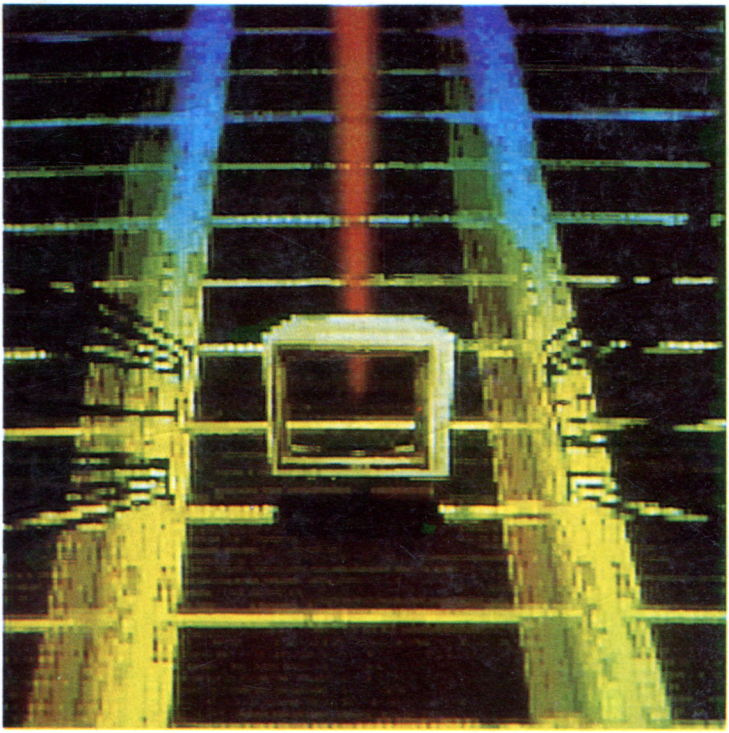
GRAN BIBLIOTECA

AMSTRAD



BASES DE DATOS

LOS BANCOS DE LA INFORMACIÓN

EDICIONES INGELEK

GRAN BIBLIOTECA AMSTRAD

5

BASES DE DATOS

Director editor:
Antonio M.ª Ferrer Abelló

Director de producción:
Vicente Robles

Director de la obra:
Fernando López Martínez

Redactor técnico:
Carlos de la Ossa Villacañas

Colaboradores:
Data-3 Informática, S. A.
Pilar Manzanera Amaro
Amelia Polo García

Diseño:
Bravo/Lofish

Maquetación:
Carlos González Amezúa

Dibujos:
José Ochoa

ISBN del tomo: 84-7708-018-6.
ISBN de la obra: 84-7708-004-6.
Fotocomposición: Andueza, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-39189-1986.
Precio en Canarias, Ceuta y Melilla: 435 ptas.

BASES DE DATOS

Introducción	5
Datos y más datos	7
Comenzando con dBASE II	21
Creando una base de datos	23
Introducción de datos	33
Salida de los datos	37
Actualización de los ficheros	45
Búsquedas y clasificaciones	61
Variables de memoria y funciones	67
Ficheros indexados	81
Programación en dBASE II	87
Diseño de pantallas e impresos	105
Consejos hardware	119
Apéndice A	121
Apéndice B	130
Apéndice C	131
Apéndice D	132
Apéndice E	133

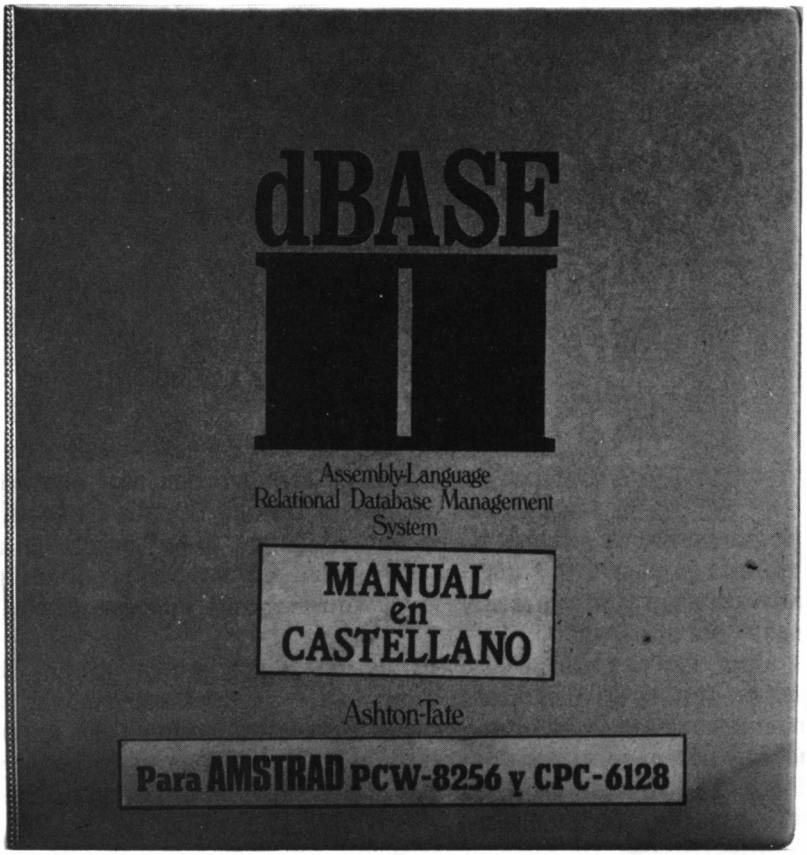
INTRODUCCION

En el mundo actual la información es uno de los bienes más preciados. La cantidad de datos que nos bombardean constantemente ha llegado a adquirir tales dimensiones, que se ha hecho necesaria la intervención de la Informática para evitar que un trabajo tan lento y costoso como la estructuración y organización de las informaciones dependa directamente del ser humano.

Para ello se crearon las bases de datos: potentes programas de gestión destinados a almacenar y procesar de una manera racional, la ingente cantidad de información que la sociedad moderna nos impone. Los ordenadores Amstrad no son una excepción a esta corriente y dotan al usuario de las herramientas necesarias para la automatización que su actividad precisa.

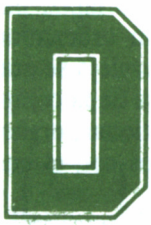
dBASE II es sin duda un excelente programa, que cumple con creces todas las exigencias que se le pueden plantear a un sistema de gestión de bases de datos. Creado por Ashton-Tate Inc. y distribuido en España por la firma MICROBYTE, es sin duda un elemento indispensable en la biblioteca de aplicaciones de cualquier profesional.

Se trata de un formidable programa que sorprende en todo momento por su potencia y versatilidad, llegando a poner a nuestro alcance el manejo más sencillo de cualquier base de datos, con una magnífica presentación, rapidez de acceso y buena gestión en general, a un precio más que asequible dadas sus prestaciones netamente profesionales.



5.0.1. dBASE II de Ashton-Tate, distribuido en España por Microbyte es, sin duda, un programa excelente en el campo de las bases de datos.

DATOS Y MAS DATOS



entro del mundillo de la Informática es fácil escuchar frases como «proceso de datos», «tratamiento automático de la información» o «base de datos». A veces se hace difícil establecer una barrera entre lo que significa «dato», «información» u otras muchas palabras relacionadas con todo lo que supone y abarca el vocablo «ordenador».

Dato es cualquier elemento simple que de por sí signifique para nosotros algo específico. Por ejemplo, un número de teléfono, el equipo ganador del último mundial de fútbol, la dirección de un familiar, el precio del kilo de naranjas o la distancia entre la tierra y la luna.

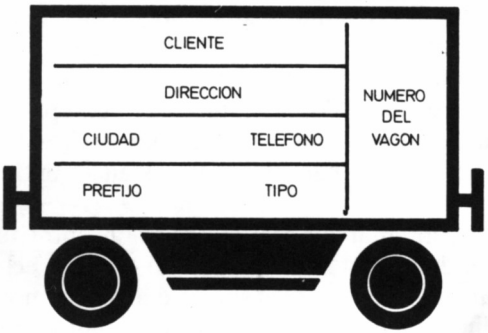
Como vemos, intentar buscar una relación entre todos los datos anteriores es sin duda bastante complicado, incluso echándole imaginación al asunto. Pero entre ellos sí existe algo que nos inclina a memorizarlos conjuntamente, a pesar de su variopinto significado: transmiten una cierta información, pequeña, pero información a fin de cuentas, de la cual deberemos valernos en múltiples circunstancias de la vida cotidiana.

Los problemas acucian cuando ya no se trata de memorizar sólo cinco datos, sino miles de ellos. Cuando se llega a este extremo, los problemas surgen a raudales:

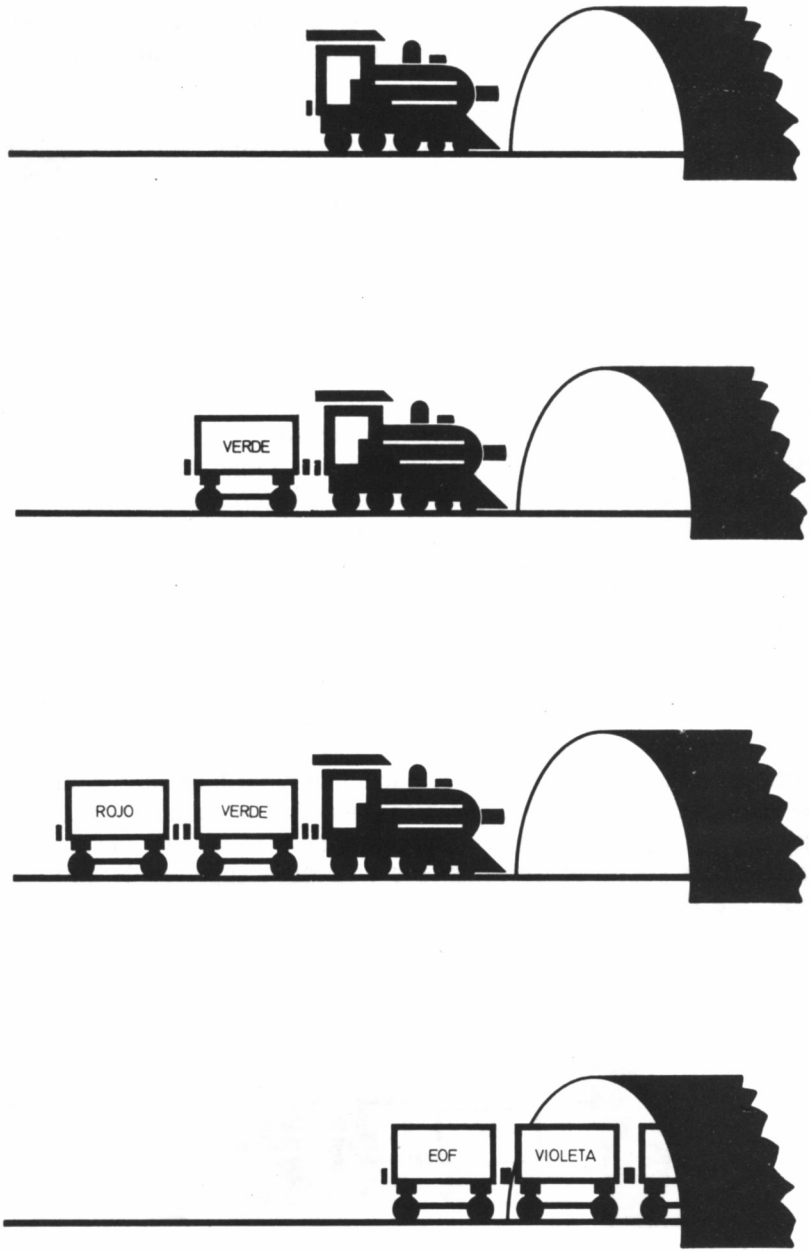
— Información que nada tienen en común unas con otras (¿qué tiene que ver el tocino con la velocidad?).
— Datos que cambian casi diariamente y, por tanto, difíciles de memorizar (que se lo pregunten a los políticos que pululan entre su partido y el grupo mixto).
— Otros sencillos, pero en los que una pequeña variación acarrea una cadena de consecuencias imprevisibles sobre otros (¡la que se lía cada vez que sube la gasolina!).
— Listas de datos con una clara relación entre todos los que la componen, pero de un volumen tal que es fácil que cunda el desorden (quien sepa recitar en voz alta y sin olvidar ni uno solo de los reyes godos, es que es profesor de Historia).

En definitiva, cuando se trabaja manejando enormes cantidades de información, lo más fácil es que nos invada el caos si no obramos con cierta precaución y nos hacemos fuertes tomando lápiz, papel, unas fichas, y su correspondiente archivador... o un ordenador.

Naturalmente, no sólo se necesita un Amstrad (perpleja se quedaría nuestra máquina si la interrogáramos sobre la cesta de la compra), sino también una serie de programas destinados a gestionar, organizar y presentar ordenadamente todo el volumen de datos con el cual la



5.1.1. Posible contenido de un vagón (registro).1.



5.1.2. Creación de un fichero secuencial.

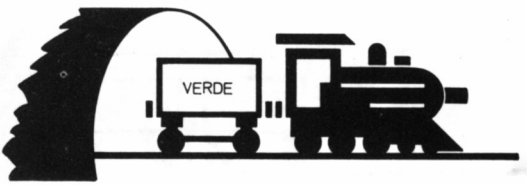
pidamos que trabaje. Toda esta serie de programas se denomina DBMS, siglas de *DataBase Management System,* o sistema de gestión de una base de datos.

Ya en este punto surgen bastantes controversias: ¡no existe un DBMS definitivo, superior a los demás! ¿Por qué? Fácil, si hemos comprendido lo que se deriva de los párrafos anteriores; las relaciones que se pueden presentar entre los datos son difíciles de encontrar, y cada sistema de gestión de la base de datos los maneja de una forma diferente.

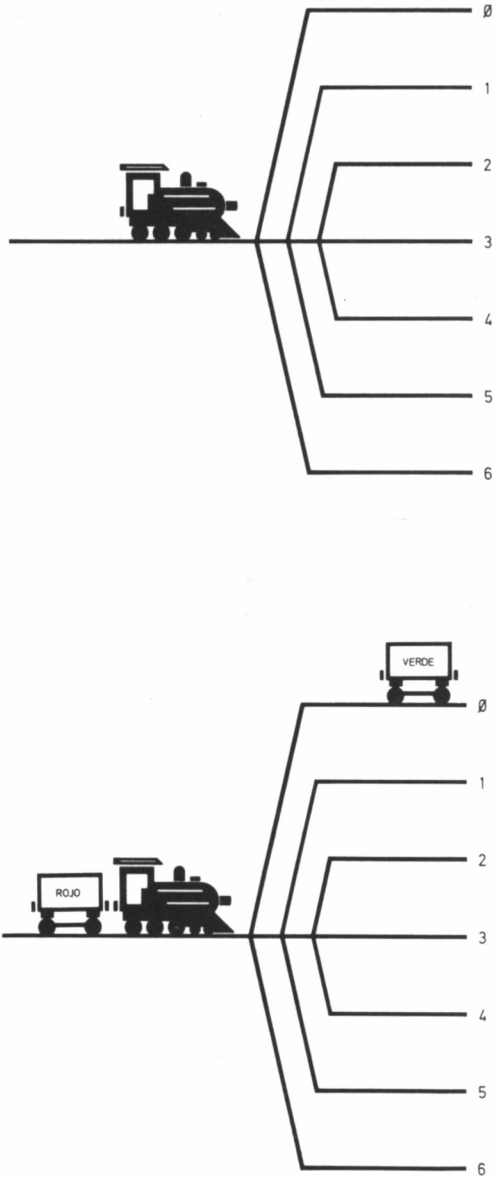
Fundamentos De Una Base De Datos

Como ya expusimos en el capítulo octavo del primer volumen de esta colección (Claves de la Informática hoy. Guía práctica. Archivos y ficheros), cuando almacenamos los datos referentes a cualquier tema concreto a través del ordenador, no lo hacemos de una manera anárquica, sino que nos atenemos a un determinado sistema, que nos permita posteriormente recuperar la información almacenada. Esta, como ya sabemos, está básicamente constituida por caracteres (bytes).

Las agrupaciones de caracteres que tienen un determinado significado para el usuario de un programa se denominan CAMPO. Así, por ejemplo, los octetos 67, 65, 82, 76, 79 y 83 correspondientes a los caracteres C, A, R, L, O y S, se pueden agrupar formando el núcleo de información CARLOS, que bien puede ser un campo nombre. Si diferentes campos son agrupados conforme a unas determinadas características comunes, se constituye un bloque superior de información que denominamos REGISTRO.



5.1.3. Lectura de un fichero secuencial.



5.1.4. Creación de un fichero directo.

En el siguiente ejemplo, varios campos se agrupan, teniendo en común su relación con nuestro amigo Carlos, constituyendo el registro de dirección del mismo:

Campo	**Contenido**
CARLOS	Nombre
DE LA OSSA	Primer apellido
VILLACAÑAS	Segundo apellido
AV RIOJANA, 15	Dirección
28031 - MADRID	Código postal y localidad

Por último, varios registros se pueden asociar pasando a formar un fichero o archivo. Siguiendo el ejemplo anterior, podríamos agrupar en un fichero todos los datos concernientes a las direcciones de nuestros amigos.

En la terminología informática, denominaremos «abrir un fichero» a la acción por la cual se notifica al ordenador nuestra intención de manejar una serie de datos, agrupados bajo el denominador común del nombre del fichero, con el fin que éste les reserve espacio en su memoria. Por tanto, es condición imprescindible abrir un fichero, antes de poder acceder a los datos que contiene. Por otra parte, la operación de «cerrado de fichero», consiste en el volcado de todos los datos relacionados con el archivo a cerrar, desde la memoria al soporte escogido: generalmente, cinta o disco.

Independientemente del significado que su contenido tenga para nosotros, los ficheros se dividen en tres tipos fundamentales, según el sistema de acceso que el ordenador siga con ellos; secuenciales, directos e indexados, formando estos dos últimos el grupo denominado de archivos aleatorios.

FICHEROS SECUENCIALES

En este tipo de archivos, los registros se escriben y se leen registro tras registro, siguiendo forzosamente la secuencia ascendente a partir del comienzo del fichero. De esta forma, si por cualquier motivo supiéramos que el registro concreto que queremos leer es, por ejemplo, el número 25 del archivo, no podríamos acceder directamente a él, sino que necesitaríamos leer previamente los 24 anteriores.

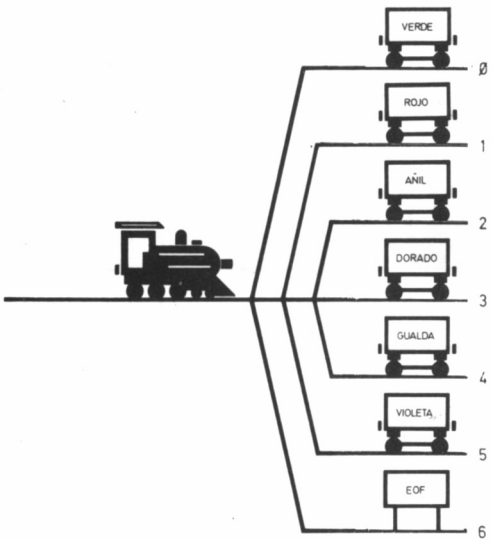
Este método presenta inconvenientes manifiestos, aunque en sistemas de cinta es el único del que se puede disponer, dada la imposibili-

dad de obtener la precisión suficiente en el avance rápido y rebobinado de la cinta, como para que la cabeza de lectura y grabación del casete se sitúe en el lugar en que se encuentra el registro a tratar.

En general, el problema no consiste en leer un registro posterior a aquel en que nos encontramos, siempre y cuando la distancia entre ambos no sea muy grande, sino en tener que leer un registro anterior al actual, ya que esto implica el cerrado y reapertura del archivo, para volver a posicionarse en su principio, con la consiguiente pérdida de tiempo.

Por otra parte, el problema llega a la hora de añadir algún registro a un archivo de este tipo, ya que en la mayoría de las ocasiones, los sistemas obligan a tener que realizar la grabación completa de todo el fichero, desechándose la copia anterior.

Pese a todo lo dicho, los archivos secuenciales son bastante útiles en el tratamiento de un pequeño volumen de datos, que no vaya a sufrir modificación una vez grabado. Tengamos en cuenta, que estos pueden ser cargados desde el soporte a la memoria y manejados una vez en ésta de forma no secuencial. En todo caso, en los sistemas de cinta, repetimos que se trata del único tipo de fichero del que podemos disponer, y siempre es mejor que carecer de ninguno.



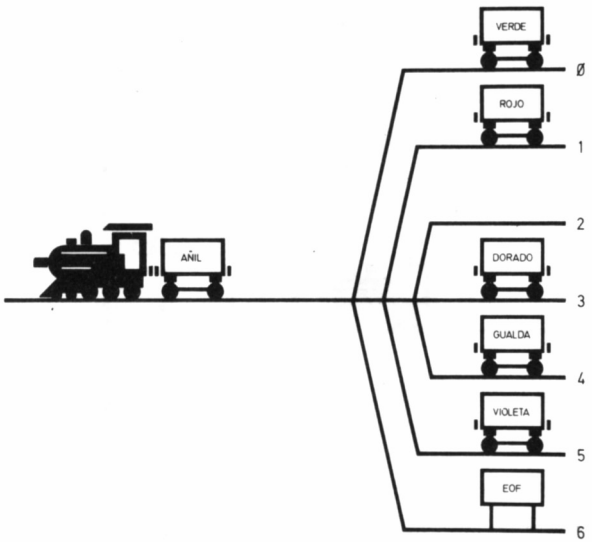
5.1.5. Situación de un fichero aleatorio cerrado.

EL FICHERO DIRECTO

Si disponemos de un sistema de disco, en la mayoría de los casos dejaremos de lado los ficheros secuenciales para recurrir a los de acceso directo, en los cuales podemos obtener directamente la información de un registro, con sólo saber qué número de orden hace dentro del fichero, sin necesidad de leer todos los anteriores.

Así pues, el acceso directo nos permite obtener la información de un determinado registro a partir de la posición de éste en la secuencia de registros, pero aún así, en el caso de desconocer dicha posición, los archivos directos se manifiestan bastante inútiles.

Si por ejemplo queremos acceder a la información sobre uno de nuestros amigos en el fichero de agenda, pero no conocemos su número de registro, sino su nombre, nos veremos ante la necesidad de leernos todo el fichero desde el principio, hasta localizar el nombre buscado; es decir, operar del mismo modo que lo haría un fichero secuencial, con la única ventaja de no tener que realizar las operaciones de cerrado y reapertura de fichero para volver el puntero de lectura al principio, y tener la posibilidad de introducir nuevos datos en cualquier posición del fichero.



5.1.6. Ejemplo de acceso a un registro (número 2) en un fichero directo.

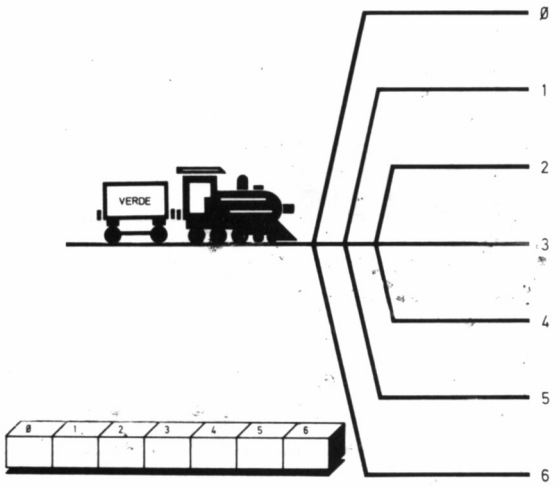
FICHEROS INDEXADOS

Se trata de otro tipo de archivo aleatorio, que se estructura como un fichero directo, en el cual cada registro puede ser accedido por uno o varios campos del mismo, que son conocidos como claves del registro. Gracias a este sistema, es posible, por ejemplo, localizar un registro de nuestra agenda indicando tan sólo el primer apellido de la persona a buscar, corriendo esta búsqueda por cuenta del ordenador.

Apoyémonos en un ejemplo para observar el desarrollo de este proceso con una clave cualquiera, por ejemplo, la localización de la clave DE LA OSSA en nuestro fichero de agenda.

Para comenzar, supongamos que los registros con los datos de nuestros amigos se encuentran en un gran archivo directo, ordenados alfabéticamente, mientras que disponemos al mismo tiempo de un fichero secuencial, cuyos registros contienen simplemente una letra y el número del primer registro del fichero directo, cuya clave comienza por esa letra.

Para encontrar la ficha de DE LA OSSA, el sistema buscará primeramente en el archivo secuencial la clave D, averiguando a través de éste cuál es la primera clave dentro del archivo directo en que



5.1.7. Fichero indexado: situación antes de la escritura del registro «verde».

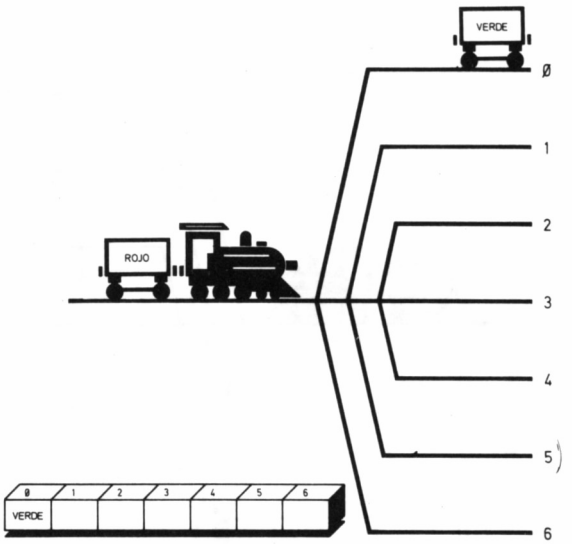
comienzan los apellidos con la letra D. Ya sólo nos restará situar el puntero de lectura en el lugar que indique el contenido del secuencial, para así leer únicamente los apellidos de aquellas personas que comiencen por la misma letra.

