CAPÍTULO 6

ARCHIVOS Y BASES DE DATOS

Como ya sabemos, para que una computadora sea realmente útil, necesita procedimientos y dispositivos que permitan almacenar los datos que no está procesando en un momento dado, pero que puede necesitar con posterioridad. Igual consideración merecen los programas, ya que los vamos a utilizar multitud de veces; el hecho de que una secuencia de instrucciones pueda ser almacenada de forma permanente y utilizada con posterioridad es la base de la existencia de la industria del software.

En el capítulo anterior discutimos varias formas de organizar los datos dentro de un programa y cómo estas estructuras se estructuran dentro de la memoria principal. Ahora nos centraremos en las formas de organización que utilizan los sistemas operativos para el almacenamiento de datos y programas en los dispositivos de memoria masiva, así como la forma como podemos acceder a ellos por medio de un lenguaje de programación. A lo largo de este capítulo haremos abstracción de las características físicas de los dispositivos de almacenamiento secundario, cuya descripción dejamos para el correspondiente capítulo dedicado al soporte físico del computador.

6.1. ARCHIVOS: DEFINICIONES Y CONCEPTOS

El concepto de archivo, aparece no sólo como forma de utilización de información presente en un medio de almacenamiento permanente, sino además, éstos permiten superar las limitaciones impuestas por el tamaño de la memoria principal a la hora de manipular grandes volúmenes de datos. Dada la ingente cantidad de datos que puede manejar un computador en la actualidad, el uso de los dispositivos de almacenamiento como memoria secundaria permite que la cantidad de datos que puede tratar un programa no esté limitada por el tamaño de la memoria principal. La propia aparición de la **memoria virtual** (capacidad de un procesador de acceder a un medio de almacenamiento masivo, tal como un disco duro, y manejarlo como si fuera memoria principal) deriva de esta misma idea, en el sentido de que una parte del dispositivo de almacenamiento es usado como una ampliación de la memoria principal.

Los datos que se encuentran en memoria masiva suelen organizarse en archivos, y por tanto entenderemos como **archivo** o **fichero** a *un conjunto de informaciones sobre un mismo tema tratado como una unidad de almacenamiento y organizado de forma estructurada para la búsqueda de un dato individual*. Como ya hemos dicho, los archivos pueden contener instrucciones de programas o información creada o usada por un programa.
Como tal, un archivo no es más que una agrupación de datos, cuya estructura interna es la que el usuario, el programador o el sistema operativo, le haya conferido implícitamente (la organización del fichero no es intrínseca a la existencia del mismo). En este punto debe aclararse que los archivos son independientes de los programas y de hecho, un mismo archivo creado por un programa puede ser usado por otros. El único requisito para obtener información de un archivo cualquiera es conocer como se han organizado los datos en el mismo, pues de lo contrario el resultado es una colección de bytes aparentemente sin sentido. Lo anterior pone de manifiesto que el fichero no contiene información explícita sobre su organización y por tanto esto tiene que ser tenido en cuenta al procesarlos.

Las estructuras con que se organizan los archivos no son inmanentes a los mismos ni son propias del sistema operativo, por lo que debe ser el software el responsable del mantenimiento de dichas estructuras. En ocasiones tendremos a nuestra disposición programas específicos llamados **sistemas de gestión de archivos** que permiten diseñar archivos con determinadas estructuras y realizar recuperaciones y actualizaciones eficazmente para dichas estructuras. Los programas que los utilicen pueden abstraerse del mantenimiento de las estructuras, delegando esta labor en el sistema de gestión de archivos.

Sin embargo una simple colección de ficheros no organiza óptimamente la totalidad de informaciones que pretendemos gestionar, ya que existen relaciones entre los distintos datos presentes en los diversos ficheros, que además deben estar disponibles para distintas aplicaciones. Por ello a toda colección de ficheros a las que pueda accederse por un conjunto de programas y que contienen todos ellos datos relacionados entre sí la llamaremos **base de datos.**

Acabamos de indicar cómo se pueden organizar los conjuntos de archivos, veamos ahora como se organiza un archivo internamente. La concepción clásica por la cual un fichero es un conjunto de informaciones estructuradas en unidades que en el capítulo anterior denominamos registros viene heredada de los sistemas manuales de tratamiento de la información y por tanto muchas de las ideas y de los términos que se usan en este campo derivan de la terminología empleada con los ficheros manuales. Según aquellos, tanto si los ficheros se almacenan en un archivador, en un disco magnético o en otro medio de almacenamiento, un **fichero** o archivo es una colección ordenada de datos, cuya unidad elemental que lo compone es el

registro. Un registro se corresponde con una ficha, o con una hoja de papel en un fichero manual, siendo una colección de información, normalmente relativa a una entidad particular, (p.e. información acerca de un estudiante) puede contener varios datos y cada registro de un mismo fichero tiene, en general, la misma estructura que los demás. Los datos individuales ocupan **campos** dentro de los registros. Un campo puede tener una longitud fija o variable, normalmente representa un ítem o elemento de datos elementales (Ver Figura 6.1). Los registros que componen un archivo pueden tener longitud fija o variable; en los segundos, debe existir alguna manera de indicar la longitud del registro, bien mediante un campo que lo indique, bien mediante un carácter especial que indique el final de un registro. El término **longitud del campo** hace referencia al tamaño máximo de dicho campo. Los campos pueden ser de diferente **tipo:** numérico, cadena, moneda, fecha o código, etc., ...

Fig. 6.1. *Ejemplo de campos de un registro*

La forma más común de identificar un registro es eligiendo un campo dentro del registro que llamaremos **clave** del registro. La localización de un registro se hace a través de los valores que tiene el campo clave, siendo a veces necesario utilizar varias claves bien por existir valores iguales de la clave en distintos registros, bien por estar interesados en buscar los registros por distintos campos (p.e. por nombre y por número de asiento en el ejemplo de la figura 6.1). Por esta razón se permite utilizar más de un campo como clave de un registro. En este caso los campos se denominan **clave primaria**, **clave secundaria**, etc.

Aclaremos que esta concepción de archivo no engloba todos los tipos de archivos existentes. No obstante esta concepción es útil, como estructura de datos, para organizar grandes volúmenes de datos en memoria masiva, donde los datos deben disponerse de un modo sistemático que facilite las operaciones más comunes sobre

los ficheros, en particular la de búsqueda. Los conceptos de carácter, campo, registro, archivo y base de datos son conceptos lógicos que se refieren a la forma como el usuario ve los datos y su organización

6.2. SOPORTE Y ACCESO A LOS ARCHIVOS

Un archivo, dependiendo principalmente del uso o capacidad que vaya a tener, puede estructurarse en un soporte (disco o cinta) de distintas formas. De hecho al hablar de archivos hay que retener que conviven dos visiones de los mismos, por un lado existe un sistema físico, más o menos complejo, donde realmente se almacenan los datos y por otro una visión lógica adecuada al usuario, que trabaja, sin preocuparse excesivamente sobre las características físicas de este almacenamiento. El sistema operativo permite que el usuario pueda utilizar solamente ésta última, haciendo de interfaz entre ambas (ver figura 6.2).

