de lectura de un archivo secuencial va a ciegas leyendo líneas y necesita saber qué significado tiene cada línea según su orden.

  Pero, ¿qué pasa si queremos grabar más datos de cada cliente? Por ejemplo,su teléfono o su código fiscal, o sus condiciones de pago…

  La primera consecuencia es que debemos rehacer todos los programas de lectura para tener en cuenta estos nuevos campos. Y esto puede representar mucho trabajo si tenemos que acabarlo de un día para otro. La segunda consecuencia es que ese archivo ya no nos serviría por ejemplo para el programa que nos habían escrito los informáticos para imprimir etiquetas. Tendríamos que guardar un archivo central con todos los datos y, a partir de éste, extraer en otro archivo sólo los datos que deseamos imprimir. Tras ello ya podríamos lanzar el nuevo listado por la impresora. Por tanto,debemos crear un nuevo proceso y complicar el que ya teníamos.

- **El modo de apertura condiciona lectura o escritura**. Las lecturas y las escrituras dependen del modo en que se haya abierto un archivo secuencial. En el momento de abrir un archivo, dependiendo del modo que hayamos escogido, quedaremos autorizados a leer o a escribir, pero no alas dos cosas.

- **Lecturas parciales pero escrituras totales**. Las operaciones de lectura pueden ser parciales, pero las operaciones de escritura conllevan obligatoriamente la escritura total de toda la secuencia completa de registros. Dicho de otro modo: al abrir un archivo en modo escritura estamos borrando su contenido anterior si lo hubiere.

  Por esta razón algunos lenguajes de programación contienen un modo de apertura llamado Append que escribe al final de un archivo existente en vez de reescribirlo de nuevo.

- **La marca de final de archivo (EOF)**. Las operaciones de lectura deben comprobar siempre que no rebasan el final del archivo secuencial, mediante la comparación del carácter de final de archivo (EOF o End Of File). Este carácter se coloca siempre y de modo implícito al cerrar un archivo secuencial en modo escritura.

- **Registros de longitud variable**. Los registros de un archivo secuencial tienen una longitud variable porque también sus campos tienen una longitud variable. No ocupa el mismo espacio un cliente que se llame Juan Sánchez, que otro cliente que se llame José María González de Castro y Santos de Carballar. Esto implica que a priori no vamos a saber el tamaño de las variables que debemos usar en memoria para almacenar estos contenidos. Para superar este tipo de problemas apareció el archivo de acceso aleatorio, que veremos más adelante.

## 5 Ficheros Aleatorios

Si los archivos secuenciales se emparejan en el tiempo con el hardware de los tambores de cinta magnética, los archivos de acceso aleatorio aparecen con el disquete y el disco duro.

Los archivos de acceso aleatorio no están sujetos a la esclavitud de pasar por el inicio hasta llegar a la porción de la información que nos interesa. En los archivos de acceso aleatorio podemos colocarnos en una posición determinada expresada por un número. Por ejemplo, podemos ir directamente a la posición 100, volver atrás hasta la posición 20, avanzar ahora hasta la 200 y así tantas veces como queramos.

Esto implica que si utilizamos una estructura delimitada en longitud, podemos calcular la posición en la que debemos situarnos para leer un registro.

La medida básica de la posición del puntero de lectura es el byte; primer byte, segundo byte y así sucesivamente.

Esto quiere decir que, si empleamos caracteres de dos bytes (por ejemplo, en Unicode), las posiciones para leerlos serán la 0, la 2, la 4, etc., avanzando cada vez de dos en dos.

Pongamos por caso que tenemos un registro como éste codificado en ANSI (1 byte por carácter):
Nombre: 80 caracteres ANSI
Dirección: 100 caracteres ANSI
Población: 50 caracteres ANSI

Su longitud será de 230 caracteres. Esto implica que el primer registro se encontrará en la posición 0; el segundo registro se encontrará en la posición 230; el tercero estará en la posición 460 y así sucesivamente.

