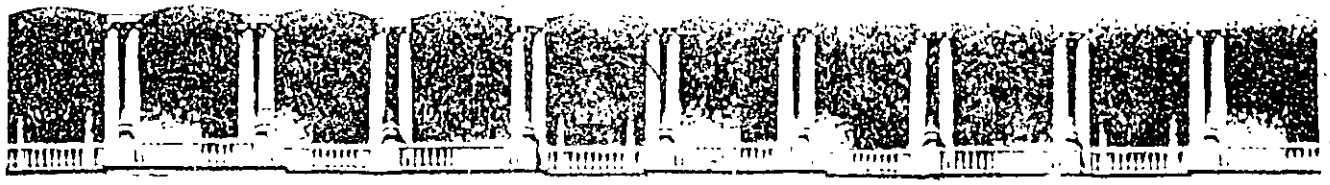


FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

MATERIAL DIDACTICO DEL CURSO

VISUAL FOXPRO

OCTUBRE, 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico, D.F. APDO. Postal M-2285
Telefonos: 512-8955 512-5121 521-7335 521 1987 Fax 510-0573 521-4020 AL 26

Prólogo

Estoy contento de exponer unas cuantas palabras a favor de mi buen amigo Les Pinter y su guía personal a Visual FoxPro. Esta nueva herramienta fascinante de desarrollo de software proporciona una poderosa recopilación de nuevas tecnologías, envueltas dentro de un conjunto de herramientas ágiles y fáciles para poner todo el poder de esas tecnologías en manos de una nueva generación de desarrolladores de aplicaciones.

En este libro, Les y John lo llevan en una visita guiada personal a través de Visual FoxPro, compartiendo con usted su perspectiva única. Su considerable experiencia en el desarrollo de aplicaciones lo dirigirán a través de la práctica. Usted aprenderá las habilidades medulares del desarrollo de aplicaciones básico y cómo seleccionar y usar la herramienta correcta dentro de Visual FoxPro para obtener un resultado que funcione.

Hay muchísimos conceptos, términos técnicos y componentes nuevos dentro de este nuevo producto. No importa si usted es novato en Visual FoxPro o está haciendo la transición a una nueva versión, tener una mano experimentada que lo guíe por lo esencial agilizará su aprendizaje y le dará una seguridad orientada y productiva en el nuevo entorno mucho más rápido. El énfasis está en aprender los elementos más esenciales, paso a paso. Disfrútelo.

Alan Schwartz
Desarrollador de FoxFire! de MicroMega

Introducción

Pinter Consulting es una empresa de desarrollo de software que se especializa en diseñar y programar aplicaciones de FoxPro. Mi hijo John y yo escribimos este libro para compartir algunas de las técnicas que utilizamos para los proyectos de nuestros clientes con otros programadores de FoxPro.

Éste es mi sexto libro sobre FoxPro. El primero trató sobre FoxPro 1.0, los siguientes dos sobre la versión 2.0 y el resto sobre la versión 2.5. A pesar de que cada versión introdujo muchas, muchas características nuevas al lenguaje, es justo decir que la diferencia entre la versión previa y Visual FoxPro es mucho más grande que la diferencia entre cualquiera de las dos versiones anteriores.

La diferencia principal es el soporte de Visual FoxPro de programación orientada a objetos (OOP: *Object Oriented Programming*). He escuchado acerca de la programación orientada a objetos por años y siempre me sorprendía cuando se introducía en mi vida. Bueno, hela aquí y estoy gratamente sorprendido. Le permite escribir mejor software en menos tiempo y con menos esfuerzo que nunca antes.

Una vez escuché una broma acerca de un vendedor cuya nueva esposa lo dejó después de la noche de bodas porque se sentó a la orilla de la cama durante toda la noche contándole lo bien que se la iban a pasar. Es así como me siento respecto a los artículos interminables de OOP que leo en los periódicos. Todos los artículos empiezan con páginas y páginas de discusiones teóricas sobre la diferencia entre

código lineal y objetos que saben cómo imprimirse a sí mismos. Simplemente no tiene sentido.

Ahora que he mirado más de cerca, empiezo a entender el problema de la programación orientada a objetos. Usted tiene que pensar en forma diferente para encontrar soluciones en Visual FoxPro. Me tomó años desarrollar enfoques para programar en FoxPro 2.5 y 2.6, pero con el tiempo, fui capaz de escuchar a futuros clientes y diseñar la solución para cuando hubieran terminado de hablar. Me tomó sólo un año ser capaz de hacer lo mismo con Visual FoxPro.

La aplicación desarrollada en este libro muestra la facilidad con la que puede darse vida a un proyecto cuando usted toma un enfoque simple. El proyecto, un sistema de entrada de órdenes para una librería, se hizo en tres etapas. Primero, usted utiliza los asistentes que vienen con Visual FoxPro para construir pantallas simples y conectarlas con un programa principal y de menú. Luego, después de aprender acerca de las características en el lenguaje, usted reelabora el proyecto tomando ventaja de lo que ha aprendido. En seguida, después de una introducción a la construcción y al uso de clases, usted reescribe de nuevo la aplicación completa. De esta manera, usted domina nuevas habilidades, luego ve usted mismo cómo las características cada vez más complejas valen la pena el esfuerzo que implica aprenderlas.

Este libro pretende iniciarlo rápidamente en Visual FoxPro. No hay teoría de la programación en él, no hay analogías sobre teléfonos o bolas rojas. Es un ejemplo simple y sencillo, programado en tres etapas que toma ventaja creciente de las nuevas características del lenguaje. Usted puede codificar desde cero la aplicación que está codificada al final del capítulo, en aproximadamente dos horas.

Y ahora algo completamente diferente...

Puse los ojos en Visual FoxPro por primera vez en mayo de 1994, y me tomó al menos un año antes de que comenzara a hacer las suposiciones correctas al analizar los problemas de programación. Ahora, se ha vuelto un juego ver que entre menos código pueda hacer, puedo exceder las expectativas de los clientes y alcanzar el presupuesto más bajo.

He aquí lo que descubrí: como se dice, no trates de enseñar a un cerdo a cantar; fallarás y además fastidias al cerdo. A Visual FoxPro le gusta hacer cosas en una forma determinada. Si usted trata de programar en Visual FoxPro utilizando el mismo estilo de codificación que la versión 2.6, se esforzará mucho y aún obtendrá resultados insatisfactorios. En lugar de ello, busque nuevas formas de hacer las viejas cosas.

Existen muchas fuentes de aplicaciones de muestra. El ejemplo TasTrade que viene con Visual FoxPro es una de ellas, pero el subdirectorio SAMPLES probablemente será más valioso para programadores intermedios. Busque ejemplos que asemejen la forma en que usted desearía que luciera su software, luego examínelos para ver qué los hace funcionar. Quedará sorprendido de ver cuántas de las cosas que yo solía codificar ahora están integradas al lenguaje.

En el pasado, un libro era la mejor manera de compartir código y técnicas. Mi libro *Applications Programming*, el cual ha sido traducido al chino, ruso y portugués fue un recurso utilizado ampliamente por los programadores de FoxPro y usted podía ordenar un disco que contenía el código fuente para las aplicaciones desarrolladas en el libro. Pero no fue enteramente satisfactorio. Impuso un retraso entre el momento que usted veía lo que quería hacer en la página impresa y el día en que el disco del libro llegaba. Y el gasto adicional, incluso si era un pequeño precio a pagar para ahorrarse el tecleo, no siempre se apreció.

Pero ahora existe una mejor forma. Si usted se conecta a CompuServe y entra a mi foro con GO PINTER, encontrará no sólo el código para este libro, sino muchos ejemplos adicionales. Espero que esto lo aliente a comenzar a utilizar estos enfoques de inmediato.

Existe un beneficio adicional asociado con la distribución del código en CompuServe. Este libro puede proporcionarle las oportunidades para demostrar las técnicas, pero el foro tiene docenas más. En el Foro Pinter encontrará muchos ejemplos adicionales de técnicas útiles e interesantes. Por ejemplo, mientras que los ejemplos del libro no utilizan barras de herramientas, por razones de espacio y simplicidad, el foro contiene varias aplicaciones de ejemplo que demuestran el valor de usar barras de herramientas comunes para controlar los formularios de su aplicación. El consorcio compuesto por CodeBook de Alan Giver, FoxFire!, FoxExpress y otros utiliza barras de herramientas exclusivamente y ellos hacen un desarrollo enorme y sencillo. Así que usted encontrará ejemplos más complejos y sofisticados en el foro que pueden incluirse en un libro introductorio como éste.

Este libro pretende ser una introducción práctica a la programación en Visual FoxPro. No hace hincapié en la teoría de la programación orientada a objetos y no se muestran varios enfoques distintos para resolver un problema. Hay muchas maneras diferentes de hacer la mayoría de las cosas en Visual FoxPro, pero en general sólo se muestra una forma simple.

Si usted utilizó Visual FoxPro 2.6, está familiarizado con las cláusulas WHEN y VALID. En el lenguaje de Visual FoxPro éstas fueron dos *eventos* soportados. VFP ha añadido docenas de eventos nuevos, cada uno de los cuales funciona en la

misma forma general; cuando llega el momento de un evento, se ejecuta el código correspondiente. Pero la riqueza del conjunto de eventos le permite un control preciso de los objetos en sus pantallas. Usted estará tentado a exagerar esto al principio. Aquí es donde esperamos que le sea útil este libro.

No quiero hacer hincapié en las diferencias entre el diseño de pantallas en FoxPro 2.6 y Visual FoxPro. Por alguna razón, algunos lectores no tendrán experiencia en FoxPro 2.6 y sólo lo encontrarán confuso. Pero la razón principal es que las diferencias son tan grandes que no tienen comparación. Nuestra empresa de consultoría actualmente está involucrada en media docena de escritos de aplicaciones de 2.6 a VFP, y la pantalla prototípica VFP no tiene más del cinco por ciento del código de su predecesor. Leyó bien; en promedio, ¡eliminamos el 95 por ciento del código! Y el programa resultante es infinitamente más fácil de depurar y mantener.

Quiero decir unas cuantas palabras sobre mis amigos en Rusia. Mi camarada Dimitry Artemov y yo comenzamos la edición rusa de *The Pinter FoxPro Letter* para ayudar a los programadores de la antes Unión Soviética a generar negocios y así liberarlos de la tiranía de la dependencia que es el más terrible legado del modelo soviético. Hoy en día, tengo cientos de amigos y conocidos en Rusia, Ucrania, Kazajkstán, Belarús y otros miembros de CEI quienes tienen una forma de vida holgada y ayudan a desarrollar un clima empresarial en sus países.

Hace algunos años, me senté a cenar con Bill Gates en su casa en Readmond y le sugerí que Microsoft bajara el precio de FoxPro de 300 dólares a 50 dólares para abrir oportunidades profesionales. Dos meses después, el precio bajó a alrededor de 50 dólares. Nunca supe si esa conversación tuvo algo que ver con el cambio de precio, pero quiero agradecer a Microsoft por sus políticas prudentes y generosas, las cuales han abierto las puertas a muchos programado/res que hablan ruso a desarrollar negocios y añadir valor en su sociedad. En lo personal, haré lo que esté al alcance de mi mano para asegurar su éxito ya que es importante, para ellos y para nosotros.

Escribimos este libro para dar a los programadores una ayuda en la construcción de un negocio. Hemos publicado *The Pinter FoxPro Letter* desde 1989 en Estados Unidos y desde 1993 en Rusia. He escuchado muchas veces hablar acerca de alguien que utilizó algo contenido en la publicación que le sirvió para terminar su trabajo y ganar dinero.

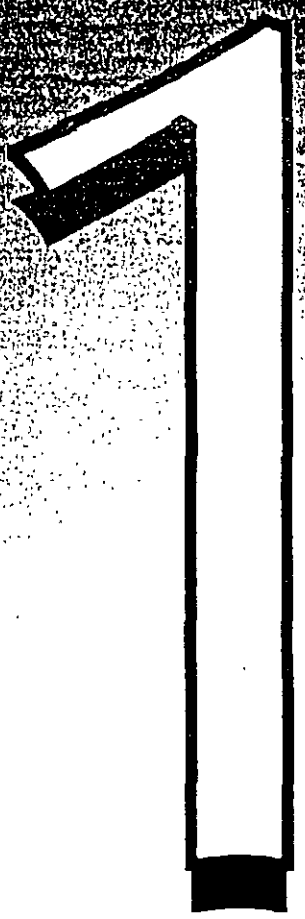
Me preocupó también que algunos enfoques publicados del desarrollo de software FoxPro fueron complicados innecesariamente. Y la mayor parte de los problemas de programación de bases de datos tiene soluciones simples. Las soluciones

académicas a menudo se discuten a un nivel etéreo que se asemeja poco al problema en forma práctica. Además, muchos desarrolladores de bases de datos tienen bases académicas distintas a las de la ciencia de la computación y están buscando algo que les funcione.

Las aplicaciones de trabajo son la mejor forma de aprender a enfocarse y resolver el tipo de problemas que todos enfrentamos. Es el motivo de este libro. Espero que pueda encontrar formas de generar aplicaciones de bases de datos basadas en enfoques simples.

Escribir un libro es un esfuerzo que consume tiempo, que en ocasiones se traduce en beneficios económicos, pero hasta que no consiga que todos se suscriban a *The Pinter FoxPro Letter*, los libros como éste son la mejor forma de ayudar a otros programadores a alcanzar sus metas. Me hace sentir bien. Y sólo duele un momento breve.

Les Pinter
en Rusia



Primeros pasos

CAPITULO 1

ISUAL FoxPro es un entorno de desarrollo para escribir aplicaciones de bases de datos. Proviene de la generación xBASE de lenguajes de programación, que incluye dBASE II y III, Clipper, FoxBASE y FoxPro, entre otros.

El propósito principal de las aplicaciones de bases de datos es mantener una o más tablas de información almacenadas en el disco duro de una computadora. En el argot de xBASE, a las tablas generalmente se les llama archivos .DBF, debido a que sus nombres contienen la extensión .DBF (*database file*: archivo de base de datos). Las tablas se caracterizan por tener columnas en las que toda la información contenida en cada columna es del mismo tipo. A las columnas normalmente se les llama *campos*.

Tal vez la característica principal de las tablas xBASE es que contienen información que describe al archivo en los primeros caracteres del archivo mismo, en un área llamada *encabezado del archivo (file header)*. Cuando un programa de FoxPro abre una tabla, lee el encabezado para averiguar qué hay en el archivo.