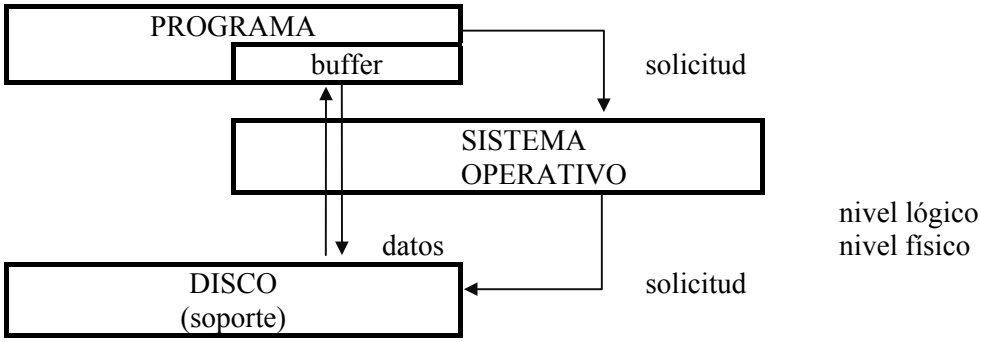

Fig. 6.2. *Acceso a los archivos*

En efecto, desde el punto de vista hardware, para almacenar datos o programas en la memoria masiva sólo existen **direcciones físicas**. Por ejemplo, en un disco toda información se graba o lee en forma de bloques (sectores) referenciados por (número de unidad)/(superficie)/(pista)/(sector). El sistema operativo posibilita que el usuario no tenga que utilizar direcciones físicas, realizando la transformación de la **dirección lógica** de un dato en el archivo (posición relativa de un dato dentro del mismo) a la **dirección física** (en qué punto concreto del soporte se encuentra el dato) y efectuando los accesos necesarios al dispositivo donde se encuentra el archivo para transferir la información del archivo al programa o a la inversa.

El tiempo que se tarda en acceder a una determinada información almacenada en memoria masiva (y por tanto contenida en un soporte magnético) es variable,

dependiendo, entre otras cosas, del lugar previo donde se ha accedido a la última información. Cuando el acceso a las informaciones de un archivo se produce en un orden determinado y fijo se conoce como **acceso secuencial**; el acceso a los bloques de datos se hace de forma sucesiva, uno tras otro., tal como se reproducen las canciones grabadas en una cinta. Cuando el acceso a determinada información contenida en el archivo puede hacerse sin acceder a las informaciones anteriores o posteriores se dice que se trata de un **acceso directo**, como ocurre al querer acceder a una determinada canción grabada en un disco.
Los soportes de memoria secundaria se clasifican según el modo de acceso que permiten en soportes secuenciales y soportes direccionables. Un **soporte secuencial** (o no direccionable) únicamente permite el acceso secuencial, es decir, el acceso a un bloque concreto del mismo requiere leer o pasar los bloques anteriores a aquél; es el caso de las cintas magnéticas. Los **soportes direccionables** o aleatorios permiten tanto el acceso directo como el secuencial: se puede acceder a un bloque físico sin más que dar su posición, sin necesidad de recorrer o leer otros bloques, aunque también puede hacerse de esa manera. Los discos magnéticos son ejemplo de soportes direccionables.

En los primeros tiempos de la informática la estructura de los ficheros estaba determinada en gran medida por los medios de almacenamiento disponibles. Hoy día la situación se ha invertido. La naturaleza de la aplicaciones determina la estructura de los ficheros, que es la que define los medios de almacenamiento que hay que utilizar. Obviamente los soportes secuenciales presentan limitaciones respecto a los direccionables, pero también su coste es mucho menor que el de éstos, aunque esta es una estimación que quizás tenga que revisarse en un futuro no muy lejano. La elección de uno u otro soporte dependerá del uso que se vaya a dar a la información que deban contener. No obstante, la tendencia actual presenta un uso ampliamente mayoritario de los soportes direccionables como dispositivos de memoria secundaria, mientras los soportes secuenciales quedan como medio de almacenamiento para copias de seguridad.

En el resto del capítulo trataremos los archivos desde el punto de vista lógico del programador y de su uso como memoria secundaria, ciñéndonos a estructuras típicamente asociadas a los soportes direccionables (los discos), obviando, por tanto, los detalles físicos subyacentes.

6.3. EL SISTEMA OPERATIVO Y LA GESTIÓN DE ARCHIVOS

Sobre la estructura de periféricos de almacenamiento disponibles en el ordenador, el sistema operativo construye dos abstracciones. La primera es la **creación de archivos**, ello aislará al usuario de los problemas físicos de almacenamiento. Así, cuando deseemos referirnos a un conjunto de información del mismo tipo, como una unidad de almacenamiento única, bastará con crear un

archivo dándole el nombre que considere oportuno. La segunda abstracción es lo que se denomina **directorio**. Los directorios son conjuntos de archivos agrupados de acuerdo con un libre criterio (teniendo en cuenta los usuarios que lo han creado, o el contenido de los archivos, etc.). El sistema operativo crea y mantiene por cada directorio un índice con los nombres de los archivos que éste contiene, así como información sobre cada uno de ellos, tal como espacio que ocupa, tipo, fecha y hora en que por última vez se ha escrito sobre él, dirección física donde se encuentra, etc. De este modo la estructura global del sistema de archivos es un árbol en el que los nodos internos (los que constituyen la raíz de un subárbol) son directorios y los nodos hoja son archivos.

Sobre estas estructuras de datos, el sistema operativo permite al usuario realizar, entre otras, las siguientes funciones:

- **creación** de archivo o directorio.
- **borrado** de archivo o directorio.
- **listar** el índice de un directorio.
- **copiar** de archivo o directorio.

Al ser estas funciones propias del sistema operativo y en general los lenguajes de programación han sido diseñados para que los programas no tengan que realizar directamente ninguna de estas operaciones, ya que basta con invocar al sistema operativo para que las lleve a cabo cuando sean necesarias.

6.4. ORGANIZACIÓN DE ARCHIVOS

Hay diferentes formas de estructurar u organizar los registros que componen un archivo sobre un soporte de información. La eficiencia en la utilización del archivo depende de la organización del mismo; por ello se debe optar por una u otra organización atendiendo a la forma en que se va a usar el archivo. Las principales organizaciones de archivos son:

Secuencial. Los registros se encuentran en cierto orden, yuxtapuestos consecutivamente y por tanto han de ser leídos, necesariamente, según este orden. En la organización secuencial los registros carecen de un orden especial, estando situados según el orden temporal de su inclusión en el archivo; si se desea que estén ordenados según otro criterio, el campo clave por ejemplo, debe hacerse programándolo adecuadamente. Por lo general, en un archivo secuencial al final del archivo físico, se graba la marca de final de fichero (end-of-file, **EOF**), que en la mayoría de lenguajes se asocia a una función lógica, **eof**, verdadera cuando se alcanza el final de fichero y falsa en caso contrario. Si un archivo está vacío, sólo contiene la marca final de archivo.

Los archivos secuenciales son los que ocupan menos memoria y son útiles cuando se desconoce a priori la cantidad de datos a almacenar (además se pueden utilizar para manejar registros de longitud variable). En general, son muy empleados para el almacenamiento de información, cuyos contenidos sufran pocas modificaciones en el transcurso de su vida útil. Un caso especial de los archivos secuenciales son los **archivos de texto** donde cada registro es simplemente un carácter o código de control. Es decir, un archivo de texto es simplemente una secuencia de caracteres que incluyen ciertos caracteres especiales o de control (p.e. el carácter fin_de_línea). Una de las aplicaciones más comunes de este tipo de archivos es la de contener documentos creados por programas como los procesadores de texto y los editores. El código de los programas también se almacena en archivos de texto, aunque la estructura del código sea desconocida para el sistema de archivos.

Secuencial Indexada.- En esta organización se dispone de una tabla de índices adicional; entenderemos como índice, una referencia que permite obtener de forma automática la ubicación de la zona del archivo físico donde se encuentra el registro buscado. Este permite localizar un registro por medio de su clave sin recorrer previamente todos los que le preceden. Un diccionario sería un ejemplo de archivo secuencial indexado, ya que en cada página tenemos dos niveles, el superior que nos dice cual es la letra inicial de la palabra y el inferior la cabecera de cada página, de forma que en un ordenador, guardaríamos en la tabla de índices las letras y las cabeceras, que nos dicen en que pagina ir a buscar la palabra deseada. La organización secuencial indexada implica un mantenimiento de las tablas de índices y una previsión inicial de la cantidad máxima de registros que va a contener.