Como podemos comprobar, este sistema ahorra una gran cantidad de tiempo, ya que evita la lectura secuencial de todos los registros cuyo campo de apellido comienza por letras anteriores a la «D».

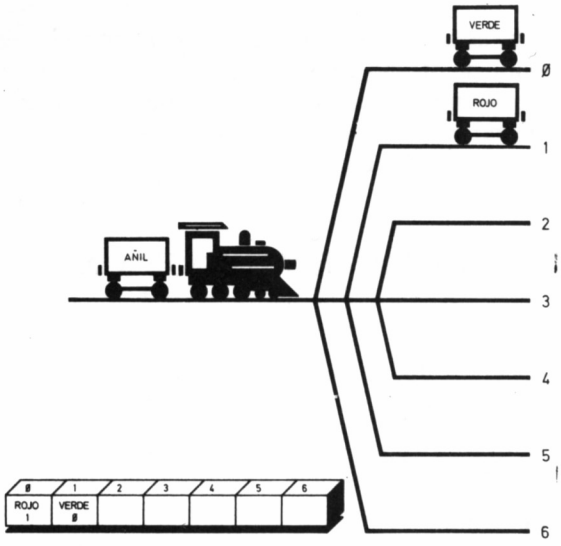
Aunque los sistemas de gestión de indexados no funcionan exactamente de esta forma tan sencilla, a fin de reducir al mínimo el número de registros que se ha de leer hasta encontrar el buscado, servirá el ejemplo como botón de muestra de la utilidad de los mismos.

En todo caso, por si nos enfrentamos con la tarea de confeccionarnos un fichero indexado para uso propio, como es muy probable que nos suceda trabajando con dBASE II, es aconsejable que seleccionemos cuidadosamente el campo que emplearemos como clave, con el fin de estar seguros que el dato será convenientemente localizado.

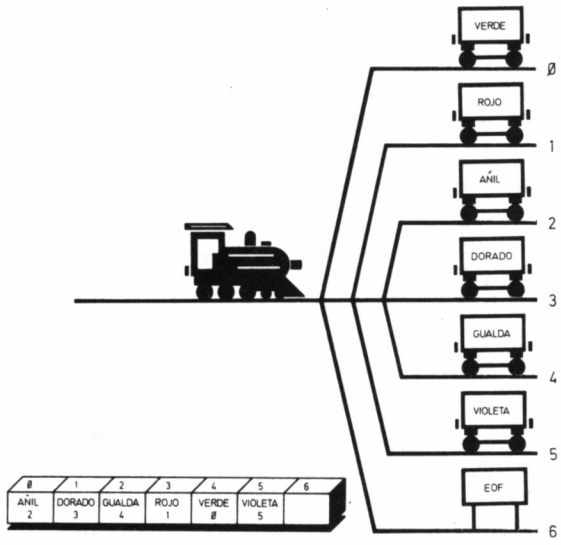
Así, por ejemplo, no es oportuno escoger como campo de búsqueda un nombre o una calle, ya que éstos se encuentran sujetos a multi-



5.1.8. Fichero indexado tras la actualización de la clave del registro «verde».



5.1.9. Fichero indexado después de la actualización de la clave «rojo».



5.1.10. Fichero indexado tras la grabación de toda la información.

tud de abreviaturas que el programa lógicamente no va a reconocer, y nosotros probablemente no conseguiremos recordar. Si ponemos por caso una búsqueda por clave-nombre de María Pilar, podemos encontrar que el mismo nombre ha podido ser escrito de muchas formas diferentes: M. Pilar, M.ª Pilar, María Pilar, Pilar...

Sistemas Para La Gestión De Una Base De Datos

Cualquier sistema lógico para el almacenamiento masivo de la información debe ser capaz de soportar las siguientes operaciones:

— Entrada de los datos.
  - Captura.
  - Depuración.
— Organización (secuencial, aleatoria, indexada...).
— Búsquedas y clasificaciones.
— Totalizaciones.
— Salidas (listados comprensibles de la información almacenada).

ENTRADA DE DATOS

Puesto que uno de los objetivos primordiales en toda base de datos es mantener volúmenes elevados de información, debe contar con los medios necesarios para permitir conducir los datos desde el exterior del sistema hasta los medios de almacenamiento masivos (cinta, disco, etc.).

Para ello, las rutinas encargadas de presentar los diferentes elementos que configuran la base de datos deben encargarse de la depuración de todas las entradas, en evitación de posibles errores, como por ejemplo, la aceptación de un contenido literal para un campo numérico, o de un dato de excesiva longitud.

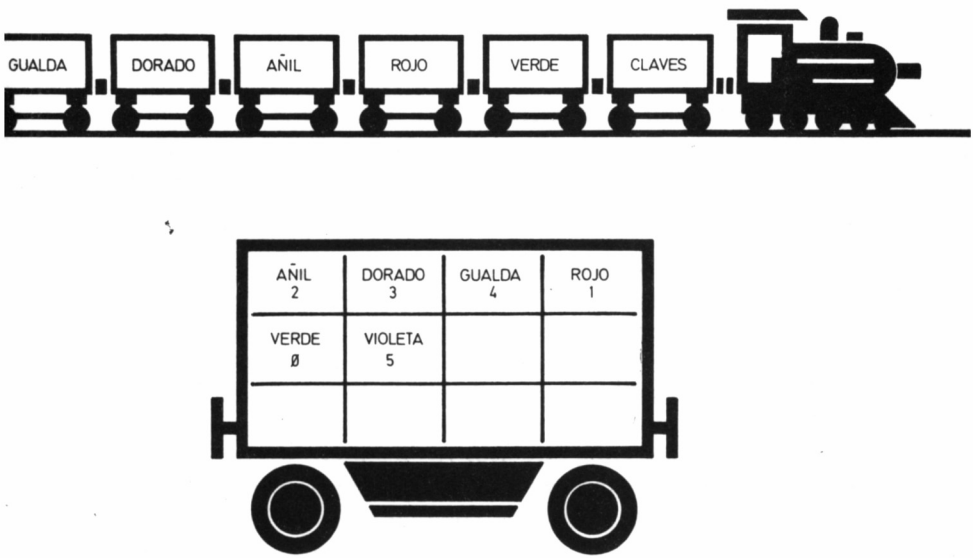
Por otra parte, ya hemos visto las diferentes estructuras que pueden emplearse en un fichero, y, por tanto, un DBMS debe contemplarlas para escoger la más adecuada. Asimismo, ha de ser tarea del programa disponer los datos de la forma en que al usuario le resulte más comprensible. Por ejemplo, puede que queramos obtener un listado de nuestra agenda telefónica por orden alfabético de apellidos.

No obstante, no es suficiente con tener una lista ordenada de ele-

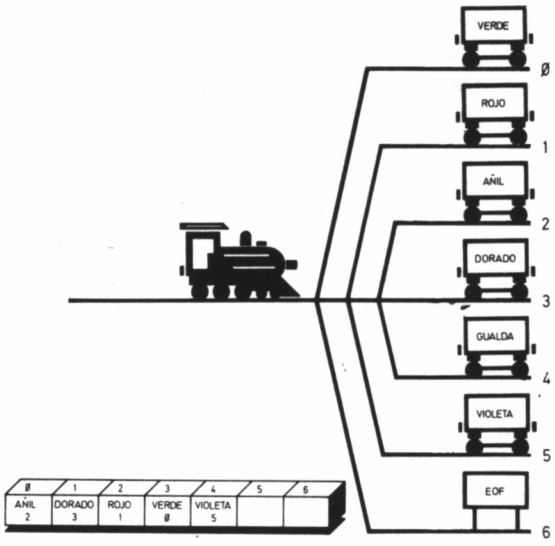
mentos, máxime cuando ésta es de gran longitud, ha de estar contemplada también la posibilidad de encontrar cualquiera de ellos en un tiempo mínimo. Recordemos que el ahorro de tiempo es una de las finalidades más trascendentales en todo proceso informático.

Ahora bien, de nada nos va a servir mantener un enorme volumen de datos, si el programa que los gestiona no es capaz de generar informes claros y de fácil interpretación por parte del usuario. Es más, éstos deberán ir acompañados de determinados subtotales o totales de los campos numéricos a modo de resumen, que aporten una información definitiva sobre los datos contenidos en los ficheros.

Supongamos que disponemos de una base de datos de gran tamaño, con una brillante gestión, que nos permite clasificar la información por cualquier criterio, modificarla, actualizarla o consultarla con gran comodidad y rapidez. Pues bien, si no fuera posible obtener informes que presentaran resultados globales, del tipo de listados, emisión de etiquetas, etc., carecería de utilidad práctica.



5.1.11. En el fichero indexado se puede distinguir un «vagón de claves», cuyo contenido, como se aprecia en la segunda parte de la figura, es el orden en el que le siguen «los vagones de información».



5.1.12. Situación del fichero índice tras la supresión del registro «gualdo».

COMENZANDO CON DBASE II

Establecidos ya los elementos fundamentales que configuran una base de datos, nos adentraremos ahora en dBASE II, uno de los programas que más facilidades ponen a nuestro alcance cuando de gestionar un enorme volumen de datos se trate.

El sistema de manejo de ficheros de dBASE II es de tipo relacional, es decir, cuando efectuemos un requerimiento de información, no será necesario estar familiarizados con la estructura de la base de datos creada, y será el programa el que busque las relaciones efectivas entre unos archivos y otros para mostrar los datos solicitados.

Comenzaremos por estudiar todos los mandatos en modo directo, mientras que en los capítulos del final del libro, nos introduciremos en el campo de la programación de esta formidable base de datos y en la construcción de ficheros de mandatos que permitan trabajar interactivamente con el ordenador.

Asimismo, aprenderemos a confeccionar menús, formatos de pantalla, salidas impresas, y en fin, a estructurar las informaciones de manera que consigamos, sea cual sea el tipo de datos sobre los que trabajemos, obtener el máximo rendimiento de ellas.

CARGA DE DBASE II

Una vez encendido el ordenador, cargaremos el CP/M. Tras aparecer el prompt del sistema operativo, escribiremos dbase, sustituyendo el disco del sistema por el propio de dBASE II antes de pulsar RETURN.

El programa se inicia con el mensaje indicativo de la versión bajo la que nos encontramos, solicitándosenos a continuación la fecha de trabajo, en el formato día, mes y dos últimas cifras del año, separadas por barras o dos puntos *(Enter today's date or return for none,* Entre la fecha de hoy o return para ninguna).

Acto seguido hará su aparición el mensaje de copyright de Ashton-Tate y el punto inductor de mandatos del programa.

```
dBASE II. Version AMSTRAD CP/M 3.0.

Enter today's date or return for none
 (DD/MM/YY) :14:10:86

Copyright (C) 1983 Ashton-Tate Inc.

***  dBASE II      Ver 2.41  1 February 1984
. _
```

CREANDO UNA BASE DE DATOS

El primer paso para realizar un empleo útil de una base de datos, es diseñarla de acuerdo a las necesidades concretas que con ella esperemos cubrir. Para ello, nada mejor que plantearnos con anterioridad los campos que formarán cada registro, su longitud, y el tipo de información (numérica, decimal, literal) que deberán contener. Todo este conjunto de parámetros que debemos comunicar a dBASE II es lo que se conoce con el nombre de ESTRUCTURA de la base de datos.

LA ESTRUCTURA

En lo que resta del libro, hemos considerado que la mejor forma de ilustrar el funcionamiento de un programa aparentemente tan complejo como el dBASE II, es a través de ejemplos. Para ello, crearemos una sencilla agenda telefónica, cuyos datos reuniremos en un fichero denominado TELEF.

En la creación del fichero TELEF, se debe precisar en primera instancia todas las características de los campos (nombre, tipo y longitud). El mandato que posibilita la definición de estos parámetros es CREATE.

Tecleando tras el prompt de dBASE II (.) dicha orden, el programa contesta emitiendo el mensaje *Enter Filename:* (Introduzca el nombre del fichero:) a lo que nosotros contestaremos B:TELEF. En el caso que el fichero TELEF ya existiera sobre dicha unidad, el programa presenta el mensaje:

Do You want to delete previous file (y/n)

el cual nos pregunta si queremos borrar el fichero TELEF existente anteriormente, a lo que deberemos contestar Y en caso afirmativo o N si lo pensamos conservar. Naturalmente, la decisión de eliminarlo debe tomarse con la precaución que se merece, por razones que se hacen evidentes.

B: indica al sistema que pretendemos crear el fichero en la unidad de disco referenciada por B. Si no hubiéramos especificado nada, por defecto, dBASE II considerará que queremos crear el fichero en A. También directamente, se hubiera podido escribir CREATE B: TELEF, lo que hubiera ahorrado al programa presentar la cuestión anterior. Por tanto, la sintaxis correcta de este primer mandato es la siguiente:

CREATE [d:] [nombre del fichero]

Lo dicho hasta el momento es aplicable también a los sistemas dotados de una única unidad de disco, dado que será el Sistema Operativo el que se encargue en el momento apropiado de solicitar al usuario el diskette de programas o datos (unidad A o B), según convenga.

Nótese que no hemos asignado extensión al nombre del fichero, debido a que dBASE II automáticamente añade una propia (.DBF, *DataBase File,* o fichero de base de datos).

Una vez aceptado el mandato CREATE, el sistema queda dispuesto a recibir la estructura de nuestra base de datos, haciéndolo patente mediante la indicación:

Enter record structure as follows:
Field Name, Type, Width, Decimal places
001

en la cual se nos pide sucesivamente el nombre de cada campo, su tipo, el tamaño y los lugares decimales en el caso de señalar tipo

numérico. Construyamos entonces la estructura de nuestro fichero. Para ello, indicaremos sucesivamente los siguientes datos, separados por comas: nombre del campo (máximo de ocho caracteres), tipo, definido por la inicial correspondiente (C=Caracteres, N=Numérico, L=Lógico) y longitud máxima ocupada por su contenido. Adicionalmente, en el caso de campos numéricos, se pueden señalar el número de cifras decimales.

```
. create b:telef
Enter record structure as follows:
Field   Name, Type, Width, Decimal places
 001     nombre,c,15
 002     apell:1,c,15
 003     apell:2,c,15
 004     direcc,c,30
 005     cod:post,n,5
 006     poblac,c,25
 007     prov,c,15
 008     prefijo,n,3
 009     telef:,n,7
 010     caract:,1
```

El final de la creación de la estructura, se le indica al programa pulsando RETURN cuando el cursor esté al principio de un campo vacío de toma de datos, en nuestro caso, tras introducir el campo CARACT:, el cursor se ha situado al principio del número 011 y al pulsar RETURN presenta el mensaje:

Input data now?

con lo cual nos sugiere si ya queremos comenzar la introducción de los datos o preferimos dejarlo para un momento posterior.

Existen dos maneras de asegurarse que el fichero se ha creado correctamente. La primera consiste en salirse del dbase II, mediante la orden QUIT, y utilizar una orden del sistema operativo (DIR, que permite la visualización de los ficheros que hay en el diskette); la segunda, emplea una orden del dBASE II para listar los ficheros *(list files)*, que no provoca el retorno al sistema operativo.

Otra opción de interés, puede ser la obtención de una copia sobre

papel de los ficheros existentes en el disco; la activación de la impresora se consigue pulsando simultáneamente las teclas CTRL (ALT, en los PCW) y P. La repetición de dicha secuencia supondrá la desactivación de esta función.

Hay que tener en cuenta, que el nombre de los ficheros está compuesto por un máximo de ocho caracteres, seguido de otros tres, que corresponden a la extensión y permiten identificar el tipo de fichero a que se refieren.

El fichero de dBASE II creado es TELEF.DBF, donde DBF, como ya hemos mencionado, significa *Data Base File,* que quiere decir fichero base de datos. Todos los ficheros de dBASE II, estructurados en campos y creados por este programa, tienen como extensión .DBF.

Para evitar tener que volver al sistema operativo, en el caso de que nuestros ficheros de base de datos no estuvieran en la unidad por defecto, se utiliza el mandato LIST FILES con la siguiente sintaxis:

LIST FILES [ON d:]

donde d es la unidad de disco afectada por la operación. También aprovechando esta misma orden, con el parámetro LIKE podremos comprobar los ficheros almacenados en la unidad seleccionada; de forma general:

LIST FILES [ON d:][LIKE <n-fich.ext>]

siendo d la unidad de disco seleccionada, n-fich el nombre del fichero, y ext su extensión. Naturalmente, estos dos últimos datos pueden especificarse manejando los símbolos comodín de CP/M (* y ?).

```
. list files on a: like *.*

MICROBYT.EGz        DBASEOVR.COM        DBASE    .COM        DSORT    .COM
FILEGEN .CMD        MENUGEN .CMD        LABELGEN.CMD         FORMGEN .CMD
DBASEMSG.TXT        DGEN    .CMD
```

Existe otra orden equivalente a LIST FILES, cuyo funcionamiento es idéntico a esta última, consistente en teclear la orden DISPLAY FILES.

```
. display files on m:

DATABASE FILES    R RCDS     LAST UPDATE
TELEF    DBF      00010      10/10/86
TELEFNEW DBF      00000      00/00/00
PEPE     DBF      00012      00/00/00
JUAN     DBF      00011      00/00/00
MELY     DBF      00000      00/00/00
COPIA    DBF      00010      09/10/86
TOTAL    DBF      00010      00/00/00
ORCOPIA  DBF      00010      00/00/00
TOT      DBF      00010      00/00/00
ORNOMBRE DBF      00010      00/00/00
SPOBLAC  DBF      00010      00/00/00
SORTINDE DBF      00010      00/00/00
BIBLIOT  DBF      00001      10/10/10
KLM      DBF      00001      10/10/10
TELEF1   DBF      00000      10/10/86
```

También es posible modificar dentro del dBASE II la unidad por defecto. Esto se consigue mediante la orden SET DEFAULT TO (definir por defecto a), indicando al final la seleccionada; en general:

SET DEFAULT TO [d:]

donde d indica la unidad de disco a señalar por defecto.

ACCESO A LOS FICHEROS

Hasta ahora, nuestro trabajo ha consistido en crear la estructura que tendrá la agenda telefónica denominada TELEF.DBF. Nuestro siguiente paso será comprobar que efectivamente los campos que componen cada registro responden exactamente a las características que nos hemos propuesto. Para ello utilizaremos otra orden LIST:

LIST STRUCTURE

Tras ejecutarla, el programa con toda seguridad habrá demostrado el mensaje *No Data Base File·In Use, Enter Filename:* (no se está manejando ningún fichero de base de datos, introduzca el nombre del fichero:). Esto es debido a que, aunque el fichero TELEF.DBF existe sobre el disco, todavía no hemos avisado a dBASE II que queremos

trabajar sobre él, ya sea sobre los datos que contiene o sobre su estructura. Por ello, antes de trabajar con un fichero, debemos indicar al programa con cuál en concreto pretendemos hacerlo, evitándose de esta manera la aparición de mensajes como el anterior.

La orden encargada de comprobar si un fichero existe sobre el disco y de ser así, permitirnos su acceso para realizar sobre él cualquièr operación posterior, es USE, y su sintaxis completa, la siguiente:

USE [[d:] <n-fich>]

donde como siempre, d es la unidad donde se encuentra almacenado el fichero a utilizar y será imprescindible escribir este parámetro siempre que ésta no corresponda con la que estamos empleando por defecto. Debemos observar, que no ha sido necesario señalar la extensión del fichero, puesto que necesariamente será .DBF, si se trata de uno perteneciente a la base de datos.

Además, el mandato USE empleado sin ningún parámetro adicional, provoca el cierre de la base de datos actual.

Una vez habilitado el acceso a nuestra agenda telefónica mediante el mandato USE TELEF, estamos en condiciones de visualizar su estructura; para ello, teclearemos la orden LIST STRUCTURE. La respuesta del programa debe tener el aspecto de la figura siguiente:

```
. list structure
Structure for file: B:TELEF    .DBF
Number of records: 00000
Date of last update: 10/10/86
Primary use database
Fld    Name    Type    Width    Dec
001      NOMBRE         C      015
002      APELL:1        C      015
003      APELL:2        C      015
004      DIRECC         C      030
005      COD:POST       N      005
006      POBLAC         C      025
007      PROV           C      015
008      PREFIJO        N      003
009      TELEF:         N      007
010      CARACT:        L      001

** Total **                00132
```

la cual nos informa acerca de diversos datos de interés sobre nuestro fichero. La primera línea nos dice la unidad donde se ha creado y el nombre que le hemos asignado. En la segunda, se nos informa del número de registros utilizados hasta el momento (naturalmente, en principio es cero, puesto que todavía no hemos introducido ningún dato). La tercera, la última fecha en que fue actualizado, en el formato mes/día/año (la que ahora aparece se corresponde con la de creación del fichero).

Finalmente, aparece una lista de todos los campos con su nombre, tipo y longitud, más una línea al final, denominada total, que indica la suma de todos los campos en bytes, es decir, la longitud de cada registro de nuestra agenda telefónica.

Como en el caso anterior, cuando de visualizar los ficheros almacenados en un determinado disco se trata, los mandatos LIST y DISPLAY conducen a idénticos resultados. Por tanto, podríamos haber escrito para comprobar la estructura de nuestro fichero la orden DISPLAY STRUCTURE.

Por otra parte, para una presentación más clara de los datos, podemos recurrir al mandato ERASE, cuya misión es el borrado de la pantalla, situando a su vez el cursor en la esquina superior izquierda.

Otra norma de uso general en dBASE II es que los comandos no tienen por qué ser escritos íntegramente, sino que para su identificación por el programa bastan sus cuatro primeras letras. Por ejemplo, DISP STRU sería suficiente para visualizar la estructura de un fichero.

Errores De Sintaxis

Dado que el sistema de edición de comandos en dBASE II es un tanto deficiente, cada vez que el programa no reconoce algún mandato por estar escrito equivocadamente, presenta el mensaje *Unkown command* (Comando desconocido), y a continuación *Correct and retry? (y/n)* (¿Corregir y reintentar?).

Si nuestra respuesta es Y, dBASE II nos interroga sobre el carácter o caracteres a partir del cual queremos modificar la orden *(Change from:)*. Una vez señalado éste, se nos preguntará por cuál o cuáles deberá ser sustituido *(Change to:)*.

Cuando ya se han seleccionado los cambios necesarios, el programa presenta el nuevo mandato corregido, junto con la indicación *More corrections (y/n)?* (¿Más correcciones?). De nuevo, el proceso se

repite si respondemos N al comprobar que su sintaxis no es la adecuada, mientras que de contestar Y, automáticamente la orden será ejecutada.

En el siguiente ejemplo describimos cómo modificar la instrucción LIST STRUCTURE, escrita erróneamente, como puede comprobarse al principio.

```
. lost syructure
*** Unknown command
lost syructure
Correct and retry? (y/n)Y
Change from: o
Change to: i
list syructure
More corrections (y/n)?Y
Change from: y
Change to: t
list structure
More corrections (y/n)?N
Structure for file: M:TELEF    .DBF
Number of records: 00010
Date of last update: 10/10/86
Primary use database
Fld   Name    Type    Width    Dec
001     NOMBRE         C     015
002     APELL:1        C     015
003     APELL:2        C     015
004     DIRECC         C     030
005     COD:POST       N     005
006     POBLAC         C     025
007     PROV           C     015
008     PREFIJO        N     003
009     TELEF:         N     007
010     CARACT:        L     001

** Total **              00132
```

Aunque este sistema pueda parecer excesivamente complicado, para corregir errores en una larga línea de mandatos, siempre resulta bastante más eficaz que tener que volver a escribirla.

CUADRO DE MANDATOS

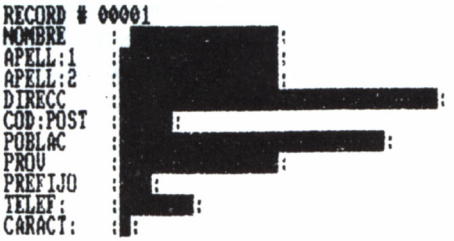
— CREATE
— QUIT

— LIST FILES
— DISPLAY FILES
— SET DEFAULT
— LIST STRUCTURE
— USE
— DISPLAY STRUCTURE
— ERASE

INTRODUCCIÓN DE DATOS

Pues bien, ahora ya está todo listo para que dBASE II sea capaz de aceptar los datos que le queramos proporcionar. El mandato encargado de activar esta opción es APPEND. Al ejecutarlo, la pantalla se borra, y aparece sobre la misma una lista de los campos elegidos durante el proceso de definición de la estructura.

La primera línea de esta pantalla nos muestra el número de registro sobre el que estamos trabajando (en nuestro caso 1, puesto que todavía no hemos introducido ningún dato).



En realidad, el mandato APPEND servirá siempre que deseemos añadir datos a nuestro fichero, ya fuera por estar vacío o a continuación de los existentes.

Conviene ahora, para mantener un criterio uniforme en lo que resta de la obra, pues muchos de los ejemplos se basan en nuestra agenda telefónica, introducir los datos que nosotros proponemos, aunque podrían ser cualesquiera otros, siempre que mantengan la misma estructura.

Cada vez que terminamos un campo, para pasar al siguiente se pulsa RETURN, aunque como se podrá comprobar, por ejemplo, al introducir los dígitos correspondientes al código postal, si se llena completamente, el salto al siguiente se produce de forma automática.

Una vez terminados todos los campos del primer registro, el programa presenta vacíos los del segundo, y así sucesivamente, hasta que por fin terminemos con todos los datos a introducir.

Para salir de este modo de trabajo y regresar al prompt de dBASE II, tenemos varias opciones: la primera consiste en pulsar RETURN cuando el cursor de la entrada de datos se encuentra al principio del primer campo de un registro vacío. La segunda forma, consiste en pulsar CTRL W, con lo cual todas las modificaciones realizadas pasan a ser parte integrante de la base de datos.