Esta brillante idea, tan simple que muchos nos preguntamos por qué no se nos ocurrió a nosotros, es la razón más importante por la que los principiantes encuentran a FoxPro tan sencillo de utilizar. Una vez que una tabla ha sido definida, simplemente se abre y se usa. Si se nos olvidan los nombres de las columnas, sólo hay que usar el comando LIST STRUCTURE y ya está.

La segunda idea brillante en los lenguajes xBASE, que afortunadamente se ha mantenido hasta hoy en día, es que son lenguajes *interpretados*. Los programas escritos en la mayoría de los lenguajes de cómputo deben ser compilados o traducidos a lenguaje de máquina, antes de poder ejecutarlos. Pero no en FoxPro, la ventana de comandos está siempre disponible, o puede ser llamada con una sola tecla, de tal manera que se pueden teclear comandos y ver qué hacen.

¿Qué hay en FoxPro?

FoxPro ha cambiado mucho desde que salió al mercado. Hoy en día tiene muchas, pero muchas características que simplifican la programación. Las ventanas Seguimiento y Depuración permiten ejecutar el código línea por línea y observar los valores vigentes de las variables o los campos de las tablas; adicionalmente, se puede hacer una pausa en el programa, no sólo en una línea específica, sino también cuando se cumpla cierta condición (por ejemplo, x=3). FoxPro permite el acceso al código durante el proceso de desarrollo, lo que lo convierte en uno de los mejores entornos de depuración existentes.

Un nuevo enfoque

Algunas áreas de Visual FoxPro han quedado prácticamente intactas de su predecesor FoxPro 2.6 para Windows -en especial el sistema de menús y el Generador de informes (reporteador). El diseño de pantallas, o diseño de formularios, como se le llama en Visual FoxPro, es radicalmente diferente. La razón es que los formularios en sí mismos son sensiblemente distintos del modelo GENSCRN.

Es aquí donde el muy mencionado modelo por objetos entra por la puerta grande. En FoxPro 2.6 la pantalla, expresada como una pareja de archivos con las extensiones .SCX y .SCT, era simplemente un depósito de instrucciones a ser usadas por el programa GENSCRN para producir un código, programa con la extensión .SPR. Para poder aprovechar totalmente el diseñador de pantallas se debía tener una buena idea de qué es lo que pasaría cuando se generara el código del programa .SPR.

En Visual FoxPro, los formularios conservan el uso de las extensiones .SCX y .SCT, pero no existe un GENSCRN, dado que *no se genera código alguno*. DO FORM *nombre del formulario* es el comando que ejecuta el formulario. Los formularios han sido ampliamente expandidos para incluir muchas especificaciones más, ahora llamadas *propiedades*, que se pueden establecer con el clic del ratón o con una asignación por medio de un comando. En lugar de desarrollar código, se desarrollan nuevas características en las pantallas simplemente con discernir qué propiedades especificar. Por ejemplo, se puede definir el color de fondo de un formulario estableciendo la propiedad BackColor. Para establecer que el color de fondo del formulario CLIENTE sea azul, use esto:

```
CLIENTE.BackColor = RGB ( 0, 0, 255 )
```

O bien, cuando se esté diseñando el formulario, simplemente se debe hacer clic en la línea BackColor de la ventana Propiedades y seleccionar el cuadrado azul del selector de colores que aparece.

Los métodos son funciones o procedimientos que se anexan a un formulario o control. La mayoría de los que normalmente se usan vienen incluidos en el formulario o control. Por ejemplo, Release es el método que libera un formulario de la memoria y de la pantalla, y los métodos se ejecutan simplemente usando su nombre precedido por el objeto que los posee; por ejemplo, para liberar el formulario CLIENTE, se incluyen las líneas CLIENTE.Release o THISFORM.Release en el procedimiento del evento clic del botón Salir.

El término THISFORM es un ejemplo de una manera genérica de referirse a un objeto. Permite obtener una copia o *instancia* del formulario con un nombre diferente, mientras se continúa escribiendo código que pueda averiguar qué formulario liberar. ¿Para qué se necesitaría hacer semejante cosa? Qué tal cuando se tienen dos instancias o copias del mismo formulario simultáneamente en la pantalla y se están viendo a dos clientes diferentes. ¡Con Visual FoxPro eso es fácil de hacer!

Usted quedaría sorprendido de cuántas cosas de las que uno hace necesitan otros desarrolladores. Los buenos amigos de Microsoft han construido el Generador de formularios de tal manera que cuantos requerimientos pudieran haber clasificado estén ya incluidos en la ventana Propiedades para formularios y controles. Sin embargo, en el caso de que hayan omitido algo, se pueden agregar propiedades a los formularios o controles y posteriormente agregar código que actúa sobre ellos. La sintaxis para referirse a las propiedades y métodos agregados es idéntica al código que se refiere a las propiedades y métodos internos de Visual FoxPro.

Programación orientada a objetos

Al permitir que el programa procese un formulario, como ocurre durante el tiempo de ejecución, en lugar de correr el código que fue generado en el tiempo de compilación, se introduce una enorme diferencia en la filosofía, dado que el formulario puede cambiar entre el tiempo en que se escribe y el momento en que el usuario lo ve. ¿Cómo? Puede modificarse instantáneamente haciendo intervenir al código. De hecho, el usuario puede cambiar las especificaciones de las propiedades de tal manera que alteren radicalmente la apariencia y el comportamiento del formulario.

Por supuesto que se puede escribir código como éste en otros lenguajes; simplemente hay que incluir 20 o 30 parámetros en la secuencia de llamada. De tal manera que si un usuario quiere cambiar el color de fondo, se debe incluir el color como un parámetro en la cadena de caracteres -el parámetro número 24 podría ser una buena opción- y entonces generar una llamada al formulario con código como éste:

```
DO FORM WITH ,,,,,,,,,,,,,,,,,,,,,,,,,RGB(0,0,1)
```

Por supuesto que se puede hacer... pero, ¿por qué molestarse? Yo he utilizado software de este tipo en mi vida anterior, como programador estadístico, y ¡era horrible! No puedo recordar el número de veces que el cliente dijo: "¿Por qué no simplemente fija la variable que controla los gráficos?" Obviamente, no existía

dicha variable. La podría haber programado, pero el costo hubiera sido mucho mayor de lo que mis estimados clientes hubieran considerado aceptable, y nunca realmente me creyeron cuando les describí las complicaciones que podría implicar.

La apertura de la arquitectura orientada a objetos hace posible el diseñar software flexible, dado que ni siquiera se acerca a lo difícil que solía ser. Los clientes definitivamente reciben mejor software puesto que hoy en día es tan fácil como ellos siempre imaginaron que debería haber sido. Si existe alguna manera mejor de desarrollar software, no la conozco.

Componentes de interfaz

Éstas son algunas de las herramientas más utilizadas. Algunas son totalmente nuevas o mejoradas de la versión anterior de FoxPro, mientras que otras han permanecido virtualmente intactas:

Ventana Examinar Una vista, tipo hoja de cálculo, de una tabla.

Ventana Código Para desplegar código asociado a varios eventos en los formularios y controles. Cuando el evento se dispara el código se ejecuta.

Ventana Depuración Permite examinar variables de memoria o valores de campos y establecer puntos de interrupción. La ejecución del programa se detiene cuando una variable de memoria o una expresión con un punto de interrupción cambia de valor.

Cuadro de diálogo Generador de expresiones Lo lleva, paso a paso, a través del proceso de construcción de expresiones complejas, proporcionando tanto nombres de campos como de funciones.

Comando Opciones (menú Herramientas) Permite controlar la configuración de docenas de características en el entorno FoxPro, incluidos todos los comandos SET, así como plantillas y bibliotecas de clases.

Ventana Propiedades Le permite establecer propiedades en una buena cantidad de generadores, incluidos los generadores de formularios, informes, etiquetas y bases de datos, y proporciona acceso a propiedades, métodos y código de eventos.

Administrador de proyectos Un diseño completamente nuevo del administrador de proyectos de FoxPro 2.6, este administrador de proyectos

organiza todos los componentes de un proyecto en cinco grupos: Datos (bases de datos, tablas, tablas libres, vistas locales y remotas, conexiones, consultas y procedimientos almacenados), Documentos (formularios, informes y etiquetas), Bibliotecas de clases, Código como programas, bibliotecas API (*Application Program Interface:* Interfaz de programas de aplicación) y otras aplicaciones, y Otros (menús, archivos de texto y otros archivos; por ejemplo, gráficos .BMP).

Generador de consultas Una recodificación completa del RQBE (*relational query by example*: consulta relacional ejemplificada), esta herramienta maneja todos los aspectos de construir una consulta.

Barras de herramientas FoxPro proporciona a los generadores de aplicaciones más de una docena de barras de herramientas para colocar todas las herramientas para varias tareas justo al alcance de sus dedos. Además, usted puede diseñar sus propias barras de herramientas en conjunción con formularios, para proporcionar a los usuarios el mismo tipo de acceso instantáneo a las herramientas.

Ventana Seguimiento Visual FoxPro le permite observar la ejecución de su código. Cuando se utiliza en conjunción con la opción de punto de ruptura contenida en la ventana Depuración, generalmente se pueden encontrar los problemas, cuando mucho en unos cuantos minutos.

Ventana Ver Una vista se ve como una consulta, pero dado que las vistas en Visual FoxPro están ligadas a sus fuentes de datos (los cambios que los usuarios hacen a los datos de la consulta se utilizan para actualizar las tablas fuente), esto la convierte en una herramienta extraordinariamente poderosa en el diseño de aplicaciones.

Generadores

La mayoría de las tareas complejas se manejan por alguno de los muchos generadores (*designers*) disponibles en FoxPro. Los generadores son entornos de trabajo en los que se construyen componentes de una aplicación de FoxPro. La siguiente lista proporciona una idea de los generadores disponibles:

Generador de clases Para construir objetos reutilizables.

Generador de conexiones Para facilitar el acceso a datos remotos.

Generador de bases de datos Para organizar los datos en tablas y documentar las relaciones entre tablas.

Generador de formularios Para diseñar las pantallas de la aplicación.

Generador de consultas Para construir conjuntos de datos utilizados en reportes y en pantallas de sólo lectura.

Generador de vistas Para construir conjuntos de datos que, cuando se cambien en los formularios, también cambien en las tablas de las que provienen.

Generador de informes Para construir informes para la pantalla o la impresora.

Generador de etiquetas Para dar formato a salidas de etiquetas impresas.

Generador de menús Construye el sistema de menús que ejecuta una aplicación.

Generador de tablas Administra el formato de las tablas utilizadas en la aplicación.

Asistentes

Asistentes son lo que la mayoría de la gente quiere cuando oprime una tecla de ayuda. Un asistente es un programa de FoxPro diseñado para manejar alguna tarea específica, como construir un tipo particular de control o una etiqueta para correo postal. Una buena cantidad de asistentes estaba disponible a la fecha de redacción de este libro y muchos más surgirán sin duda.

Asistente para formularios Construye "pantallas instantáneas" con la estructura de las tablas basándose en clases prediseñadas, incluidos efectos especiales en las pantallas y botones de navegación interconstruidos.

Asistente para documentación Documenta la aplicación.

Asistente para importar Facilita la traducción de datos de otros formatos. Si su primer desarrollo en Visual FoxPro incluye convertir una aplicación originalmente escrita en otro lenguaje, muy probablemente utilizará este asistente antes que cualquier otra cosa.

Asistente para etiquetas Construye aplicaciones de impresión de etiquetas.

Asistente para combinar correspondencia Construye un archivo de combinación de correspondencia ya sea en formato de MS Word o bien

delimitado por comas, luego prepara el documento de Word para combinarlo con la lista de correspondencia y llama a MS Word para que termine el trabajo.

Asistente para formularios Construye un formulario "plano" usando la estructura de la tabla de datos.

Asistente para formularios uno a varios Construye una pantalla tipo factura usando el objeto de cuadrícula de FoxPro.

Asistente para tablas dinámicas Crea una tabulación cruzada en Microsoft Excel, instalándolo opcionalmente como un objeto OLE en un formulario. Se deben tener Excel y MS Query en la computadora para poder usar este asistente.

Asistente para consultas Construye la sintaxis SQL necesaria para extraer un subconjunto de datos de las tablas y desplegarlo.

Asistente para informes Diseña informes, sencillos o complejos, utilizando un poco más que la estructura de las tablas.

Asistente para tablas Si no se han diseñado las tablas de datos y se requiere una sugerencia inicial, Visual FoxPro proporciona más de 40 copias predefinidas de estructuras de tablas. Usted puede modificarlas o añandirles cosas, pero probablemente nunca necesite hacerlo.

Asistente para upsizing Una vez que se ha construido una aplicación en la computadora de escritorio, se pueden mover todas las referencias a datos hacia un servidor remoto utilizando esta sencilla herramienta.

La figura 1-1 muestra un ejemplo del Asistente para tablas en acción. Los asistentes simplifican muchas tareas en ambos sentidos; hacen la tarea más sencilla y crean objetos que no están tan pulidos como si usted hubiera tenido el tiempo de crearlos a mano. Aun así, son excelentes herramientas de aprendizaje y en muchos casos el resultado final es perfectamente utilizable, o cuando menos, proporciona un buen comienzo.