Si en cambio codificamos los caracteres en Unicode (UTF-16) de forma que su longitud es de 16 bits por carácter, esto es 2 bytes, el registro quedará así:
Nombre: 80 caracteres UTF-16.
Dirección: 100 caracteres UTF-16.
Población: 50 caracteres UTF-16.

Su longitud será de 460 caracteres para un solo registro, ya que cada carácter ocupa 2 bytes.

Por tanto, podemos decir que la posición será:

Pos = NumRegistro * LongitudRegistro

¿Qué pasa si un campo de nuestro fichero es numérico? ¿Ocupa los mismo que los alfabéticos? En los archivos secuenciales se guardaba el número con su formato expresado en decimal: dígito a dígito. Por tanto, el número “10” ocupaba 2 caracteres y el número “1234567” ocupaba 7 caracteres. En los archivos aleatorios no se guardan los dígitos, sino el *valor binario*. Estos datos ya no son legibles con un procesador de texto.

Si intentamos leer los datos de estos ficheros desde un procesador de texto nos aparecerán caracteres extraños, con posibilidad de que el contenido no aparezca entero, ya que es frecuente encontrar un valor binario que coincida con el valor EOF (final de archivo)

Las principales características de los archivos de acceso aleatorio son:

- Posicionamiento inmediato. Permiten situar el puntero de lectura o escritura sobre una posición concreta del archivo sin necesidad de pasar por las posiciones anteriores, con el consiguiente incremento de rapidez. Una sola apertura proporciona la capacidad de avanzar y retroceder sin necesidad de reabrir el archivo.

- Registros de longitud fija. Todos los registros tienen la misma longitud, ya que se utiliza siempre una estructura rígida dimensionada a la máxima longitud para cada campo.

- Apertura para lectura/escritura. Permiten su apertura en modo mixto (lectura y escritura), de forma que con una sola operación de apertura podemos leer o escribir a voluntad en cualquier posición según nos convenga.

- Dimensionamiento máximo al ser creados. Los archivos de acceso aleatorio deben dimensionarse hasta un número de registros máximo en el momento de crearse

- Permiten el uso concurrente (multiusuario). Al establecerse zonas específicas y limitadas de actuación para lectura y escritura, diversos usuarios pueden acceder y escribir en diferentes porciones del archivo de forma simultánea. Esto lleva a considerar la posibilidad de concurrencia de dos usuarios en el mismo registro. Para ello se utilizan algoritmos de gestión de los bloqueos. Dichos algoritmos administran zonas (que no siempre coinciden con registros) y las marcan para evitar dicha concurrencia de forma temporal. El tema del bloqueo lo trataremos más adelante dentro de los capítulos dedicados a bases de datos relacionales

Los archivos de acceso aleatorio están especialmente indicados para aquellos casos en que el código del registro (cliente, proveedor, etc.) pueda ser directamente el número de registro en el que se guardan los datos.

Al trabajar con archivos de acceso aleatorio deberemos tener en cuenta no rebasar nunca el final de archivo. Si nuestro archivo está dimensionado hasta 100 registros, intentar que no haya ningún usuario que nos pregunte por el 101 o por el 200. Para ello deberemos establecer nuestros sistemas de prevención desde el interfaz o desde el sistema de cálculo de la posición del puntero de lectura / escritura.
Los archivos de acceso aleatorio son un paso más que nos conduce por el camino de los archivos indexados, puesto que constituyen el depósito de los datos.

## 6 Ficheros Indexados

Los archivos indexados son básicamente archivos de acceso aleatorio a los que se añade una utilidad para acceder al registro deseado a través de una clave.

En el caso de los archivos de acceso aleatorio, la clave es siempre el número de registro.

Los archivos indexados pueden buscar un cliente según su nombre, según su código de identificación fiscal o según cualquier otro campo. Para ello se construye una estructura de índice por cada campo que se desea indexar.

Esta estructura de índice se mantiene en la memoria RAM de forma ordenada, de manera que podamos encontrar rápidamente una relación entre el apellido de un cliente y el número de registro en el que se encuentra.