MODIFICACIONES

Naturalmente, es fácil cometer errores mientras se están introduciendo los datos y para permitir la modificación de cualquier campo

erróneo, o para desplazarnos de unos a otros dentro de un registro, dBASE II dispone de una serie de teclas o combinaciones de éstas, destinadas a poder situar el cursor en el lugar deseado de la pantalla.

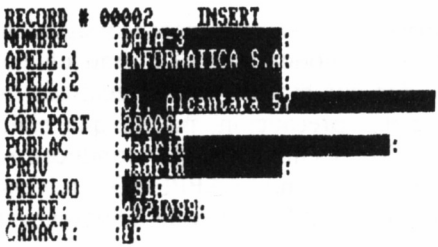
CTRL S desplaza el cursor una posición hacia la izquierda. Si está situado al principio de un campo, saltará al primer carácter del campo anterior y si además se trataba del primero dentro de un registro, se emplazará sobre el primero del registro siguiente.

CTRL D desplaza el cursor una posición hacia la derecha; si éste se encontraba situado en el último carácter del último campo de un registro, su pulsación provocará un salto del cursor hasta la posición inicial del primer campo del registro siguiente.

CTRL E mueve el cursor de abajo arriba, situándolo cada vez en el primer carácter del campo anterior.

CTRL X actúa de forma idéntica al anterior, pero en sentido inverso.

CTRL V bascula activando y desactivando el modo de inserción de texto, circunstancia que el programa pone de manifiesto, mostrando en la línea que contiene el número de registro, el mensaje INSERT. Naturalmente, podremos insertar tantos caracteres dentro de un campo como queden en blanco a partir del último almacenado originalmente. Si intentamos sobrepasar este tope, los nuevos caracteres sobreescribirán a los anteriores.



CTRL G borra el carácter situado bajo el cursor.

CTRL Y borra el contenido del campo sobre el que está situado el cursor. Si éste no se encontraba sobre el primer carácter de dicho campo, la acción de estas teclas lo conducirá en primera instancia

hasta dicha posición, y una vez allí, pulsándolas de nuevo, provocarán la eliminación de los datos contenidos en el campo.

CTRL V borra un registro completo. Al hacer uso de este mandato, en la zona superior del registro editado se muestra la inicación DELETED (borrado), aunque si ésta no era nuestra intención, todavía estamos a tiempo de recuperar su contenido, pulsando nuevamente CTRL V.

```
RECORD # 00003                    DELETED
NOMBRE    :Juan          :
APELL:1   :Cuesta        :
APELL:2   :Nuin          :
DIRECC    :Pza. de los desamparados 18,1o:
COD:POST  :28003:
POBLAC    :Madrid                :
PROV      :Madrid        :
PREFIJO   : 91:
TELEF:    :2742142:
CARACT:   :t:
```

CTRL C provoca un salto al registro siguiente.

CTRL R en este modo tiene el mismo efecto que CTRL C.

Finalmente, dos combinaciones de teclas de edición que deben emplearse con cierta precaución, puesto que aunque de comportamiento parecido, los resultados a que conducen son bien distintos:

CTRL Q abandona el modo APPEND y retorna a dBASE II, sin contemplar cualquier modificación que hubiéramos podido realizar en la base de datos.

CTRL W provoca la salida del modo, pero en este caso, todas las modificaciones son aceptadas en el archivo.

CUADRO DE COMANDOS

— APPEND
— Teclas en modo APPEND

SALIDA DE LOS DATOS

Una de las grandes ventajas de dBASE II, reside en sus posibilidades de visualización e impresión de los resultados. Para ello, dispone de una serie de comandos, que en combinación con unos parámetros específicos, cumplen a las mil maravillas el trabajo que se les ha encomendado.

El mandato LIST (con nuestros conocimientos hasta el momento, idéntico resultado se consigue empleando en su lugar la orden DISPLAY), provoca la impresión de los datos contenidos en el fichero, ya sea hacia la pantalla o hacia la impresora (recordemos que bastaría pulsar CTRL P, tras escribir LIST, para que los resultados fueran también escritos en este periférico).

También es posible dirigir la salida de información hacia la impresora, utilizando el mandato SET PRINT ON, previamente a la ejecución de la orden LIST. Naturalmente, SET PRINT OFF, tendrá el mismo efecto que si volviéramos a pulsar CTRL P con la impresora activa, es decir, deshabilitará la salida hacia el mencionado periférico.

LISTAS DE DATOS

Cuando ejecutamos el mandato LIST sin otros parámetros, dBASE II visualiza el conjunto completo de los datos almacenados en el

```
. set print on
. list
00001  Fernando          Lopez             Martinez          Cl. Vallefermoso 23, 5o.
       28004 Madrid                        Madrid              91  4488425 .T.
00002  DATA-3            INFORMATICA S.A                     Cl. Alcantara 57
       28006 Madrid                        Madrid              91  4021099 .F.
00003  Juan              Cuesta            Nuin              Pza. de los desamparados
18,1o  20003 Madrid                        Madrid              91  2742142 .T.
00004  MICROBYTE                                             Paseo de la Castellana 17
9, 1o  28046 Madrid                        Madrid              91  4425433 .F.
00005  Rafael de la      Ossa              Coloma            Cl. General Aranaz 1
       28027 Madrid                        Madrid              91  7243392 .T.
00006  Pedro Miguel      Lozano            Castellote        Miguel de Cervantes 8
       16400 Tarancon                      Cuenca             966   112039 .T.
00007  EDICIONES         INGELEK                             Pza. Republica Ecuador 2,
 1o    28016 Madrid                        Madrid              91  4579424 .F.
00008  Carlos de la      Ossa              Villacañas        Pza. de la Gramatica Lati
na 1   28027 Madrid                        Madrid              91  7422933 .T.
00009  Amelia            Polo              Garcia            Cl. Onesimo Redondo
        8017 Barcelona                     Barcelona           92  3365422 .T.
00010  Pilar             Manzanera         Amaro             Cl. del Videojuego 1, 1o
       28027 Madrid                        Madrid              91  7421590 .T.
```

fichero. El listado comienza desde el primer registro del mismo, y para detenerlo momentáneamente, se debe pulsar CTRL S. Para reanudarlo, bastará con pulsar cualquier otra tecla.

Cuando el tamaño de los registros supera las 80 columnas en los CPC o las 90 en los PCW, el programa nos muestra líneas sucesivas sobre la pantalla y de igual manera en la impresora, según el número de caracteres que pueden ser representados en cada periférico.

LIST ofrece también la posibilidad de obtener los datos de solamente un campo o un conjunto de éstos, los cuales deberán ir a continuación del mandato y separados por comas.

```
. list nombre,apell:1,apell:2,telef:
00001  Fernando          Lopez             Martinez          4488425
00002  DATA-3            INFORMATICA S.A                     4021099
00003  Juan              Cuesta            Nuin              2742142
00004  MICROBYTE                                             4425433
00005  Rafael de la      Ossa              Coloma            7243392
00006  Pedro Miguel      Lozano            Castellote         112039
00007  EDICIONES         INGELEK                             4579424
00008  Carlos de la      Ossa              Villacañas        7422933
00009  Amelia            Polo              Garcia            3365422
00010  Pilar             Manzanera         Amaro             7421590
```

EXPRESIONES

Una de las funciones que amplía la potencia y flexibilidad de los mandatos que dBASE II maneja, es la capacidad de expandirlos manejando expresiones. Una expresión puede tomar múltiples formas y dentro de ellas se pueden incluir todo tipo de operadores, cadenas de caracteres, datos numéricos o lógicos, e incluso algunos caracteres especiales cuya función comprobaremos. La sintaxis de estas órdenes es la siguiente:

LIST FOR campo=<expresión>

siendo campo el nombre de alguno de los definidos en la estructura del fichero y expresión un dato que puede responder a algunas de las siguientes características:

— Una cadena de caracteres encerrada entre comillas o apóstrofes.
— Un dato numérico relacionado con el campo mediante alguno de los operadores menor (<), mayor (>), igual (=), distinto (<>), menor o igual que (<=) o mayor o igual que (>=).
— Un dato lógico referenciado por una T (true) o una Y (yes), cuando es verdadero o por una F (false) o una N (no), cuando es falso.

```
. list for poblac='Madrid'
00001   Fernando        Lopez           Martinez        Cl  Vallefermoso 23, 5o.
        28004 Madrid                    Madrid            91  4488425 .T.
00002   DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
        28006 Madrid                    Madrid            91  4021099 .F.
00003   Juan            Cuesta          Nuin            Pza. de los desamparados
18,1o   20003 Madrid                    Madrid            91  2742142 .T.
00004   MICROBYTE                                       Paseo de la Castellana 17
9, 1o   28046 Madrid                    Madrid            91  4425433 .F.
00005   Rafael de la    Ossa            Coloma          Cl. General Aranaz 1
        28027 Madrid                    Madrid            91  7243392 .T.
00007   EDICIONES       INGELEK                         Pza. Republica Ecuador 2,
 1o     28016 Madrid                    Madrid            91  4579424 .F.
00008   Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Lati
na 1    28027 Madrid                    Madrid            91  7422933 .T.
00010   Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
        28027 Madrid                    Madrid            91  7421590 .T.
```

OPERADORES

dBASE II maneja tres tipos de operadores: relacionales lógicos y aritméticos. En el primer grupo están englobados todos los comentados en el apartado anterior.

Los lógicos permiten efectuar entre los operandos la suma, multiplicación y negación. Siguen una sintaxis un tanto especial, puesto que debe colocarse al principio y al final de ellos un punto (.), como a continuación se describe:

- AND. efectúa el Y lógico entre los operandos (multiplicación).
- OR. es el 0 lógico (suma).
- NOT. realiza la negación del operando sobre el que actúa.

Manejando estos tres operadores se pueden construir expresiones lógicas bastante complejas. Debemos tener en cuenta que la longitud máxima de una línea que es capaz de aceptar dBASE II es de 254 caracteres. Por ello, cuando en una orden se vaya a superar el ancho máximo de la pantalla, situando un punto y coma (;) al final de la línea, dBASE II interpretará que lo que sigue en la próxima pertenece a la misma orden.

Los operadores aritméticos suma, resta, multiplicación y división pueden actuar sobre cualquier campo específico.

También entre las posibilidades de dBASE II se cuenta su utilización como calculadora en modo directo; para ello, cualquier expresión que deseemos evaluar deberemos especificársela al programa tras un signo de interrogación (?).

OTROS PARÁMETROS

Hasta el momento hemos mencionado una serie de expresiones y operaciones que pueden manejarse dentro del mandato LIST, aunque asociadas al total de los registros que componen la base de datos. No obstante, en muchas ocasiones, sólo pretenderemos trabajar dentro de un determinado margen y para ello el programa permite en conjunción con LIST limitar dicho rango.

LIST ALL provocará la visualización de todos los registros que contenga la base de datos. Esta característica se considera por defecto cuando no se indica otra cosa dentro del mandato LIST.

```
. list off
Fernando        Lopez           Martinez        Cl. Vallefresnoso 23, 5o.       28004 Madrid                    Madrid          91  448
8425 .T.
DATA-3          INFORMATICA S.A                 Cl. Alcantara 57                28006 Madrid                    Madrid          91  402
1099 .F.
Juan            Cuesta          Nuin            Pza. de los desamparados 18,1o  20003 Madrid                    Madrid          91  274
2142 .T.
MICROBYTE                                       Paseo de la Castellana 179, 1o  28046 Madrid                    Madrid          91  442
5433 .F.
Rafael de la    Ossa            Coloma          Cl. General Aranaz 1            28027 Madrid                    Madrid          91  724
3392 .T.
Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8           16400 Tarancon                  Cuenca          966  11
2039 .T.
EDICIONES       INGELEK                         Pza. Republica Ecuador 2, 1o    28016 Madrid                    Madrid          91  457
9424 .F.
Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Latina 1   28027 Madrid                    Madrid          91  742
2933 .T.
Amelia          Polo            Garcia          Cl. Onesimo Redondo              8017 Barcelona                 Barcelona       92  336
5422 .T.
Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o        28027 Madrid                    Madrid          91  742
1590 .T.
```

LIST OFF inhibe la aparición del número de registro como principio de los datos que lo configuran.

LIST NEXT n efectúa un listado de los n registros situados a partir de la posición indicada por el puntero interno manejado por dBASE II.

LIST RECORD n muestra el contenido del registro número n.

Para modificar la posición ocupada por el puntero dentro de la base de datos, basta con señalar su número tras el prompt de dBASE II. No obstante, existen una serie de mandatos de posición que permiten desplazar el puntero con toda rapidez hasta cualquier registro de la base de datos.

GO TO lo sitúa sobre el primer registro de la base de datos.

GO BOTTOM lo posiciona sobre el último registro de la base de datos.

```
. go bottom
. list record 10
00010  Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
       28027 Madrid                    Madrid            91  7421590 .T.
```

GO n provoca idéntico resultado que cuando tras el punto inductor señalamos un número, es decir, ubica el puntero en el registro referenciado por n.

```
. go 4
. list nombre,apell:1,direcc next 5
00004  MICROBYTE                          Paseo de la Castellana 179, 1o
00005  Rafael de la      Ossa             Cl. General Aranaz 1
00006  Pedro Miguel      Lozano           Miguel de Cervantes 8
00007  EDICIONES         INGELEK          Pza. Republica Ecuador 2, 1o
00008  Carlos de la      Ossa             Pza. de la Gramatica Latina 1
```

SKIP n hace avanzar al puntero el número de posiciones especificadas por n. Esta cantidad puede ser también una negativa, con lo cual se provocará el retroceso del puntero el número de posiciones especificadas.

```
. 6
. display
00006  Pedro Miguel     Lozano          Castellote      Miguel de Cervantes 8
       16400 Tarancon                   Cuenca           966   112039 .T.
. skip
Record: 00007
. disp
00007  EDICIONES        INGELEK                         Pza. Republica Ecuador 2,
 1o    28016 Madrid                    Madrid            91  4579424 .F.
. skip -3
Record: 00004
. disp
00004  MICROBYTE                                       Paseo de la Castellana 17
9, 1o  28046 Madrid                    Madrid            91  4425433 .F.
```

DIFERENCIA ENTRE LIST Y DISPLAY

Como hasta el momento presente hemos podido comprobar, los efectos producidos por los mandatos LIST y DISPLAY eran absoluta-

mente idénticos, no obstante, existe una pequeña diferencia entre la función realizada por ambas órdenes.

Mientras que LIST visualiza todos los registros del fichero en curso, DISPLAY sólo muestra aquel al que se encuentra señalando el puntero interno de la base de datos.

La sintaxis completa de estos mandatos es la siguiente:

LIST [<opción>] [FOR <expresión>] [<nombre de campos>] [OFF]

DISPLAY [<opción>] [FOR <expresión>] [<nombre de campos>] [OFF]

EDICIÓN POR VENTANAS

Tanto para la modificación como para la visualización de los datos contenidos en el fichero, existe un mandato que los muestra sin el retorno de línea cuando la longitud de los registros supera el ancho de la pantalla. Para ello basta con ejecutar el mandato BROWSE.

Tras ello, la primera línea nos indica el número de registro sobre el que nos encontramos y la segunda el nombre de los campos que aparecen columnados.

```
. browse

RECORD # :0001
NOMBRE--------- APELL:1-------- APELL:2-------- DIRECC-----------------------
Fernando        Lopez           Martinez        Cl. Valleformoso 23, 5o.
DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
Juan            Cuesta          Nuin            Pza. de los desamparados 18,1o
MICROBYTE                                       Paseo de la Castellana 179, 1o
Rafael de la    Ossa            Coloma          Cl. General Aranaz 1
Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8
EDICIONES       INGELEK                         Pza. Republica Ecuador 2, 1o
Carlos de la    Ossa            Villacalas      Pza. de la Gramatica Latina 1
Amelia          Polo            Garcia          Cl. Onesimo Redondo
Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
```

Dentro de este modo, todas las funciones de edición comentadas cuando se trató la orden APPEND siguen siendo válidas y será necesario añadirles las dos siguientes:

CTRL B desplaza la ventana un campo hacia la derecha.
CTRL Z la desplaza un campo hacia la izquierda.

Téngase en cuenta, que en este modo están habilitadas todas las funciones de modificación de los datos de un determinado campo, por lo que puede ser utilizada no sólo para visualizar el contenido de la base de datos, sino también para corregirla o actualizarla, aunque no será posible añadir nuevos registros a los que ya la constituyen.

CUADRO DE MANDATOS

— DISPLAY
— SET PRINT ON
— SET PRINT OFF
— GO
— SKIP
— BROWSE

ACTUALIZACIÓN DE LOS FICHEROS

Tras haber creado un fichero y mostrado los mandatos destinados a la visualización de los datos que contienen, durante este capítulo analizaremos la forma en que dBASE II permite actualizar los registros de la base de datos, admitiendo inserciones, supresiones, modificaciones y copias de bloques de información.

Como ya quedó claro en el capítulo destinado a explicar la creación de ficheros, éstos están formados por dos bloques de información independientes; de una parte, su estructura y, por otra, los propios datos que almacenan.

ACTUALIZACIÓN DE LA ESTRUCTURA

Aunque ya indicamos que como paso previo a la definición de una estructura, debe efectuarse un análisis completo de los requerimientos que esperamos satisfacer con nuestra base de datos, es muy posible que aparezcan ciertos extremos que no habían sido previstos anteriormente.

De poca utilidad nos sería un programa como dBASE II, si tras introducir cientos de registros en cualquiera de los ficheros creados, comprendiéramos que, por cualquier circunstancia, hemos olvidado incluir un campo de relativa importancia, y como consecuencia de ello, tuviéramos que repetir todo el proceso de introducción de datos.

dBASE II, como excelente programa para la gestión de base de datos, contempla que pudiera suceder tal eventualidad y desde luego nos evita, como a continuación comprobaremos, el monótono proceso de reinsertar todos los datos existentes.

Para modificar los datos referentes a la estructura de un determinado fichero, se deben seguir los siguientes pasos:

— Efectuar una copia de la estructura del fichero antiguo en uno nuevo.
  USE <n-fich ant>
  COPY STRUCTURE TO [x:] <n-fich nuevo>
— Crear la nueva estructura efectuando modificaciones en la anterior.
  USE <n-fich nuevo>
  MODIFY STRUCTURE.

Tras ello, el programa presenta el mensaje *Modify Erases All Records... proceed? (y/n)* (la modificación borrará todos los registros... ¿está de acuerdo?). Si contestamos negativamente (n) a esta pregunta, se abandona el mandato retornándose a la pantalla de órdenes.

Por el contrario, si nuestra respuesta es afirmativa (y), los datos que pudiera contener el fichero sobre el que estamos trabajando serán destruidos y se nos presentarán los campos que configuran la estructura, englobados en una pantalla de edición, de manera tal que puedan modificarse o ser añadidos los que consideremos convenientes.

Para ello, se utilizan las teclas ya conocidas, tanto para desplazar el cursor de un lado a otro, como para abandonar esta pantalla de edición de la estructura del fichero.

Si deseamos añadir nuevos campos, basta con introducir sus características (nombre, tipo y longitud), en alguno de los espacios vacíos situados a partir del último campo presente en la estructura.

Si lo que pretendemos es insertar uno nuevo entre dos ya existentes, deberemos situar el cursor en el lugar en el que pretendamos posicionarlo. Una vez allí, pulsando CTRL N aparecerá dicho campo vacío; los restantes, incluyendo el situado bajo el cursor, se habrán desplazado un lugar hacia abajo en la pantalla.

```
. use telef
. copy structure to telefn
. use telefn
. modify structure
Modify erases all records,,, proceed? (y/n)Y
```

```
            NAME        TYP LEN  DEC
FIELD 01  :NOMBRE       C  015  000  :
FIELD 02  :APELL:1      C  015  000  :
FIELD 03  :APELL:2      C  015  000  :
FIELD 04  :DIRECC       C  030  000  :
FIELD 05  :COD:POST     N  005  000  :
FIELD 06  :POBLAC       C  025  000  :
FIELD 07  :PROV         C  015  000  :
FIELD 08  :PREFIJO      N  003  000  :
FIELD 09  :TELEF:       N  007  000  :
FIELD 10  :CARACT:      L  001  000  :
FIELD 11  :                          :
FIELD 12  :                          :
FIELD 13  :                          :
FIELD 14  :                          :
```

Cuando ya hayamos completado todas las modificaciones, pulsando CTRL W se grabará esta nueva estructura en el fichero con el que estamos trabajando. Si por el contrario, abandonamos la edición mediante CTRL Q, no será tenida en cuenta ninguna de las modificaciones efectuadas. Ambos extremos deberían verificarse mediante el mandato LIST STRUCTURE, ya conocido.

— Trasladar los datos del fichero antiguo al nuevo.
  USE <n-fich nuevo>
  APPEND FROM <n-fich antiguo>

De esta manera, se consigue que los datos pertenecientes al fichero antiguo, no tengan que ser nuevamente tecleados. Debe tenerse en cuenta, que la adición de los datos sólo se efectuará con aquellos campos que tengan el mismo nombre, tanto en el fichero origen como en el destino.

Para comprobar que efectivamente la transferencia se ha realizado correctamente, basta con ejecutar el mandato LIST. Por otra parte, debe considerarse que si la longitud de los nuevos campos es mayor que la de los antiguos, la zona restante se completará con blancos cuando son alfabéticos, o con asteriscos si es que son numéricos. Si se da el caso contrario, es decir, menor longitud en los nuevos que en los

```
. use telefn
. append from telef
00010
 records added
```

viejos, al hacer uso del fichero actualizado, los datos que aparecen en dichos campos estarán truncados por la derecha, circunstancia que se ha de tener presente si no se quiere perder parte de la información.

Otra manera de utilizar el mandato APPEND FROM, es restringir su acción a sólo un determinado grupo de campos que cumplan una condición específica; la sintaxis sería:

APPEND FROM [d:] <n-fich antiguo> FOR <expresion>

Téngase también en cuenta, que para manejar con éxito esta última orden, será imprescindible que los campos especificados en la expresión estén presentes, tanto en el fichero origen como en el destino.

En este instante, tendremos dos ficheros, cada uno con una estructura y unos datos asociados. Es posible que el antiguo ya no nos sirva de nada y queramos deshacernos de él, para lo cual será preciso ejecutar la orden DELETE FILE, cuya sintaxis completa es la siguiente:

DELETE FILE [d:] <n-fich.ext>

En este caso, a diferencia de lo que ocurre con el mandato USE, es posible especificar junto con el nombre del fichero a borrar su extensión, separados ambos datos por un punto (.), lo cual nos permite especificar un archivo que carezca de la extensión .DBF. Además, dBASE II no permitirá eliminar un fichero que se estuviese manejando en el momento de ejecutar dicha orden y, por tanto, antes de hacerlo, debe cerrarse introduciendo un mandato USE o CLEAR.

Circunstancialmente, también puede resultar interesante, una vez borrado el fichero antiguo, asignar al nuevo el nombre que el anterior tenía, para lo cual se ejecuta la siguiente orden:

RENAME [d:] <n-fich actual> TO [d:] <n-fich nuevo>

El mandato RENAME debe utilizarse con cierta precaución, puesto que a todo fichero que no se le especifique extensión, dBA-

SE II le asignará la .DBF, aunque no tenga por qué ser un fichero de base de datos.

ACTUALIZACIÓN DE LOS DATOS

Hasta el momento, conocemos cómo añadir más registros a los existentes en un fichero, mediante la orden APPEND. Nuestro próximo objetivo, consiste en describir todas aquellas órdenes que dBASE II pone a nuestra disposición, para actualizar el contenido de cualquier campo dentro de un registro perteneciente a la base de datos.

Antes de poder acceder a un determinado registro, es preciso informarle al programa que esa es nuestra intención. Para ello, como siempre, debemos ejecutar el mandato USE sobre el fichero con el que vamos a trabajar.

El mandato EDIT N, siendo N el número de registro que pretendemos modificar, presentará sobre la pantalla los datos que configuran sus diferentes campos, pudiéndose manejar a partir de este momento las conocidas teclas de edición, para posicionarse o modificar cualquier carácter. Estas teclas son:

CTRL S desplaza el cursor una posición hacia la izquierda. Si está situado al principio de un campo, saltará al primer carácter del campo anterior y si además se trataba del primero dentro de un registro, se emplazará sobre el primero del registro anterior.

CTRL D desplaza el cursor una posición hacia la derecha; si éste se encontraba situado en el último carácter del último campo de un registro, su pulsación provocará un salto del cursor hasta la posición inicial del primer campo del registro siguiente.

CTRL E mueve el cursor de abajo a arriba, situándolo cada vez en el primer carácter del campo anterior.

CTRL X actúa de forma idéntica al anterior, pero en sentido inverso.

CTRL V bascula activando y desactivando el modo de inserción de texto, circunstancia que el programa pone de manifiesto, mostrando en la línea que contiene el número de registro, el mensaje INSERT. Naturalmente, podremos insertar tantos caracteres dentro de un campo como queden en blanco a partir del último almacenado originalmente. Si intentamos sobrepasar este tope, los nuevos caracteres sobreescribirán a los anteriores.

CTRL G borra el carácter situado bajo el cursor.

CTRL Y borra el contenido del campo sobre el que está situado el

cursor. Si éste no se encontraba sobre el primer carácter de dicho campo, la acción de estas teclas lo conducirá en primera instancia hasta dicha posición, borrando previamente el contenido del campo a partir de la situación del cursor, y una vez allí, pulsándolas de nuevo, provocarán la eliminación de los datos contenidos en el campo.

CTRL U borra un registro completo. Al hacer uso de este mandato, en la zona superior del registro editado se muestra la indicación DELETED (borrado), aunque si ésta no era nuestra intención, todavía estamos a tiempo de recuperar su contenido, pulsando nuevamente CTRL U.