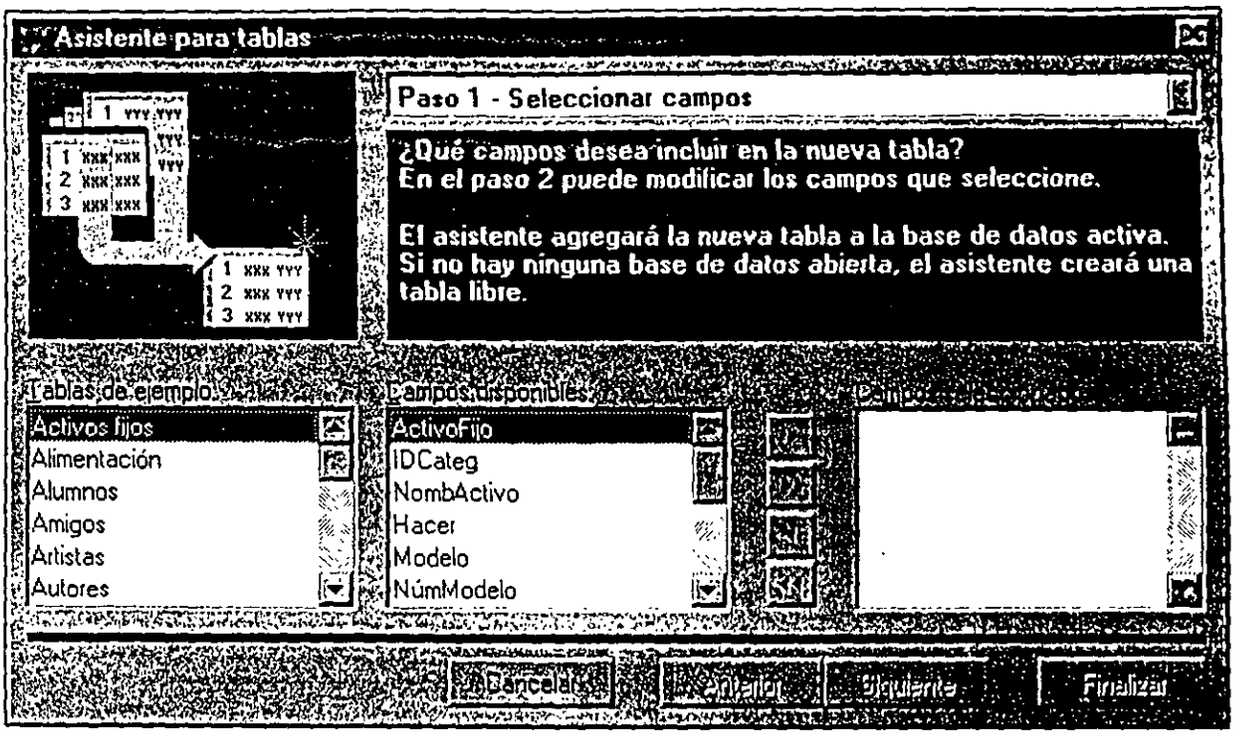


Figura 1-1

Asistente para tablas.

Generadores

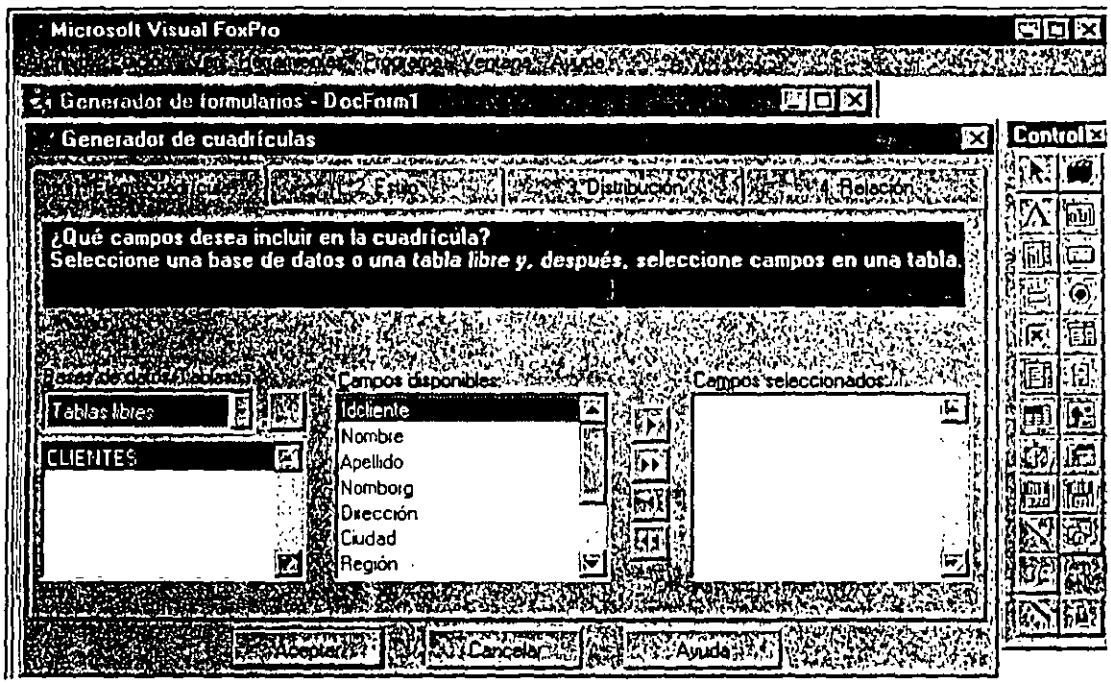
Uno de los toques finales que todavía debe ser adicionado a Visual FoxPro -y uno que desafortunadamente no se presentará con detalle en este libro- es la proliferación de generadores (*builders*). Éstos son herramientas que corren en alguna de las superficies de diseño y usan la información disponible para construir el control. El Generador de cuadrículas, mostrado en la figura 1-2, es un ejemplo de qué se puede esperar. A la fecha de esta publicación, existían los siguientes:

Generador de autoformato Aplica un estilo, de un conjunto predeterminado, a los controles seleccionados en el formulario.

Generador de formularios Éste es probablemente el generador que más utilizará. Emplea los campos de una tabla para llevar a cabo el primer intento de diseño de un formulario. Posteriormente se pueden introducir las modificaciones necesarias al conjunto de objetos generados.

Generador de cuadrículas Permite aprender cómo las configuraciones de la propiedad de cuadrícula del objeto controlan la operación de la cuadrícula.

Figura 1-2



Generador de cuadrículas.

Generador de integridad referencial Escribe código en el campo memo de procedimientos almacenados en el contenedor de la base de datos, de tal manera que las operaciones que afectan a los registros relacionados, se ejecutan automáticamente.

Generador de cuadros de edición Éste y los otros cinco generadores permiten seleccionar combinaciones útiles de opciones para las clases relacionadas. Los cinco generadores relacionados son:

- El Generador de cuadros de texto
- El Generador de cuadros de lista
- El Generador de grupos de opciones
- El Generador de cuadros combinados
- El Generador de grupos de comandos

Hay una proliferación de generadores inteligentes disponibles en CompuServe y en la literatura. La arquitectura abierta de FoxPro lo permite y todos nos beneficiaremos con ello.

Barras de herramientas

FoxPro ha ido un paso más allá de los menús y ha creado un conjunto de barras de herramientas para tareas específicas que ayudan en la ejecución de distintas tareas. La idea es sencilla: las herramientas que más se utilizan para cada trabajo se colectan en barras de herramientas -grupos de iconos en los que simplemente se hace clic para obtener la herramienta que se necesita. Es en el uso de barras de herramientas donde FoxPro más se asemeja a una mesa de trabajo. Usted estira la mano para alcanzar el desarmador de cruz que siempre deja en el mismo lugar y, ahí está. Además puede personalizar las barras de herramientas quitando opciones que no necesite y agregando las que sí, agrupando todo lo necesario para el trabajo en un sólo menú contextual que esté siempre sobre la mesa de trabajo. Las barras de herramientas actuales incluyen las siguientes:

Paleta de colores Permite la selección visual de colores y posteriormente inserta el resultado en la propiedad seleccionada del formulario en una función RGB, con sus tres parámetros numéricos (rojo, verde, azul), basados en el color elegido.

Generador de bases de datos Contiene iconos que manejan el entorno de datos, iconos para: crear una tabla, agregar una tabla, quitar una tabla, crear una nueva vista remota o local, modificar una tabla, examinar una tabla, o editar procedimientos almacenados en el contenedor de la base de datos.

Controles de formularios Inserta controles o texto en un formulario. Además un bloqueo del generador le permite seleccionar el mismo control repetidamente sin necesidad de regresar a la barra de herramientas.

Generador de formularios Durante el diseño de formularios, esta barra de herramientas permite el paso rápido de uno a otro entre varios elementos usados en el diseño de pantallas: el entorno de datos, la ventana Propiedades, la ventana Código, la barra de herramientas Controles de formularios, la paleta de colores, la barra de herramientas Distribución, el Generador de formularios (el cual selecciona campos del entorno de datos y aplica estilos de la biblioteca de asistentes) y la herramienta de autoformato (quien aplica estilos a los controles existentes en el formulario).

Distribución Alinea y modifica el tamaño de los objetos que estén en la pantalla. Esta importante mejora de FoxPro 2.6 permite insertar campos o etiquetas en la pantalla y posteriormente seleccionar grupos de objetos y alinearlos vertical u horizontalmente.

Presentación preliminar Una vez que se ha diseñado un informe, se puede ver en la pantalla, moverlo hacia adelante o hacia atrás y ampliar o reducir el tamaño desplegado. En efecto, esta barra de herramientas agrega la posibilidad de "emitir informes sin papel" a las aplicaciones.

Generador de consultas Contiene iconos para agregar o eliminar una tabla, agregar una combinación, desplegar la ventana de SQL, maximizar la vista de la tabla y fijar el destino de la consulta.

Controles de informes Permite la selección de objetos, así como iconos, para agregar los seis tipos de objetos que pueden ser colocados en un informe: etiquetas, campos, líneas, rectángulos, rectángulos redondeados y controles de imágenes u OLE dependientes. Un bloqueo del generador facilita añadir varios controles del mismo tipo, uno tras otro, sin necesidad de volver a hacer clic en el icono apropiado.

Generador de informes Proporciona acceso instantáneo a los controles de informes, a la paleta de colores y la barra de herramientas Distribución, y a la característica de agrupación de datos del menú desplegable Informes.

Estándar Está presente cuando se inicia FoxPro, proporciona acceso al Generador de formularios y al Generador de informes, a bases de datos de impresión, tablas, consultas, conexiones, vistas, vistas remotas, formularios, informes, etiquetas, programas, clases, archivos de texto y menús. Cuando así lo requiera, puede seleccionar un asistente para que le ayude en el proceso.

Generador de vistas Contiene iconos para agregar o eliminar una tabla, agregar una combinación, desplegar la ventana de SQL y maximizar la vista de la tabla del Generador de vistas. Las vistas, a diferencia de las consultas, constituyen una conexión permanente entre el cursor y sus fuentes.

El Administrador de proyectos

Los proyectos de FoxPro se encuentran integrados por el Administrador de proyectos, quien mantiene la pista de los componentes de la aplicación. Una pantalla del Administrador de proyectos se muestra en la figura 1-3. Conforme se agregan componentes a un proyecto, FoxPro los colecta bajo alguno de los siguientes encabezados principales:

Datos Las bases de datos (y todos los elementos que pueden describir), incluidas las tablas, vistas locales y remotas, conexiones y procedimientos almacenados, así como tablas libres y consultas.

Documentos Formularios, etiquetas e informes.

Bibliotecas de clases Repositorios de objetos usados en la aplicación.

Código .PRG que contienen código que no está asociado con un formulario, así como bibliotecas API y archivos llamados por la aplicación.

Otros Menús, archivos de texto y otros, incluyendo mapas de bits.

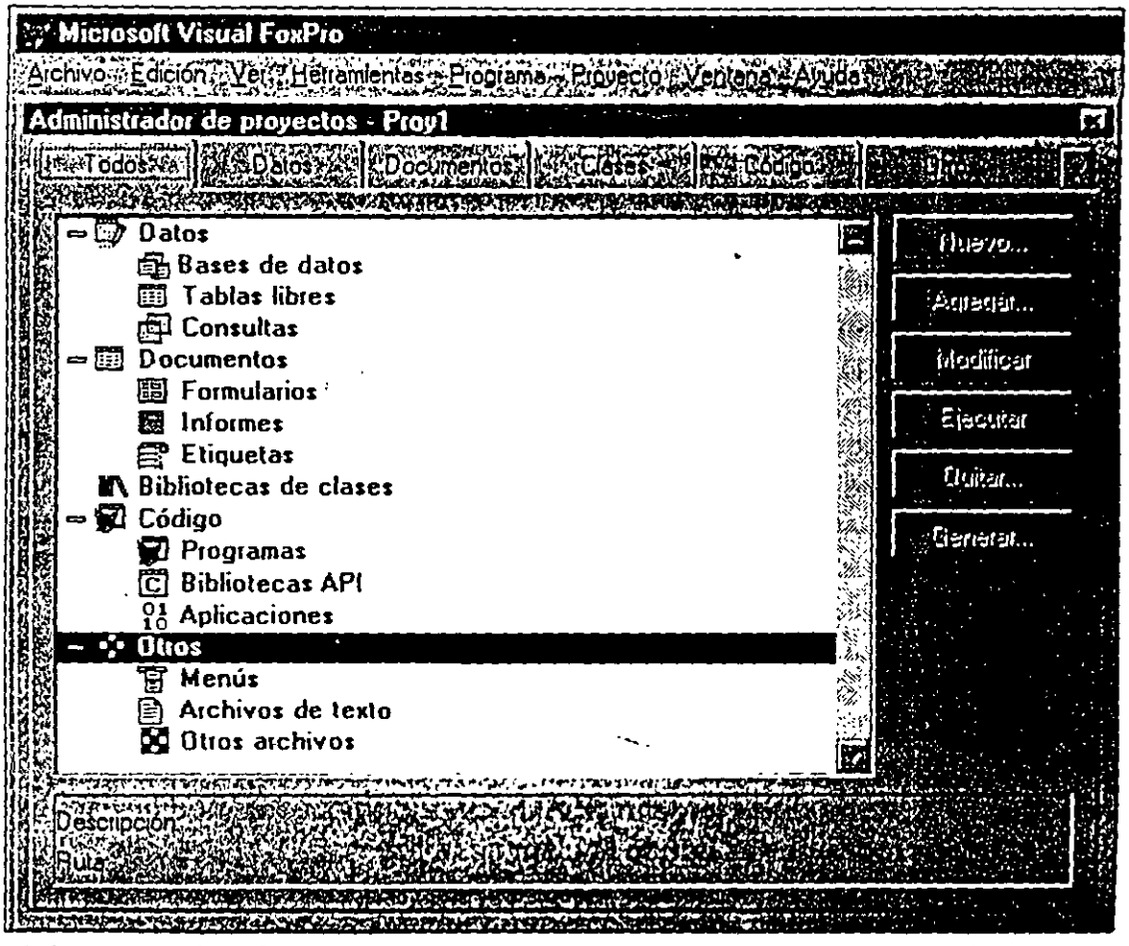


Figura 1-3

Administrador de proyectos.

Se designa un programa principal (usualmente llamado PRINCIPAL.PRG) como el punto de partida de la aplicación y luego se construye la aplicación en uno de los varios formatos, por lo general un archivo .APP para el cual se requiere una copia de Visual FoxPro, o un .EXE que pueda ejecutarse ya sea desde FoxPro o con la biblioteca de soporte de Visual FoxPro (runtime) disponible en la edición profesional de Visual FoxPro.