En general, al igual que un diccionario, cada archivo secuencial indexado consta de dos archivos el de índices y el de datos, el primero es secuencial y contiene las claves del último registro de cada bloque físico del archivo y la dirección de acceso al primer registro del bloque y en el segundo, los registros de datos, clasificados en orden ascendente por el campo clave.

Directa.- En está organización, la ubicación del registro en el soporte físico, se obtiene directamente a partir de funciones que la obtienen a partir del valor de la clave, mediante un algoritmo de transformación **(hashing)** de ésta. Un archivo para que pueda estar dotado de una organización directa tiene que cumplir dos condiciones: a) que sus registros sean de longitud fija y b) su propio tamaño tiene que estar prefijado, lo que determina la distribución de la información, al tiempo que limita la cantidad de registros que podrá contener. Conviene saber que las funciones *hash* de conversión de clave a dirección, son numerosas y están basadas en diferentes métodos, aunque su estudio supera el ámbito de este texto.

Digamos finalmente, que la organización más sencilla y más comúnmente empleada es la secuencial , aunque no sea la más eficiente. Todos los lenguajes de

programación permiten tratar con archivos secuenciales, mientras que el tratamiento de archivos indexados y archivos de organización directa no está previsto en todos los lenguajes.

6.5. OPERACIONES SOBRE ARCHIVOS

Desde el punto de vista del programador, los archivos interesan porque los programas típicamente operan sobre ellos, para leer ó escribir en los mismos. Sin embargo, ya hemos dicho que no debe preocuparse de los detalles del hardware dado que la manipulación directa del archivo corresponden al sistema operativo. Los programadores deben emplear determinados procedimientos o funciones para comunicarle al S.O. la operación a realizar y obtener una respuesta de éste.

El sistema operativo debe mantener ciertas informaciones sobre cada fichero que esté manipulando, tales como el soporte en que se encuentra el archivo, el tipo de organización del mismo, el lugar donde éste empieza o la posición actual dentro de un archivo (que indica dónde se va a producir el próximo acceso al archivo, si se trabaja con acceso secuencial). Todo ello esta contenido en un **descriptor de fichero** asociado a cada archivo que se esté utilizado en un momento determinado. Vamos a ver los procedimientos básicos, que con la ayuda del sistema operativo, los programas pueden llevar a cabo sobre los distintos tipos de ficheros.

6.5.1 APERTURA Y CIERRE DE UN ARCHIVO

Para que un programa pueda operar directamente sobre un archivo, la primera operación que debe realizar es la **apertura** del mismo. En la misma, el programa emplea una subrutina identificando el archivo con el que quiere trabajar (mediante un nombre y, según el caso, el soporte donde se encuentra) y el modo en que va a emplearlo (este segundo aspecto varía según el lenguaje con que se trabaje). El sistema operativo construye a partir de estas informaciones un descriptor de fichero, de manera que el programa ya no se referirá al archivo por su nombre (que es un identificador externo al programa), sino por un número o variable asociado a este descriptor, que a partir de ahora será un identificador interno del programa. Hacemos notar que el modo de apertura determina las operaciones que se podrán realizar sobre el mismo: por ejemplo no se puede escribir en un archivo si su descriptor de fichero 'sabe' que ha sido abierto para 'sólo lectura'. Además, cuando se abre un fichero, la posición actual dentro del archivo es el comienzo del mismo.

Cuando un programa no vaya a acceder más a un archivo, es necesario indicar al sistema operativo esta circunstancia. Con ello el sistema operativo libera el descriptor de fichero y se asegura que el archivo queda debidamente almacenado en la memoria secundaria. Para cerrar un archivo simplemente se utiliza la

subrutina de **cierre** indicando el archivo por medio de su identificador interno (número o variable).

Evidentemente, para poder utilizar un archivo, éste tiene que existir. Por ello el fichero deberá haber sido creado en algún momento, y recordemos que la **creación de un fichero** es una tarea propia del sistema operativo. Asimismo, al abrir un fichero para su lectura, las informaciones de este archivo tienen que haber sido almacenadas sobre un soporte y ser utilizables; un intento de lectura en un fichero inexistente produce indefectiblemente un error.

La situación de escritura es diferente: si en la apertura damos el nombre de un fichero para escribir datos en él y no existe ningún fichero con ese nombre, en muchos lenguajes ello significará que el sistema operativo lo creará de modo automático (en caso contrario lo habrá hecho el programador previamente). Si ya existiera un fichero con el mismo nombre, sus contenidos serían borrados y empezaría a escribiese desde un principio. La forma de evitar el borrado de contenidos si se desea añadir datos nuevos a los ya existentes en un archivo es abrirlo en modo 'añadir': los datos nuevos se escribirán en el archivo sin borrar los anteriores, en el bien entendido que haya espacio disponible para ello.

En los casos en que la apertura requiere la creación de un nuevo archivo por parte del sistema operativo, éste necesita saber:

- *nombre dispositivo*: indica el soporte donde se situará el archivo;
- *nombre del archivo*: que lo identifica entre los restantes en el mismo soporte;
- t*amaño del archivo*: indica el espacio necesario para la creación del archivo;
- *organización del archivo*: tipo de organización del archivo;
- *tamaño del bloque o registro físico*: cantidad de datos que se leen o escriben en cada operación de entrada/salida (E/S).

además de algunas indicaciones acerca del directorio al que se va a incorporar.

Algunos de estos datos tienen valores por defecto, esto es, valores que tomarán si no se indica nada distinto. El proceso de creación puede no ser posible por generar *una serie de errores* entre los que se pueden señalar:

- Otro archivo con el mismo nombre ya existía en el soporte.
- El dispositivo no tiene espacio disponible para crear otro nuevo archivo.
- El dispositivo no está operativo.
- Existe un problema de hardware que hace abortar el proceso.
- Uno o más de los parámetros de entrada en la instrucción son erróneos.

6.5.2 LECTURA Y ESCRITURA EN UN ARCHIVO

Básicamente, la **lectura** de un archivo consiste en transferir información del archivo a la memoria principal, mientras que la **escritura** en un archivo es la transferencia de información de la memoria principal a un archivo. Para ello, invocaremos la subrutina de lectura o escritura respectiva indicando a que archivo queremos acceder (mediante su identificador interno del programa) y la información que queremos transferir. En general, para una lectura debemos indicar el lugar de la memoria principal (indicado bien por el nombre de una variable, bien por un puntero) donde se desea situar los datos procedentes del archivo. En el caso de escritura indicamos igualmente qué datos de la memoria principal deseamos transferir al archivo (determinados asimismo, por la variable que los contiene o por el puntero que indica su posición en la memoria). Esta descripción genérica de las operaciones debe particularizarse para cada tipo de organización de archivo posible.

En los archivos secuenciales, la lectura o escritura se realizan en el archivo a partir de la posición actual en el mismo (contenida en el descriptor de fichero) que es en principio la posición siguiente a la del último acceso al archivo que se hubiera producido. En caso de que el soporte lo permita (soporte direccionable) y conociendo la posición de un dato determinado dentro del archivo podremos cambiar la posición actual dentro del mismo llamando a la subrutina de **posicionamiento** y efectuar el siguiente acceso a partir de la nueva posición. Es especialmente importante en la lectura de estos archivos, detectar cuando se ha alcanzado el final del fichero y no hay más datos que leer. En algunos lenguajes, la propia subrutina de lectura indica cuando se encuentra la marca de final de fichero. En otros se dispone de una subrutina de **detección de fin de fichero** que conviene consultar antes de realizar una lectura.