CTRL C provoca un salto al registro siguiente.

CTRL R en este modo tiene el mismo efecto que CTRL C.

Finalmente, dos combinaciones de teclas de edición que deben emplearse con cierta precaución, puesto que aunque de comportamiento parecido, los resultados a que conducen son bien distintos:

CTRL Q abandona el modo edición y retorna a dBASE II, sin contemplar cualquier modificación que hubiéramos podido realizar en la base de datos.

CTRL W provoca la salida del modo, pero en este caso, todas las modificaciones son aceptadas en el archivo.

ACTUALIZACIONES MÚLTIPLES

El mandato EDIT permite la modificación de cualquier campo dentro del registro especificado. Sin embargo, cuando las circunstancias impliquen introducir en el mismo campo de un grupo de registros el mismo dato, resulta bastante poco operativo.

Para solventar este inconveniente, dBASE II dispone del mandato REPLACE, el cual permite la sustitución múltiple de los campos especificados. Su sintaxis completa es la siguiente:

REPLACE [<opción>] [<nombre de campo 1>] WITH [<expresión 1>], [<nombre de campo 2>] WITH [<expresión 2>],... FOR [<expresión>]

En principio parece temible construir correctamente una orden REPLACE, pero analizando cuidadosamente las zonas que la conforman veremos que no hay motivo para no utilizarla.

Opción puede ser ALL, NEXT N o RECORD N, lo que determinará que la orden actúe sobre todos, los N siguientes a la posición ocupada por el puntero o el registro número N, respectivamente. A continuación se especifica el nombre de los campos a modificar, situando detrás de WITH la expresión que lo sustituirá. Además, opcionalmente se puede indicar al programa que solamente los que cumplan la expresión definida tras FOR sean modificados.

```
use telefn
. list
00001   Fernando        Lopez           Martinez        Cl. Vallefermoso 23, 5o.
        28004 Madrid                    Madrid            91  4488425 .T.
00002   DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
        28006 Madrid                    Madrid            91  4021099 .F.
00003   Juan            Cuesta          Nuin            Pza. de los desamparados
18,1o   20003 Madrid                    Madrid            91  2742142 .T.
00004   MICROBYTE                                       Paseo de la Castellana 17
9, 1o   28046 Madrid                    Madrid            91  4425433 .F.
00005   Rafael de la    Ossa            Coloma          Cl. General Aranaz 1
        28027 Madrid                    Madrid            91  7243392 .T.
00006   Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8
        16400 Tarancon                  Cuenca           966  112039 .T.
00007   EDICIONES       INGELEK                         Pza. Republica Ecuador 2,
 1o     28016 Madrid                    Madrid            91  4579424 .F.
00008   Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Lati
na 1    28027 Madrid                    Madrid            91  7422933 .T.
00009   Amelia          Polo            Garcia          Cl. Onesimo Redondo
         8017 Barcelona                 Barcelona         92  3365422 .T.
00010   Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
        28027 Madrid                    Madrid            91  7421590 .T.
. replace poblac with 'El Foro' for poblac='Madrid'
00008
 replacement(s)
. list
00001   Fernando        Lopez           Martinez        Cl. Vallefermoso 23, 5o.
        28004 El Foro                   Madrid            91  4488425 .T.
00002   DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
        28006 El Foro                   Madrid            91  4021099 .F.
00003   Juan            Cuesta          Nuin            Pza. de los desamparados
18,1o   20003 El Foro                   Madrid            91  2742142 .T.
00004   MICROBYTE                                       Paseo de la Castellana 17
9, 1o   28046 El Foro                   Madrid            91  4425433 .F.
00005   Rafael de la    Ossa            Coloma          Cl. General Aranaz 1
        28027 El Foro                   Madrid            91  7243392 .T.
00006   Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8
        16400 Tarancon                  Cuenca           966  112039 .T.
00007   EDICIONES       INGELEK                         Pza. Republica Ecuador 2,
 1o     28016 El Foro                   Madrid            91  4579424 .F.
00008   Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Lati
na 1    28027 El Foro                   Madrid            91  7422933 .T.
00009   Amelia          Polo            Garcia          Cl. Onesimo Redondo
         8017 Barcelona                 Barcelona         92  3365422 .T.
00010   Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
        28027 El Foro                   Madrid            91  7421590 .T.
```

Como ya vimos en el capítulo anterior, la orden BROWSE también puede utilizarse para la actualización de los datos contenidos en un registro, aunque mediante el mandato CHANGE es posible alterar solamente algunos caracteres dentro de un determinado campo. Su sintaxis responde a la siguiente estructura:

CHANGE [<opción>] FIELD lista de campos [FOR <expresión>]

siendo como antes, opción, alguna de las anteriores, lista de campos, el nombre de los que se pretende modificar y expresión, si es que se especifica, una determinada condición que han de verificar para que los que la cumplan sean alterados.

```
. go top
. display
00001  Fernando        Lopez           Martinez        Cl. Vallefermoso 23, 5o.
       28004 El Foro                   Madrid            91  4488425 .T.
. change field nombre for prov='Madrid' .and. apell:1='Lopez'

Record: 00001


NOMBRE: Fernando
CHANGE? F
TO       Ph

NOMBRE: Phernando
CHANGE?
. go 1
. display
00001  Phernando       Lopez           Martinez        Cl. Vallefermoso 23, 5o.
       28004 El Foro                   Madrid            91  4488425 .T.
```

Un campo puede ser borrado completamente, pulsando CTRL I tras el mensaje *CHANGE?* Por otra parte, este mandato puede ser cancelado mediante ESC (SAL, en los PCW).

INSERCIÓN DE REGISTROS

Entre las posibilidades de dBASE II, también se cuenta la de insertar un nuevo registro en cualquier posición de la base de datos. Ya estudiamos anteriormente cómo añadir uno más al final de los exis-

tentes, mediante APPEND, pero en muchas ocasiones será necesario ubicarlo en cualquier otro lugar.

Para ello, se han de seguir los siguientes pasos:

— Situar el puntero en el lugar deseado, para lo cual se pueden emplear los mandatos GO o SKIP.

```
. list
00001   Phernando       Lopez           Martinez        Cl. Vallefermoso 23, 5o.
        28004 El Foro                   Madrid            91  4488425 .T.
00002   DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
        28006 El Foro                   Madrid            91  4021099 .F.
00003   Juan            Cuesta          Nuin            Pza. de los desamparados
18,1o   20003 El Foro                   Madrid            91  2742142 .T.
00004   MICROBYTE                                       Paseo de la Castellana 17
9, 1o   28046 El Foro                   Madrid            91  4425433 .F.
00005   Raphael de la   Ossa            Coloma          Cl. General Aranaz 1
        28027 El Foro                   Madrid            91  7243392 .T.
00006   Justo           Pardillo        Pardillo        Cl. del Suspiro Verde, 3
        28000 Orejilla del Sordete      Oregon            45       1 .T.
00007   Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8
        16400 Tarancon                  Cuenca           966   112039 .T.
00008   EDICIONES       INGELEK                         Pza. Republica Ecuador 2,
 1o     28016 El Foro                   Madrid            91  4579424 .F.
00009   Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Lati
na 1    28027 El Foro                   Madrid            91  7422933 .T.
00010   Amelia          Polo            Garcia          Cl. Onesimo Redondo
         8017 Barcelona                 Barcelona         92  3365422 .T.
00011   Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
        28027 El Foro                   Madrid            91  7421590 .T.
```

— Una vez fijado el puntero, la orden INSERT abrirá un espacio para un registro vacío, en la siguiente posición a la que ocupa el puntero. En ese momento se pueden rellenar los diferentes campos que configuran el registro. Para abandonar en cualquier momento el modo de inserción, basta con pulsar CTRL W, si lo que queremos es que el programa acepte los nuevos datos, o CTRL Q si consideramos que no son válidos.

— Para añadir un registro en una posición anterior a la ocupada por el puntero, se debe utilizar la orden INSERT BEFORE y actuar de la forma descrita en el párrafo anterior.

— Otra posibilidad consiste en la inserción de un registro vacío, tanto antes como después del que ocupa la posición determinada por el puntero. Para ello, se ejecutarán los mandatos INSERT BEFORE BLANK o INSERT BLANK respectivamente. La inclusión de registros vacíos dentro de la base de datos, normalmente tiene como objetivo reservar espacio para ciertas informaciones, que se habrán de almacenar con posterioridad.

```
RECORD # 00006
NOMBRE    :Justo          :
APELL:1   :Pardillo       :
APELL:2   :Pardillo       :
DIRECC    :Cl. del Suspiro Verde, 3      :
COD:POST  :28000:
POBLAC    :Orejilla del Sordete     :
PROV      :Oregon         :
PREFIJO   : 45:
TELEF:    :      1:
CARACT:   :t:
```

```
. list
00001  Phernando        Lopez            Martinez        Cl. Vallefermoso 23, 5o.
       28004 El Foro                     Madrid            91  4488425 .T.
00002  DATA-3           INFORMATICA S.A                  Cl. Alcantara 57
       28006 El Foro                     Madrid            91  4021099 .F.
00003  Juan             Cuesta           Nuin            Pza. de los desamparados
18,1o  20003 El Foro                     Madrid            91  2742142 .T.
00004  MICROBYTE                                        Paseo de la Castellana 17
9, 1o  28046 El Foro                     Madrid            91  4425433 .F.
00005  Raphael de la    Ossa             Coloma          Cl. General Aranaz 1
       28027 El Foro                     Madrid            91  7243392 .T.
00006  Pedro Miguel     Lozano           Castellote      Miguel de Cervantes 8
       16400 Tarancon                    Cuenca           966   112039 .T.
00007  EDICIONES        INGELEK                          Pza. Republica Ecuador 2,
 1o    28016 El Foro                     Madrid            91  4579424 .F.
00008  Carlos de la     Ossa             Villacañas      Pza. de la Gramatica Lati
na 1   28027 El Foro                     Madrid            91  7422933 .T.
00009  Amelia           Polo             Garcia          Cl. Onesimo Redondo
        8017 Barcelona                   Barcelona         92  3365422 .T.
00010  Pilar            Manzanera        Amaro           Cl. del Videojuego 1, 1o
       28027 El Foro                     Madrid            91  7421590 .T.
```

```
. go 5
. insert before blank
. skip 3
Record: 00008
. insert blank
. list
00001  Fernando         Lopez            Martinez        Cl. Vallefermoso 23, 5o.
       28004 Madrid                      Madrid            91  4488425 .T.
00002  DATA-3           INFORMATICA S.A                  Cl. Alcantara 57
       28006 Madrid                      Madrid            91  4021099 .F.
00003  Juan             Cuesta           Nuin            Pza. de los desamparados
18,1o  20003 Madrid                      Madrid            91  2742142 .T.
00004  MICROBYTE                                        Paseo de la Castellana 17
9, 1o  28046 Madrid                      Madrid            91  4425433 .F.
00005
            0                                               0        0 .F.
00006  Rafael de la     Ossa             Coloma          Cl. General Aranaz 1
       28027 Madrid                      Madrid            91  7243392 .T.
00007  Pedro Miguel     Lozano           Castellote      Miguel de Cervantes 8
       16400 Tarancon                    Cuenca           966   112039 .T.
00008  EDICIONES        INGELEK                          Pza. Republica Ecuador 2,
 1o    28016 Madrid                      Madrid            91  4579424 .F.
00009
            0                                               0        0 .F.
00010  Carlos de la     Ossa             Villacañas      Pza. de la Gramatica Lati
na 1   28027 Madrid                      Madrid            91  7422933 .T.
00011  Amelia           Polo             Garcia          Cl. Onesimo Redondo
        8017 Barcelona                   Barcelona         92  3365422 .T.
00012  Pilar            Manzanera        Amaro           Cl. del Videojuego 1, 1o
       28027 Madrid                      Madrid            91  7421590 .T.
```

Cuando se añade un registro vacío, los campos de tipo alfabético se ponen a blancos, mientras que los numéricos se rellenan con un cero. Además, cualquier campo de tipo lógico que pudiera existir toma la característica de falso (F).

Puesto que en el momento de añadir un registro en un lugar intermedio de la base de datos, el código numérico de los que se encuentren por debajo de él se debe aumentar en una unidad, en el caso de un fichero con gran cantidad de registros, la actualización de este índice puede llevar un tiempo considerable, en vista de lo cual no es aconsejable utilizar esta orden cuando la base de datos tiene ya un cierto volumen.

BORRADO Y ELIMINACIÓN DE REGISTROS

Aunque no es lo habitual, en el caso de dBASE II, los términos BORRAR y ELIMINAR no tienen un mismo significado. Por motivos de seguridad, el programa no borra directamente un registro que nosotros hayamos indicado, sino que lo marca con un asterisco (*), posibilitando volvernos atrás en esta operación, hasta que confirmemos la eliminación definitiva mediante otro mandato. Esta circunstancia es de especial utilidad, ya que no sólo nos permite recuperarlos, sino incluso copiar en otro fichero los así marcados, evitándose de esta forma lamentables accidentes.

Ya vimos que desde la pantalla de edición, pulsando CTRL U es posible seleccionar el registro afectado como borrado (en el modo BROWSE o de ventana también se puede hacer uso de esta función). Asimismo, dBASE II actúa directamente sobre un registro o un grupo de ellos, que cumplan una determinada condición.

Para señalar un registro o un grupo de éstos como listo para ser borrado, se utiliza el mandato DELETE, con lo que la marca asterisco aparece junto al número de registro en cualquier visualización que efectuemos del mismo. La sintaxis del comando es:

DELETE [<opción>] [FOR <expresión>]

o también...

DELETE [<opción>] [WHILE <expresión>]

es decir, podremos hacer actuar esta orden sobre el registro o grupo de registros definidos en opción y además, si así lo deseamos, sobre

los que cumplan la condición especificada por expresión.

```
. delete for poblac='Madrid'
00008
 deletion(s)
. list
00001 *Fernando          Lopez              Martinez         Cl. Vallefermoso 23, 5o.
       28004 Madrid                        Madrid             91  4488425 .T.
00002 *DATA-3            INFORMATICA S.A                     Cl. Alcantara 57
       28006 Madrid                        Madrid             91  4021099 .F.
00003 *Juan              Cuesta             Nuin             Pza. de los desamparados
18,1o  20003 Madrid                        Madrid             91  2742142 .T.
00004 *MICROBYTE                                            Paseo de la Castellana 17
9, 1o  28046 Madrid                        Madrid             91  4425433 .F.
00005 *Rafael de la      Ossa               Coloma           Cl. General Aranaz 1
       28027 Madrid                        Madrid             91  7243392 .T.
00006  Pedro Miguel      Lozano             Castellote       Miguel de Cervantes 8
       16400 Tarancon                      Cuenca            966  112039 .T.
00007 *EDICIONES         INGELEK                             Pza. Republica Ecuador 2,
 1o    28016 Madrid                        Madrid             91  4579424 .F.
00008 *Carlos de la      Ossa               Villacañas       Pza. de la Gramatica Lati
na 1   28027 Madrid                        Madrid             91  7422933 .T.
00009  Amelia            Polo               Garcia           Cl. Onesimo Redondo
        8017 Barcelona                     Barcelona          92  3365422 .T.
00010 *Pilar             Manzanera          Amaro            Cl. del Videojuego 1, 1o
       28027 Madrid                        Madrid             91  7421590 .T.
```

Para recuperar un grupo de registros marcados como borrados y devolverlos al estado inicial, se utiliza el mandato RECALL. Como la mayoría de las órdenes de dBASE II, RECALL también puede actuar de forma selectiva, en función de lo señalado en opción y expresión, según la estructura:

RECALL [<opción>] FOR [<expresión>]

Puesto que los registros marcados con el carácter asterisco siguen estando presentes en la base de datos, ocupan un lugar que en cierto momento puede ser necesario desalojar, a fin de incluir nueva información; para ello, empleamos el mandato PACK. Por motivos obvios, dicha orden debe manejarse con la precaución que se merece.

COPIA DE FICHEROS

En cualquier tratamiento informático, duplicar el contenido de los ficheros con datos de cierta importancia es un trabajo que a nadie

```
. recall for .not. caract:
00003
 recall(s)
. list
00001 *Fernando          Lopez              Martinez        Cl. Vallefermoso 23, 5o.
       28004 Madrid                        Madrid            91  4488425 .T.
00002  DATA-3            INFORMATICA S.A                    Cl. Alcantara 57
       28006 Madrid                        Madrid            91  4021099 .F.
00003 *Juan              Cuesta             Nuin            Pza. de los desamparados
18,1o  20003 Madrid                        Madrid            91  2742142 .T.
00004  MICROBYTE                                           Paseo de la Castellana 17
9, 1o  28046 Madrid                        Madrid            91  4425433 .F.
00005 *Rafael de la      Ossa               Coloma          Cl. General Aranaz 1
       28027 Madrid                        Madrid            91  7243392 .T.
00006  Pedro Miguel      Lozano             Castellote      Miguel de Cervantes 8
       16400 Tarancon                      Cuenca           966   112039 .T.
00007  EDICIONES         INGELEK                            Pza. Republica Ecuador 2,
 1o    28016 Madrid                        Madrid            91  4579424 .F.
00008 *Carlos de la      Ossa               Villacañas      Pza. de la Gramatica Lati
na 1   28027 Madrid                        Madrid            91  7422933 .T.
00009  Amelia            Polo               Garcia          Cl. Onesimo Redondo
        8017 Barcelona                     Barcelona         92  3365422 .T.
00010 *Pilar             Manzanera          Amaro           Cl. del Videojuego 1, 1o
       28027 Madrid                        Madrid            91  7421590 .T.
```

```
. pack
Pack complete00005
 records copied
. list
00001  DATA-3            INFORMATICA S.A                    Cl. Alcantara 57
       28006 Madrid                        Madrid            91  4021099 .F.
00002  MICROBYTE                                           Paseo de la Castellana 17
9, 1o  28046 Madrid                        Madrid            91  4425433 .F.
00003  Pedro Miguel      Lozano             Castellote      Miguel de Cervantes 8
       16400 Tarancon                      Cuenca           966   112039 .T.
00004  EDICIONES         INGELEK                            Pza. Republica Ecuador 2,
 1o    28016 Madrid                        Madrid            91  4579424 .F.
00005  Amelia            Polo               Garcia          Cl. Onesimo Redondo
        8017 Barcelona                     Barcelona         92  3365422 .T.
```

debe pesarle, puesto que en caso de accidente, mantener copias de seguridad será la única manera de no perder horas de trabajo.

Para realizar una copia exacta de un fichero, debe informarse primero mediante el mandato USE a dBASE II cuál es el que pretendemos duplicar y seguidamente, ejecutar el mandato COPY, es decir, la secuencia de acontecimientos sería la siguiente:

USE <n-fich>
COPY TO <n-fich nuevo>

De esta manera, se consigue obtener una copia exacta del fichero implicado, incluyendo aquellos registros que pudieran estar marcados como borrados.

El mandato COPY permite, como ya es norma habitual, la actuación selectiva sobre determinados campos e incluso una vez definido este rango, sobre los que además cumplan una condición específica. La sintaxis completa sería la siguiente:

COPY TO [d:] <n-fich nuevo> [<opción>] [FIELD <lista de campos>] [FOR <expresión>] [SDF] [WHILE <expresión>] [DELIMITED WITH <separador>]

El mandato COPY admite algunas cláusulas, las cuales permiten acomodar el formato del nuevo fichero, de manera tal que pueda ser utilizado por otros programas.

Si se añade el identificativo SDF *(System Data Format,* formato de datos del sistema), se creará un fichero de extensión .TXT, en el cual sólo están duplicados los datos del original, en formato estándar ASCII, ignorándose su estructura, lo cual posibilitará el empleo por procesadores distintos al dBASE II.

También puede utilizarse la cláusula DELIMITED, con lo que los diferentes campos del fichero destino quedarán delimitados por apóstrofes (') y separados unos de otros por comas; de esta manera, es posible acceder a ellos de forma secuencial, desde programas escritos en otros lenguajes.

Si además empleamos la cláusula WITH, podremos seleccionar el separador que deseemos. En caso de ser la coma (,) el discriminador elegido, los blancos a la derecha de los campos de caracteres serán eliminados, al igual que los situados a la izquierda de los numéricos; por otra parte, los literales no estarán encerrados entre comillas.

```
use ntelef
. copy to telef-1 field nombre,apell:1,apell:2,direcc sdf
00010
 records copied
. copy to telef-2 field nombre,apell:1,apell:2,direcc delimited with :
00010
 records copied
```

```
A>type m:telef-1.txt
Fernando        Lopez           Martinez        Cl. Vallefermoso 23, 5o.
DATA-3          INFORMATICA S.A                 Cl. Alcantara 57
Juan            Cuesta          Nuin            Pza. de los desamparados 18,1o
MICROBYTE                                       Paseo de la Castellana 179, 1o
Rafael de la    Ossa            Coloma          Cl. General Aranaz 1
Pedro Miguel    Lozano          Castellote      Miguel de Cervantes 8
EDICIONES       INGELEK                         Pza. Republica Ecuador 2, 1o
Carlos de la    Ossa            Villacañas      Pza. de la Gramatica Latina 1
Amelia          Polo            Garcia          Cl. Onesimo Redondo
Pilar           Manzanera       Amaro           Cl. del Videojuego 1, 1o
```

```
A>type m:telef-2.txt
:Fernando:,:Lopez:,:Martinez:,:Cl. Vallefermoso 23, 5o.:
:DATA-3:,:INFORMATICA S.A:,: :,:Cl. Alcantara 57:
:Juan:,:Cuesta:,:Nuin:,:Pza. de los desamparados 18,1o:
:MICROBYTE:,: :,: :,:Paseo de la Castellana 179, 1o:
:Rafael de la:,:Ossa:,:Coloma:,:Cl. General Aranaz 1:
:Pedro Miguel:,:Lozano:,:Castellote:,:Miguel de Cervantes 8:
:EDICIONES:,:INGELEK:,: :,:Pza. Republica Ecuador 2, 1o:
:Carlos de la:,:Ossa:,:Villacañas:,:Pza. de la Gramatica Latina 1:
:Amelia:,:Polo:,:Garcia:,:Cl. Onesimo Redondo:
:Pilar:,:Manzanera:,:Amaro:,:Cl. del Videojuego 1, 1o:
```

CUADRO DE MANDATOS

— COPY STRUCTURE
— MODIFY STRUCTURE
— APPEND FROM
— DELETE FILE
— RENAME
— EDIT
— REPLACE
— CHANGE
— INSERT (BEFORE) (BLANK)

BORRADO Y ELIMINACION DE REGISTROS

— DELETE

— RECALL
— PACK

COPIA DE FICHEROS

— COPY TO
— CLAUSULAS SDF Y DELIMITED

BÚSQUEDAS Y CLASIFICACIONES

uando el tamaño de la base de datos aumenta considerablemente, intentar recordar el número que ocupa un determinado registro, o más aún, su contenido es poco más que imposible. Para estos inoportunos olvidos, y en previsión de la pérdida de tiempo que supone la consulta manual, registro a registro, hasta encontrar el que nos interesa, dBASE II dispone de una serie de funciones que permitirán localizar el que nos interese, aunque sólo recordemos algunos datos sobre él.

BÚSQUEDAS DE INFORMACIÓN

El mandato LOCATE ofrece la posibilidad de buscar un campo perteneciente a un determinado registro, por cualquier criterio. Su sintaxis completa responde a la siguiente construcción:

LOCATE [<opción>] FOR <expresión>

donde opción como ya es costumbre puede ser ALL, NEXT N, o RECORD, y expresión, cualquiera sintácticamente correcta, y de una longitud máxima de 128 caracteres. Por ejemplo,

```
. use telef
. locate for nombre='Pilar'
Record: 00010
. display
00010  Pilar          Manzanera       Amaro          Cl. del Videojuego 1, 1o
       28027 Madrid                   Madrid           91  7421590 .T.
```

El programa contesta mostrando el número de registro en el cual se da la ocurrencia de expresión. Este puede ser visualizado mediante LIST o DISPLAY.

Naturalmente, es frecuente que el contenido definido en expresión, no sólo se encuentre en un determinado registro, sino en varios de la base de datos. Cuando este sea el caso, el mandato CONTINUE reanuda la búsqueda informándonos de cualquier aparición posterior, o en caso contrario, presenta el mensaje *End of file encountered* (Localizado el final del fichero), indicando que no se halló ninguno más que coincidiera con el descrito.

```
. locate for prov<>'Madrid'
Record: 00006
. disp
00006  Pedro Miguel    Lozano          Castellote     Miguel de Cervantes 8
       16400 Tarancon                  Cuenca          966   112039 .T.
. continue
Record: 00009
. display
00009  Amelia          Polo            Garcia         Cl. Onesimo Redondo
        8017 Barcelona                 Barcelona        92  3365422 .T.
. continue
End of file encountered
```

Cuando esto ocurra, conviene puntualizar que el puntero de registro, queda señalando al último de la base de datos. Además, si la opción especificada en la orden fue NEXT N, de no acabar con éxito

la búsqueda, dBASE II muestra la indicación *End of LOCATE* (Fin del mandato LOCATE), quedando el puntero sobre el último registro afectado.

CLASIFICACIONES

Entre las funciones de ayuda al usuario con que toda base de datos debe contar, están las destinadas a crear, una vez almacenada la información, una lista clasificada según el criterio elegido (nombre, dirección, fecha, distintos códigos, etc.) de los datos que contiene.