Los ejecutables de FoxPro son grandes; los tamaños inician en alrededor de 1.5MB y pueden ser del doble o triple de este tamaño; sin embargo, no se preocupe, ya que las aplicaciones de FoxPro se cargan razonablemente rápido y se comprimen a la mitad de su tamaño o menos.

Un nuevo paradigma

Visual FoxPro agrega una nueva dimensión a xBASE: programación manejada por eventos. Desde la invención del ratón (mouse) y otros dispositivos apuntadores, los usuarios han preferido apuntar y hacer clic en lugar de seguir un conjunto de pasos. La programación manejada por eventos significa que los programas pueden responder al clic de un ratón o a otro evento en cualquier momento.

Desde el principio de la programación, el código ha sido escrito y ejecutado de manera lineal. Así es como funcionaban las computadoras. Hizo al código fácil de entender y seguir, pero restringió la manera en que funcionaban los programas. Visual FoxPro agrega docenas de eventos que pueden detonar código relacionado, lo que significa que usted debe estar atento a aquello que pueda interrumpir la ejecución de un programa. Es un poco más complejo que esto, pero poco a poco lo comprenderá y los resultados serán verdaderamente impresionantes.

Finalmente, la introducción de la programación orientada a objetos ha creado una encarnación robusta de Visual FoxPro. Si se diseña una pantalla u otro "objeto" que se desee reutilizar en otra parte de la aplicación o en otras aplicaciones, se guarda como clase. Se pueden usar clases en las aplicaciones con mejoras o modificaciones locales que no tocan la copia original del objeto (clase). De esta manera, una clase base puede servir como modelo para muchas variantes de la misma idea. Y si se modifica la definición de clase original, las modificaciones se llevan a cabo en todas partes donde la clase se utiliza.

Es posible que se tenga que acostumbrar a un nuevo vocabulario. El término adecuado para *la copia de una clase es instancia*. No se copia una clase, se crea una instancia del objeto definido por la clase. En el pasado, copiábamos pantallas

o código de programas y realizábamos cambios a la copia, pero la programación orientada a objetos pretende eliminar esta práctica. Usaremos la nueva terminología, pero se puede decir "copia de clase" si se desea.

Análogamente, se pueden agregar propiedades y métodos a los objetos. Aunque hay diferencias, se puede pensar en ello como simplemente agregar variables y funciones locales y no se estará lejos de la realidad.

Configuración del entorno

Antes de que instale Visual FoxPro hay algunas cosas que necesita saber. FoxPro ama la memoria, en especial FoxPro para Windows. La cantidad de memoria mínima requerida para correr FoxPro es 12 megabytes y conforme a nuestra experiencia 16 es notablemente mejor.

¿Por qué FoxPro requiere tanta memoria? La razón principal es que FoxPro ha sido construido para lograr velocidad. Depende en gran medida del uso de la memoria caché -almacenando en la memoria la información recientemente leída del disco- no sólo los datos, sino también el código de programas, índices y cualquier otra cosa que utilice. El acceso a la memoria es muchísimo más rápido que el acceso al disco. Afortunadamente los precios de la memoria van en constante declinación. Esperamos que uno de estos días regalen SIMM de 4MB en cajas de cereal; pero aun así, dado que usted acaba de comprar esa nueva computadora, ¿quién quiere gastar otros 500 dólares en chips de memoria?

Lo compadecemos por tener que comprar más memoria, pero el hecho es que los desarrolladores no nada más usan computadoras, también usan software. Por lo general codificamos, compilamos y probamos. La codificación toma el mismo tiempo en cualquier máquina ya que no se puede teclear más rápido que la computadora, pero hemos visto compilaciones que toman varios minutos en una máquina y sólo dos segundos en una Pentium-90. Los tiempos de carga y ejecución también pueden incrementarse dramáticamente. El hecho es que ¡su equipo de 3,000 dólares puede correr FoxPro dos o tres veces más rápido si le aumenta la memoria a 16 megabytes! Usted compró esa nueva computadora porque su tiempo es valioso, así que si tiene menos de 16MB vaya a comprar más memoria.

Una vez que tiene suficiente memoria debe decirle a FoxPro cómo utilizarla. A FoxPro le gusta la memoria extendida, del tipo que rebasa los 1024K. También le gusta la memoria convencional, hasta 640K. FoxPro no puede utilizar los 384K de memoria arriba de los primeros 640K, pero el comando DOS LOADHIGH sí

puede. Así que ponga al principio de su archivo CONFIG.SYS las siguientes líneas:

```
DEVICE=DOS\HIMEM.SYS
DEVICE=DOS\EMM386.EXE NOEMS
```

El parámetro NOEMS le indica a DOS que no use la memoria arriba de 1024K como memoria expandida (paginada), lo cual se solía hacer en una época, pero ya no. Ahora, cargue los adaptadores de red DOSKEY y cualquier otra cosa que quiera en la memoria alta, como sigue:

```
DEVICEHIGH=DOSKEY.EXE
DEVICEHIGH=NET\NE2000.DRV
...
```

O bien, para archivos ejecutables se puede hacer lo siguiente, en AUTOEXEC.BAT:

```
LOADHIGH DOSKEY.EXE
LOADHIGH HYPERKEY.EXE   && programa acelerador del teclado
```

Finalmente, FoxPro debe tener un número suficiente de manipuladores de archivo, de manera que siempre incluya esto en CONFIG.SYS:

```
FILES=99
```

Instalación de Visual FoxPro

Visual FoxPro utiliza el programa SHARE.EXE, de DOS, para manejar OLE y otros procesos, así que necesita cargarlo antes de proceder. Se debe usar la siguiente sintaxis:

```
LOADHIGH DOS\SHARE.EXE /F:8192 /L:200
```

El parámetro F: asigna espacio de archivo en bytes para compartir información de archivos de DOS; mientras que el parámetro L: define el número de archivos que pueden ser bloqueados al mismo tiempo. Estos números generalmente funcionan bien, para mí.

Si utiliza Windows 3.x, haga clic en el menú Archivo y después haga clic en Ejecutar. Posteriormente teclee A:\SETUP (si está instalando desde la unidad B:, entonces teclee B:\SETUP), o haga clic en Examinar, cambie a la unidad A:, haga

clic en SETUP.EXE y luego en Aceptar. Aparecerá la pantalla de instalación de FoxPro.

Si utiliza Windows 95, haga clic en el botón Inicio y seleccione la opción Ejecutar. En el cuadro Abrir del cuadro de diálogo Ejecutar que se despliega, teclee A:\SETUP (si está instalando desde la unidad B, entonces teclee B:\SETUP), y haga clic en Aceptar.

El directorio predeterminado es \VFP. A menos que tenga una muy buena razón para cambiarlo, use ese nombre de directorio. Inserte los distintos discos conforme vayan siendo solicitados. Visual FoxPro construye una estructura de directorios bajo el directorio VFP. Dicha estructura es parecida a la de la figura 1-4. Los subdirectorios son los siguientes:

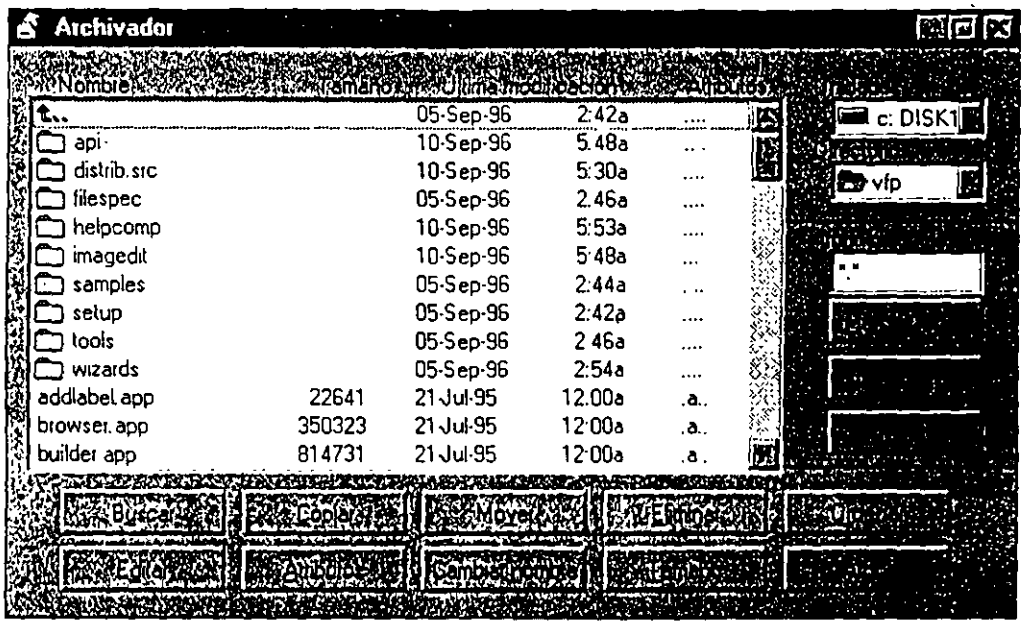


Figura 1-4

Estructura de directorios de Visual FoxPro.

* FILESPEC

Contiene dos tablas con descripciones de los campos contenidos en los archivos de FoxPro, incluyendo información de múltiples plataformas.

* TOOLS

CPZERO Contiene un programa de utilería para establecer la página de código para una tabla. Las páginas de código determinan qué conjunto de caracteres

nacionales se almacenan en una tabla. Para los que hablan inglés, la página de código 1252 es la que normalmente se usa. El byte de la página de código se almacena en el byte 29 del área de encabezado en una tabla de FoxPro.

GENDBC Contiene una utilería que lee un contenedor de una base de datos y posteriormente construye un programa que recrea las tablas e índices descritos ahí.

* SAMPLES

Contiene proyectos de ejemplo completos con código fuente, para demostrar todas las capacidades de FoxPro. Este directorio, a su vez, contiene seis subdirectorios:

CONTROLS Contiene SAMPLES.APP, que demuestra los distintos controles. Los ejemplos y su fuente se localizan en los siguientes subdirectorios bajo CONTROLS:

GRAPHICS Dibuja líneas y figuras en un formulario y demuestra un graficador de ecuaciones.

EVENTS Muestra la secuencia en que se disparan los eventos.

OBJECTS Muestra cómo modificar el tamaño de los objetos de manera dinámica, activa la opción de arrastrar y soltar (drag and drop) y ejecuta múltiples formularios.

PGFRAME Demuestra el uso de diferentes marcos de página y un cuadro de diálogo con fichas. Los marcos de página son uno de los elementos de diseño más poderosos de Visual FoxPro.

LISTS Muestra cómo cambiar RowSourceType, seleccionar múltiples elementos en un cuadro de lista, agregar interactivamente elementos a un cuadro de lista, desplegar múltiples columnas y mover elementos entre cuadros de lista.

GRID El objeto de cuadrícula de Visual FoxPro es lo que se suponía era el BROWSE y aún más. Se pueden construir formularios de uno a varios que operen como usted espera, cambiar las propiedades de la cuadrícula e incluso colocar controles como casillas de verificación en la columna de una cuadrícula.

TIMER Cronómetro (swatch.prg) e intervalos de cronómetro (timecomm.prg) demuestran cómo se puede utilizar el cronómetro en las aplicaciones.

CSAPP Contiene un ejemplo cliente-servidor. Visual FoxPro hace que el desarrollo de aplicaciones cliente-servidor sea sumamente sencillo. Para algunas aplicaciones, todo lo que se tiene que hacer es construir una vista remota y los datos en el servidor se actualizan conforme los cambios en el cursor local se envían de regreso de forma automática.

DATA Contiene datos de muestra utilizados en los ejemplos.

MAINSAMP Contiene la aplicación de muestra Importadores Tastrade, TASTRADE.APP. Puede ejecutarla para ver una aplicación interesante escrita en Visual FoxPro.

OLE Contiene ejemplos de OLE (Object Linking and Embeding: incrustación y vinculación de objetos). Para poder correr esta demostración Microsoft Excel y Microsoft Word deben estar instalados en la computadora.

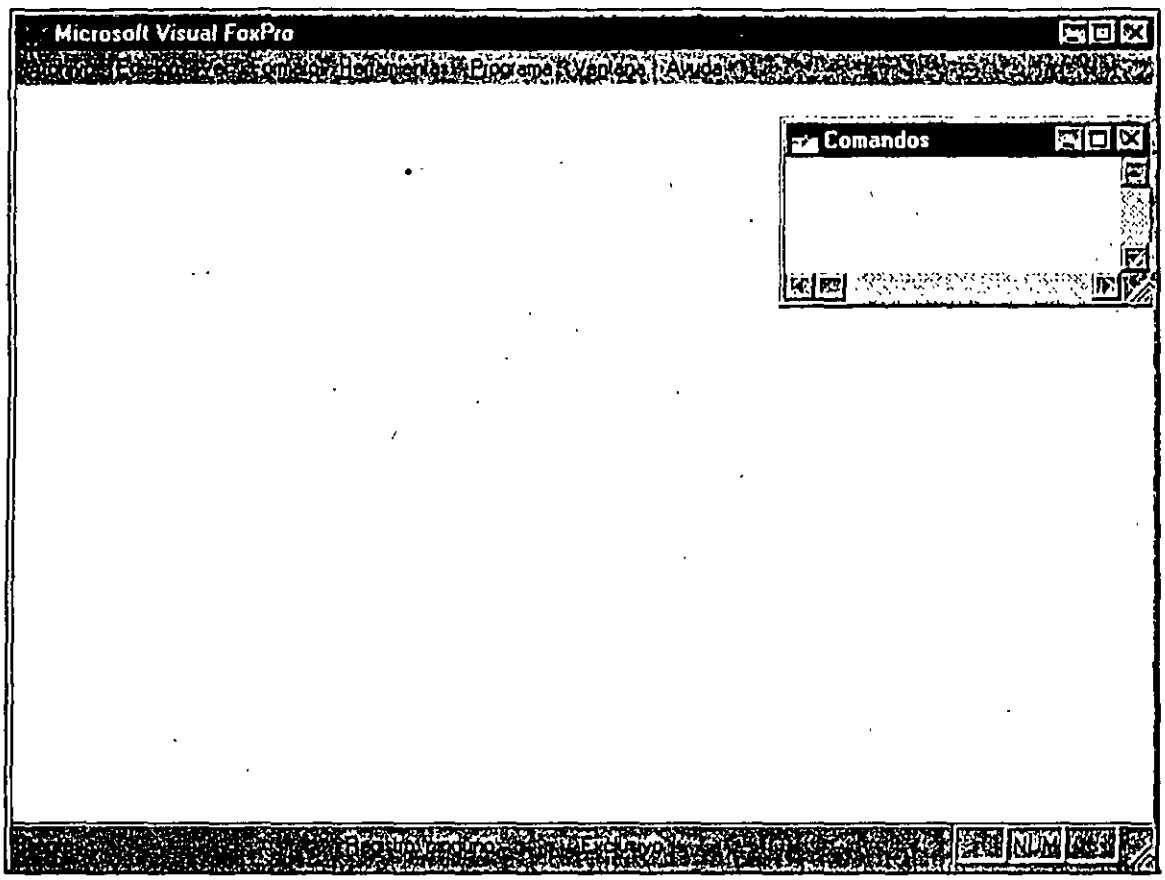
* WIZARDS

Contiene los asistentes de FoxPro, los cuales lo llevan paso a paso a través de muchas de las tareas comunes requeridas para construir aplicaciones. El directorio VFP contiene VFP.EXE, así como muchos otros programas y archivos. El archivo de ayuda HELP de FoxPro es casi tan grande como VFP.EXE mismo y contiene una enorme cantidad de información sobre comandos y funciones de FoxPro.