En los archivos secuenciales indexados los accesos no utilizan la posición actual, sino que deben indicar el valor del campo clave para buscar el registro, de modo que si no se encuentra un registro con dicha clave, hay que tener en cuenta esta circunstancia. Igualmente la escritura se realiza indicando el valor de la clave y el registro se sitúa en el archivo en la posición adecuada, actualizando las tablas de índices si fuera necesario.

En los archivos de organización directa, que según indicamos eran de tamaño prefijado y registros de longitud fija, usando la función *hash* que utilice el archivo, a partir de cada clave se obtiene la posición del archivo donde debe encontrarse el registro. El problema específico de esta organización (genérico de toda función *hash*) es que la relación no es biunívoca, esto es, claves distintas pueden dar lugar a la misma posición, lo que en la organización directa suscita el problema de las colisiones, que son resueltas usando técnicas especiales.

Para la escritura indicamos el registro con su campo clave lo cual determina la posición en el archivo. Sin embargo, si un campo con clave distinta ya ocupara dicha posición (a consecuencia de tener el mismo resultado de la función *hash*) se produce una colisión. La solución parcial a las colisiones es la existencia de una zona de rebose donde emplazar el registro que ha colisionado. Para la lectura indicaremos el campo clave, lo que permite calcular la posición donde debe encontrarse el registro, comprobándose que efectivamente la clave del registro encontrado sea la solicitada. Si no fuera así, se busca el registro en la zona de rebose. Si tampoco se encuentra allí, se indicará el error correspondiente.

6.6. PROCESAMIENTO DE ARCHIVOS

La vida de todo archivo comienza cuando se crea y acaba cuando se borra. Durante su existencia es objeto de constante procesamiento, que con mucha frecuencia incluye acciones de **consulta** o búsqueda y de **actualización**. En el caso de la estructura archivos, entenderemos como **actualización,** además de las operaciones, vistas para vectores y listas enlazadas en el capitulo anterior, de introducir nuevos datos (**altas**) o de eliminar alguno existente (**bajas**), la **modificación** de datos ya existentes, (operación muy común con datos almacenados). En esencia, es la puesta al día de los datos del archivo.

Una operación de **alta** en un archivo consiste en la adición de un nuevo registro. En un archivo de empleados, un alta consistirá en introducir los datos de un nuevo empleado. Para situar correctamente un alta, se deberá conocer la posición donde se desea almacenar el registro correspondiente: al principio, en el interior o al final de un archivo. El algoritmo de ALTAS debe contemplar la comprobación de que el registro a dar de alta no existe previamente.

Una **baja** es la acción de eliminar un registro de un archivo. La baja de un registro puede ser lógica o física. Una baja lógica supone el no borrado del registro en el archivo. Esta baja lógica se manifiesta en un determinado campo del registro con una bandera, indicador o *"flag"* -carácter *. $, etc.,-, o bien con la escritura o rellenado de espacios en blanco en el registro dado de baja.

Una **modificación** en un archivo consiste en la operación de cambiar total o parcialmente el contenido de uno de sus registros. Esta fase es típica cuando cambia el contenido de un determinado campo de un archivo; por ejemplo, la dirección de un empleado. El proceso consiste en la lectura del registro procedente del archivo, modificación de su contenido y reescritura, total o parcial del mismo en el archivo.

Estas acciones las realizan los programas actuando a nivel de registro a partir de las operaciones básicas descritas en el apartado anterior. La mayor parte de estas

acciones de procesamiento implican la localización de un registro concreto, para luego actuar sobre él (leerlo, escribir o cambiar parte de él, borrarlo, etc.). La facilidad de ejecución de cada una de ellas y su eficiencia en término de coste temporal depende de la organización del fichero, como veremos a continuación.

6.6.1 PROCESAMIENTO DE ARCHIVOS SECUENCIALES

En archivo secuencial los registros se insertan en el archivo en orden cronológico de llegada al soporte, es decir, un registro se almacena inmediatamente a continuación del registro anterior. Puesto que los archivos secuenciales terminan con una marca final de archivo, cuando se tengan que añadir registros a un archivo secuencial, éstos se añadirán en las marcas fin de archivos.

6.6.1.1 PROCESO PARA LA ESCRITURA DE REGISTROS

Este proceso es secuencial, y para ello tendremos que ejecutar un programa que permite la entrada de datos del archivo desde el terminal. El sistema usual es el *interactivo* en el que el programa solicita los datos al usuario que los introduce por teclado, hasta que se introduce una marca final de archivo EOF o FF) que supone el final físico del archivo. Esta operación tiene dos variantes:

- utilizar por primera vez el archivo.
- añadir datos al archivo ya creado, después del último registro del mismo.

La primera supone que el archivo tiene que ser creado durante el proceso, mientras que la segunda, asume que se trabaja con un registro que ya contiene datos y que por tanto ya existe. Dejando aparte la cuestión de la creación del archivo, el proceso que nos interesa requerirá los siguientes pasos:

* abrir el archivo;
* leer datos del registro.
* grabar registro.
* cerrar archivo.

El algoritmo de creación con inclusión del menú de opciones es el siguiente:

```
algoritmo creación
   inicio
     {menú de opciones}
     escribir `1 creación archivo nuevo´.
     escribir `2 añadir datos al archivo`.
     leer opción
     si opción = 1
        entonces
```

```
            abrir archivo nuevo para creación
         sino
            abrir archivo para añadir datos
      fin_si
      {introducción de datos en el archivo}
      mientras no se alcance el fin de archivo (EOF) hacer
         leer datos de un registro
         escribir (grabar) registro en el archivo
      fin_mientras
      cerrar archivo
fin
```

6.6.1.2 CONSULTA

El proceso de búsqueda o consulta de una información en este tipo de archivo, se debe efectuar obligatoriamente en modo secuencial. Por ejemplo, si se desea consultar la información contenida en el registro 50, se deberán leer previamente los 49 primeros registros que le preceden en orden secuencial. En el caso de un archivo de personal si desean buscar un registro determinado correspondiente a un determinado estudiante, será necesario recorrer, -leer- todo el archivo desde el principio hasta encontrar el registro que se busca o la marca final de archivos, si el registro correspondiente al estudiante buscado, no se encuentra en el archivo.

Así, para el caso de un archivo de *n* registros, el número de lecturas de registros efectuadas son:

- mínimo 1, si el registro buscado es el primero del archivo;
- máximo *n*, si el registro buscado es el último o no existe dentro del archivo.

Por término medio, el número de lecturas necesarias para encontrar un determinado registro es: (n+1)/2

El tiempo de acceso al periférico será influyente en las operaciones de lectura/escritura**.** Así, mientras en el caso de una lista o vector de *n* elementos almacenados en memoria central puede suponer tiempos de microsegundos o nanosegundos, en el caso de un archivo de *n* registros en un disco los tiempos de acceso son de milisegundos o fracciones/múltiplos de milisegundos. Esto supone un tiempo de acceso de 1.000 a 100.000 veces más grande una búsqueda de información en un soporte externo que en memoria central.

Para obtener un algoritmo de consulta en un archivo secuencial, se requerirá un diseño previo de la presentación de la estructura de los registros correspondientes en el dispositivo de salida, de acuerdo con el número y longitud de los campos que

lo integran. En el algoritmo siguiente hay que distinguir la lectura desde una terminal del registro buscado, de las lecturas de registros, pertenecientes al archivo, que se efectúan secuencialmente.

```
algoritmo consulta
  inicio
    leer registro buscado (campo x)  {registro buscado con un campo clave x}
    encontrado ← falso
    abrir archivo para lectura
    leer registro
    N ← 1
    mientras registro < > FF hacer    {FF, fin de fichero}
      si  registro (campo x) = registro buscado (campo x)
        entonces
          escribir "el registro buscado existe"
          encontrado ← verdadero
          registro = FF
        sino
          leer registro
          N ← N + 1
      fin-si
    fin_mientras
    si encontrado ← falso
        entonces
        escribir 'registro no encontrado, después de consultar' N-1 'registros'
    fin-si
cerrar archivo
fin
```

6.6.1.3 ALTAS

La operación de dar de alta un determinado registro es similar a la operación ya descrita anteriormente de añadir datos a un archivo. Es importante remarcar que en un archivo secuencial sólo permite añadir datos al final del mismo. En otro caso, si se quiere insertar un registro en medio de los ya presentes en el archivo, sería necesaria la creación nueva del archivo.