Con dBASE II, las clasificaciones pueden efectuarse en orden creciente o decreciente, siguiéndose la presente construcción:

SORT ON [<nombre campo>] TO [d:] [<n-fich>] [ASCENDING/DESCENDING]

De esta manera, se consigue que los registros que configuran nuestro fichero, resulten ordenados ascendente o descendentemente, según el contenido del campo especificado en la instrucción, construyéndose un nuevo fichero (n-fich) que los aloje en este nuevo aspecto, para que el original no sufra ninguna modificación.

```
. sort on nombre to clnombre
SORT COMPLETE
. use clnombre
. list nombre,apell:1,apell:2
00001   Amelia          Polo            Garcia
00002   Carlos de la    Ossa            Villacañas
00003   DATA-3          INFORMATICA S.A
00004   EDICIONES       INGELEK
00005   Fernando        Lopez           Martinez
00006   Juan            Cuesta          Nuin
00007   MICROBYTE
00008   Pedro Miguel    Lozano          Castellote
00009   Pilar           Manzanera       Amaro
00010   Rafael de la    Ossa            Coloma
```

```
. sort on nombre to clnombre descending
SORT COMPLETE
. use clnombre
. list nombre,apell1:1,apell1:2
00001   Rafael de la     Ossa              Coloma
00002   Pilar            Manzanera         Amaro
00003   Pedro Miguel     Lozano            Castellote
00004   MICROBYTE
00005   Juan             Cuesta            Nuin
00006   Fernando         Lopez             Martinez
00007   EDICIONES        INGELEK
00008   DATA-3           INFORMATICA S.A
00009   Carlos de la     Ossa              Villacañas
00010   Amelia           Polo              Garcia
```

Es importante tener presente, que el utilizar el mismo fichero como origen de los datos sin clasificar y a la par como destino de los clasificados, conducirá al error *File is currently open* (el fichero está abierto en la actualidad).

Las clasificaciones se efectúan según el código ASCII *(American Standard Code for Information Interchange,* Código Americano Normalizado para el Intercambio de Información). En este sentido, es de destacar, que a efectos de ordenaciones, los caracteres en minúsculas son diferentes de sus correspondientes en mayúsculas, puesto que generan códigos distintos.

Por este motivo, es importante plantearse antes de introducir los datos, un criterio uniforme (mayúsculas o minúsculas) en los campos alfabéticos. Es más, tengamos en cuenta que si como destino de una información numérica escogemos un campo del tipo alfabético, dos datos como 98 y 971, una vez clasificados en sentido creciente, nos darán el inesperado resultado que 39 es mayor que 398, puesto que las comparaciones se efectúan carácter a carácter, en nuestro caso, el 9 con el 9, el 8 con el 7, y, por tanto, el programa considerará mayor la primera cadena que la segunda. Mucho cuidado en este sentido.

También es posible realizar clasificaciones sucesivas por varios criterios. Para ello, se debe indicar una serie de mandatos SORT, con los campos que deseamos emplear para la ordenación, comenzando por el menos significativo.

CUADRO DE MANDATOS

— LOCATE
— CONTINUE
— SORT

```
. use telef
. sort on apell:2 to sort-1
SORT COMPLETE
. use sort-1
. sort on apell:1 to sort-2
SORT COMPLETE
. use sort-2
. sort on poblac to sort-3
SORT COMPLETE
. use sort-3
. list nombre,apell:1,apell:2,poblac
00001  Amelia          Polo             Garcia          Barcelona

00002  MICROBYTE                                        Madrid

00003  Juan            Cuesta           Nuin            Madrid

00004  DATA-3          INFORMATICA S.A                  Madrid

00005  EDICIONES       INGELEK                          Madrid

00006  Fernando        Lopez            Martinez        Madrid

00007  Pilar           Manzanera        Amaro           Madrid

00008  Rafael de la    Ossa             Coloma          Madrid

00009  Carlos de la    Ossa             Villacañas      Madrid

00010  Pedro Miguel    Lozano           Castellote      Tarancon
```

VARIABLES DE MEMORIA Y FUNCIONES

i repasamos las características más relevantes de dBASE II, encontraremos entre ellas el comportarse como algo más que una simple base de datos; además, permite diseñar programas que gestionen de una forma óptima los ficheros, automatizándose de esta manera todo el mantenimiento de los archivos, almacenar resultados intermedios, incluso someter los datos a la acción específica de algunas funciones que este estupendo programa es capaz de evaluar.

En el presente capítulo, nos centraremos en estos dos últimos aspectos y en los mandatos de dBASE II que nos permitirán utilizarlos. Cuando tratemos el tema de la programación, valoraremos en toda su extensión la posibilidad de incluir entre las instrucciones que configuren un programa, totalizadores, funciones y variables de cualquier tipo, tal como que se manejan desde un lenguaje de programación cualquiera.

VARIABLES EN DBASE II

Una variable es una zona de memoria reservada para alojar ciertos datos y referenciada o reconocible por un nombre que la define uní-

vocamente. Puede almacenar informaciones de tipo numérico, alfabético o lógico, y el número de ellas que dBASE II es capaz de mantener simultáneamente en memoria es de 64, aunque como en seguida comprobaremos, no es óbice para disponer de un número superior, puesto que pueden ser grabadas en el disco, y hacer uso de ellas en el momento que las necesitemos.

Las variables alfabéticas o de caracteres admiten un máximo de 254 posiciones, mientras que en las numéricas se considerarán significativas las diez primeras cifras (una cantidad numérica decimal sigue el mismo criterio, y además el punto decimal cuenta como otra posición), sustituyéndose por ceros todas aquellas situadas a partir de la décima posición.

Para la creación y asignación de un determinado valor a una variable, se utiliza el mandato STORE, según la estructura siguiente:

STORE <expresión> TO <variable>

Si la variable implicada en el mandato ya existía, su valor es sustituido por el especificado en expresión y su tamaño pasa a ser el del último dato introducido.

Naturalmente, expresión puede ser cualquiera de las manejadas por dBASE II, es decir, una cantidad constante, el resultado de una operación sobre otras variables, el contenido de algún campo del fichero sobre el que se está trabajando, etc., como puede comprobarse en los siguientes ejemplos.

```
. use telef
. store 123 to var1
  123
. store nombre to var2
Fernando
. go 9
. store nombre to var3
Amelia
. store 20*100 to var4
  2000
. display memory
A           (N)     123
B           (C)  Fernando
VAR1        (N)     123
VAR2        (C)  Fernando
VAR3        (C)  Amelia
VAR4        (N)    2000

** Total **      06 VARIABLES USED  00069 BYTES USED
```

De ellos, hemos de obtener algunas conclusiones de importancia. En primer lugar, en el nombre asignado a la variable sólo son significativos los primeros «xx» caracteres. Además, para introducir una cadena de caracteres **constante** en una variable, debe encerrarse su contenido entre comillas (") o apóstrofes ('). Por otra parte, puede suceder que a una variable que previamente contenía un valor numérico, en una operación posterior le asignemos uno del tipo alfabético, o viceversa. dBASE II se encarga automáticamente de discernir el tipo de variable utilizado.

El mandato DISPLAY MEMORY o LIST MEMORY, visualiza una lista completa de las variables actualmente almacenadas en la memoria, precisando su tipo (C=Caracteres, N=Numérica, L=Lógica) y su contenido. Finalmente, en la última línea, se nos informa del número total de variables definidas en la actualidad y del montante de memoria que ocupan.

Si las circunstancias nos obligan a manejar más de 64 variables, podemos salvaguardar el contenido de las presentes en la memoria, en el disco, mediante la orden SAVE TO, indicando al final el nombre del fichero (n-fich) que deseamos las almacene, según la sintaxis siguiente:

SAVE TO <n-fich> [ALL LIKE <plantilla>]

Este último adquirirá la extensión .MEM, indicándonos que su contenido responde a la zona de memoria reservada para almacenar el valor de las variables. La plantilla, es cualquier grupo de caracteres que nos permite señalar un conjunto de variables a grabar, y no la totalidad de la zona; a tal fin, podemos emplear los símbolos comodín (interrogación y asterisco).

```
. save to variabl all like var?
. list memory
A             (N)     123
B             (C)  Fernando
VAR1          (N)     123
VAR2          (C)  Fernando
VAR3          (C)  Amelia
VAR4          (N)     2000

** Total **      06 VARIABLES USED  00069 BYTES USED
```

Ahora ya estamos en condiciones de desalojar el espacio que estas variables ocupaban y utilizar una nueva serie. Para ello se le indica al programa la orden:

RELEASE [<lista de variables> / ALL]

siendo lista de variables el nombre asignado a cada una de ellas, separados cada uno del siguiente por una coma (,). Para limpiar completamente el contenido de la zona de memoria reservada para el almacenamiento de las variables, basta con ejecutar la orden RELEASE ALL.

```
. release all
. display memory

** Total **      00 VARIABLES USED  00000 BYTES USED
```

Por otra parte, para recuperar desde el disco un bloque de variables previamente almacenados en un fichero .MEM, se utiliza la orden:

RESTORE FROM <n-fich>

siendo n-fich el nombre del fichero que las contenía. Al ejecutar este mandato, el último grupo de valores sustituye a cualquier otro que pudiera encontrarse previamente en la memoria. Por ello, antes de efectuarlo, conviene asegurarse de haberlo grabado, o si no es esa nuestra intención, cerciorarse que la información que almacenan no nos sirve para nada, puesto que será destruida al cargar el nuevo fichero de variables.

```
. restore from variabl
. list memory
VAR1         (N)     123
VAR2         (C)  Fernando
VAR3         (C)  Amelia
VAR4         (N)    2000

** Total **      04 VARIABLES USED  00046 BYTES USED
```

FUNCIONES DE dBASE II

Con el objetivo de ser utilizadas tanto en órdenes directas como dentro de cualquier procedimiento incluido en un programa, dBASE II pone a nuestra disposición una serie de funciones auxiliares, que nos facilitan cualquier tratamiento de la información contenida en la base de datos. Podemos clasificarlas en los siguientes grupos:

a) Funciones de propósito general.
   — Función clase de expresión, TYPE (<expresión>).
   Devuelve un carácter identificativo del tipo de datos utilizados en expresión:

   - C → Caracteres alfabéticos.
   - N → Numéricos puros o decimales.
   - L → Lógicos.
   - U → Indefinidos *(Undefined),* por ejemplo, al interrogar sobre el contenido de una variable a la que no se le ha asignado todavía valor alguno.

   — Cálculo del puntero de registro, #.
   Obtiene el número de registro al cual apunta, en el momento de evaluar esta función, el puntero interno de la base de datos. Si no se está trabajando sobre ningún fichero, contiene la última anotación que le pudiera haber llegado. Nótese que su valor no es modificable directamente, sino que dBASE II, lo actualiza de forma automática, cada vez que se altera el puntero del fichero en curso.
   — Fecha del sistema, DATE().
   En cierto modo, más que como una función podríamos considerar DATE como una variable interna de dBASE II, puesto que su misión consiste en almacenar la fecha en el formato Mes/Día/Año en que fue definida a la entrada del programa. Es importante puntualizar que dicha fecha nada tiene que ver con la que es posible seleccionar desde CP/M mediante la orden DATE SET.
   — Análisis de la existencia de ficheros, FILE (expresión).
   Precisa si el fichero cuyo nombre se define mediante expresión, se encuentra sobre la unidad de disco especificada, haciéndolo notar con .T. *(True,* verdadero) si es que lo encuentra o con .F. *(False,* falso), en caso contrario.

```
. ? date()
13/10/86
. ? file('telef')
.T.
. ? file('noexiste')
.F.
```

— Evaluación lógica de una expresión, TEST (<expresión>).

Determina si expresión existe o no (se está manejando en ese momento, ya sea por tratarse del nombre de una variable o de un campo de base de datos en curso), devolviendo un valor diferente de cero en caso afirmativo, o cero, en caso contrario.

```
. ? test (nombre='Fernando')
  1
. ? test (var5)
  0
```

— Fin de fichero, EOF

La función EOF *(End Of File)* devuelve el valor lógico .T. cuando el puntero interno de la base de datos ha alcanzado el final del fichero, o .F., en caso contrario. Se utiliza con frecuencia, cuando en el desarrollo de un programa se leen secuencialmente los registros que componen un determinado archivo y se desea tomar una acción posterior al llegar al final de éste.

b) Funciones de acción sobre cadenas de caracteres.

— Cálculo de la longitud, LEN (<expresión>).

```
. ? len('cadena de caracteres')
  20
. ? len(direcc)
  30
```

Evalúa la longitud de la cadena de caracteres situada entre los paréntesis. Esta puede ser el resultado de una operación sobre una serie de variables o cadenas. No es posible aplicar la función LEN sobre datos de tipo numérico o lógico.

— Conversión de minúsculas a mayúsculas, ! (<expresión>).

```
. ? !('las letras entre parentesis se convertiran en mayusculas')
LAS LETRAS ENTRE PARENTESIS SE CONVERTIRAN EN MAYUSCULAS
```

Cambia todos los caracteres de la «a» a la «z» que pudiera contener expresión a sus equivalentes letras mayúsculas, dejando inalterados los que se encontraran dentro de este rango. En las rutinas de toma de datos implementadas, es conveniente su uso para fijar un criterio uniforme, que evite ambigüedades en el almacenamiento de la información. Téngase en cuenta, por ejemplo, que tres palabras como Fernando, fernando o ferNando, serán completamente diferentes para dBASE II, por lo cual es imprescindible unificar criterios de forma que, a la larga, nos eviten lamentables pérdidas de tiempo.

— Conversión de una expresión a su valor numérico, VAL (<expresión>).

Transforma una cadena de caracteres formada por una serie de dígitos, incluyendo su signo (+ ó −) y el punto decimal si existiera, a su correspondiente expresión numérica. Esta función debe manejarse con cierta precaución, pues los resultados a los que conduce, no siempre son los esperados como se desprende de los siguientes ejemplos.

— Extracción de subcadenas, $(<expresión>, <comienzo>, <longitud>).

Obtiene a partir de la cadena de caracteres definida por expresión, una subcadena formada por el número de caracteres especificado en «longitud» a partir del que ocupa la posición «comienzo».

— Análisis de la existencia de una subcadena perteneciente a otra principal, (<expresión1>) $ (<expresión2>).

Devuelve el resultado lógico .T. (verdadero), si la cadena definida por «expresión1» está contenida en la especificada en «expresión2», o .F. (Falso), en caso contrario.

— Búsqueda de subcadenas, @ (<expresión1>, <expresión2>).

Proporciona como resultado la posición a partir de la cual se sitúa, dentro de la cadena definida por «expresión2», la subcadena «expresión1». Si no la encuentra, el resultado de ejecutar esta función es 0.

— Eliminación de blancos, TRIM (expresión)

Desprecia todos los espacios en blanco situados en una cadena, desde el último carácter significativo de ésta, hacia la derecha. De esta manera, es posible agrupar datos contenidos en variables o campos alfabéticos sin que queden espacios entre ellos, mejorándose la presentación de los resultados.

```
. ? $(var2,4,3)
nan
. ? 'nan' $ 'Fernando'
.T.
. ? 'fer' $ 'Fernando'
.F.
. ? @('nan',var2)
   4
. ? @('fer',var2)
   0
. list nombre,apell:1
00001  Fernando         Lopez
00002  DATA-3           INFORMATICA S.A
00003  Juan             Cuesta
00004  MICROBYTE
00005  Rafael de la     Ossa
00006  Pedro Miguel     Lozano
00007  EDICIONES        INGELEK
00008  Carlos de la     Ossa
00009  Amelia           Polo
00010  Pilar            Manzanera
. list trim(nombre),apell:1
00001  Fernando Lopez
00002  DATA-3 INFORMATICA S.A
00003  Juan Cuesta
00004  MICROBYTE
00005  Rafael de la Ossa
00006  Pedro Miguel Lozano
00007  EDICIONES INGELEK
00008  Carlos de la Ossa
00009  Amelia Polo
00010  Pilar Manzanera
```

— Cálculo del código ASCII de un carácter, RANK (<expresión>).

Obtiene la versión decimal correspondiente al código ASCII del primer carácter de la cadena definida en «expresión». Como en el siguiente apartado comprobaremos, su efecto es inverso al realizado por la función CHR.

— Función de macrosustitución, & (nombre de variable).

Esta potente función provoca que el nombre de la variable de caracteres que le sigue, sea sustituido por su contenido. De esta manera, cuando una expresión de relativa complejidad se maneja repetidamente, puede almacenarse como cadena en una variable de este tipo, y haciendo actuar la función & sobre ella, ejecutarse con el consiguiente ahorro de tiempo y pulsaciones. Conviene prestar especial atención al ejemplo siguiente:

```
. ? rank(var2)
  70
. store 'display files like *.dbf' to disp
display files like *.dbf
. &disp

PEPE     .DBF        JUAN     .DBF        MELY     .DBF        COPIA    .DBF
TOTAL    .DBF        ORCOPIA  .DBF        TOT      .DBF        ORNOMBRE.DBF
SPOBLAC  .DBF        SORTINDE.DBF         BIBLIOT  .DBF        AAA      .DBF
TELEFNEW.DBF         BBB      .DBF        CCC      .DBF        SORT-1   .DBF
SORT-2   .DBF        SORT-3   .DBF        TELEF    .DBF
```

— Suma total de cadenas, (<expresión1>)+(<expresión2>)+...

Efectúa la unión de las cadenas definidas por «expresión1», «expresión2», etc., en una única, cuya longitud será la suma de los tamaños de las implicadas.

— Suma parcial de cadenas (<expresión1>)−(<expresión2>)−...

```
. use telef
. ? nombre+apell:1+apell:2
Fernando        Lopez           Martinez
. ? nombre-apell:1-apell:2
FernandoLopezMartinez
. store nombre+apell:1 to var5
Fernando        Lopez
. store nombre-apell:1 to var6
FernandoLopez
. ? len(var5)
  30
. ? len(var6)
  30
```

Realiza la unión de las cadenas implicadas en la orden, aunque en este caso, los blancos por la derecha que pudieran contener son despreciados. Sin embargo, la longitud de la expresión final es suma de los tamaños de cada expresión particular.

c) De acción sobre datos numéricos.

— Cálculo de la parte entera, INT (<expresión>).

Desprecia los decimales que pudiera contener el valor especificado por «expresión», obteniéndose su parte entera. Los términos contenidos en «expresión» pueden ser directamente dígitos numéricos, o una serie de variables o campos que conduzcan a un resultado de este tipo.

— Conversión de una expresión numérica en cadena de caracteres, STR (<expresión>,<longitud>,<decimales>).

Transforma el valor dado por «expresión» a una cadena de caracteres de tamaño definido en «longitud», tomando del valor original el número de dígitos situados a la derecha del punto decimal señalados en «decimales».

```
. store 1234.56 to decimal
 1234.56
. ? int(decimal)
     1234
. store str(decimal,7,2) to cadena
1234.56
. ? type(cadena)
C
. ? chr(65)
A
```

— Conversión a ASCII, CHR (<expresión>).

Evalúa el contenido de «expresión» y convierte el valor obtenido en el carácter ASCII correspondiente a dicho valor. Manejando la función CHR se pueden provocar acciones específicas del sistema, cuando los códigos generados no son imprimibles (de control), como el borrado de la pantalla, la activación del vídeo inverso, etc.

```
. use telef
. list nombre,apell:1,poblac
00001  Fernando          Lopez              Madrid
00002  DATA-3            INFORMATICA S.A    Madrid
00003  Juan              Cuesta             Madrid
00004  MICROBYTE                            Madrid
00005  Rafael de la      Ossa               Madrid
00006  Pedro Miguel      Lozano             Tarancon
00007  EDICIONES         INGELEK            Madrid
00008  Carlos de la      Ossa               Madrid
00009  Amelia            Polo               Barcelona
00010  Pilar             Manzanera          Madrid
. count for poblac='Madrid' to capital
Count =  00008
. ? capital
         8
```

TOTALIZADORES

Una vez revisadas todas las funciones que pone a nuestra disposición dBASE II, continuaremos con una serie de mandatos encaminados a realizar totalizaciones sobre los campos de un determinado fichero, que cumplan alguna condición específica que nosotros mismos les impondremos, si es que éste es nuestro deseo, y que mantengan una estrecha relación con las variables de memoria y funciones antes discutidas.

El mandato COUNT proporciona el número de registros que cumplen una condición específica. Su sintaxis completa responde a la siguiente estructura:

COUNT [<opción>] [FOR <expresión>] [TO <variable>]

donde como ya es habitual, «opción» puede ser ALL, NEXT N, o RECORD N, «expresión» cualquiera válida y «variable» el nombre asignado a la variable a que pretendemos enviar el resultado.

Si el mandato se ejecuta solo (COUNT), sin ninguna de sus opciones, devuelve el número total de registros en el fichero investigado. Si además elegimos una variable como destino del resultado, su valor quedará almacenado en ella y podrá consultarse en cualquier momento mediante la orden DISPLAY MEMORY.

Cuando dBASE II procesa esta orden, también totaliza aquellos registros que cumpliendo la condición especificada, se encuentren marcados con el signo *, como borrados.

Cuando se trata de totalizar las cantidades almacenadas en una serie de campos del fichero, sometidos o no a alguna condición, dBASE II pone a nuestra disposición el mandato SUM. Su estructura muestra bastante bien las posibilidades que conlleva:

SUM <lista de campos> [TO <lista de variables>] [opción] [FOR <expresión>]

Analicémosla con más detenimiento. Tras el mandato SUM, «lista de campos» es el nombre de aquellos sobre los cuales deseamos que actúe la orden. Deberán separarse por comas (,) cada uno del siguiente, si es que escogemos más de uno.

En principio, esto sería suficiente, pero como la mayoría de los mandatos de dBASE II, se nos permite «exprimir» bastante más la instrucción. Si tras TO indicamos el nombre de una serie de variables, en ellas quedarán alojados los resultados obtenidos. Naturalmente, el total del primer campo se conservará en la primera variable, el del segundo en la situada en esta misma posición tras TO, y así, sucesivamente. Si el número de variables es menor al de campos a totalizar, dBASE II creará una variable de nombre propio para almacenar el resultado.

```
. sum cod:post,prefijo,telef: to tcp,tpref,ttel for poblac<>'Madrid'
  24417   1058   3477461
. ? tcp
     24417
. ? tpref
      1058
. ? ttel
    3477461
```

Nuevamente, con «opción» podemos restringir el campo de acción a sólo una determinada zona del fichero, y además, «expresión» permite que únicamente los campos que la cumplan sean contabilizados.

En el caso de este mandato y a diferencia de lo ocurrido al ejecutar COUNT, los registros marcados para ser borrados, NO serán contabilizados. Por otra parte, el número de campos a totalizar que se pueden incluir dentro de una instrucción SUM, queda limitado a cinco.

CUADRO DE MANDATOS

- — VARIABLES EN dBASE II
  - STORE
  - DISPLAY MEMORY
  - LIST MEMORY
  - SAVE TO
  - RELEASE
  - RESTORE FROM
- — FUNCIONES DE dBASE II
  - DE PROPOSITO GENERAL
    - — TYPE
    - — #
    - — DATE
    - — FILE
    - — TEST
    - — EOF
  - DE ACCION SOBRE CADENAS DE CARACTERES
    - — LEN
    - — !
    - — VAL
    - — $
    - — @
    - — TRIM
    - — RANK
    - — &
    - — CONCATENACION +, –
  - DE ACCION SOBRE DATOS NUMERICOS
    - — INT
    - — STR
    - — CHR

— TOTALIZADORES
  - COUNT
  - SUM

FICHEROS INDEXADOS

uando el tamaño de los ficheros que constituyen la base de datos aumenta desmesuradamente, las búsquedas de información siguiendo técnicas secuenciales, retrasan en exceso la respuesta del sistema.

Para evitar este inconveniente y optimizar los procesos relacionados con la búsqueda y clasificación de la información, dBASE II permite la creación de ficheros indexados, a partir de nuestra base de datos original.

CREACIÓN DE UN FICHERO INDEXADO

La creación de un índice se efectúa aplicando el mandato INDEX sobre el fichero con el cual nos encontramos trabajando. Para ello, se sigue la regla sintáctica...

INDEX ON <clave> TO <n-fich>

es decir, el indexado del fichero en curso se define a partir de alguna de las claves o campos que lo forman, aunque también es posible utilizar variables o expresiones complejas de hasta un máximo de no-

venta y nueve caracteres de largo. El nuevo fichero de índices creado (n-fich), adquirirá la extensión .NDX (iNDeXed).

```
. use telef
. index on nombre to inombre
00010
 records indexed
. index on apell:1 to iapell:1
00010
 records indexed
. index on poblac to ipoblac
00010
 records indexed
. display files like *.ndx

IPOBLAC .NDX          INOMBRE .NDX          IAPELL:1.NDX
```

Mientras que la clasificación de un fichero respecto de un determinado criterio efectúa su proceso asignado a cada registro un número correlativo, el indexado respeta el que tuvieran definidos los registros del archivo original, creándose una tabla de índices, ordenada respecto al argumento elegido.

Se pueden crear sobre un determinado fichero tantos índices como se desee, incluso uno para cada campo que componga su estructura. Pero en la memoria del ordenador a lo sumo deben encontrarse activos siete.

Por otra parte, el fichero creado a partir del mandato SORT tendrá el mismo tamaño que el original, lo que puede conducir al desbordamiento de la capacidad del disco, cuando de un archivo de gran tamaño se trate. Sin embargo, los ficheros indexados son de considerable menor tamaño.

No obstante, las búsquedas de información sólo pueden realizarse sobre el indexado especificado en primer lugar, manteniéndose los restantes para proceder a su actualización si es preciso.

El acceso a las claves definidas se realiza a través del mandato USE, determinando tras él los argumentos (índices) con los que se va a trabajar, según la siguiente estructura:

USE <n-fich> INDEX <n-fich ind1>, <n-fich ind2>,...

siendo «n-fich ind1», «n-fich ind2» el nombre dado durante el proceso de indexación a los ficheros de extensión .NDX que contienen las claves.