En modo predeterminado, Visual FoxPro se instala bajo su propio grupo de programas en Windows, con varios iconos en el grupo. El primero es el conocido logotipo del zorro; haga clic en él para ejecutar el programa. Aparecerá la ventana de licencia seguida por el entorno tipo mesa de trabajo de escritorio que se exhibe en la figura 1-5.

Los elementos más significativos de la mesa de trabajo son la pantalla principal, el escritorio, la ventana Comandos, el menú y la barra de herramientas estándar. La pantalla del escritorio es una verdadera ventana, con los controles usuales para cerrar, minimizar y maximizar. Hacer clic en la esquina superior izquierda es una de las formas para salir de FoxPro. ALT-F4 es otra. Teclear QUIT en la ventana Comandos es otra manera. Ahora sabe cómo salir del programa. ¿Recuerda WordPerfect 5.1? ¿Quién fue el científico de cohetes que decidió utilizar F7, Yes, No para salir del programa?

Figura 1-5



El escritorio de Visual FoxPro.

Configuración de Visual FoxPro

Cuando se carga FoxPro, lee un archivo llamado CONFIG.FPW en el directorio VFP. Si desea que lea un archivo de configuración localizado en algún otro lugar, utilice la siguiente declaración SET en AUTOEXEC.BAT:

```
SET FOXPROCFG=C:\MIDIR\CONFIG.FPW
```

CONFIG.FPW es un archivo de texto ASCII y contiene instrucciones de la forma:

```
NOMBREDELPARAMETRO=VALOR
```

Estos parámetros se leen al momento de iniciar y se convierten en los valores predeterminados de la sesión de FoxPro, aunque usted puede cambiarlos en cualquier momento. He aquí algunos de los elementos que los desarrolladores fijan con frecuencia en CONFIG.FPW:

```
SAFETY=OFF                          && No avisa antes de sobrescribir archivos
TALK=OFF                            && No despliega el progreso
ECHO=OFF                            && Inicialmente desactiva el rastreo
DEBUG=ON                            && Habilita el rastreo y la depuración
RESOURCE=FOXUSER                    && Elige un archivo de recursos
RESOURCE=OFF                        && No se use hasta que se solicite
COMMAND=DO PROG                     && Corre PROG al inicio
PATH=DBFS; MENUS; SCREENS; PRGS     && Simplifica pantallas, pruebas/
                                    && depuración
```

Si se especifica SET RESOURCE ON, FoxPro guarda detalles de muchas de las cosas que usted hace en un archivo llamado FOXUSER. Por ejemplo, si se cambia el tamaño y la ubicación de la ventana Comandos, sus nuevos parámetros se almacenan en un registro en FOXUSER. La próxima vez que se inicie FoxPro, se leerán y usarán los últimos valores de los parámetros.

Hemos visto archivos de recursos con miles de registros en ellos. FoxPro puede volverse lento si FOXUSER es demasiado grande, pero pueden eliminarse de él líneas indeseables. Teclee:

```
SET RESOURCE OFF
USE \VFP\FOXUSER
COPY TO RESBACK
BROWSE
```

Se observará una columna etiquetada ID, misma que contiene datos como LABELLYT, WINDBROW, WINDSNIP y FORMINFO. La siguiente columna a la derecha se llama Name. Si contiene nombres de elementos de sus proyectos, como nombres de pantallas o programas, puede hacer clic en la parte más a la izquierda del registro y marcarlo, para eliminarlo. Posteriormente haga clic en el botón Cerrar en la esquina superior izquierda de la ventana Examinar y teclee PACK en la ventana Comandos. Ahora teclee:

```
USE
SET RESOURCE ON
```

Si hizo algo de lo que se arrepiente terriblemente, se puede restaurar el archivo de recursos entero tecleando lo siguiente:

```
SET RESOURCE OFF
USE \VFP\FOXUSER
ZAP
APPEND FROM RESBACK
USE
SET RESOURCE TO \VFP\FOXUSER
SET RESOURCE ON
```

En el próximo capítulo se explicará qué significan estos comandos. El hecho es que FoxPro utiliza las tablas de FoxPro para almacenar datos, pero también las utiliza para todos sus archivos de sistema. Manipular archivos en FoxPro es especialmente fácil, por lo que sus desarrolladores decidieron que también los usarían para manejar el mismo FoxPro. *Voilà* -otra idea brillante.

El entorno Visual FoxPro

Dado que usted pasará todo el día programando en el entorno Visual FoxPro, probablemente sea una buena idea empezar por familiarizarse con su contexto. Usted se sorprendería de saber cuántos programadores siguen descubriendo herramientas fundamentales meses después de haber empezado a trabajar con un producto nuevo. Esto hará que usted empiece con el pie derecho.

El escritorio de FoxPro

El escritorio de FoxPro es la ventana principal de Visual FoxPro. Se pueden aplicar varios de los comandos que se usan en las ventanas de usuarios al escritorio de FoxPro.

Usted puede modificar el tamaño de la ventana de escritorio. Si hace clic en cualquiera de los bordes laterales o en las esquinas, el cursor del ratón cambiará de forma para indicar la dirección en la que se modificará el tamaño. Por ejemplo, si usted desea que la ventana sea más angosta y más corta en la misma proporción, mueva el puntero del ratón hacia la esquina inferior derecha del escritorio hasta que aparezca una flecha inclinada de noroeste a sudeste, con puntas en ambos extremos, entonces haga clic con el botón primario del ratón (botón izquierdo; si su ratón está configurado para usuarios zurdos, entonces use el derecho), mantenga el botón oprimido y arrastre al ratón hacia arriba y a la izquierda. La ventana irá reduciendo su tamaño continuamente hasta que usted suelte el botón principal del ratón. Si se especifica SET RESOURCE ON, los valores establecidos se guardarán en FOXUSER y aparecerá con la misma configuración la próxima vez que corra FoxPro.

La ventana Comandos es otro componente importante del escritorio. En esta ventana se pueden teclear comandos para que FoxPro los ejecute inmediatamente. A pesar de que se puede tener acceso a la mayoría de los elementos de FoxPro desde el menú, muchas veces es útil simplemente teclear un comando. Si se abren una o más ventanas, se puede usar ALT-N para jalar el menú Ventana del menú de FoxPro y después oprimir V para la ventana Comandos.

La primera vez que ejecute FoxPro, probablemente verá una barra de estado en la parte inferior de la pantalla. Esta área gris despliega información acerca de la situación actual de la tabla en uso: su nombre, el número del registro actual, el número de registros en la tabla, el estado de uso de multiusuarios (ya sea Exclusivo o Registro bloqueado/desbloqueado, dependiendo de si se está utilizando la tabla actual de manera exclusiva o compartida) y tres ventanas de estado para las teclas SOB, NUM y MAY. Generalmente esta barra de estado no es de interés para los programadores, de manera que se puede utilizar SET STATUS OFF para deshacerse de ella. Para que la barra de estado no aparezca en forma predeterminada, incluya en CONFIG.FPW la línea siguiente:

```
STATUS BAR=OFF
```

El menú de FoxPro

La mayoría de las cosas que se hacen en FoxPro empieza con el menú de FoxPro. Inicialmente se presenta como una serie de ocho opciones, llamadas *menús principales*, a lo largo de la parte superior del escritorio: Archivo, Edición, Ver, Herramientas, Programa, Ventana y Ayuda.

Como en otros programas se puede "apuntar y hacer clic" en una selección de los menús desplegables que viven bajo sus respectivos nombres. Para usar el menú, primero debe activarlo, ya sea oprimiendo la tecla F10 o la tecla ALT. Se pueden usar las teclas de flecha izquierda o derecha para resaltar una de las opciones de menú y luego presionar ENTER. Entonces, se pueden usar las teclas de flecha arriba o flecha abajo para desplazarse hasta resaltar la selección que se quiera hacer en el menú (llamada *opción de menú*) y presionar ENTER una vez más para seleccionarlo. Si una opcion de menú tiene opciones adicionales, la punta de una flecha en el lado derecho del menú dará indicios de su existencia; usted puede oprimir la tecla ENTER o la tecla flecha derecha para ver el siguiente nivel.

Las letras subrayadas en los menús de FoxPro son *teclas de acceso rápido,* a las que se puede acceder oprimiendo la tecla ALT y la tecla subrayada, simultáneamente. Por ejemplo, para abrir el menú Archivo, mantenga oprimida la tecla ALT y presione la tecla A. Para los usuarios que se adaptan más al teclado, con frecuencia esto es preferible a oprimir F10 y después usar las teclas de flechas y ENTER para navegar a través del sistema de menús. Cuando se despliega un menú de la barra de menús, se puede observar que cada opción de menú tiene un carácter subrayado. Nuevamente, el carácter subrayado de la opción de menú es

una tecla de acceso rápido mientras esté visible, oprimirla es equivalente a resaltarla y presionar ENTER, sólo que mucho más rápido.

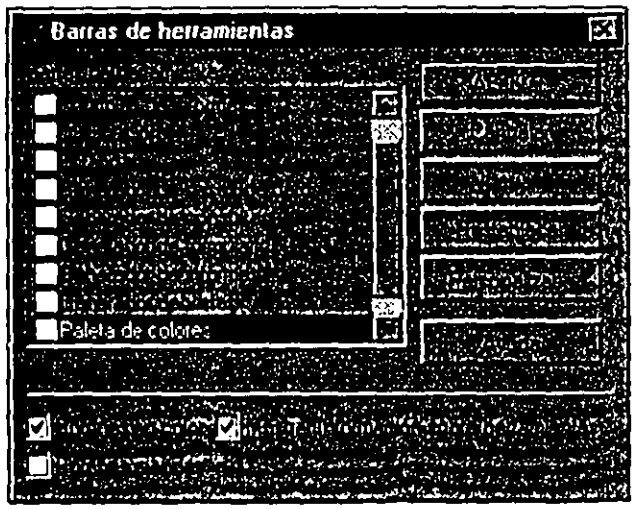
Finalmente, una combinación de teclas aparece a un lado de algunas opciones de menú, cuando éstas se despliegan; por ejemplo, la opción Imprimir en el menú Archivo tiene a su derecha CTRL+P. Esto significa que se puede usar la combinación de teclas de acceso rápido CTRL+P sin necesidad de activar el menú del todo. CTRL+F2, la combinación de teclas de acceso rápido (o método abreviado) que activa la ventana Comandos, es una tecla de acceso rápido de menú comúnmente usada.

Una vez que usted se familiarice tanto con el sistema de menús, como con el del ratón, probablemente preferirá el del ratón.

La barra de herramientas

Justo abajo del menú de FoxPro hay una serie de aproximadamente 20 iconos cuadrados que se extienden a lo largo de la pantalla sobre un fondo gris. Ésta es la barra de herramientas Estándar de FoxPro. También existen otras barras de herramientas. Si se hace clic en el menú Ver, aparece una sola opción, Barras de herramientas. Haga clic en Barras de herramientas y aparecerá la pantalla que se exhibe en la figura 1-6.

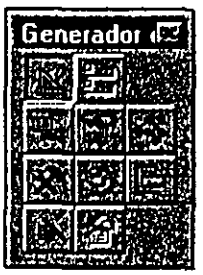
ıra 1-6



El cuadro de diálogo Barras de herramientas.

Se pueden activar una o más barras de herramientas, cada una de las cuales contiene opciones que simplifican el trabajo. Por ejemplo, el diseño de formularios requiere del uso de la barra de herramientas Generador de formularios mostrada en la figura 1-7. Una barra de herramientas adicional, Distribución, lleva a cabo todas aquellas tediosas operaciones de alinear objetos en la pantalla, operaciones que usted odiaba hacer en FoxPro 2.6 para Windows. Hablaremos con detalle de las barras de herramientas en el capítulo 4.

Figura 1-7



Barra de herramientas Generador de formularios.

Cada icono representa alguna función u operación comúnmente usada, algo que usted probablemente seleccionaría de uno de los menús. Para utilizar una selección de barra de herramientas, simplemente haga clic en el icono.