El algoritmo para dar de alta un registro al final del fichero es como sigue:

```
algoritmo altas
    leer registro de alta
    inicio
```

```
    abrir archivo para añadir
    mientras haya mas registros hacer {algunos lenguajes ahorran este bucle}
        leer datos del registro
    fin_mientras
        escribir (grabar) registro de alta en el archivo
    cerrar archivo
fin
```

6.6.1.4 BAJAS

Existen dos métodos para dar de baja a un registro en un archivo secuencial, donde no es fácil eliminar un registro situado en el interior de una secuencia: Para ello podemos seguir dos métodos:

1) Utilizar y por tanto crear un segundo archivo auxiliar transitorio, también secuencial, copia del que se trata de actualizar. Se lee el archivo completo registro a registro y en función de su lectura se decide si el registro se debe dar de baja o no. En caso afirmativo, se omite la escritura en el archivo auxiliar. Si el registro no se va a dar de baja, este registro se reescribe en el archivo auxiliar (Ver Figura 6.3).

Tras terminar la lectura del archivo original, se tendrán dos archivos: *original* (o maestro) *y auxiliar*. El proceso de *bajas* del archivo concluye borrando el archivo original y cambiando el nombre del archivo auxiliar por el del inicial.

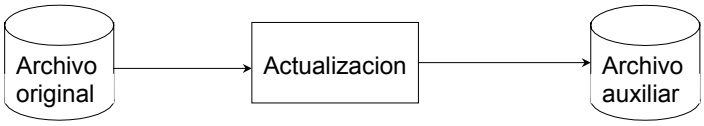

Fig. 6.3. *Borrado y Modificación*

2) Guardar o señalar los registros que se desean dar de baja con un indicador o bandera que se guarda en un array; de esta forma los registros no son borrados físicamente, sino que son considerados como inexistentes. Inevitablemente, cada cierto tiempo, habrá que crear un nuevo archivo secuencial con el mismo nombre, en el que los registros marcados no se grabarán.

6.6.1.5 MODIFICACIONES

El proceso de modificación de un registro consiste en localizar este registro, efectuar dicha modificación y a continuación reescribir el nuevo registro en el archivo. El método que se utiliza es el mismo que el visto para bajas.

Vamos a suponer que recorremos el archivo, que llamaremos maestro y que preguntamos al operador si desea modificar cada uno de sus registros. No habría ninguna dificultad, en el caso de que la modificación se quisiera hacer en un solo registro, para que éste se localizara individualmente (como hicimos durante la consulta) utilizando el correspondiente campo clave. El algoritmo correspondiente seria:

```
algoritmo modificación
   inicio
     abrir archivo maestro para lectura
     abrir archivo auxiliar para creación
     leer registro del maestro
     mientras registro del maestro <> EOF hacer
        escribir "Modificar (S/N)"
        leer respuesta
        si respuesta = "S" entonces
             llamar_a subprograma de modificación
        fin_si
        escribir registro en el archivo auxiliar
     fin _mientras
     cerrar archivo maestro
     cerrar archivo auxiliar
     borrar archivo maestro
     cambiar nombre del archivo auxiliar por nombre del archivo maestro
fin
```

El subprograma de modificación de un registro consta de unas pocas instrucciones en las que se debe introducir por teclado el registro completo con indicación de todos sus campos o, por el contrario, el campo o campos que se desea modificar. El subprograma en cuestión podría ser:

```
subprograma modificar
inicio
   escribir "Escribir R, para modificar el Registro completo"
   escribir" Escribir C, si solo se quiere modificar algunos Campos individuales"
   leer opción
   {validar opción válida}
   según_la opción hacer
      R:        visualizar campos
        introducir todos los campos del registro
```

```
        C:      solicitar campos a modificar
                elegir campos
                introducir campos
    fin_según
fin
```

6.6.2 PROCESAMIENTO DE FICHEROS SECUENCIALES INDEXADOS

Debido a que esta organización de archivos no está soportada por todos los lenguajes de programación, no vamos a tratar en detalle su correspondiente procesamiento. Recordemos que estos archivos constan de un área de datos que agrupa los registros y un área de índices. Una de las ventajas de la utilización de índices en un fichero secuencial reside en que la búsqueda de registros puede hacerse de forma mucho mas rápida que la que hemos visto hasta ahora. El almacenamiento del área de índices en memoria principal, mientras el programa se ejecuta, permite ejecutar la búsqueda binaria, con las ventajas que sabemos que ésta conlleva.

Las operaciones que se pueden llevar a cabo con un archivo secuencial indexado, son las habituales, solo que en su programación hay que tener en cuenta que se tiene que construir y manejar la tabla de índices (claves-direcciones) y que los registros se graban en el orden de las claves.

6.6.3 PROCESAMIENTO DE FICHEROS DE ORGANIZACIÓN DIRECTA

Las operaciones con archivos de acceso directo son las usuales, ya vistas anteriormente, aunque teniendo en cuenta que para cualquier acceso a un registro inicialmente hay que buscar en la tabla de índices una clave igual a la correspondiente al elemento que se desea (para obtener la dirección de la clave y poder acceder el registro correspondiente) y que desde el punto de vista del programador, solo existen los registros alcanzables a través de los índices que existen en cada momento.

En un soporte direccionable -normalmente un disco-, cada posición se localiza por su *dirección absoluta*-número de pistas y número de sector en el disco-. Los archivos de acceso directo manipulan direcciones relativas en lugar de absolutas, lo que hace al programa independiente de la posición absoluta del archivo en el soporte. La función de conversión transformará las claves en direcciones relativas. Suponiendo que existen N posiciones disponibles para el archivo, la conversión de clave producirá una dirección relativa en el rango 1 a N. En el caso en que dos

registros distintos produzcan la misma dirección, se dice que se produce una colisión que por otro lado son inevitables y que se controlan mediante dos métodos básicos (sobre los que no insistiremos mas):

Buscar una nueva dirección libre en el mismo espacio del archivo.
Asignar el registro a la primera posición libre de la zona de excedentes.-

6.6.3.1 PROCESO DE ESCRITURA DE REGISTROS

El proceso de introducción de datos en un archivo directo o aleatorio, que en un abuso de lenguaje llamaremos creación, consiste en ir introduciendo los sucesivos registros en el soporte que los va a contener y en la detección obtenida resultante del algoritmo de conversión. Si al introducir un registro se encuentra ocupada la dirección, el nuevo registro deberá ir a la zona de sinónimos o de excedentes, sobre cuyos detalles ya hemos dicho que no entraríamos. Bastará con saber que estos archivos necesitan dos zonas de almacenamiento, la principal constituida por registros de longitud constante y la de sinónimos donde secuencialmente se almacenan aquellos registros cuyas claves coinciden o son direcciones sinónimas de registros grabados en la zona principal.

Veamos un algoritmo que desarrolla este proceso de escritura:

```
algoritmo escri-reg
    inicio
      abrir archivo
      leer registro
      mientras < > FF hacer
          calcular dirección mediante algoritmos de conversión
          escribir "dirección libre S/N"
          leer respuesta
          si respuesta = "S"
            entonces
                grabar registros
            sino
        buscar espacio en área de sinónimos
        grabar registro
    fin_si
  leer registro
  fin_mientras
fin
```

6.6.3.2 CONSULTA.-

A partir de la entrada del número o números de registros a consultar. Las operaciones a realizar son:

- definir clave del registro buscado.
- aplicar algoritmo de conversión clave a dirección.
- lectura del registro ubicado en la dirección obtenida.
- comparación de las claves de los registros leído y buscado.
- exploración secuencial del área de excedentes, si no se encuentra el registro en este área es que no existe.