Si se quiere indexar un fichero según varios campos (ordenamiento múltiple), será suficiente unir el nombre de los elegidos mediante el signo más (+). En este sentido, se debe prestar atención a la clase de los campos que se pretende relacionar, puesto que deben ser coherentes (del mismo tipo).

```
. use telef index inombre,iapell:1,ipoblac
. list nombre,apell:1,poblac
00009  Amelia          Polo              Barcelona
00008  Carlos de la    Ossa              Madrid
00002  DATA-3          INFORMATICA S.A   Madrid
00007  EDICIONES       INGELEK           Madrid
00001  Fernando        Lopez             Madrid
00003  Juan            Cuesta            Madrid
00004  MICROBYTE                         Madrid
00006  Pedro Miguel    Lozano            Tarancon
00010  Pilar           Manzanera         Madrid
00005  Rafael de la    Ossa              Madrid
```

```
. use telef index iapell:1
. list nombre,apell:1,poblac
00004  MICROBYTE                         Madrid
00003  Juan            Cuesta            Madrid
00002  DATA-3          INFORMATICA S.A   Madrid
00007  EDICIONES       INGELEK           Madrid
00001  Fernando        Lopez             Madrid
00006  Pedro Miguel    Lozano            Tarancon
00010  Pilar           Manzanera         Madrid
00005  Rafael de la    Ossa              Madrid
00008  Carlos de la    Ossa              Madrid
00009  Amelia          Polo              Barcelona
```

BÚSQUEDA SOBRE FICHEROS INDEXADOS

Ya hemos señalado que el objetivo principal cuando se decide la creación de ficheros indexados, es optimizar al máximo los procesos

de clasificación y búsqueda de información sobre estos. Por este motivo, aunque también utilizables, mandatos como SORT o LOCATE deben desecharse en virtud de las prestaciones y menores tiempos de respuesta que FIND, de uso exclusivo en ficheros relacionados por claves, proporciona. Una orden de búsqueda sobre un indexado ofrece el siguiente aspecto:

FIND <expresión>

Como viene siendo norma habitual en la mayoría de las órdenes de dBASE II, «expresión» puede adquirir diversas formas, como cadenas de caracteres entre comillas o apóstrofes, o sin ellos (sólo son estrictamente necesarios cuando la clave tiene blancos a la izquierda), variables de memoria, o el resultado de la evaluación de la propia expresión.

```
. use telef index inombre
. find carlos
No find
. find Carlos
. display
00008  Carlos de la     Ossa              Villacañas        Pza. de la Gramatica Lati
na 1   28027 Madrid                      Madrid              91  7422933 .T.
```

ACTUALIZACIÓN DE UN FICHERO INDEXADO

Para la actualización de ficheros indexados se emplea el mandato UPDATE. Este nos permite reemplazar determinados registros por otros procedentes de una segunda base de datos. La sintaxis completa de este comando es la siguiente:

UPDATE FROM <n-fich> ON <clave> [ADD <lista de campos>] [RANDOM] [REPLACE [<lista de campos>]] [<campo> WITH <campo origen>]

donde «clave» es el campo por cuya coincidencia se efectúa la actualización del registro, y la cláusula RANDOM es la asumida por defecto. Esto supone que simplemente se realiza la sustitución de la línea en cuestión cuando existe una coincidencia entre el campo clave buscado en el archivo a partir del cual se ejecuta la actualización. Esta

identidad entre el campo clave y el leído para la modificación ha de ser completa, en todos y cada uno de sus caracteres, así como en su longitud.

Las cláusulas ADD y REPLACE permiten además añadir o reemplazar, respectivamente, los campos especificados en «lista de campos» del fichero señalado tras FROM al de origen.

PRECAUCIONES TRABAJANDO CON INDEXADOS

Cuando se utilizan ficheros indexados, algunas órdenes de las comentadas en capítulos anteriores pueden conducir a resultados diferentes de los esperados, puesto que la base de datos está organizada según un sistema distinto del secuencial. Entre ellas cabe destacar:

APPEND: Conforme se añaden nuevos registros a los existentes en la base de datos, los índices se van actualizando respecto a aquellas claves que fueron fijadas en el último mandato USE. Los demás índices que tuvieran relación con el fichero en cuestión, deberán ser creados de nuevo.

COPY: Efectúa el duplicado del fichero original y no así de los índices que pudiera tener asociados. Por tanto, cuando trabajemos sobre la copia, si se pretende disponer de los indexados, será preciso crearlos.

DELETE: Al igual que sucede con el mandato APPEND, solamente los índices especificados en USE, sufrirán la adecuada actualización; los restantes deberán desecharse y volver a ser definidos.

INSERT: Su efecto sobre los índices es similar a lo sucedido con APPEND, actualizándose los que estuvieran abiertos en la orden USE, conforme insertamos nuevos registros, pero siendo imprescindible volver a crear los restantes.

MODIFY STRUCTURE: Como se ha señalado, esta orden debe manejarse con precaución puesto que los datos del fichero actual se pierden. Por tanto, cuando se modifique la estructura de un fichero de base de datos, nunca está de más volver a crear los índices correspondientes.

UPDATE: Puesto que este mandato actualiza los datos del fichero en curso, mediante los presentes en un segundo que se utiliza para modificarlo, los índices activos, quedarán sometidos a dichas alteraciones, y como ya es norma habitual, los restantes será preciso adaptarlos a la nueva situación.

No obstante, y para evitar molestias innecesarias cuando quera-

mos volver a crear los índices, dBASE II incorpora entre sus mandatos la orden REINDEX, cuyo efecto es actualizar todas las claves una vez que se hayan realizado modificaciones sobre el fichero de datos.

Cuadro De Mandatos

— INDEX ON
— USE... INDEX
— FIND
— UPDATE
— REINDEX

PROGRAMACIÓN EN DBASE II

uando a fuerza de paciencia y práctica se llegan a manejar con soltura los mandatos en modo directo de dBASE II, este sistema resulta especialmente ineficaz y sobre todo lento. Alguien podría pensar que las posibilidades de dBASE II finalizan aquí; sin embargo, una de las cualidades que distinguen a este programa, son las facilidades que incorpora para diseñar ficheros de mandatos: ¡es posible programarla!

Además, entre las instrucciones disponibles, se encuentran aquellas estructuras que permiten optimizar al máximo el diseño modular de los programas, facilitándose por este motivo su construcción, prueba y puesta a punto final.

Las cuatro estructuras principales de las cuales parten la mayoría de los programas son las siguientes:

— Secuencial: Las instrucciones se procesan una tras otra en el orden que el programa las va encontrando.
— Iterativa: Basada en la repetición de una determinada serie de instrucciones contenidas en un bucle DO WHILE-ENDDO.
— Condicional: Permiten bifurcar el control del programa en función del resultado lógico (verdadero o falso) obtenido al eva-

luar una determinada premisa o grupo de éstas. Son instrucciones del tipo IF-ELSE-ENDIF.
— Llamadas: Ceden el control a un nuevo archivo de comandos, que procesa su propia lista de instrucciones.

CREACIÓN DE UN PROGRAMA

Para la construcción de un programa, es decir, para el diseño de una secuencia de órdenes unidas de forma lógica, dBASE II dispone de su propio editor (podría utilizarse alguno exterior, como WORDSTAR o equivalentes), al que se accede a través del mandato MODIFY COMMAND. La sintaxis es la siguiente:

MODIFY COMMAND <nombre de programa>

donde «nombre de programa» es el elegido para almacenar el fichero de órdenes a desarrollar a continuación, que adquirirá la extensión .CMD (CoMmanD).

Si sobre el disco está presente alguno con idéntico nombre, la acción del mandato MODIFY conduce a la edición del mismo, para su posterior modificación. dBASE II, en estos casos, crea otro de idéntico nombre, pero de extensión .BAK, para salvaguardarlo en caso de accidente.

Un vez situados en el editor de programas, las teclas señaladas en el apéndice B de este volumen, se encuentran a nuestra entera disposición. De no haberse alterado las condiciones iniciales de dBASE II, el editor muestra una línea principal en vídeo inverso con el cursor en modo normal, sobre la cual podemos comenzar a desarrollar nuestro algoritmo.

Una línea de programa de dBASE II acepta todo tipo de mandatos, aunque algunos como CREATE o APPEND, necesitan de la intervención del usuario, por encima del contenido del archivo de programa. Por tal motivo, deben ser manejados con cierta precaución cuando son incluidos en un fichero de este tipo.

Las órdenes se escriben sin ajustarse a ningún formato especial, aunque ciertas limitaciones han de tenerse en cuenta, como que cada línea de mandatos debe finalizar con la pulsación de RETURN (INTRO), o que la longitud máxima de línea es de 77 caracteres. Si el último carácter de una línea es el signo punto y coma (;), éste indicará a dBASE II que el mandato no ha finalizado y, por tanto, que continúa en la línea siguiente.

Los programas creados mediante el mandato MODIFY, se ejecutan con la orden DO <nombre del programa>, no siendo necesario especificar la extensión .CMD que todos los ficheros de mandatos asumen por defecto.

Desde un programa puede realizarse una llamada a otro, para que este último tome el control de la aplicación. dBASE II permite a lo sumo, mantener 16 ficheros abiertos a la vez, en otros términos, 16 niveles de subrutina o anidamiento. Por tanto, si tenemos uno activo mediante USE, sólo quince más quedarán disponibles como máximo.

Tengamos en cuenta también, que ciertos mandatos se sirven de algunos ficheros adicionales para llevar a cabo su trabajo. SORT, por ejemplo, precisa de dos, mientras que órdenes como COPY, RESTORE, MODIFY, PACK o REPORT, manejan un fichero adicional. Todas estas circunstancias deberemos tenerlas presentes durante el diseño de la aplicación.

Cuando un fichero de programa encuentra el mandato RETURN, o se llega al fin de fichero (EOF), el archivo de órdenes se cierra, con lo que el espacio liberado queda disponible para cualquier otro programa o mandato que lo reclame.

A continuación, describimos los mandatos más utilizados en el desarrollo de programas sobre dBASE II. A nadie se le ocultará que aun no tratándose de un lenguaje de programación, las facillidades proporcionadas para el diseño de aplicaciones permiten obtener excelentes resultados, cuando su objetivo principial sea la gestión y organización de enormes volúmenes de datos.

INICIALIZACIONES Y COMENTARIOS

En todo programa, por mucho que nos esforcemos en diseñarlo de forma estructurada, a poco que su complejidad aumente, es fácil perderse cuando al cabo del tiempo necesitemos revisarlo para actualizarlo o modificarlo.

En previsión de tal eventualidad, resulta bastante útil incluir en él líneas de comentarios que describan la función de cada una de las rutinas que a continuación se desarrollan. Para ello, se utiliza el mandato NOTE.

Una línea que comience por NOTE permite escribir a continuación y separado de éste como mínimo por un espacio, cualquier tipo de descripción que precisemos. Esta no será mostrada en la pantalla, ni en la impresora, si es que la salida va dirigida hacia allí, cuando el programa sea ejecutado.

Otra opción que dBASE II acepta es sustituir NOTE por un signo asterisco (*), puesto que su acción conduce a resultados equivalentes. Sea cual sea el elegido, no deberán emplearse como líneas de comentario aquellas que contengan mandatos.

REMARK permite escribir un comentario tras él, pero a diferencia de las órdenes anteriores, si será presentado en la pantalla (impresora, si este es el caso). Su utilidad se hace patente cuando en algún momento durante la ejecución de un programa, el usuario debe tomar una decisión, añadiéndose a cada una de las posibles acciones, una breve descripción del resultado al cual conducirá.

La última línea de cualquier programa debería contener un mandato RETURN. Este posibilita que sea accedido desde algún otro programa, a modo de subrutina. Cuando trabajemos con uno solo, al encontrarse la instrucción RETURN se regresa a la pantalla de mandatos de dBASE II.

INSTRUCCIONES CONDICIONALES

Con bastante frecuencia, durante la ejecución de un programa es preciso plantearse si una determinada condición se cumple, y en función del resultado obtenido, procesar grupos de instrucciones diferentes.

La estructura de los mandatos que permiten tomar en dBASE II decisiones sencillas es la siguiente:

```
IF <condición> [.AND. <cond 2> .OR. <cond 3>...]
   [instrucción 1]
   [instrucción 2]
ENDIF
```

Las condiciones pueden contener cualquier expresión lógica de como máximo 254 caracteres, que permitan producir un resultado lógico verdader (.T.) o falso (.F.).

Cuando la condición o conjunto de condiciones especificadas tras IF se cumplen, las instrucciones que le siguen serán ejecutadas, mientras que de no verificarse, el control del programa pasa automáticamente a la línea siguiente al ENDIF.

Si el cumplimiento o no de la condición conduce a dos acciones específicas, pero diferentes, se puede incluir dentro de esta estructura el término ELSE («en caso contrario»), con lo que la estructura de control condicional adquirirá el siguiente aspecto:

```
IF <condiciones>
    [acción 1]
ELSE
    [acción 2]
ENDIF
```

En este caso, el cumplimiento de las condiciones conducirán a la ejecución del grupo de instrucciones que hemos denominado «acción 1». De no verificarse, las procesadas serán las situadas a continuación de ELSE («acción 2»), y siempre, sea cual sea la decisión tomada, una vez completadas las instrucciones de una u otra acción, el programa continuará con las siguientes al ENDIF.

Naturalmente pueden tomarse decisiones de múltiples alternativas, es decir, es posible anidar estructuras IF...ELSE. Un procedimiento de este tipo seguiría la sintaxis que a continuación describimos:

```
IF <condiciones 1>  (1)
    <acción 1>
ELSE
    IF <condiciones 2>  (2)
        <acción 2>
    ELSE
    .....................
    ENDIF (2)
ENDIF (1)
```

En este caso, el no cumplimiento del grupo de condiciones 1, obligaría al programa a comprobar si las condiciones 2 son ciertas, y así sucesivamente, hasta encontrar alguna que se verificara. De no hacerlo, el grupo de mandatos tras el último ELSE sería ejecutado.

Nótese que por cada instrucción IF aparece al final del procedimiento otra ENDIF, norma imprescindible para la correcta sintaxis de la orden.

Por otra parte, dBASE II no tiene en cuenta los caracteres presentes tras un ENDIF, por lo cual puede utilizarse como zona de comentarios para mejorar la claridad del programa.

La forma de diversos entrantes en que hemos presentado la estruc-

tura anterior también mejora considerablemente su legibilidad, puesto que permite relacionar fácilmente cada uno de los grupos IF...ELSE...ENDIF, técnica conocida como indentación.

PROCEDIMIENTOS CASE

El mandato CASE puede considerarse como una particularización de DO, puesto que determina la ejecución de diferentes procedimientos en virtud del cumplimiento o falsedad de las condiciones indicadas en la instrucción.

Además, en algunas aplicaciones en las que sea preciso una toma de múltiples decisiones, resulta más eficaz que las estructuras IF...ELSE...ENDIF, al clarificar la acción a desarrollar en cada uno de los casos. Esta estructura en dBASE II, se construye de la manera siguiente:

```
DO CASE
    CASE <condición>
      <acción 1>
    CASE <condición 2>
      <acción 2>
    .....................
    OTHERWISE
      <acción n>
ENDCASE
```

El número de líneas CASE no está limitado dentro del mandato DO CASE, pero veamos a continuación cómo es evaluada esta expresión.

Una vez que el programa llega a la instrucción DO CASE, examina cada una de las líneas CASE siguientes, hasta encontrar la primera que haga verdadera la condición que lleva asociada. De encontrarla, ejecutará las acciones que le siguen, saltando finalmente, una vez que su proceso haya concluido, hasta la línea ENDCASE y posteriormente a las instrucciones que pudieran seguir. Por tanto, cuando coincidan varias condiciones ciertas, solamente la primera hallada será tenida en cuenta.

Si todas las condiciones resultaran falsas, entonces se procesan las

instrucciones situadas tras las cláusulas OTHERWISE, concluyéndose de nuevo el proceso, llegando el programa a ENDCASE.

La cláusula OTHERWISE es opcional. De no utilizarla, si ninguna de las condiciones presentes en las líneas CASE resultara verdadera, el control del programa saldrá de esta estructura sin ejecutar instrucción alguna.

Otra particularidad a tener en cuenta, es que si se incluyen instrucciones entre DO CASE y la primera línea CASE, NO serán ejecutadas.

INSTRUCCIONES DE ITERACIÓN

El mandato DO WHILE permite al usuario obtener el máximo rendimiento de su ordenador, puesto que es posible repetir un determinado proceso el número de veces que sea preciso. Esta nueva estructura de dBASE II se construye de la forma siguiente:

```
DO WHILE <condiciones>
    <instrucción 1>
    <instrucción 2>
    .....
ENDDO
```

Si una vez evaluadas la condición o condiciones situadas tras el DO WHILE, resultan ser ciertas, las instrucciones almacenadas antes de ENDDO serán procesadas, volviéndose a comprobar a continuación, si la condición sigue manteniéndose verdadera y repitiéndose así el bucle sucesivamente. Cuando se determina la condición como falsa, el programa continúa con las sentencias que siguen al mandato ENDDO.

En el interior de un bucle pueden alojarse estructuras IF...ELSE...ENDIF. En estos casos, un ENDDO no debe aparecer antes del ENDIF.

CUADRO DE MANDATOS

— MODIFY COMMAND
— DO

— NOTE
— *
— REMARK
— IF-ELSE-ENDIF
— DO CASE-OTHERWISE-ENDCASE
— DO WHILE-ENDDO

5.10.1. Listados de gestión de una agenda telefónica programados en dBASE II.

PROGRAMA DE PRESENTACION DE CAMPOS

```
* Programa.: TE-ALG.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
IF *
   @ 1,55 SAY "BORRADO"
ELSE
   @ 1,55 SAY "       "
ENDIF
@  4,11 SAY Nombre
@  5,11 GET Apell:1
@  6,11 GET Apell:2
@  7,11 GET Direcc
@  8,11 GET Cod:post
@  9,11 GET Poblac
@ 10,11 GET Prov
@ 11,11 GET Prefijo
@ 12,11 GET Telef:
@ 13,11 GET Caract:
RETURN
* EOF: TE-ALG.CMD
```

PROGRAMA DE BORRADO

```
* Programa.: TE-BORR.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
ERASE
@ 2, 0 SAY "B O R R A D O    T E L E F"
@ 2,72 SAY DATE()
@ 3, 0 SAY "========================================"
@ 3,40 SAY "========================================"
STORE "NO" TO selec
@ 5,0 SAY "Confirma el borrado? [SI/NO] ";
      GET selec PICTURE "!!"
```

```
READ NOUPDATE
IF selec <> "SI"
   RETURN
ENDIF
@ 6,0 SAY "B:TELEF.OLD sera su fichero de seguridad."
IF FILE( "B:TELEF.OLD" )
   STORE " " TO selec
   @ $+1,0 SAY "Borro la copia antigua? (S/N) ";
           GET selec PICTURE "!"
   READ NOUPDATE
   IF selec <> "S"
      RETURN
   ENDIF
   DELETE FILE B:TELEF.OLD
ENDIF
USE
RENAME B:TELEF.DBF TO B:TELEF.OLD
@ $+1,0 SAY " "
*
USE B:TELEF.OLD
SET TALK ON
SET ECHO ON
COPY TO B:TELEF
USE
USE B:TELEF
* --- Recreacion del fichero de indices
INDEX ON NOMBRE TO B:INOMBRE
USE
SET ECHO OFF
SET TALK OFF
STORE " " TO selec
@ 22,0 SAY borralinea
@ 22,0 SAY "Pulse cualquier tecla...";
       GET selec
READ NOUPDATE
RETURN
* EOF: TE-BORR.CMD
```

PROGRAMA DE BUSQUEDA

```
* Programa.: TE-BUSC.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
STORE " " TO expresion,cadena
* --- seccion BUSQUEDA
DO WHILE expresion = " "
   @ 15,0 SAY 'EJEMPLO: nombre="Carlos"'
   @ 17,0 SAY borralinea
   *
   @ 16,0 SAY "-"
   ACCEPT "Entre expresion para BUSQUEDA" TO expresion
   @ 15,0 SAY borralinea
   STORE TRIM(expresion) TO expresion
   DO CASE
```

```
      CASE expresion = " "
      * --- Salida
         RETURN
      CASE 0 = TEST(&expresion)
      * --- EXPRESION NO VALIDA
         @ 18,0 SAY borralinea
         STORE " " TO selec
         @ 18,0 SAY "EXPRESION NO VALIDA: "+;
                    "Pulse cualquier tecla... ";
                GET selec
         READ NOUPDATE
         @ 18,0 SAY borralinea
         STORE " " TO expresion
      OTHERWISE
      * --- BUSQUEDA del registro
         * --- Cierra el fichero indice para una rapida BUSQUEDA
         SET INDEX TO
         LOCATE FOR &expresion
         IF .NOT. EOF
            * --- Encuentra un registro identico
            STORE # TO ultimoreg
            * --- Reapertura del fichero indice
            SET INDEX TO B:INOMBRE
            GOTO ultimoreg
         ELSE
            * --- Reapertura del fichero indice y
            * --- puesta a verdadero (.T.) de la marca EOF
            SET INDEX TO B:INOMBRE
            GO BOTTOM
            SKIP
         ENDIF
   ENDCASE
ENDDO
IF posopcion = "S"
   * --- Retorno al programa de llamada si solo BUSQUEDA
   RETURN
ENDIF
*
* --- seccion MUESTRA
STORE " " TO cadena
DO WHILE cadena = " "
   @ 15,0 SAY "EJEMPLO: nombre"
   @ 17,0 SAY "E"
   ACCEPT "Entre cadena para MUESTRA" TO cadena
   @ 15,0 SAY borralinea
   STORE TRIM(cadena) TO cadena
   DO CASE
      CASE cadena = " "
      * --- Salida
         @ 17,0 SAY borralinea
         @ 18,0 SAY borralinea
         RETURN
      CASE 0 = TEST(&cadena)
      * --- EXPRESION NO VALIDA
         @ 18,0 SAY borralinea
         STORE " " TO selec
         @ 18,0 SAY "EXPRESION PARA MUESTRA NO VALIDA: "+;
                    "Pulse cualquier tecla... ";
                GET selec
         READ NOUPDATE
         @ 18,0 SAY borralinea
         STORE " " TO cadena
```

```
   ENDCASE
ENDDO
* --- Ahora, MUESTRA la expresion
STORE F TO es:eof,es:alg
DO WHILE .NOT. es:eof
   * --- El siguiente grupo de comandos de dBASE II sirven
   * --- para borrar el final de la pantalla. Si tiene un
   * --- PC, puede reemplazarlos por el comando simple
   * --- @ 4,0 ERASE
   *
   @ 4,0 SAY borralinea
   @ 5,0 SAY borralinea
   @ 6,0 SAY borralinea
   @ 7,0 SAY borralinea
   @ 8,0 SAY borralinea
   @ 9,0 SAY borralinea
   @ 10,0 SAY borralinea
   @ 11,0 SAY borralinea
   @ 12,0 SAY borralinea
   @ 13,0 SAY borralinea
   @ 14,0 SAY borralinea
   @ 15,0 SAY borralinea
   @ 16,0 SAY borralinea
   @ 17,0 SAY borralinea
   @ 18,0 SAY borralinea
   @ 19,0 SAY borralinea
   @ 20,0 SAY borralinea
   @ 21,0 SAY borralinea
   @ 22,0 SAY borralinea
   @ 20, 0 SAY "----------------------------------------"
   @ 20,40 SAY "----------------------------------------"
   STORE 4 TO fila
   DO WHILE .NOT. EOF .AND. fila-3 <= 15
      STORE T TO es:alg
      @ fila,0 SAY &cadena
      STORE fila + 1 TO fila
      CONTINUE
   ENDDO
   * --- Se utiliza una variable logica para detectar el
   * --- fin de fichero.  La funcion EOF no se puede emplear
   * --- hasta que este se reinicializa con la instruccion EOF
   STORE EOF TO es:eof
   IF .NOT. es:alg
      * --- No hay registros que coincidan
      @ 4,0 SAY "*** NO HAY REGISTROS COINCIDENTES ***"
   ENDIF
   STORE " " TO selec
   @ 21,0 SAY "Pulse cualquier tecla... ";
          GET selec
   READ NOUPDATE
   @ 21,0 SAY borralinea
ENDDO
* --- El siguiente grupo de comandos de dBASE II sirven
* --- para borrar el final de la pantalla. Si tiene un
* --- PC, puede reemplazarlos por el comando simple
* --- @ 4,0 ERASE
*
@ 4,0 SAY borralinea
@ 5,0 SAY borralinea
@ 6,0 SAY borralinea
@ 7,0 SAY borralinea
@ 8,0 SAY borralinea
@ 9,0 SAY borralinea
```

```
@ 10,0 SAY borralinea
@ 11,0 SAY borralinea
@ 12,0 SAY borralinea
@ 13,0 SAY borralinea
@ 14,0 SAY borralinea
@ 15,0 SAY borralinea
@ 16,0 SAY borralinea
@ 17,0 SAY borralinea
@ 18,0 SAY borralinea
@ 19,0 SAY borralinea
@ 20,0 SAY borralinea
@ 21,0 SAY borralinea
@ 22,0 SAY borralinea

GO TOP
RETURN

* EOF: TE-BUSC.CMD
```

PROGRAMA EDITOR DE CAMPOS

```
* Programa.: TE-CUAD.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5
*
@  2, 0 SAY "----------------------------------------"
@  2,40 SAY "----------------------------------------"
@  4, 0 SAY "Nombre....:"
@  5, 0 SAY "Apell:1...:"
@  6, 0 SAY "Apell:2...:"
@  7, 0 SAY "Direcc....:"
@  8, 0 SAY "Cod:post..:"
@  9, 0 SAY "Poblac....:"
@ 10, 0 SAY "Prov......:"
@ 11, 0 SAY "Prefijo...:"
@ 12, 0 SAY "Telef:....:"
@ 13, 0 SAY "Caract:...:"
@ 16, 0 SAY "----------------------------------------"
@ 16,40 SAY "----------------------------------------"
RETURN
* EOF: TE-CUAD.CMD
```