Personalización del entorno

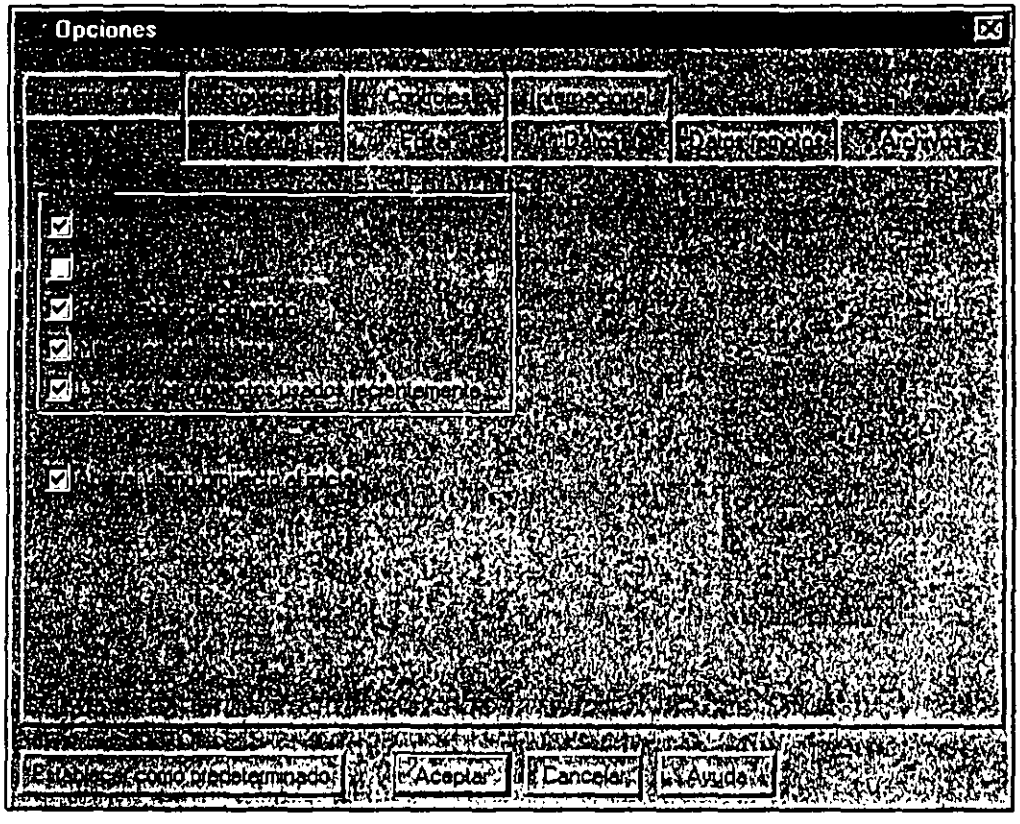
FoxPro se configura con el menú Herramientas. Seleccione Opciones haciendo clic y aparecerá la pantalla de la figura 1-8. El marco de página Opciones consiste en diez "páginas" que se pueden seleccionar haciendo clic en sus fichas. Si se selecciona una ficha en una fila que no es la primera fila, se convierte en la primera fila. Las páginas disponibles son:

Archivos Ubicación de archivos de ayuda y recursos de FoxPro; así como generadores, asistentes y el Generador de menús.

Formularios Líneas de la cuadrícula y el tamaño de los espacios, las opciones de orden de tabulación, el área de diseño máxima, VCX para usarlos como plantillas para formularios y el bloqueo del generador. Cuando el bloqueo del generador está activado las opciones seleccionadas se convierten en predeterminadas para tareas relacionadas, evitando la necesidad de seleccionar la misma opción una y otra vez cuando se requiere realizar la misma operación varias veces.

Proyectos Establece las opciones a usar cuando se invoca el Generador de proyectos.

Figura 1-8



Marco de página Opciones.

Controles Aquí es donde se listan las bibliotecas de clases visuales (VCX: Visual Class Libraries) y los controles OLE.

Internacional Para desarrolladores de fuera de los Estados Unidos o para aquellos que desarrollan aplicaciones que se venden o usan en distintos países, esta pantalla permite cambiar el trato que se da a la moneda, números, fechas y horas.

Ver Fija la barra de estado y el reloj, y permite seleccionar el comportamiento predeterminado durante una sesión de edición y cuando se trabaja con archivos de proyectos.

General Establece diversas opciones de entorno, incluyendo el bip y la respuesta del programa a la tecla ESCAPE (esencial durante el proceso de depuración, pero que necesita desactivarse antes de que los usuarios metan las manos en el software). Esta pantalla contiene muchas de las opciones que antes se manejaban utilizando el comando Set, como Set Carry, Set Safety, Set Bell, Set Compatible y Set Confirm. Aún se puede usar Set, pero esta pantalla organiza las cosas adecuadamente.

Editar Permite fijar el editor de FoxPro de acuerdo con las preferencias personales. Una vez que se establezcan conforme a sus gustos, probablemente no las volverá a cambiar nunca.

Datos Esta pantalla nuevamente contiene muchos de los comandos Set utilizados para controlar el acceso a las tablas. Se hablará de estas opciones otra vez en el capítulo 13, en la sección de consideraciones de multiusuario. Esperamos tener una agradable sorpresa para usted.

Datos remotos Uno de los cambios principales de Visual FoxPro es la incorporación de la tecnología cliente-servidor. La página Datos remotos controla aspectos del acceso a datos remotos que probablemente esté usted viendo por primera vez, a menos que ya haya trabajado con otros lenguajes de programación. Tan extraño como parezca este mundo, Visual FoxPro lo hace lo más sencillo posible. Usted simplemente llene los campos requeridos e inténtelo, si funciona, ya la hizo.

Macros

La opción Macros en el menú Herramientas abre el editor de macros, el cual le permite administrar su propia colección de golpes de tecla memorizados. Volveremos a hablar de esto más adelante, pero créame: las macros pueden eliminar mucho del tedio de la vida de un programador (eso, y un buen juego de bocinas para su CD-ROM -el disco compacto de James Taylor, *Never Die Young*, se tocó alrededor de unas mil veces durante la producción de este manuscrito).

Mucha gente cree que las macros se usan para evitar la agonía de teclear la palabra *append*, dado que es la macro predeterminada de la tecla F9 en la ventana Comandos; pero son mucho, mucho más que eso. Por un lado, las macros permiten hacer cambios repetitivos sin esfuerzo. Si se cambia el nombre de un archivo después de haber diseñado una pantalla con 50 campos en ella y todos ellos tienen el nombre original del archivo dentro del campo Propiedades en el archivo .SCX, una pequeña macro puede hacer los cambios requeridos en alrededor de 60 segundos, incluido el tiempo necesario para diseñar la macro. Una vez que haya experimentado con algunas, ya no podrá trabajar sin ellas.

Detección de problemas

Si se tienen problemas con FoxPro después de haberlo instalado, existen algunos lugares probables para revisar.

El primero es la memoria. La mayoría de los problemas que hemos tenido con Visual FoxPro han consistido en haber cambiado la configuración de la memoria de la máquina de una manera que a FoxPro no le gustó. Revise nuevamente el comentario anterior sobre EMM386.

El segundo problema puede ser el controlador de video que venía con su tarjeta de video. No obstante que Microsoft ha establecido lineamientos de diseño para los fabricantes de controladores de video, son tan complicados que pocas empresas han podido cumplir con todos ellos. Como resultado, algunos programas -no sólo FoxPro- simplemente fallan de manera rutinaria e impredecible en Windows.

La primera solución es llamar al sistema de tableros de boletines electrónicos (BBS: Bulletin Board System) del fabricante de la tarjeta de video y bajar la versión más reciente de sus controladores para Windows y reinstalarlos. No toma mucho tiempo y ellos reciben gran cantidad de retroalimentación cuando sus controladores no funcionan. La caja en la que venía su tarjeta de video pudo haber estado en un estante durante meses y nuevos controladores salen al mercado continuamente.

La segunda solución, que es costosa pero fundamentalmente superior, es utilizar Windows NT. Desde que nosotros lo usamos, no hemos tenido el menor problema del mismo tipo que sospechábamos era causado por un conflicto con los controladores de video. NT también tiene capacidades de red, de tal manera que si está pensando en interconectar sus computadoras, échele un vistazo.

Esto concluye la configuración y visita guiada inicial por Visual FoxPro. Espero que usted esté entusiasmado con este pequeño panorama preliminar. Su ambiente de trabajo está a punto de sufrir una sacudida importante, pero creemos que lo va a disfrutar.

2

Tablas de dato
de FoxPro

CAPITULO 2

EN este capítulo se verá cómo se almacenan los datos en FoxPro. Éste es, después de todo, un ambiente de desarrollo de base de datos. Y al final, hay una sorpresa.

Los datos de Visual FoxPro se almacenan en tablas, pero existe una capa adicional llamada *contenedor de base de datos* (DBC: *Database Container*). El DBC contiene información sobre tablas, índices, relaciones, desencadenantes y algunas otras cosas más. Primero observaremos las tablas, luego las formas en que el contenedor de base de datos las mantiene juntas.

Estructura de un DBF

Los datos de FoxPro se almacenan en forma de tablas, muy parecido a una hoja de cálculo. Cada columna representa un elemento de dato único, como un nombre, una dirección o un número telefónico. Cada fila es un registro, un grupo que contiene un elemento de cada columna en la tabla. Por ejemplo, el registro 1 puede contener el nombre, la dirección y el número telefónico de Jesús Altuve, y el registro 2 puede contener el nombre, la dirección y el número telefónico de María Pérez. No se pueden tener diferentes tipos de registro en la misma tabla. Aceptar el hecho de que datos similares van en la misma tabla, tan simple como parece, es una de las claves más importantes para simplificar la programación de bases de datos. Hablaremos sobre esto más adelante.

Las tablas de datos de FoxPro obedecen las convenciones de nomenclatura del DOS, de manera que los nombres deben ser de uno a ocho caracteres de largo y deben tener la extensión predeterminada .DBF (*database file*: archivo de base de datos). Se pueden usar otras extensiones, pero generalmente no vale la pena la confusión. Además, si se usa la extensión .DBF, no se tendrá que proporcionar la extensión en la mayoría de las operaciones con tablas.

Visual FoxPro le permite asignar nombres gigantescos a las tablas, y todas las referencias internas al archivo reflejarán el nombre que usted le dio. Sin embargo, si se crea una tabla con el nombre MILISTADECLIENTES.DBF, se dará cuenta de que DOS le permite usar únicamente los primeros ocho caracteres, por ejemplo, MILISTAD.DBF. El contenedor de datos resuelve cualquiera de esos mapeos. Por otro lado, Windows NT (y Windows 95) permiten nombres de archivo largos.

Los DBF empiezan con una breve descripción de los datos que están en la tabla. La figura 2-1 contiene un vaciado hexadecimal de un DBF de FoxPro. Empezando en la quinta línea, se puede ver un patrón regular de nombres de elementos de

datos: NOMBRE, DIRECCION y así sucesivamente. Ésa es la descripción de la tabla. Los bytes precedentes indican a FoxPro qué tan largo es el encabezado, cuántos registros hay en la tabla y algunos otros bits de información importante. La estructura lógica de una tabla de FoxPro es, tal vez, la razón por la que xBASE se popularizó en un principio. Dado que la parte superior de la tabla de datos contiene su descripción, cada vez que un programa abre una tabla, sabe qué campos contiene la misma. En la mayoría de los otros lenguajes, se tiene que incluir una estructura para cada tabla en el código fuente de cada programa y volver a compilar el programa (traducir a lenguaje de máquina) si cambia la estructura de la tabla. (Esto significa, por supuesto, que usted debe estar al tanto de la estructura de los datos, algo que también es cierto para xBASE, pero menos importante.) No sucede así con FoxPro. Ni siquiera le importa si usted cambia la estructura de las tablas, mientras que los campos mencionados en los programas permanezcan en la tabla de datos cuando se ejecute el programa.

```
00000000:  30 61 05 0C 00 00 00 00 - 88 01 1F 00 00 00 00 00 0a♣♀....ê.▼.....
00000010:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 03 00 00 ................
00000020:  4E 4F 4D 42 52 45 00 00 - 00 00 00 43 01 00 00 00 NOMBRE.....C....
00000030:  0A 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000040:  44 49 52 45 43 43 49 4F - 4E 00 00 43 0B 00 00 00 DIRECCION..C♂...
00000050:  0A 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000060:  54 45 4C 45 46 4F 4E 4F - 00 00 00 43 15 00 00 00 TELEFONO...C§...
00000070:  0A 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000080:  0D 64 3A 5C 64 6F 63 5C - 70 65 72 73 6F 6E 61 6C ♪d:\doc\personal
00000090:  5C 74 72 61 64 75 63 5C - 65 6A 65 72 63 69 6F 5C \traduc\ejercio\
000000A0:  70 69 6E 74 65 72 31 2E - 64 62 63 00 00 00 00 00 pinter1.dbc.....
000000B0:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000000C0:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000000D0:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000000E0:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000000F0:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000100:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000110:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000120:  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
```

Figura 2-1

Una vista hexadecimal de un encabezado DBF.

Tipos de campos

Las tablas de FoxPro consisten en campos. Los campos son análogos a los encabezados de columnas en un formato de hoja de cálculo. Cada columna tiene un nombre, llamado *nombre de campo*, y a cada columna se le conoce como *campo*.

En versiones previas de FoxPro, los nombres de los campos podían tener hasta diez caracteres, pero en Visual FoxPro, pueden ser gigantescos -hasta 254 caracteres. Sin embargo, no se entusiasme demasiado, porque no se pueden usar espacios en ningún lado dentro de un nombre de campo de FoxPro; así que probablemente muy rara vez irá más allá del viejo límite. Pero es el principio el que cuenta.

Los nombres de campos no pueden contener caracteres especiales, fuera del carácter "guión bajo", y tampoco pueden empezar con un número. Muchas personas solían agregar un prefijo de dos caracteres, derivado del nombre del DBF, como recordatorio del origen de la tabla y para evitar sobrescribir copias de los datos en variables de memoria, pero eso es parte del pasado. Los campos pueden ser cualesquiera de varios tipos:

Carácter Cualquier cosa que se pueda teclear en el teclado puede ir en un campo de caracteres, así como caracteres que no se ven en el teclado. Un carácter es lo mismo que un byte y un byte puede tener cualquier valor ASCII hasta 255, con un par de excepciones.

Numérico Éste puede ser de hasta 18 dígitos, con tantas cifras decimales como se desee. Los enteros se almacenan en este formato y un nuevo tipo de moneda se usa generalmente para dinero.

Flotante Este formato, incluido por compatibilidad con dBASE V, es equivalente al formato numérico.

Fecha FoxPro almacena fechas en un formato interno numérico, pero su representación y validación se controlan con este tipo de campo. Vea Fecha-hora, más adelante, para información más precisa sobre el control de la fecha y la hora.

Lógico Éste, ya sea verdadero o falso, es representado por .T. o .F., para verdadero o falso respectivamente (no olvide los puntos).

Memo Representado en su estructura de registros como un campo de caracteres de longitud 10. Estos campos de hecho se almacenan en una segunda tabla con la

extensión .FPT. Pueden ser casi de cualquier tamaño. Los campos memo, en su forma usual, se usan para almacenar texto, pero no están limitados a texto.

General Éstos también son campos memo, excepto que usualmente almacenan otros tipos de objetos además de texto: gráficos .BMP o .PCX, por ejemplo.