Veamos un esquema que facilita el proceso de consulta:

```
mientras < > EOF hacer
   leer registro R
   ir a subprograma de obtención de la dirección
   leer registro S
   si R=S
   entonces
     llamar_a subprograma consulta
   sino
     leer área de sinónimos
     si FF en área de sinónimos   ¨{no hay ningún sinónimo}
        entonces
            escribir "registro no existe"
        sino llamar_a subprograma consulta
       fin_si
   fin_si
fin_mientras
```

6.6.3.3 ALTAS

Para dar de alta a un registro, se debe introducir su número de orden y contenido. La inserción de un registro en el archivo, supone utilizar un campo adicional, *alta/baja* del registro, SW (interruptor) que tome el valor 1 ó 0, según que el registro esté dado de alta o de baja, ya que como veremos a continuación, la baja es solo de caracter lógico y es posible que demos de alta un registro que ya existe, pero que fue dado de baja anteriormente.

El algoritmo para llevar a cabo el procedimiento de dar de alta es:

```
algoritmo altas
inicio
   SW = 0
   repetir
      leer" número de registro a dar de alta". NR
      si 1<NR<ALTO  {ALTO es el número máximo de registros permitidos}
         entonces
            leer registro NR
            si SW = 1
               entonces
                  escribir "registro ya existe"
               sino
                  SW=1
                  leer datos del registro
                  escribir SW y datos en el registro NR
            fin_si
         sino
           escribir "el numero de registro no es correcto."
      fin_si
   hasta_que no se deseen más altas
fin
```

6.6.3.4 BAJAS

Para realizar una baja, basta con indicarlo en el campo SW del registro correspondiente, indicando, en el caso de que el registro exista en el archivo, que su valor pase de 1 a 0. Este tipo de baja es solo lógica, lo que significa que, pese a usar un registro dado de baja, éste sigue ocupando el mismo espacio que si estuviera dado de alta.

El algoritmo para llevar a cabo el procedimiento de dar de baja es:

```
algoritmo baja
repetir
   leer NR                { número de registro}
   si 1 ≤ NR ≤ TOTAL
      entonces
         leer registro NR
         si SW=0
           entonces
              escribir" el registro no existe"
           sino
            escribir registro
```

```
                si la opción de baja es correcta
                  entonces
                    SW = 0
                    escribir SW en el registro NR
                fin_si
            fin_si
        sino
            escribir "número de registro no correcto".
    fin_si
  hasta-que no mas bajas
fin
```

6.6.3.5 MODIFICACIONES.-

En un archivo aleatorio para modificar un determinado registro hay que localizarlo, para lo que hay que introducir el correspondiente número de registro, modificar el contenido y reescribirlo. Veamos su algoritmo correspondiente:

```
algoritmo modificación
inicio
    repetir
        leer "número de registro", NR
        si 1 ≤ NR ≤ TOTAL
        entonces
            leer registro NR
            si SW= 0
              entonces
                  escribir "registro no existe"
              sino
                  escribir registro
                  leer modificaciones
                  escribir nuevo registro
              fin_si
            sino
              escribir "número de registro no correcto"
        fin_si
    hasta_que no haya más modificaciones.
fin
```

6.7. TIPOS DE ARCHIVOS

En una aplicación informática se pueden utilizar archivos para realizar funciones diversas. Conocer la función que va a desempeñar un archivo concreto es fundamental a la hora de decidir cómo se debe organizar éste. En una primera aproximación podemos clasificar los archivos en la forma que se indica en la Tabla 6.1.

Un **archivo permanente** contiene información relevante para una aplicación, es decir, los datos necesarios para el funcionamiento de la misma. Su vida es larga al menos comparable a la de la aplicación para la que ha sido creado) y generalmente no puede generarse de una forma inmediata a partir de otros archivos.

Tabla 6.1.- *Clasificación de los archivos según el uso que se hace de ellos.*

-ARCHIVOS PERMANENTES
Archivos maestros
Archivos constantes
Archivos históricos
-ARCHIVOS TEMPORALES
Archivos intermedios
Archivos de maniobras
Archivos de resultados

Un **archivo temporal** contiene información relevante para un determinado proceso o programa, pero no para el conjunto de la aplicación. Se genera a partir de los datos de archivos permanentes o para actualizar éstos, y su vida es generalmente muy corta.

Los archivos permanentes se pueden clasificar en:

Archivos maestros.- Un archivo maestro, tal como lo hemos venido viendo, contiene el estado actual de los datos susceptibles de ser modificados durante la aplicación informática que se programa. Es el núcleo central de la aplicación. Todos los procesos están, en general, orientados a actualizar el archivo maestro o a obtener resultados de él. Un ejemplo de este tipo de archivo es el archivo de clientes de un banco, en el que los registros contienen información de identificación de clientes, su saldo en cuenta, etc.

Archivos constantes.- Un archivo constante es aquel que contiene datos fijos para la aplicación. En él las modificaciones son infrecuentes, normalmente se accede a él sólo para consultar. Serán archivos constantes lo que contengan los intereses para distintos tipos de cuentas bancarias, la ubicación de estantes en una biblioteca, la capacidad de las aulas de un centro, una tabla de números primos, etc.

Archivos históricos.- Un archivo histórico es aquel que contiene datos que fueron actuales en tiempos anteriores. Se conservan para poder reconstruir la situación actual o situaciones anteriores. En algunos casos puede estar formado simplemente por los registros borrados del archivo maestro. Un archivo histórico puede contener, por ejemplo, los clientes que se han dado de baja en una entidad bancaria.

Los archivos temporales se pueden clasificar en:

Archivos intermedios.- Se utilizan para almacenar resultados de un programa que han de ser utilizados por otro, dentro de una misma tarea.

Archivos de maniobras.- Se utilizan para almacenar datos propios de un programa que éste no puede conservar en memoria principal, por falta de espacio en ésta. Se encuentran normalmente en programas de cálculo numérico, compiladores y editores. Su vida es siempre menor que es el tiempo de ejecución del programa.

Archivos de resultados.- Se utilizan para almacenar datos elaborados que van a ser transferidos a un dispositivo de salida. Por ejemplo, un **archivo de impresión**, que contiene datos que van a ser transferidos a una impresora.

Veamos un ejemplo próximo a una aplicación real, para ver el papel que juega cada uno de los tipos de archivo.

Ejemplo 1:

Supongamos que se desea almacenar y procesar los datos relativos a ocupación y cobros de un hotel. Será necesario, en primer lugar, almacenar una serie de datos fijos, como son las características de cada habitación, la tarifa de precios, etc. Esta información estará contenida en archivos, que por su naturaleza serán **permanentes y constantes**. Los registros de estos archivos pueden tener la siguiente estructura:

FDES: Archivo de descripción de habitaciones

- -Número de habitación.
- -Número de camas.
- -Categoría
- -Exterior/interior.

FPRE: Archivo de precios

- -Categoría de la habitación.
- -Número de camas.
- -Precio por día en temporada alta.

-Precio por día en temporada baja.

Además, se deberán crear otros archivos que contendrán datos variables. Concretamente, debe existir un archivo con información sobre el estado actual de ocupación del hotel, que será un archivo **permanente**, éste será el archivo **maestro** de la aplicación. Este archivo podrá tener la siguiente estructura:

HOTEL: Archivo de ocupación del hotel
-Número de habitación
-Inservible (si/no)
-Ocupada (sí/no).
-Número de personas.
-DNI del inquilino.
-Apellidos y nombre del cliente
-Fecha prevista para la partida.
-Fecha de ocupación.