PROGRAMA DE EDICION DE CAMPOS

```
* Programa.: TE-EDIC.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
DO WHILE T
   STORE " " TO edicopcion
   @ 17,0 SAY "COMANDO: (E)dita, (B)orra, (R)ecupera, "+;
```

```
            "(C)ontinua, (S)itua ";
         GET edicopcion PICTURE "!"
   READ NOUPDATE
   CLEAR GETS
   @ 17,0 SAY borralinea
   DO CASE
      CASE edicopcion = " "
      * --- Salida
         RETURN
      CASE edicopcion = "S"
      * --- (S)itua
         STORE "X" TO posopcion
         DO WHILE posopcion <> " "
            DO TE-pos
         ENDDO
      CASE edicopcion = "B"
      * --- (B)orra
         DELETE
         @ 1,55 SAY "BORRADO"
      CASE edicopcion = "R"
      * --- (R)ecupera
         RECALL
         @ 1,55 SAY "        "
      CASE edicopcion = "E"
      * --- (E)dita
         @ 17,0 SAY "Pulse <control-W> para salir"
         IF # <> 0
            DO TE-alg
            READ
         ENDIF
      CASE edicopcion = "C"
      * --- (C)ontinua hasta el proximo registro
         STORE # TO ultimoreg
         CONTINUE
         * --- Comprueba el FIN DE FICHERO (EOF)
         IF .NOT. EOF
            DO TE-leer
            CLEAR GETS
         ELSE
            * --- EOF encontrado
            GOTO ultimoreg
            @ 17,0 SAY borralinea
            @ 17,0 SAY "FIN DE FICHERO encontrado"
            STORE " " TO selec
            @ 18,0 SAY "Pulse cualquier tecla...";
                   GET selec
            READ NOUPDATE
            @ 17,0 SAY borralinea
            @ 18,0 SAY borralinea
         ENDIF
   ENDCASE
ENDDO
* EOF: TE-EDIC.CMD
```

PROGRAMA DE PRESENTACION DE CAMPOS

```
* Programa.: TE-ALG.CMD
* Autor....: Carlos de la Ossa Villacañas
```

```
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
IF *
   @ 1,55 SAY "BORRADO"
ELSE
   @ 1,55 SAY "       "
ENDIF
@  4,11 SAY Nombre
@  5,11 GET Apell:1
@  6,11 GET Apell:2
@  7,11 GET Direcc
@  8,11 GET Cod:post
@  9,11 GET Poblac
@ 10,11 GET Prov
@ 11,11 GET Prefijo
@ 12,11 GET Telef:
@ 13,11 GET Caract:
RETURN
* EOF: TE-ALG.CMD
```

PROGRAMA PRINCIPAL

```
* Programa : TE-MENU.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5
* Variables: selec, selecnum, edicopcion, posopcion,
*            error, busctec, expresion, cadena, es:eof
*            borralinea, intopcion, es:alg, ultimoreg
*

USE B:TELEF
INDEX ON nombre TO B:INOMBRE

SET TALK OFF
SET BELL OFF
SET COLON OFF

* --- Utiliza blancos para limpiar hasta el final de linea
STORE $(STR(0,81),1,80) TO borralinea

DO WHILE T

ERASE
@  1, 0 SAY "========================================="
@  1,40 SAY "========================================="
@  2, 0 SAY "¦¦"
@  2,19 SAY "T E L E F    M E N U    P R I N C I P A L"
@  2,78 SAY "¦¦"
@  3, 0 SAY "========================================="
@  3,40 SAY "========================================="
@  4, 0 SAY "¦¦"
@  4,78 SAY "¦¦"
@  5, 0 SAY "¦¦"
@  5,78 SAY "¦¦"
@  6, 0 SAY "¦¦"
@  6,78 SAY "¦¦"
@  7, 0 SAY "¦¦"
@  7,78 SAY "¦¦"
@  8, 0 SAY "¦¦"
```

```
@  8,78 SAY "¦¦"
@  9, 0 SAY "¦¦"
@  9,78 SAY "¦¦"
@ 10, 0 SAY "¦¦"
@ 10,78 SAY "¦¦"
@ 11, 0 SAY "¦¦"
@ 11,78 SAY "¦¦"
@ 12, 0 SAY "========================================"
@ 12,40 SAY "========================================"
@  5,31 SAY " 0. Salida"
@  6,31 SAY " 1. Consulta"
@  7,31 SAY " 2. Introduccion"
@  8,31 SAY " 3. Edicion"
@  9,31 SAY " 4. Borrado"
STORE 5 TO selecnum
DO WHILE selecnum < 0 .OR. selecnum > 4
   STORE " " TO selec
   @ 12,33 SAY " selec : : "
   @ 12,42 GET selec PICTURE "#"
   READ
   STORE VAL(selec) TO selecnum
ENDDO

   DO CASE
      CASE selecnum= 0
         CLEAR
         SET COLON ON
         SET BELL ON
         SET TALK ON
         RETURN
      CASE selecnum= 1
      *  DO consulta
         USE B:TELEF INDEX B:INOMBRE
         ERASE
         @ 1, 0 SAY "C O N S U L T A    T E L E F"
         @ 1,72 SAY DATE()
         DO TE-cuad
         IF # = 0
            * --- El fichero de datos esta vacio
            STORE " " TO selec
            @ 17,0 SAY "FICHERO VACIO"
            @ 18,0 SAY "Pulse cualquier tecla...";
                   GET selec
            READ NOUPDATE
         ELSE
            * --- El fichero de datos contiene registros
            DO TE-leer
            CLEAR GETS
            STORE "X" TO posopcion
            DO WHILE posopcion <> " "
               DO TE-pos
            ENDDO
         ENDIF
         USE
      CASE selecnum= 2
      *  DO intro
         USE B:TELEF 'INDEX B:INOMBRE
         COPY STRUCTURE TO B:TELEF.int
         SELECT SECONDARY
         USE B:TELEF.int
         ERASE
```

```
      @ 1, 0 SAY "I N T R O D U C C I O N    T E L E F"
      @ 1,72 SAY DATE()
      DO TE-cuad
      @ 17,0 SAY "Pulse <control-W> para salir"
      STORE "X" TO intopcion
      DO WHILE intopcion <> " "
         APPEND BLANK
         DO TE-leer
         READ
         * --- NOMBRE no puede estar en blanco
         STORE TRIM( NOMBRE ) TO intopcion
      ENDDO
      DELETE
      USE
      SELECT PRIMARY
      APPEND FROM B:TELEF.int
      USE
   CASE selecnum= 3
   *  DO edic
      USE B:TELEF INDEX B:INOMBRE
      ERASE
      @ 1, 0 SAY "E D I C I O N    T E L E F"
      @ 1,72 SAY DATE()
      DO TE-cuad
      IF # = 0
         * --- El fichero de datos esta vacio
         STORE " " TO selec
         @ 17,0 SAY "FICHERO VACIO"
         @ 18,0 SAY "Pulse cualquier tecla...";
                GET selec
         READ NOUPDATE
      ELSE
         * --- El fichero de datos contiene registros
         DO TE-leer
         CLEAR GETS
         DO TE-edic
      ENDIF
      USE
   CASE selecnum= 4
   *  DO borr
      DO TE-borr
ENDCASE

ENDDO T
* EOF: TE-MENU.CMD
```

PROGRAMA DE POSICIONAMIENTO

```
* Programa.: TE-POS.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
*
STORE " " TO posopcion
@ 17,0 SAY borralinea
@ 17,0 SAY "COMANDO: (M)uestra, (B)usca, "+;
           "(S)itua, (C)ontinua, (A)vanza ";
       GET posopcion PICTURE "!"
```

```
READ NOUPDATE
CLEAR GETS
@ 17,0 SAY borralinea
IF .NOT. (posopcion $ "MBSCA")
   RETURN
ENDIF
IF posopcion = "B"
   * --- (B)usca
   @ 16,0 SAY "-"
   ACCEPT "Entre NOMBRE" TO buscclv
   @ 17,0 SAY borralinea
   STORE TRIM(buscclv) TO buscclv
   IF buscclv = " "
      RETURN
   ENDIF
   STORE # TO ultimoreg
   FIND &buscclv
   IF (# <> 0)
      DO TE-leer
      CLEAR GETS
   ELSE
      * --- NO ENCONTRADO
      GOTO ultimoreg
      @ 17,0 SAY borralinea
      @ 17,0 SAY '"'+buscclv+'"'+" no en indice"
      STORE " " TO selec
      @ 18,0 SAY "Pulse cualquier tecla...";
             GET selec
      READ NOUPDATE
      @ 17,0 SAY borralinea
      @ 18,0 SAY borralinea
   ENDIF
ELSE
   * --- (A)vanza, (C)ontinua, (M)uestra, o (S)itua
   STORE # TO ultimoreg
   DO CASE
      CASE posopcion = "A"
      * --- (A)vanza
         SKIP
      CASE posopcion = "C"
      * --- (C)ontinua
         CONTINUE
      OTHERWISE
      * --- (M)uestra o (S)itua
         DO TE-busc
         IF expresion = " "
            RETURN
         ENDIF
         IF posopcion = "M"
            IF cadena = " "
               RETURN
            ENDIF
            DO TE-cuad
         ENDIF
   ENDCASE
   * --- Comprobacion de FIN DE FICHERO (EOF)
   IF .NOT. EOF
      DO TE-leer
      CLEAR GETS
   ELSE
      * --- EOF encontrado
      GOTO ultimoreg
```

```
      @ 17,0 SAY borralinea
      @ 17,0 SAY "FIN DE FICHERO encontrado"
      STORE " " TO selec
      @ 18,0 SAY "Pulse cualquier tecla...";
             GET selec
      READ NOUPDATE
      @ 17,0 SAY borralinea
      @ 18,0 SAY borralinea
   ENDIF
ENDIF
RETURN
* EOF: TE-POS.CMD
```

DISEÑO DE PANTALLAS E IMPRESOS

En los capítulos anteriores nos hemos ocupado de analizar el comportamiento de dBASE II desde el punto de vista de la gestión (creación, actualización, modificación y organización) de los ficheros presentes en el disco de trabajo, sin preocuparnos expresamente del aspecto de los resultados finales, ni de las tomas de datos que todo programa deberá mantener durante su conversación con el usuario.

Si en casi todas las vertientes anteriormente consideradas, existían mandatos especialmente flexibles para facilitar la gestión de los ficheros, cuando se trata del diseño de las salidas de información, ya sea hacia la pantalla o la impresora, dBASE II pone a nuestro alcance una serie de procedimientos directamente destinados a conseguir una presentación adecuada.

CAPTURA DE DATOS

La gestión interactiva de la base de datos, es decir, el diálogo que el usuario debe mantener con el ordenador, ha de estar definido de forma clara para evitar errores de interpretación, que puedan provocar equivocaciones durante la ejecución de los programas.

Existen varios mandatos en dBASE II destinados a aceptar las entradas que el operario realizará mientras trabaja con un programa determinado. El primero de ellos, INPUT, se encarga de almacenar en la variable que le sigue la información escrita mediante el teclado. Una instrucción INPUT sigue el esquema siguiente:

INPUT [<«comentario»>] TO <variable>

El tipo que adquirirá la variable utilizada se determina automáticamente, en función del primer carácter introducido, es decir, si éste es un apóstrofe (') o unas comillas ("), dBASE II asumirá que se trata de una serie alfanumérica; de ser un dígito numérico, será numérica la definida, mientras que si este carácter es T, S, F o N, se asumirá el distintivo L (tipo lógico).

Es importante precisar que cuando se efectúa la entrada de una cadena de caracteres, éstos deberán estar limitados por las comillas o apóstrofes correspondientes. Si por error, tras una serie de dígitos numéricos se escribe alguno alfabético, dBASE II despreciará lo tecleado desde esta posición en adelante. En todo caso, INPUT acepta la entrada, y la almacena en una variable del tipo definido en el primer carácter, y los restantes son asimilados, siempre y cuando sean coherentes con éste.

Por el propio funcionamiento del mandato, su utilización debe limitarse a la captura de datos numéricos o lógicos, mientras que para aceptar cadenas de cualesquiera caracteres resulta más apropiado ACCEPT.

La sintaxis de una orden que incluya este último mandato es idéntica a la seguida por INPUT

ACCEPT [<«comentario»>] TO <variable>

aunque, sea cual sea la naturaleza de los caracteres tecleados, la variable utilizada, siempre adquirirá el tipo C.

Como norma común a cualquier entrada por el teclado que se realice manejando alguno de estos dos comandos, los caracteres escritos deberán finalizarse pulsando las teclas RETURN o INTRO, proceso conocido como validación.

Cuando solamente se trate de capturar un único carácter, el mandato adecuado es WAIT. Seguido de la cláusula TO y una variable de memoria, provoca que la tecla pulsada sea automáticamente recogida, sin necesidad de pulsar RETURN. La variable utilizada adquirirá el tipo C en todos los casos.

Si la opción TO no se emplea, su ejecución provoca la detención incondicional del programa, hasta que una tecla cualquiera se pulsa, presentándose el mensaje *Waiting* (esperando).

Los Formatos De Pantalla

La pantalla de los ordenadores Amstrad, trabajando con dBASE II, se divide en 24 filas de 80 columnas en el caso de los CPC, o en 31 de 90 caracteres, en los PCW. A cada uno de los formatos anteriores, deberemos ajustar el diseño de los comentarios y salidas que produzcan tanto los datos, como la ejecución de las instrucciones procedentes de los programas que los gestionan.

Para la confección de formatos de pantalla, son suficientes un operador (@) y un mandato (SAY). Su sintaxis completa responde al esquema siguiente:

@ <fila,col.> [SAY <expresión> [USING <plant 1>]] [GET <var> [PICTURE <plant 2>]]

Analicémosla detenidamente para comprobar que su temible aspecto no lo es en realidad tanto. Los parámetros «fila» y «col.» son las coordenadas de una posición de la pantalla, teniendo en cuenta que «fila» debe variar de 0 a 23 (0-30, en los PCW) y «col.» ha de estar comprendida entre 0 y 79 (0-89, en los PCW). El origen (0,0), está situado en la esquina superior izquierda de la pantalla. En la localización así definida, se escribirá el comentario («expresión») que sigue a SAY.

Las coordenadas no tienen por qué ser valores numéricos constantes; también pueden venir dadas por la evaluación de una determinada expresión que conduzca a un valor de este tipo. Es más, se puede utilizar direccionamiento relativo, es decir, definir la posición incrementando a la última una cantidad específica. Para ello, emplearemos el símbolo dólar ($) como referencia de la posición del cursor de escritura.

Por ejemplo, si escribimos @ 10,20 SAY «AMSTRAD», este comentario será impreso en las coordenadas absolutas señaladas, mientras que si a continuación ejecutamos la instrucción @ $,$+4 SAY «ORDENADOR», esta última palabra aparecerá en la fila 10, columna 24.

El mandato SET FORMAT TO SCREEN designa que la salida

sea dirigida hacia la pantalla, situación normal cuando comenzamos a trabajar con dBASE II. Como en seguida comprobaremos, fácilmente puede ser desviada hacia la impresora.

Cuando el monitor es destino de la información, las instrucciones @ pueden ejecutarse en cualquier orden, es decir, la escritura primero en la fila 10, y posteriormente, en la 7, es perfectamente válida. El mismo comentario cabe repetir cuando de las columnas se trate.

PLANTILLAS DE FORMATO

El parámetro opcional USING se utiliza para ajustar la salida a un determinado formato, el cual se define mediante una serie de caracteres de significado especial, que deben encerrarse entre comillas. Se trata de los siguientes:

9 define dígitos numéricos y sus signos (+ ó −) si los hubiera.
determina posiciones numéricas, espacios en blanco y signos.
X define una posición para cualquier tipo de carácter.
$ envía un carácter $ a las posiciones nulas de un campo numérico.
* idéntico que el anterior, pero empleando el signo (*).

CAPTURAS Y PRESENTACIÓN SOBRE PANTALLA COMPLETA

Ya hemos mencionado que las frases utilizadas en SAY pueden dirigirse tanto hacia la pantalla como a la impresora. Sin embargo, si también incluimos la cláusula GET, solamente serán reconocidas cuando se trate del monitor.

GET se encarga de mostrar el contenido actual de un campo de la base de datos o de una variable de memoria. Necesariamente, ésta habrá sido definida con anterioridad, por ejemplo, manejando STORE. El valor puede posteriormente modificarse, si tras la instrucción que contiene a GET añadimos la cláusula READ.

Al igual que SAY se apoyaba en USING para definir un formato determinado, GET puede manejarse en combinación con PICTURE. La cadena de caracteres que ahora seleccionemos, estipula el tipo de los datos que serán aceptados. Se plantean los siguientes casos:

ó 9 autorizan la entrada de cifras numéricas con su signo y espacio en blanco si lo hubiera.

X acepta la entrada de cualquier tipo de carácter.

A sólo permite la entrada de caracteres alfanuméricos.

$ ó * recoge los datos tal y como los encuentra, sin seleccionarlos.

! convierte el carácter siguiente a este signo en su correspondiente mayúscula.

Cuando se desarrolla un programa, pueden encontrarse activos simultáneamente un máximo de 64 GETs. En este sentido, para liberar espacio de memoria, se manejan los mandatos ERASE o CLEAR GETS.

Si el tipo de variable o del campo es lógico, para su validación solamente se podrán utilizar las letras T, Y, F o N.

Como hemos mencionado, si además incluimos la cláusula READ, GET se convierte en un potente editor de variables o campos en cualquier lugar de la pantalla que sea designado. Los cambios que entonces se efectúen, serán automáticamente ejecutados, tanto en la base de datos en uso, como en la variable de memoria afectada.

De esta manera, conseguimos sustituir el mandato EDIT cuando trabajemos en modo interactivo durante la ejecución de un programa. Además, si estamos manejando indexados, tras cada actualización, dBASE II comprueba si se ha alterado alguna clave para proceder a actualizarla también. En ocasiones, sobre todo al trabajar con muchos índices y siendo la base de datos de considerable tamaño, este proceso puede tomar bastante tiempo.

Si de antemano conocemos que la acción de READ no provocará cambio alguno en las claves, se debe incluir tras este mandato la cláusula NOUPDATE, con lo que el proceso de actualización de los índices será suprimido.

La Salida Impresa

La orden @ admite que la información sea dirigida tanto hacia la pantalla como a la impresora. Para desviarla hacia este periférico, se debe ejecutar la instrucción SET FORMAT TO PRINT.

Si este es el caso, las coordenadas tanto de fila como de columna aumentan su rango de acción para situarse en cualquier valor entre 0 y 254. Ahora, el origen (0,0) corresponde con la esquina superior izquierda del papel.

Naturalmente, han de tomarse ciertas precauciones, y los mandatos @ deben emitirse respetando escrupulosamente el orden de izquierda a derecha y de arriba a abajo. Por ejemplo, todos los comentarios que afecten a la línea 3 han de emitirse antes que los de la 4. De no hacerlo así, nos expondremos, además de no coincidir lo previsto con lo resultante, a desaprovechar injustificadamente el papel.

El mandato SET FORMAT aporta una tercera posibilidad:

SET FORMAT <n-fich de formato>

es decir, también dBASE II admite que pudiéramos tener algún formato previamente definido, sin necesidad de manejar las instrucciones @.

Su diseño parte del mandato REPORT, el cual crea ficheros de extensión .FMT, que trataremos en detalle en el siguiente epígrafe.

GENERACIÓN DE INFORMES

El mandato REPORT se emplea para diseñar formatos de informes, ya sea por pantalla o impresora, para acomodar los datos del archivo en curso a una determinada distribución. La sintaxis completa de esta orden es:

REPORT [FORM <n-fich formato>] [<opción>] [TO PRINT] [PLAIN] [FOR <expresión> / WHILE <expresión>]

Gracias a la cláusula FORM, podemos utilizar un formato ya creado mediante la misma orden REPORT, esta vez sin la mencionada cláusula adicional. Así pues, cuando se emplea el mandato REPORT, se pide al usuario el nombre con que almacenar el formato a crear, mientras que empleando la cláusula FORM, se reclama el «n-fich formato» para que marque la distribución seguida en la salida impresa del informe.

TO PRINT, desvía la salida hacia la impresora, mientras que «opción» o FOR / WHILE tienen el cometido habitual. Por otra parte, PLAIN se utiliza para la compatibilización de la salida de datos por REPORT con el sistema seguido normalmente en los procesadores de texto convencionales.

Al ejecutar la orden REPORT, se solicitan los siguientes datos para confeccionar el formato del informe:

Enter report form name: Nombre del fichero de formato.

Enter options, mleft margin, llines/page, wpage width: Indique opciones, mmargen izquierdo, llíneas/página, wanchura de página.

Page heading? (y/n): ¿Cabecera de página? (s/n).

En caso de respuesta afirmativa (y), se presenta el mensaje *Enter page heading* (escriba la cabecera de página).

Double space report? (y/n): ¿Informe a doble espacio?

Are totals required? (y/n): ¿Se necesitan totales?

Subtotals in report? (y/n): ¿Subtotales en el informe?

```
. use telef
. report
Enter report form name: agenda
Enter options, m=left margin, l=lines/page, w=page widthm=8,l=60,w=64
Page heading? (y/n)y
Enter page heading: LISTIN TELEFONICO
Double space report? (y/n)n
Are totals required? (y/n)n
Col   Width, Contents
001     15,nombre
Enter heading: NOMBRE
002     15,apell:1
Enter heading: 1er. APELLIDO
003     15,apell:2
Enter heading: 2o. APELLIDO
004     3,prefijo
Enter heading: PRF
005     7,telef:
Enter heading: TELEF.
006
```

```
Page no. 00001
        13/10/86

                              LISTIN TELEFONICO

          NOMBRE          1er. APELLIDO      2o. APELLIDO    PRF   TELEF.

        Fernando          Lopez              Martinez         91 4488425
        DATA-3            INFORMATICA S.A                     91 4021099
        Juan              Cuesta             Nuin             91 2742142
        MICROBYTE                                             91 4425433
        Rafael de la      Ossa               Coloma           91 7243392
        Pedro Miguel      Lozano             Castellote      966  112039
        EDICIONES         INGELEK                             91 4579424
        Carlos de la      Ossa               Villacañas       91 7422933
        Amelia            Polo               Garcia           92 3365422
        Pilar             Manzanera          Amaro            91 7421590
```

Si respondemos afirmativamente, se preguntará a continuación *Enter subtotals field:* (indique el campo de subtotales).

Summary reports only? (y/n): ¿Desea sólo un informe resumido?

Eject page after subtotals? (y/n): ¿Salto de página tras los subtotales?

Enter subtotal heading: Escriba la cabecera de subtotal.

Efectuándose a continuación la entrada de la columna, la anchura y el campo, así como la cabecera que se le asigna.