También hay algunos tipos de campos nuevos:

Fecha-hora Incluye la hora después de la fecha, en el formato {MM/DD/AA hh:mm:ss}.

Moneda Incluye un componente fijo de cuatro cifras decimales.

Imagen Usado específicamente para almacenar imágenes.

Carácter binario Campos de caracteres no sujetos a traducciones de página de códigos (como SET NOCPTRANS TO listadecampos).

Memo binario Campos memo con la misma característica NOCPTRANS.

Esta lista puede crecer en el futuro dado que hay varias fuentes de tipos de datos de estándares de la industria, pero esto debe ser suficiente por el momento.

Abra y cierre tablas

FoxPro tiene 32,767 espacios de trabajo, llamados *áreas de trabajo* porque una tabla debe ser seleccionada para volverla la tabla actual. Cuando se inicia FoxPro por primera vez, el área de trabajo 1 está seleccionada por omisión. Para abrir un DBF en el área de trabajo 1 cuando ninguna otra tabla está abierta, teclee:

```
USE nombredetabla
```

No tiene que estar en un área de trabajo para abrir una tabla. Puede especificar:

```
USE nombredetabla IN 3
```

sin embargo, con más frecuencia usted especificaría:

```
USE nombredetabla IN 0
```

lo que significa "encuentra el área de trabajo siguiente en la que no se encuentre abierta una tabla y abre nombredetabla ahí". Si se ejecuta el comando:

```
SELECT 0
USE nombredetabla
```

FoxPro buscará la tabla en el directorio actual, que es generalmente en el que estaba al arrancar Visual FoxPro. Se puede ver cuál es el directorio actual, tecleando:

```
? SYS(2003)  o ?CURDIR()
```

o abriendo la ventana Depuración mediante las teclas ALT-H, D (Herramientas, Ventana Depuración) y tecleando SYS(2003) en el panel de la izquierda. FoxPro también buscará en cualquier directorio que haya nombrado en SET PATH. Abre la primera instancia que encuentre de la tabla nombrada.

Áreas de trabajo seleccionadas

Cuando usted utiliza una tabla, FoxPro le asigna un *alias*, un nombre de hasta 10 caracteres con el cual se refiere a ella en el programa. Generalmente el alias es el mismo que el nombre de la tabla; por ejemplo, si tiene una tabla llamada Cliente, el alias también será Cliente. Si tiene varias tablas abiertas y quiere trabajar con la tabla Cliente, especifique:

```
SELECT CLIENTE
```

Si se necesita referir al campo Nombre de la tabla Cliente, llámelo Cliente.Nombre. Este asunto de los alias es muy importante. Si por alguna razón tiene un alias diferente del nombre de la tabla, debe usar el alias para referirse a los campos de datos; por ejemplo, el siguiente código es correcto:

```
USE CLIENTE IN 0 ALIAS COMPRADORES
? COMPRADORES.Nombre
```

Por otro lado, lo siguiente produciría una queja de FoxPro:

```
USE CLIENTE IN 0 ALIAS COMPRADORES
? CLIENTE.Nombre
```

FoxPro no podrá encontrar Cliente.Nombre porque, con esta sintaxis, no sabe que el alias de Compradores es el mismo que el de la tabla Cliente. El nombre de la tabla es irrelevante para FoxPro, lo único que le importa es el alias.

De hecho, esto puede ser muy útil para hacer que el código sea genérico. Considere el siguiente código para crear el nombre de una tabla de un prefijo predeterminado y la abreviación de tres caracteres del mes en curso:

```
Nombredetabla = "ULTIMO" + LEFT (CMONTH(DATE()),3)
USE ( Nombredetabla ) ALIAS MENSUAL
```

Esto encontrará la tabla llamada ULTIMOMAY si la función Date( ) obtuvo una fecha de mayo, o ULTIMOSEP, si se corrió en septiembre. De tal manera que si usted quisiera correr un informe usando la tabla de respaldo de este mes, podría referirse a todos los campos en el informe como Mensual.nombredelcampo y no le importaría qué nombre de tabla hubiera abierto. Así que el concepto de alias es bastante útil.

FoxPro tiene varias funciones que le permiten a uno trabajar con tablas. Si quiere abrir la tabla Cliente, en caso de que no esté abierta ya, use esto:

```
IF NOT USED ( "CLIENTE" )
    USE CLIENTE IN 0
ENDIF
```

Éste puede ser un buen momento para hablar de la sintaxis de FoxPro. Estas funciones de tablas requieren un nombre de tabla como *argumento*, el valor que usted pasa a la función. Cuando se refiera a un nombre de tabla, puede escribirlo como parte del código usando delimitadores (ya sea comillas simples, dobles o paréntesis cuadrados) o puede almacenar el nombre en una variable de caracteres. Las expresiones siguientes son equivalentes:

```
Nombre = "Cliente"
IF USED ( Nombre )      && sin delimitadores
    ...
ENDIF
```

y

```
IF USED ( "Cliente" )
    ...
ENDIF
```

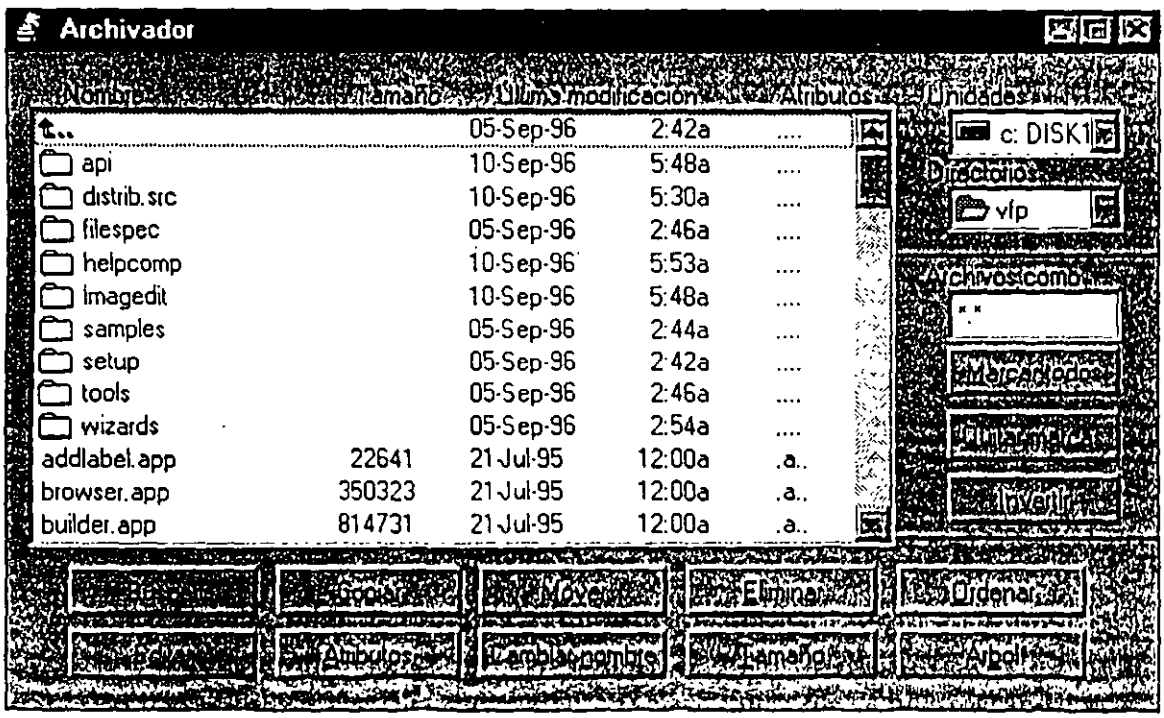
Por lo general es preferible utilizar nombres de variables, dado que uno nunca sabe cuando querrá cambiar algo y es mucho más sencillo cambiarlo una sola vez, al principio del programa, que hacer una búsqueda global y reemplazar las cadenas de caracteres. También, como en el caso del directorio mencionado previamente, puede simplificar generalizaciones que de otra manera serían complicadas. Si prepara un proyecto correctamente, puede tomar más tiempo que sus clientes le describan lo que quieren, que realizarlo.

Cree una tabla de FoxPro

Pongamos en uso parte de esta información creando una tabla. Primero cree un directorio para almacenar su trabajo. Esto se puede hacer ya sea usando comandos similares a sus contrapartes en DOS (por ejemplo, SET DEFA TO C:\, MKDIR MITRAB, CD MITRAB, MD CAP02), o bien, usando el navegador visual de directorios incluido en FoxPro, cuyo nombre es Archivador. Teclee FILER en la ventana Comandos para obtener la pantalla de la figura 2-2. Entre otras cosas, el Archivador le informará dónde se encuentra. Probablemente está en la unidad C, en el subdirectorio VFP. El Archivador tiene dos vistas, la que se muestra y la vista de árbol. Para crear un nuevo subdirectorio, debe estar en la vista de árbol, así que haga clic en el botón Árbol, en el extremo inferior derecho de la pantalla del Archivador.

Cuando la pantalla se refresque, aparecerá un desplegado como el de la figura 2-3 y la primera línea -el directorio raíz C:- estará seleccionada. Realmente no está seleccionado hasta que no se resalte con un fondo azul, así que haga clic en él y después haga clic en Crear directorio. El cuadro de diálogo de la figura 2-4 le preguntará qué nombre desea utilizar; teclee pinter o lo que usted elija y presione

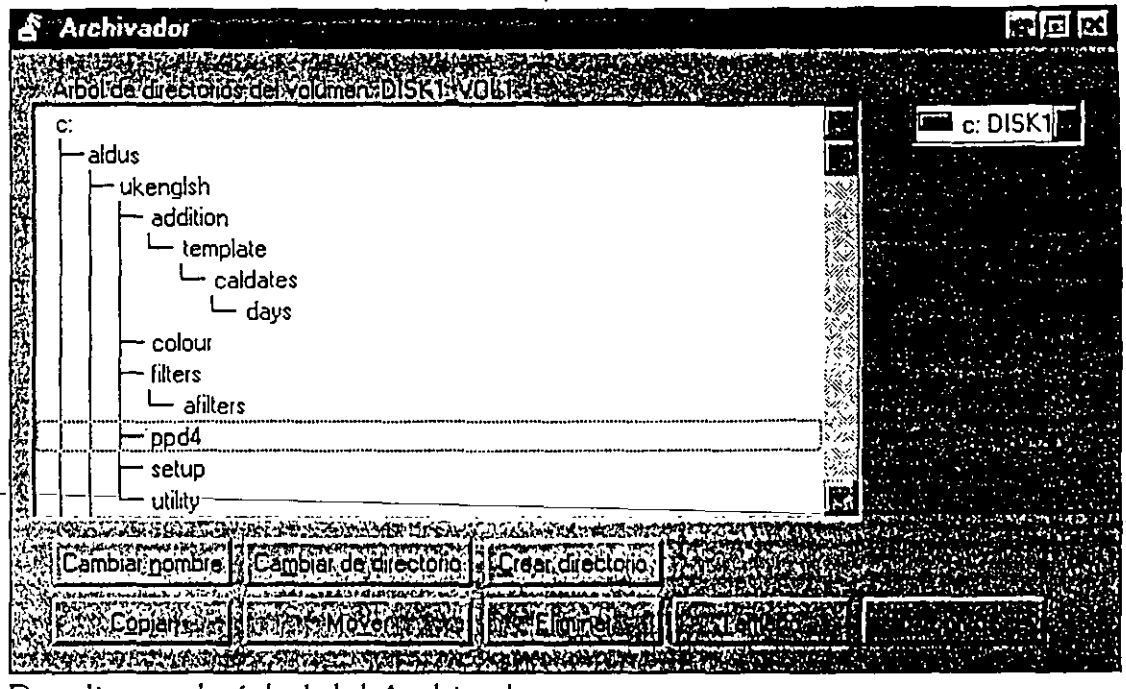
Figura 2-2



Ventana Archivador.

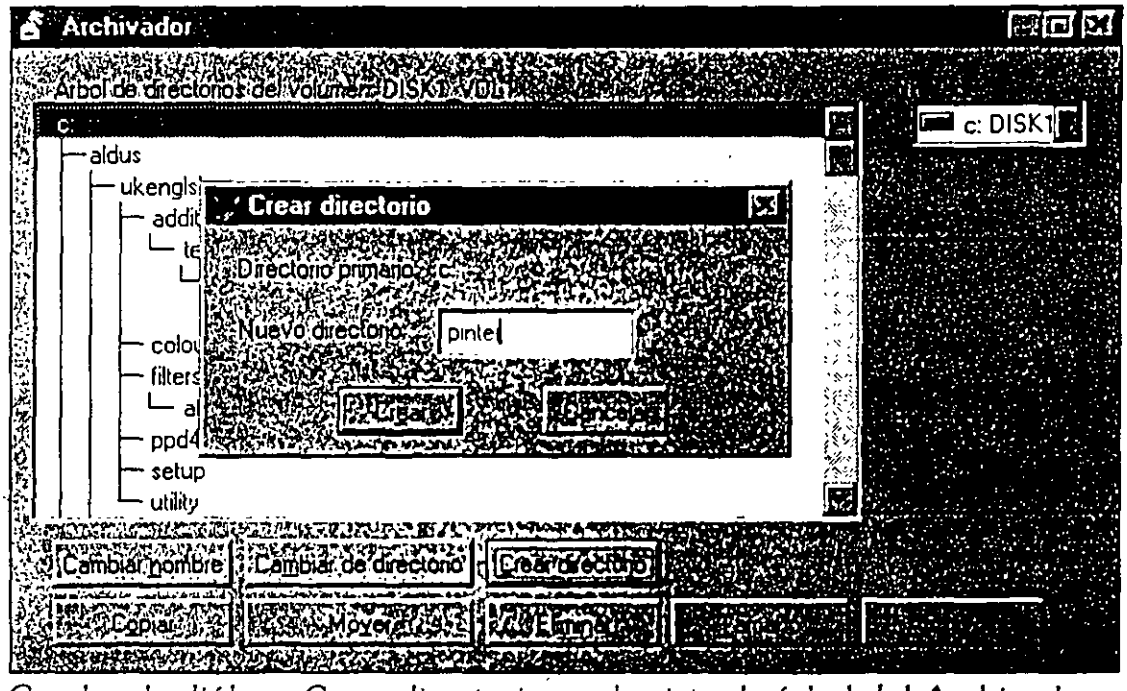
ENTER. Si quiere crear un directorio para cada cosa en este libro, entonces use nombres como CAP02, CAP03 como recordatorio de dónde estaba usted trabajando.