Cada vez que un cliente abandona el hotel se creará un archivo de **resultados** conteniendo la factura que se debe cobrar a dicho cliente. Este archivo será transferido posteriormente a impresora, después de lo cual se borrará. Se trata, por tanto, de un archivo temporal. Los datos de ocupación de habitaciones del archivo HOTEL pasarán, una vez dejada la habitación por el cliente, a un archivo **histórico**.

6.8. BASES DE DATOS

En una aplicación convencional con archivos, éstos se diseñan siguiendo las instrucciones de los correspondientes programas. Esto es, una vez planteado se decide si debe haber ó no archivos, cuántos deben ser, qué organización contendrá cada uno, qué programas actuarán sobre ellos y cómo lo harán. Esto tiene la ventaja, en principio, de que los programas son bastante eficientes, ya que la estructura de un archivo era pensada "para el programa" que lo va a usar. Sin embargo, esta forma de actuar está plagada de graves inconvenientes. Por un lado, los programas que se realizan con posterioridad a la creación de un archivo pueden ser muy lentos, al tener que usar una organización pensada y creada "a la medida" de otro programa previo, que realiza procesos diferentes. Por otra parte, si se toma la decisión de crear nuevos archivos para los programas que se han de realizar, se puede entrar en un proceso de degeneración de la aplicación, ya que:

- gran parte de la *información aparecerá duplicada* en más de un archivo (se denomina **redundancias**) ocupando la aplicación más espacio del necesario;

- al existir la misma información en varios archivos, *los procesos de actualización* se complican de forma innecesaria, dando lugar a una propagación de errores;
- se corre el riesgo de tener datos incongruentes entre los distintos archivos. Por ejemplo, tener dos domicilios diferentes de un mismo individuo en dos archivos distintos (por estar uno actualizado y el otro no).

Ejemplo 2:

La dirección de una empresa se plantea la necesidad de contar con información de sus empleados a efectos de envío de circulares y de selección de personal para determinadas tareas. Una solución aceptable para esta aplicación, puede ser crear un archivo con los siguientes campos: DNI, Nombre, Dirección, Puesto ocupado, Fecha ingreso, Fecha nacimiento. Teléfono, a partir de los cuales se puede obtener la información deseada. Si posteriormente la misma empresa decide informatizar su nómina tendrá que optar por una de las siguientes posibilidades:

- Crear una aplicación completamente independiente de la anterior, que dispondrá de nuevos archivos de datos: En este caso habrá datos duplicados y con una alta posibilidad de que en algunos casos sean incoherentes, pues los procesos de modificación serán independientes.
- Modificar los archivos existentes. Esto implica reescribir o, al menos, modificar los programas de aplicación anteriores.
- Crear archivos que contengan la información de la nueva aplicación no usada el la anterior. En estos archivos deberá haber al menos un campo de enlace con el archivo anterior, por ejemplo el DNI. Nos encontramos aquí en una situación intermedia, hay redundancias, no es necesario reescribir todos los programas, pero los nuevos programas serán más lentos.

Es una aplicación convencional con archivos aparecen, pues, los siguientes problemas:

Dificultad de mantenimiento. Si hay archivos con información parcialmente duplicada, realizar las actualizaciones necesarias en un problema complejo y costoso. Normalmente es necesario actualizar varios archivos con diferentes organizaciones. Si la actualización no se realiza correctamente se tendrán informaciones incoherentes.

Redundancia. Se dice que hay redundancia cuando hay datos que no aportan realmente información real. Es decir, su valor se puede deducir del de otros datos.

Supóngase que en el archivo de selección del ejemplo anterior se hubiese incluido un campo con la antigüedad del empleado. Este campo sería redundante, pues su valor se puede obtener del valor del campo Fecha de ingreso y de la fecha actual. Un caso trivial de redundancia se da cuando se incluye más de una vez el mismo dato. Esto no es normal que ocurra en un archivo, pero es frecuente en un sistema de archivos.

Rigidez de búsqueda. El archivo se concibe para acceder a los datos de un determinado modo. Sin embargo, en la mayoría de los casos es necesario (o al menos deseable) combinar acceso secuencial y directo por varias claves.

En el caso de archivo de dirección la organización más apropiada sería directa o indexada, utilizando como clave el DNI del empleado. Esto permite la consulta y la modificación de un empleado concreto, operaciones que serán frecuentes. Sin embargo, estas organizaciones no permiten:

- localizar a un empleado por su nombre,
- obtener una relación de empleados por orden alfabético,
- obtener empleados con fecha de ingreso anterior a una dada,
- .....

Todas estas búsquedas implican procesar el archivo secuencialmente, creando un archivo intermedio, que en algunos casos habrá que ordenar posteriormente.

Dependencia de los programas. En un archivo las relaciones existentes entre campos y registros no están plasmadas en modo alguno. Es el programa , que trabaja con el archivo, el que determina en cada caso dichas relaciones. El programa recibe cada vez que lee una cadena de caracteres. Por tanto, la información de dónde comienza y dónde acaba cada campo, su tipo, etc., es inherente al programa. Esto implica que cualquier modificación de la estructura de un archivo obliga a modificar todos los programas que lo estén usando. Esto ocurre aún en el caso de que la alteración sea ajena al programa. Así, si se aumenta la longitud de un campo, habrá que modificar incluso los programas que no lo usan.

Confidencialidad y seguridad. Uno de los mayores problemas de cualquier sistema de información es mantener la reserva necesaria sobre los datos que contiene. Esto es, por un lado, impedir el acceso de determinados usuarios a determinados datos (**confidencialidad**) y por otro, impedir modificaciones provocadas por usuarios no autorizados (**seguridad**). Si se está trabajando con archivos, el control deberá realizarlo el propio programa, por lo que no se podrá impedir que alguien construya un programa para modificar o ver el contenido del archivo, siempre que el sistema operativo le permita el acceso.

6.8.1 CONCEPTO DE BASE DE DATOS

Las bases de datos surgen como alternativa a los sistemas de archivos, intentando eliminar o al menos reducir sus inconvenientes. Podemos definir una base de datos así:

Una **base de datos** *es un sistema formado por un conjunto de datos y un paquete software para la gestión del mismo, de tal modo que se controla el almacenamiento de datos redundantes, los datos resultan independientes de los programas que lo usan, se almacenan las relaciones entre los datos junto con éstos y se puede acceder a los datos de diversas formas.*

En una base de datos se almacenan las relaciones entre datos junto a los datos. Esto, y el utilizar como unidad de almacenamiento el campo además del registro, es el fundamento de la independencia con los programas de datos.
Normalmente la definición del concepto de base de datos se realiza en forma de requisitos u objetivos que está debe cumplir. Aunque la definición anterior es suficientemente precisa, vamos a exponer algunos requisitos que debe cumplir un buen sistema de base de datos:

Acceso múltiple. Diversos usuarios pueden acceder a la base de datos, sin que se produzcan conflictos ni visiones incoherentes.

Utilización múltiple Cada usuario podrá tener una imagen o visión particular de la estructura de la base de datos.

Flexibilidad. Se podrán usar distintos métodos de acceso, con tiempos de respuesta razonablemente pequeños.

Confidencialidad y seguridad. Se controlará el acceso a los datos (incluso a nivel de campo), impidiéndoselo a los usuarios no autorizados.

Protección contra fallos. Deben existir mecanismos concretos de recuperación en caso de fallo de la computadora.

Independencia física. Se puede cambiar el soporte físico de la base de datos (modelo de discos, por ejemplo), sin que esto repercuta en la base de datos ni en los programas que la usan.

Independencia lógica. Capacidad para que se puedan modificar los datos contenidos en la base, las relaciones existentes entre ellos o incluir nuevos datos, sin afectar a los programas que lo usan.

Redundancia controlada. Los datos se almacenan una sola vez.