CUADRO DE MANDATOS

— INPUT
— ACCEPT
— WAIT
— @ SAY
— GET
— READ
— USING
— PICTURE
— ERASE
— CLEAR GETS
— SET FORMAT TO SCREEN
— SET FORMAT TO PRINT
— SET FORMAT
— REPORT
— REPORT FORM

5.11.1. *Listados generadores de informes para agenda, etiquetas y menús.*

```
PROGRAMA DE MENUS

* Programa.: MENU.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
* Reservado: selec, selecnum
*
SET TALK OFF
SET BELL OFF
SET COLON OFF

DO WHILE T

ERASE
@  1, 0 SAY "========================================"
@  1,40 SAY "========================================"
@  2, 0 SAY "¦¦"
@  2,26 SAY "M E N U    P R I N C I P A L"
@  2,78 SAY "¦¦"
@  3, 0 SAY "========================================"
@  3,40 SAY "========================================"
@  4, 0 SAY "¦¦"
@  4,78 SAY "¦¦"
@  5, 0 SAY "¦¦"
@  5,78 SAY "¦¦"
@  6, 0 SAY "¦¦"
@  6,78 SAY "¦¦"
@  7, 0 SAY "¦¦"
@  7,78 SAY "¦¦"
@  8, 0 SAY "¦¦"
@  8,78 SAY "¦¦"
@  9, 0 SAY "¦¦"
@  9,78 SAY "¦¦"
@ 10, 0 SAY "¦¦"
@ 10,78 SAY "¦¦"
@ 11, 0 SAY "¦¦"
@ 11,78 SAY "¦¦"
@ 12, 0 SAY "¦¦"
@ 12,78 SAY "¦¦"
@ 13, 0 SAY "========================================"
@ 13,40 SAY "========================================"
@  5,31 SAY " 0. SALIDA"
@  6,31 SAY " 1. INTRODUCCION"
@  7,31 SAY " 2. MODIFICACION"
@  8,31 SAY " 3. CONSULTA"
@  9,31 SAY " 4. BAJA"
@ 10,31 SAY " 5. LISTADO"
STORE  6 TO selecnum
DO WHILE selecnum < 0 .OR. selecnum > 5
   STORE " " TO selec
   @ 13,33 SAY " selec : : "
   @ 13,42 GET selec PICTURE "#"
   READ
   STORE VAL(selec) TO selecnum
ENDDO

DO CASE
   CASE selecnum= 0
      SET COLON ON
      SET BELL ON
      SET TALK ON
      CLEAR
```

```
      RETURN
   CASE selecnum= 1
   * DO INTRODUCCION
   CASE selecnum= 2
   * DO MODIFICACION
   CASE selecnum= 3
   * DO CONSULTA
   CASE selecnum= 4
   * DO BAJA
   CASE selecnum= 5
   * DO LISTADO
ENDCASE

ENDDO T
* EOF: MENU.CMD
```

PROGRAMA DE ETIQUETAS

```
* Programa.: ETIQUETA.CMD
* Autor....: Carlos de la Ossa Villacañas
* Fecha....: 15/10/86
* Copyright: Gran Biblioteca Amstrad. Numero 5.
* Reservado: selec, condicion, extra
*
SET TALK OFF
SET BELL OFF
STORE " " TO selec
USE B:TELEF
ERASE
@ 2, 0 SAY "B : T E L E F    E T I Q U E T A S"
@ 2,72 SAY DATE()
@ 3, 0 SAY "========================================"
@ 3,40 SAY "========================================"
STORE " " TO selec
@ 5,0 SAY "Salida por pantalla o impresora? [P/I] ";
      GET selec PICTURE "!"
READ
DO CASE
   CASE selec = "P"
      ERASE
   CASE selec = "I"
      SET CONSOLE OFF
      SET PRINT ON
   OTHERWISE
      ERASE
      SET BELL ON
      SET TALK ON
      RETURN
ENDCASE
* --- Entre expresion de FOR para las etiquetas, tal como,
* --- STORE "nombre='Carlos'" TO condicion
STORE " " TO condicion
DO WHILE .NOT. EOF
   IF condicion <> " "
```

```
      IF .NOT. (&condicion)
         SKIP
         LOOP
      ENDIF
   ENDIF
   STORE 0 TO extra
   ? trim(nombre)+' '+trim(apell1:1)+' '+trim(apell:2)
   ? direcc
   ? trim(poblac)+' ('+trim(prov)+')'
   ?
   ?
   DO WHILE extra > 0
      ?
      STORE extra - 1 TO extra
   ENDDO
   SKIP
ENDDO
*
SET PRINT OFF
SET CONSOLE ON
?
? "ESTO ES TODO..."
CLEAR
SET TALK ON
SET BELL ON
RETURN
* EOF: ETIQUETA.CMD
```

PROGRAMA DE AGENDA TELEFONICA

```
* Programa .: AGENDA.CMD
* Autor ....: Carlos de la Ossa Villacañas
* Fecha ....: 15/10/86
* Copyright: Gran Biblioteca Amstrad  Numero 5
* Reservado: numpag, lin, cabpag, cab:col, condicion,
*            ultimoreg
*
SET TALK OFF
SET BELL OFF
SET MARGIN TO 8
STORE 1 TO numpag
STORE 254 TO lin
STORE "AGENDA TELEFONICA" TO cabpag
STORE (64-LEN(cabpag))/2 TO cab:col
*
* --- Abre la base de datos y escribe el informe
USE B:TELEF
ERASE
@ 2, 0 SAY cabpag
@ 2,72 SAY DATE()
@ 3, 0 SAY "========================================"
@ 3,40 SAY "========================================"
STORE " " TO selec
```

```
@ 5,0 SAY "Salida por pantalla o impresora? [P/I] ";
      GET selec PICTURE "!"
READ
DO CASE
   CASE selec = "P"
      ERASE
      STORE 22 TO lonpag
   CASE selec = "I"
      SET FORMAT TO PRINT
      STORE 54 TO lonpag
   OTHERWISE
      ERASE
      SET BELL ON
      SET TALK ON
      RETURN
ENDCASE
* --- Entra FOR <expresion> para el informe, por ejemplo,
* --- STORE "nombre='Carlos'" TO condicion
STORE " " TO condicion
DO WHILE .NOT. EOF
   IF lin > lonpag
      IF selec = "P"
         ERASE
      ELSE
         EJECT
      ENDIF
      @ 0,0 SAY "PAG. NO."
      @ 0,9 SAY STR(numpag,3)
      @ 2,cab:col SAY cabpag
      *
      * --- Genera cabeceras de columnas
      @ 4,  0 SAY "NOMBRE"
      @ 4, 18 SAY "1er. APELLIDO"
      @ 4, 36 SAY "2o. APELLIDO"
      @ 4, 54 SAY "PRF"
      @ 4, 60 SAY "TELEF."
      @ 5,  0 SAY "================"
      @ 5, 18 SAY "================"
      @ 5, 36 SAY "================"
      @ 5, 54 SAY "==="
      @ 5, 60 SAY "========"
      STORE  numpag+1 TO numpag
      STORE 7 TO lin
   ENDIF
   * --- Comprueba si la condicion existe
   IF condicion <> " "
      IF .NOT. (&condicion)
         SKIP
         LOOP
      ENDIF
   ENDIF
   *
   * --- Escribe linea de detalle
   @ lin,  0 SAY $(nombre,1, 15)
   @ lin, 18 SAY $(apell:1,1, 15)
   @ lin, 36 SAY $(apell:2,1, 15)
   @ lin, 54 SAY $(STR(prefijo,3,0),1,  3)
   @ lin, 60 SAY $(STR(telef:,10,2),1,  7)
   STORE lin+1 TO lin
   SKIP
ENDDO
@ lin+1,0 SAY " "
```

```
SET FORMAT TO SCREEN
RELEASE ALL
SET TALK ON
SET BELL ON
RETURN
* EOF: AGENDA.CMD
```

CONSEJOS HARDWARE

BASE II es un excelente programa para la gestión de bases de datos; no obstante, para obtener un rendimiento adecuado del mismo, es necesario disponer de al menos 256 Kbytes de memoria central y de unos sistemas de almacenamiento externo, capaces de absorber los volúmenes de información para los cuales está diseñado el programa.

Dada la diversificación existente entre los diferentes modelos Amstrad, se hacen necesarias ciertas normas de funcionamiento, que permitan obtener el máximo partido posible de dBASE II desde una configuración hardware particular.

MODELOS CPC

Si partimos de un CPC 6128, encontraremos que la máxima capacidad de almacenamiento viene señalada por la propia de la unidad de disco, 178 Kbytes. En este caso, se manejarán dos disketes: uno, que contendrá el programa en sí, y otro destinado a albergar los datos de nuestros ficheros.

Como es evidente, serán muy frecuentes las operaciones de cambio de soporte, dado que dBASE II precisa del acceso al diskete de

programas para la ejecución de determinados mandatos, pues en la memoria central no tiene cabida toda la inmensa gama de posibilidades que pone a nuestra disposición.

El cambio de soporte en el momento adecuado, vendrá gestionado por el sistema, mediante mensajes que señalan la introducción del disco de programas o datos, en cada caso.

Si somos los afortunados poseedores de una segunda unidad de disco, el problema de intercambio frecuente de soportes se verá obviado, introduciendo en la unidad A el diskete de dBASE II y en la B el destinado a los datos.

MODELOS PCW

En su configuración mínima (8256) disponemos en primera instancia de la posibilidad de trabajar tal como indicábamos para el CPC 6128 con una sola unidad de disco, es decir, dejando a cargo del sistema la solicitud del disco apropiado en cada momento. Sin embargo, otra posible alternativa es seleccionar como unidad destino para los datos el disco virtual (SET DEFAULT TO M), soportando en la unidad de diskette el programa. Por otra parte, este último sistema, supone una reducción en el máximo volumen de datos a manejar, dado que la capacidad de disco virtual es de tan solo 112 Kbytes. Por contra, se mejora la operatividad y rapidez del sistema.

Una vez finalizado el trabajo con dBASE II, en caso de haber optado por el manejo de datos en la unidad M, deberemos recordar volcar los ficheros a un diskette, dado que se perderán al desenchufar el ordenador. A tal fin, podemos hacer uso del comando de CP/M PIP (libro 2 de esta misma colección).

Si el modelo del que disponemos es un PCW 8512, en su defecto un 8256 ampliado, se presentan varias alternativas de trabajo. Por una parte, si los datos a gestionar tienen ya un cierto volumen, utilizaremos como destino de los ficheros que los han de albergar la unidad B de 720 Kbytes. En lo referente al soporte del programa, podemos optar entre situar éste en la unidad A, o desviarlo previamente a la carga de dBASE II hacia la M. Obviamente, todas las operaciones que impliquen trabajar con el disco virtual, mejorarán sensiblemente la velocidad general de proceso.

Finalmente, siempre queda la posibilidad de utilizar la unidad M para soportar los programas de dBASE II y emplear las dos unidades de disco (A y B) como destino de los ficheros de datos.

APÉNDICE A

LISTA DE MANDATOS

@ <fila,col> [SAY <expresión> [USING <plantilla1>]] [GET <variable> [PICTURE <plantilla2>]].
Escribe en la pantalla el mensaje especificado en una posición determinada.

ACCEPT ["<comentario>"] TO <variable>.
Entrada de una cadena de caracteres desde el teclado a una variable.

APPEND
Adición de registros al final del fichero en uso.

APPEND BLANK
Adición de un registro en blanco al final del fichero en uso.

APPEND FROM [d:] <n-fich> [FOR <expresión>] [SDF] [DELIMITED [WITH <delimitador>]]

Creación de un fichero, de las características indicadas en el mandato, que es posible utilizar por otros programas distintos a dBASE II.

BROWSE
Edición de los registros en la pantalla a modo ventana, con posibilidad de modificación de sus contenidos.

BROWSE [FIELD <lista de campos>]
Edición del tipo BROWSE de determinados campos especificados en el mandato.

CANCEL
Anulación de la ejecución de un programa y vuelta al dBASE II.

CHANGE [<opción>] FIELD <lista de campos> [FOR <expresión>]
Modificación de caracteres dentro de los campos de registros que cumplan una determinada condición.

CLEAR [GETS]
Reinicialización de dBASE II y liberación del espacio de memoria ocupado por los GET.

CONTINUE
Continuación de una búsqueda tras el mandato LOCATE.

COPY STRUCTURE TO [d:] <n-fich>
Volcado en un nuevo fichero de la estructura de la base de datos en uso.

COPY TO [d:] <n-fich> [<opción>] [FIELD <lista de campos>] [FOR / WHILE <expresión>] [SDF] [DELIMITED [WITH <separador>]]
Volcado de los datos de un fichero en otro, siempre que cumplan determinadas características especificadas en el mandato.

COUNT [<opción>] [FOR <expresión>] [TO <variable>]
Totalización del número de registros que cumplan una determinada condición.

CREATE [d:] [<nombre de fichero>]
Creación de la estructura de una base de datos.

DELETE FILE [d:] <n-fich>
Borrado de una base de datos.

DELETE [<opción>] [FOR / WHILE <expresión>]
Marcado de los registros la condición para ser borrados.

DISPLAY FILES [ON <d:>] [LIKE <n-fich.ext>]
Muestra los ficheros presentes en el disco que cumplan la condición especificada.

DISPLAY MEMORY
Muestra las variables.

DISPLAY STATUS
Muestra cuál es la base de datos en curso y los parámetros de su entorno.

DISPLAY STRUCTURE
Muestra la estructura de la base de datos actual.

DISPLAY [<opción>] [FOR <expresión>] [<lista de campos>] [OFF]
Muestra los registros del fichero en uso que cumplan la condición especificada.

DO <nombre de programa>
Ejecución de un fichero de mandatos.

```
DO CASE
    CASE <condición>
      <acción1>
    CASE <condición>
      <acción2>
    .....
    [OTHERWISE]
      <acción n>
    .....
```

ENDCASE
Estructura de decisión múltiple.

```
DO WHILE <condiciones>
<instrucción1>
<instrucción2>
.....
```

ENDDO
Estructura de bucle en un programa.

EDIT [<n>]
Edición del registro especificado.

EJECT
Salto de página en la impresora.

ERASE
Borrado de la pantalla con ubicación del cursor en la esquina superior izquierda de ésta.

FIND <expresión>
Búsqueda de un registro en un fichero indexado.

GO o GOTO RECORD <n>
Desplazamiento del puntero al registro especificado.

GO o GOTO TOP
Desplazamiento del puntero al primer registro del fichero en uso.

GO o GOTO BOTTOM
Desplazamiento del puntero al último registro del fichero actual.

GO o GOTO <variable>
Desplazamiento del puntero al registro especificado por la variable.

<número>
Desplazamiento del puntero al registro especificado.

HELP <mandato>
Muestra un texto de ayuda referente al mandato.

IF <condición> [.AND. <cond2> .OR. <cond3>...]
[<instrucción1>]
[<instrucción2>]
.....
ELSE
[<instrucción n>]
.....
ENDIF
Evaluación de una decisión simple en un programa.

INDEX ON <clave> TO <n-fich>
Creación de un fichero de claves sobre la base de datos en uso.

INPUT [«<comentario>»] TO <variable>
Entrada de un dato a una variable desde el teclado.

INSERT [BEFORE] [BLANK]
Inserción de un registro en el fichero en uso.

JOIN TO <n-fich> FOR <expresión> [FIELDS <lista de campos>]
Encadenamiento de dos bases de datos en un solo fichero.

LIST FILES [ON d:] [LIKE <n-fich.ext>]
Muestra los nombres de los ficheros de la unidad de disco especificada.

LIST MEMORY
Muestra las variables.

LIST STATUS
Muestra el fichero en uso y los parámetros de su entorno de utilización.

LIST STRUCTURE
Muestra la estructura de la base de datos en curso.

LIST [<opción>] [FOR <expresión>] [<lista de campos>] [OFF]
Muestra los registros que cumplan la condición especificada.

LOCATE [<opción>] [FOR <expresión>]
Búsqueda secuencial del registro que cumpla la condición especificada.

LOOP
Vuelve al comienzo del bucle DO WHILE actual.

MODIFY COMMAND [<nombre de programa>]
Edición o creación de un programa de dBASE II.

MODIFY STRUCTURE
Modificación de la estructura del fichero en uso.

NOTE o * [<comentario>]
Comentario interno de un programa.

PACK
Borrado de los datos marcados para esta operación.

QUIT
Salida de dBASE II y cierre de todos los ficheros en curso.

QUIT [TO <lista de programas .COM o de órdenes CP/M,>]
Salida del dBASE II y ejecución de los programas CP/M mencionados en la lista.

READ
Muestra en la pantalla un fichero de formato .FMT.

RECALL [<opción>] [FOR <expresión>]
Eliminación de la marca de borrado de los registros que cumplan la condición especificada.

REINDEX <nombre de fichero de índice>
Reindexación de los ficheros de claves.

RELEASE [ALL] / [<lista de variables>]
RELEASE ALL LIKE <plantilla>
RELEASE ALL EXCEPT <plantilla>
Borrado de las variables que cumplan la condición especificada.

REMARK [<comentario>]
Muestra los comentarios de un programa.

RENAME [d:] <n-fich actual>] TO [d:] <n-fich nuevo>
Cambio de nombre de un fichero.

REPLACE [<opción>] [campo1 WITH <expresión1>] [,campo2 WITH <expresión2>...] [FOR <expresión>]
Sustitución de datos de fichero en uso que cumplan la condición especificada.

REPORT [FORM <n-fich formato>] [<opción>] [TO PRINT] [PLAIN] [FOR / WHILE <expresión>]
Generación de informes sobre el fichero en uso.

RESET
Reinicialización del sistema tras el cambio de diskette.

RESTORE FROM <n-fich> [ADDITIV]
Volcado a la memoria de variables desde un fichero.

RETURN
Retorno al programa principal.

SAVE TO <n-ficheros>
SAVE TO <n-fich> ALL LIKE <plantilla>
SAVE TO <n-fich> ALL EXCEPT <plantilla>
Grabación en un fichero de las variables.

SELECT [PRIMARY / SECONDARY]
Selección de una de las zonas de memoria que albergan las bases de datos.

SET <opción a> [ON] [OFF]
Activación y desactivación de los entornos de uso de dBASE II.

1. ALTERNATE ON/OFF: Grabación de las órdenes de un fichero de disco.
2. BELL ON/OFF: Emisión de un pitido al completarse una zona de entrada de datos.
3. CARRY ON/OFF: Transferencia de los datos del registro precedente a los del registro en uso.

4. COLON ON/OFF: Elección del delimitador dos puntos (:) para las zonas de entrada de GET.
5. CONFIRM ON/OFF: Confirmación mediante RETURN tras la entrada de cada campo.
6. CONSOLE ON/OFF: Muestra de los caracteres emitidos en la pantalla.
7. DEBUG ON/OFF: Emisión de todos los mensajes de ECHO y STEP.
8. DELETE ON/OFF: Consideración de los registros marcados para el borrado.
9. ECHO ON/OFF: Visualización en la pantalla de los mandatos emitidos por un fichero de órdenes.
10. EJECT ON/OFF: Salto de página antes de la emisión de un informe.
11. ESCAPE ON/OFF: Cancelación de la ejecución del programa.
12. EXACT ON/OFF: Exactitud entre las dos cadenas con que se efectúan las comparaciones en una búsqueda.
13. INTENSITY ON/OFF: Presentación de los campos en vídeo normal o inverso.
14. LINKAGE ON/OFF: Independencia entre los ficheros primario y secundario.
15. PRINT ON/OFF: Emisión de los caracteres por impresora.
16. RAW ON/OFF: Separación de los campos mediante un espacio.
17. SCREEN ON/OFF: Operaciones a pantalla completa.
18. STEP ON/OFF: Ejecución paso a paso de un programa.
19. TALK ON/OFF: Muestra en la pantalla los resultados de los mandatos.

SET <parámetro> TO <opción>
Modificación de la configuración de dBASE II.

1. ALTERNATE TO <nombre de fichero>: Grabación en disco de las informaciones mostradas en la pantalla.
2. DATE TO <expresión de fecha>: Definición de la fecha del sistema.
3. DEFAULT TO <d:>: Unidad de disco por defecto.
4. FORMAT TO [SCREEN] [PRINT] [<n-fich de formato>]: Definición del periférico en uso y el fichero de formato.
5. HEADING TO <expresión>: Definición de la línea de cabecera de un informe.
6. INDEX TO <índice principal> [,<índice secundario>...]: Iden-

tificación y creación de ficheros de índices (7 como máximo) para la base de datos en uso.
7. MARGIN TO <expresión>: Definición del margen para un informe.

SKIP [+/− <expresión>]
Desplazamiento del puntero interno de la base de datos.

SORT ON <campo> TO [d:] <n-fich> [ASCENDING / DESCENDING]
Clasificación del fichero actual.

STORE <expresión> TO <variable>
Almacenamiento de un valor en una variable.

SUM <lista de campos> [TO <lista de variables>] [<opción> [FOR <expresión>]
Totalización de los campos numéricos de la base de datos actual.

TEXT
<texto>

ENDTEXT
Muestra el texto especificado hasta encontrar un ENDTEXT.

TOTAL ON <clave> TO <n-fich> [<lista de campos>] [FOR <expresión>]
Grabación en disco de las totalizaciones del fichero en uso según las condiciones especificadas.

UPDATE FROM <n-fich> ON <clave> [ADD <lista de campos>] [RANDOM] [REPLACE <lista de campos>] [<campo> WITH <campo origen>]
Actualización en bloque de un fichero clasificado o indexado a partir de los datos de otro archivo.

USE
Cierra todos los ficheros activos.

USE [<n-fich>]
Asigna la de datos en curso.

USE [<n-fich>] [INDEX <n-fich ind1>, <n-fich ind2>,...]
Asignación del fichero en curso junto a sus correspondientes índices.

WAIT
Detención hasta la pulsación de una tecla.

WAIT [TO <variable>]
Detención hasta la pulsación de una tecla y almacenamiento del carácter generado en la variable especificada.

APÉNDICE B

ORDENES DEL EDITOR

CTRL +	TECLAS DE EDICION
S	Retroceso de un carácter a la izquierda.
D	Avance de un carácter a la derecha.
E o A	Retroceso al campo anterior.
X o F	Avance al campo siguiente.
G	Borrado del carácter bajo el cursor.
Y	Borrado de campo.
V	Báscula del modo inserción.
W	Retorno a dBASE II teniendo en cuenta las modificaciones.
Q	Retorno a dBASE II sin tener en cuenta las modificaciones.

CTRL +	MODO CREATE, EDIT, BROWSE
C	Escritura de registro y avance al siguiente.
R	Escritura de registro y retroceso al anterior.

CTRL +	MODO MODIFY
C	Avance al campo siguiente.
R	Retroceso al campo anterior.
N	Inserción de un campo.
T	Borrado de un campo.

CTRL +	MODO EDIT, BROWSE
U	Marca para borrado.

CTRL +	MODO BROWSE
B	Avance un campo hacia la derecha.
Z	Retroceso un campo hacia la izquierda.

APÉNDICE C

DATOS GENERALES DEL dBASE II

Máximo número de índices en USE	7
Número de campos por registro	32
Número de registros por base de datos	65535
Máxima precisión numérica en cifras	10
Máximo número admisible	1.8E63
Mínimo número admisible	1.8E-63
Máximo número de variables simultáneas	64
Máximo número de instrucciones GET	64
Longitud máxima de una línea de mandatos	254
Número de caracteres por registro	1000
Máximo número de expresiones para SUM	5
Longitud máxima de una cabecera de informe	254
Máximo número de campos en un informe	24
Longitud máxima de una clave indexada	99
Máximo número de ficheros de mandatos abiertos	16
Número de caracteres por campo	254
Número de zonas de memoria	2

APÉNDICE D

TIPOS DE FICHEROS EN DBASE II

dBASE II gestiona ocho tipos diferentes de ficheros.

Extensión	Tipo de fichero
.DBF	Fichero de base de datos.
.MEM	Fichero de variables.
.TXT	Fichero de texto.
.NDX	Fichero indexado.
.CMD	Fichero de comandos.
.FRM	Fichero de informe.
.FMT	Fichero de formato.
.BAK	Fichero back up.

APÉNDICE E

MENSAJES DE ERROR EN dBASE II

Bad decimal width field
Número incorrecto de decimales en el campo (repetir la definición de decimales).

Bad file name
Nombre incorrecto de fichero (error de sintaxis en el nombre de fichero).

Bad name field
Nombre incorrecto para el campo (error de sintaxis en el nombre del campo).

Bad type field
Tipo incorrecto de campo (debe ser (C) carácter (N) numérico o (L) lógico).

Bad width field
Tamaño incorrecto de campo (la longitud del campo no debe sobrepasar los 254 caracteres).

Cannot insert-there are no records in database file
No puede realizarse ninguna inserción porque el fichero está vacío (para añadir registros a la base de datos utilice APPEND).

Cannot open file
El fichero no puede abrirse (error interno, consulte a su distribuidor).

Command file cannot be found
El fichero de mandatos no ha sido encontrado (comprobar el nombre del fichero).

Data item not found
No se ha encontrado el dato.

Database in use is not indexed
La base de datos no está indexada (FIND es sólo admisible en ficheros indexados).

Directory is full
El directorio está completo (se ha agotado el número de reseñas libres en el directorio).

Disk is full
El disco está lleno (no hay sitio en el disco para más datos).

End of file found unexpectedly
Fin de fichero inesperado (el formato de la base de datos en uso no es correcto).

«Field» phrase not found
Palabra «Field» no encontrada (error de sintaxis en la línea de órdenes).

File already exists
Fichero ya existente.

File does not exist
Fichero inexistente (verifique el disco introducido en la unidad).

File is currently open
El fichero ya está abierto (utilice USE o CLEAR para cerrarlo).

Format file cannot be opened
El fichero de formato no puede abrirse.

Format file has not been set
El fichero de formato no ha sido definido (utilizar SET).

Illegal data type
Tipo de dato incorrecto.

Illegal goto value
Valor para goto incorrecto (goto se debe ejecutar entre 1 y 65535).

Illegal variable name
Nombre de variable incorrecto (error de sintaxis en la variable).

Index does not match database
El índice no se corresponde con la base de datos.

Index file cannot be opened
El archivo índice no puede abrirse (error de sintaxis o no indexación de la base de datos).

Join attempted to generate more than 65,534 records
JOIN intentó generar más de 65534 registros (demasiados registros).

Keys are not the same length
Las claves no son de la misma longitud (UPDATE precisa que las claves sean de la misma dimensión).

Macro is not a character string
La macro-instrucción no está formada por una cadena de caracteres.

More than 5 fields to sum
La orden SUM no puede utilizar más de 5 campos a la vez.

Nesting limit violation exceeded
Rebasado el máximo anidamiento permitido.

No «FOR» phrase
Falta el «FOR» en la línea de mandato.

No «FROM» phrase
Falta el «FROM» en la línea de mandato.

No find
No encontrado (dBASE II no ha localizado el registro o clave).

Non-numeric expression
Expresión no numérica.

Nonexistent file
Fichero inexistente.

«ON» phrase not found
Falta el «ON» en la línea de mandatos.

Out of memory for memory variables
Se ha sobrepasado el espacio de memoria para las variables.

Record length exceeds maximum size (of 1000)
Longitud de registro superior a 1000 caracteres.

Record not in index
Registro no indexado (utilizar REINDEX).

Record out of range
Registro fuera de márgenes.

Source and destination data types are different
Los datos de origen y destino son de diferente tipo.

Syntax error
Error de sintaxis.

Sintax error in format specification
Error de sintaxis en la especificación de formato.

Sintax error, re-enter
Error de sintaxis, vuelva a introducir.

«TO» phrase not found
Falta el «TO» en la línea de mandatos.

Too many characters
Demasiados caracteres (acorte la línea de mandato).

Too many files are open
Demasiados ficheros abiertos (el máximo de ficheros abiertos simultáneamente es de 16).

Too many memory variables
Tan sólo se puede utilizar un máximo de 64 variables de memoria simultáneamente.

Too many returns encountered
Se han encontrado demasiadas órdenes «RETURN» (revisar un posible error en la estructura del fichero de comandos).

«WITH» phrase not found
Falta el «WITH» en la línea de comandos.

Unassigned file number
Número de fichero sin asignar.

Unknown command
Mandato desconocido.

Variable cannot be found
Variable no encontrada.

En el mundo actual la información es uno de los bienes más preciados. La cantidad de datos que nos bombardean constantemente ha llegado a adquirir tales dimensiones, que se ha hecho necesaria la intervención de la Informática para evitar que un trabajo tan lento y costoso como la estructuración y organización de las informaciones dependa directamente del ser humano. Para ello se crearon las bases de datos, que llegan hasta los ordenadores Amstrad de la mano de un magnífico programa: dBASE II.

GRAN BIBLIOTECA AMSTRAD

450 ptas.
(incluido IVA)
Precio en Canarias, Ceuta y Melilla: 435 ptas.