Interfaz de alto nivel. Existe una forma sencilla y cómoda de utilizar la base, al menos se cuenta con un lenguaje de programación de alto nivel, que facilita la tarea.

Interrogación directa ("query"). Existan facilidades para que se pueda tener acceso a los datos de forma conversacional.

6.8.2 ESTRUCTURA GENERAL DE UNA BASE DE DATOS

En una base de datos se almacena información de una serie de objetos o elementos. Estos objetos reciben el nombre de **entidades**. Entidad es cualquier ente sobre el que se almacena información. Así, en una base de datos académicos podrá haber información de las siguientes entidades: alumno, profesor, asignatura, centro, plan de estudios, curso, etc. En una base de datos comerciales de una empresa podrán aparecer las siguientes entidades: cliente, producto, vendedor, etc. De cada entidad se almacenan una serie de datos que se denominan **atributos** de la entidad. Puede ser atributo de una entidad, cualquier característica o propiedad de ésta que se considere relevante para la aplicación. Así son atributos de la entidad alumno, para una aplicación administrativa: DNI, apellido y nombre, sexo, fecha de nacimiento, nacionalidad, etc., sin que debieran figurar características como el pensamiento político o la inclinación sexual, que siendo importantes no tienen nada que ver con los requisitos de una aplicación administrativa.

Entidades y atributos son conceptos abstractos. En una base de datos, aunque la tecnología evoluciona constantemente, la información de cada entidad se almacena en **registros,** y cada atributo en **campos** de dicho registro, de forma análoga al almacenamiento en archivos. Sin embargo, cada entidad necesita registros con una estructura específica, mientras que en un archivo, todos los registros tienen la misma estructura. Esto es, en una base de datos hay **diferentes tipos de registros,** uno por entidad. Normalmente se reserva el nombre "registro" para especificar un "tipo de registro", usándose otras denominaciones para especificar cada una de las apariciones de ese registro en la base de datos, **ocurrencia de registro.** Con esta terminología se puede decir que, en la base de datos de uso para una empresa, antes mencionada, hay un registro de vendedor y tantas "ocurrencias" de dicho registro como vendedores tenga la empresa.

Normalmente no es necesario conocer los valores de todos los atributos de una entidad para determinar si dos elementos son iguales. Por lo general, es suficiente con conocer el valor de uno o varios atributos para identificar un elemento. Pues bien, diremos que un conjunto de atributos de una entidad es un **identificador** de dicha entidad si el valor de dichos atributos determina de forma unívoca cada uno de los elementos de dicha entidad y no existe ningún subconjunto de él que sea

identificador de la entidad. Por ejemplo, en la entidad alumno, el atributo DNI es un identificador de esa entidad.

Frecuentemente es necesario buscar una ocurrencia de un registro en una base de datos, conociendo el valor de uno o varios campos. Para que esta operación sea rápida, estos campos deben estar definidos en la base de datos como **clave** o **llave** de búsqueda de dicho registro. En general, podemos decir que una clave es un campo o conjunto de campos, cuyos valores permiten localizar de forma rápida ocurrencias de un registro. La clave puede corresponderse con un identificador de la entidad. Si esto no ocurre, podrá haber varias ocurrencias de registro con el mismo valor de clave. Se dice entonces que la clave admite duplicados.

En una base de datos se almacenan, además de la entidades, las **relaciones** existentes entre ellas. Así, por ejemplo, en la base de datos académicos antes citada hay relaciones entre las siguientes entidades: cursos y alumnos, alumnos y profesores, profesores y asignaturas.

6.8.3 TIPOS DE BASES DE DATOS

Las bases de datos se clasifican tradicionalmente en tres grupos: *jerárquicas, en red y relacionales*. Las dos primeras se diferencian en los tipos de relaciones que permiten. Puede decirse que la estructura jerárquica es un caso particular de la estructura de la red. Cualquier esquema que se cree para una base de datos jerárquica se puede utilizar para una base de red. Las bases de datos relacionales son conceptualmente distintas. En éstas, las relaciones se almacenan y manipulan de forma completamente distinta, como se verá más adelante.

Bases de datos jerárquicas.-

En una base de datos jerárquica sólo se pueden crear estructuras jerárquicas (esto es, en árbol). No es, pues, posible definir relaciones muchos a muchos. Para poder dar una caracterización más precisa de este tipo de base de datos se introduce un nuevo concepto: el de **conjunto**. Un conjunto es una estructura formada por dos registros ligados por una relación uno a muchos. Los registros que forman el conjunto reciben, en éste, los nombres de propietario y miembro, siendo la relación de un propietario a muchos miembros y no acepta estructuras en las que un mismo registro sea miembro de dos conjuntos distintos. Físicamente, una estructura de este tipo se almacena usando punteros como enlace entre los distintos registros de cada conjunto.

Bases de datos en red.-

En una base de datos de red no hay ninguna restricción ni en el tipo de relaciones que se pueden usar, ni en los registros que pueden intervenir en ellas. No obstante se distinguen entre bases de datos en red **simple** y bases de datos en red **compleja**, según permitan o no utilizar relaciones muchos a muchos. En una base de datos en red simple este último tipo de relaciones no está permitido. Una base de datos de red simple puede descomponer en conjuntos, al igual que una base jerárquica.

Bases de datos relacionales.-

A principios de la década de los 70, F.F. Codd planteó una alternativa a las bases de datos jerárquicas y en red, con la pretensión de obtener más flexibilidad que con las bases anteriores y más rigor en el tratamiento de los datos.

Una base de datos relacional está formada por tablas (Ver Figura 6.4). como sabemos, una tabla es una estructura bidimensional formada por una secuencia de registros del mismo tipo.

Número de vuelo	Avión	Origen	Destino	Hora de salida
27	DC-9	Granada	Madrid	8:45
404	DC-10	Madrid	Oslo	10:37
1024	B-727	Barcelona	París	9:45
114	B-727	Santiago	Sevilla	17:21
	DC-9	Madrid	Málaga	15:30

Fig. 6.4. *Ejemplo de representación de relaciones en forma de una tabla*

Si se imponen ciertas condiciones a las tablas, se pueden tratar como **relaciones** matemáticas. De ahí el nombre de este tipo de bases de datos y el hecho de que a las tablas de una base de datos relacional se les denomine **relaciones**. Las tablas deben cumplir las siguientes condiciones:

- Todos los registros de una tabla son del mismo tipo. Para almacenar registros se usan tablas distintas.
- En ninguna tabla aparecen campos repetidos.
- En ninguna tabla existen registros duplicados.
- El orden de los registros en la tabla es indiferente. En cada momento se pueden recuperar los registros en un orden particular.
- En cada tabla hay una clave, formada por uno o varios campos.

Digamos que la mayor parte de bases de datos que están disponibles en la actualidad para ordenadores personales y estaciones de trabajo son relacionales,

habiendo incorporado una serie de ventajas adicionales, aunque éste es un campo de la informática que está en constante evolución.

6.8.4 SISTEMA DE GESTIÓN DE LA BASE DE DATOS

Se denomina **sistema de gestión de la base de datos** (SGDB o **DBMS**, del inglés "Data Base Management System") al conjunto de software destinado a la creación, control y manipulación de la información de una base de datos. Concretamente, Un DBMS debe permitir la realización de las siguientes tareas:

- **Acceso** a los datos desde algún lenguaje de alto nivel.
- **Interrogación** (o recuperación de la información) directa en modo conversacional.
- **Definición** del esquema de la base y de los distintos subesquemas.
- **Organización** física de la base de datos y recuperación tras fallos del sistema.

Un Sistema de Gestión de Base de Datos, al igual que el sistema operativo, proporciona servicios tanto a los usuarios como a otros programas. A menudo, cuando el usuario piensa que está utilizando directamente el Sistema de Gestión, lo que realmente hace es usar un programa que le proporciona una interfaz de usuario para trabajar con él.