Figura 2-3



Despliegue de árbol del Archivador.

Figura 2-4



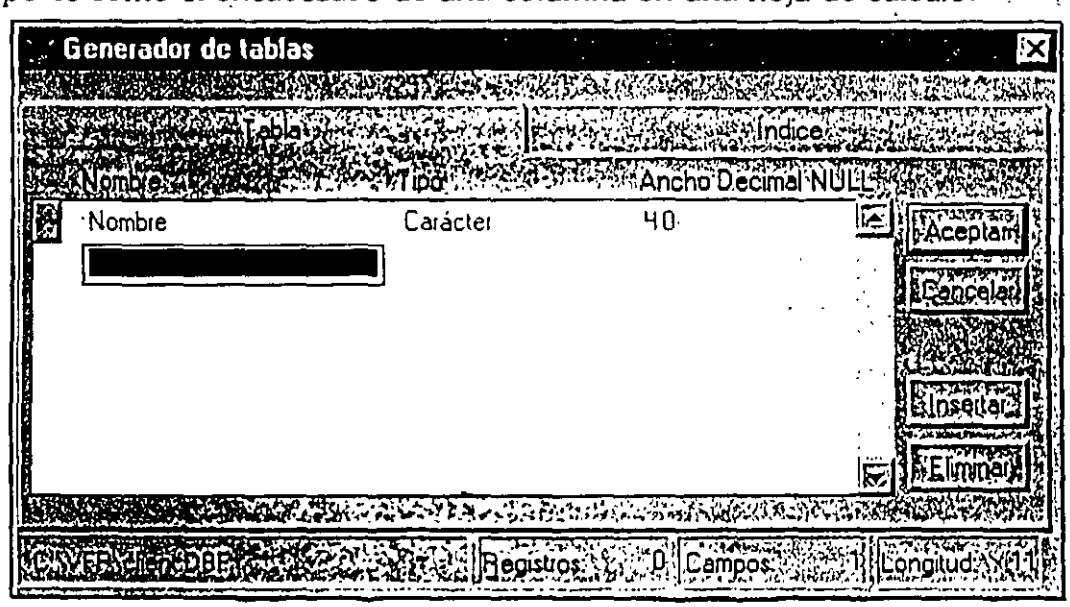
Cuadro de diálogo Crear directorio, en la vista de árbol del Archivador.

Ahora está listo para crear una tabla. Aunque el contenedor de datos permite usar nombres de tablas largos, hacerlo puede acarrearle algunos problemas, de manera que hay que usar nombres de tablas que se apeguen a las convenciones de DOS. Por ejemplo, use un nombre que empiece con una letra, que no tenga más de ocho caracteres y que no incluya espacios o caracteres especiales. De hecho, puede recurrir a ciertos caracteres especiales, pero una vez yo usé un programa que creó en mi disco duro un archivo que no tenía un nombre válido para DOS y literalmente tuve que reformatear mi disco duro para deshacerme de él. El software es más inteligente hoy en día, pero el conjunto de caracteres alfanuméricos ofrece 36 opciones diferentes a la octava potencia (alrededor de 282,000,000,000), lo cual debe ser suficiente para satisfacer sus necesidades. De hecho, usemos CLIENTE.

Para crear una tabla, usted puede teclear CREATE CLIENTE o bien, usar el menú Archivo y seleccionar Nuevo, Tabla. Si emplea el método del menú, FoxPro le permitirá usar el Asistente para tablas, para elegir una tabla de muestra que tenga los campos que usted probablemente necesitará. Pero hagámoslo de la manera difícil.

Cuando usted vea la pantalla del Generador de tablas, que se muestra en la figura 2-5, estará listo para introducir información de campos. Recuerde que cada campo es como el encabezado de una columna en una hoja de cálculo.

Figura 2-5



Pantalla del Generador de tablas.

Así que cree una tabla CLIENTE, CLIENTE.DBF. Introduzca los siguientes nombres de campos, tipos y tamaños. (La tecla TAB y la combinación de teclas

MAYÚS-TAB se usa para moverse hacia adelante y hacia atrás respectivamente, y las teclas de flecha hacia arriba y hacia abajo, para moverse hacia arriba y hacia abajo. ENTER termina el proceso de definición de la tabla.)

NOMBRE	Carácter	40
DIRECCION1	Carácter	40
DIRECCION2	Carácter	40
CIUDAD	Carácter	20
ESTADO	Carácter	3
CODPOST	Carácter	10
SALDO	Monetario	8 (suministrado por programa)

Cuando usted oprime ENTER para completar el proceso de definición de la tabla, FoxPro le pregunta ¿Desea introducir registros de datos ahora? Conteste que no porque usted tendrá que aprender acerca de esto tarde o temprano.

Si teclea LIST STRUCTURE, verá los resultados de su trabajo, pero si cuenta las longitudes de todos los campos se quedará corto por uno. El byte faltante se llama el *byte de borrar* (delete). Si usted teclea DELETE NEXT 1 o presiona CTRL-T mientras está examinando, FoxPro marcará el registro a eliminar. Llámelo nuevamente usando CTRL-T, o teclee RECALL NEXT 1 y regresará. El comando PACK o la opción Quitar registros eliminados, del menú Tabla, desecharán permanentemente los registros marcados para su eliminación.

Ahora teclee BROWSE. (Esto se analizará con más detalle después, por lo pronto considérelo como introducir datos de prueba.) Verá una tabla vacía con los nombres de columna que usted definió. Si usted ya está familiarizado con xBASE, probablemente ya oprimió CTRL-N. ¿Por qué le da todas las opciones *excepto* agregar un registro? y ¿dónde está el registro en blanco que se supone se agrega al final de la tabla cuando se presiona CTRL-N?

Bueno, las cosas cambian. CTRL-N es ahora lo mismo que seleccionar Archivo, Nuevo de la barra de menús de FoxPro y probablemente tiene más sentido, aun en la traducción al ruso se teclea como se escribe. Úselo para agregar nuevos proyectos, tablas, consultas, informes -cualquier cosa excepto nuevos registros. Utilice CTRL-Y, que parece no tener sentido, para añadir un registro. Si no desea oprimir CTRL-Y para introducir los registros, puede activar el modo de adición utilizando la opción Modo Añadir, del menú Ver. También puede oprimir ALT-V, y luego la M. En el Modo Añadir, usted agrega un nuevo registro cada vez que intenta moverse hacia el final de la tabla, con la tecla de desplazamiento de flecha abajo.

Teclee unos cuantos registros con valores razonables en ellos, de tal manera que pueda probar algunos de los comandos de la próxima sección. En particular introduzca al menos un registro con DF en el campo estado y otro con Df (D mayúscula, f minúscula), o con cualquier otro estado. Los necesitará en un ejercicio posterior.

Comandos de tablas List, Browse y Display

Otra razón de la temprana popularidad de xBASE fue la existencia de comandos a nivel tabla que facilitaron el acceso a la información. FoxPro le facilita llegar a sus datos. Hay varios comandos que rara vez se utilizan en los programas, pero son usados constantemente por los programadores para verificar que funcionen sus programas.

List es muy fácil de usar. Si usted tiene una tabla abierta en el área de trabajo actual, puede teclear LIST para observar todos los 300,000 registros en una lista continua. Éste es un buen momento para hablarle sobre el comando Set Escape On, el cual le permite oprimir la tecla Esc para detener comandos como List. Los programadores siempre deben especificar SET ESCAPE ON antes de comenzar a trabajar. Asegúrese de desactivarlo en su APP de producción.

List tiene una sintaxis más compleja, que de alguna manera le proporciona mejor control sobre sus resultados. La sintaxis exacta es:

```
LIST
      ¦ FIELDS ListaCampos
      ¦ TO PRINT
      ¦ WHILE ExpresionDominio
      ¦ FOR ExpresionDominio
      ¦ OFF
```

Si desea mostrar sólo ciertos campos, listelos después de la palabra clave Fields. To Print envía la salida directamente a la impresora. For, seguido de una expresión de dominio, lista sólo aquellos registros que coincidan con la expresión, por ejemplo, List For Estado = [DF]. While realiza lo mismo, excepto que se detiene la primera vez que se topa con un registro que no coincide con la expresión de dominio. Por ejemplo, si su tabla está ordenada por estados, List For Estado = [DF] leería la tabla entera para listar las entradas del Distrito Federal, mientras que List While Estado = [DF], si usted comenzó con la primera entrada de Distrito Federal, leería sólo la porción del Distrito Federal en la tabla. Esto puede ser *mucho* más rápido. Y la sintaxis de la expresión de dominio While se utiliza para

optimizar muchos aspectos del desempeño de FoxPro. Off desactiva la opción de numeración de registros. Si no desea ver los encabezados de campos, especifique SET HEADING OFF antes de que utilice el comando List.

Si List fue un poco más allá o le proporcionó más registros que una simple pantalla, probablemente prefiera usar Display. Este comando tiene exactamente la misma sintaxis que el comando List, con la excepción de que hace una pausa después de cada página y espera a que usted oprima ENTER. Advierta que si no especificó SET ESCAPE ON, Display esperará a que usted oprima ENTER y luego desplegará otra pantalla de datos, después otra, y otra...

Cómo funcionan las expresiones de dominio

¿Recuerda cómo introdujo varios registros con un código de Estado DF y otros con Df? Note cómo cada uno de los siguientes comandos produce diferentes resultados:

```
LIST FOR ESTADO = "DF"
```

Usted debería ver sólo los registros cuyo campo Estado contenga precisamente las letras mayúsculas D y F. El siguiente:

```
LIST FOR ESTADO = "Df"
```

sólo listaría los registros que tuvieran una D mayúscula y una f minúscula en el campo Estado. Este comando:

```
LIST FOR UPPER(ESTADO) = "DF"
```

listará todas las entradas para el Distrito Federal, debido a que convierte el contenido del campo Estado a mayúsculas, antes de hacer la comparación. DF, Df, dF y df funcionarían. Este otro:

```
LIST FOR LOWER(ESTADO) = "df"
```

produciría exactamente el mismo resultado que la condición de dominio principal, debido a que la conversión se realiza antes de la comparación. Y éste:

```
LIST FOR ESTADO        = "    DF"  && espacios antes de DF
```

no listaría ningún registro. FoxPro comienza la comparación desde el primer carácter de la expresión o cadena en el lado derecho del signo igual que, y se

detiene cuando se compara con todos los caracteres que contiene la expresión al lado derecho del signo igual que. El comando siguiente:

```
LIST FOR ESTADO= "D"
```

listaría Distrito Federal y cualquier estado que comezara con D mayúscula. La comparación se detiene después de la primera letra, debido a que la cadena contiene sólo un carácter. Esta variación interesante:

```
LIST FOR "F" $ ESTADO
```

encuentra todos los registros que contienen una letra F mayúscula en el campo Estado, sin importar la posición de la letra dentro del campo. Y éste:

```
LIST FOR "F" $ UPPER(ESTADO)
```

convierte el contenido del campo Estado a mayúsculas antes de realizar la búsqueda dentro de él, así que se listan DF y Df.

Expresiones de dominio más complejas

Usted puede mezclar y coincidir criterios. Por ejemplo, Upper(Estado) = [DF] AND Saldo > 100 listaría todos los clientes del Distrito Federal cuyo saldo excede $100.00. Lo único que debe mantener en mente es que algunos tipos de criterios pueden ser más lentos cuando usted tiene tablas de datos grandes. Afortunadamente hay una forma de controlar esto. La muy mencionada tecnología Rushmore que permite acelerar en extremo algunas operaciones de tablas en FoxPro es muy previsible y se analizará con más detalle cuando hablemos de índices.

Utilice Browse con un dominio

Browse es la herramienta fundamental de bases de datos del programador. Hemos empleado muchos años tratando de usar Browse, pero ahora hay una nueva herramienta llamada control de cuadrícula. Pero Browse es muy, muy práctico para visualizar una tabla o incluso una combinación de campos de varias tablas.

Browse tiene una sintaxis increíblemente rica. He aquí la lista completa de parámetros opcionales:

```
BROWSE
        [FIELDS ListaCampos]
        [FONT cNombreFuente [, nTamañoFuente]]
        [STYLE cEstiloFuente]
        [FOR lExpresion1 [REST]]
        [FORMAT]
        [FREEZE NombreCampo]
        [KEY eExpresion1 [, eExpresion2]]
        [LAST | NOINIT]
        [LOCK nNumeroCampos]
        [LPARTITION]
        [NAME NombreObjeto]
        [NOAPPEND]
        [NODELETE]
        [NOEDIT | NOMODIFY]
        [NOLGRID] [NORGRID]
        [NOLINK]
        [NOMENU]
        [NOOPTIMIZE]
        [NOREFRESH]
        [NORMAL]
        [NOWAIT]
        [PARTITION nNumeroColumna [LEDIT] [REDIT]]
        [PREFERENCE NombrePreferencia]
        [SAVE]
        [TIMEOUT nSegundos]
        [TITLE cTextoTitulo]
        [VALID [:F] lExpresion2 [ERROR cTextoMensaje]]
        [WHEN lExpresion3]
        [WIDTH nAnchoCampo]
        [[WINDOW NombreVentana1]
        [IN [WINDOW] NombreVentana2
        | IN SCREEN]]
        [COLOR SCHEME nNumeroEsquema
        | COLOR ListaParesColores]
```

Los parámetros se listan aquí para satisfacer su apetito. Éstos se describen completamente en el sistema de ayuda de FoxPro y en la documentación de referencia del lenguaje.

Mientras Browse está activo, usted puede usar el ratón para modificar el tamaño de columnas individuales. Incluso reduciéndolas a nada. Puede reacomodar las columnas y modificar el tamaño de la ventana Examinar a la derecha o hacia abajo y moverla alrededor de la pantalla. Puede minimizar la ventana Examinar y colocarla al lado de la pantalla, fuera de su camino, hasta que necesite verla otra vez. Hasta que se cierra la ventana, ésta retiene su conexión con la tabla o las tablas que son el objeto de la examinación con Browse.

Cuando lleguemos a Set Relation, le mostraremos unos cuantos trucos más. Pero piense en Browse como una herramienta del programador y un menú de datos ocasional.

Cálculos de tablas Sum, Count, Total, etcétera

Similares a los comandos List y Display son aquellos que operan en la tabla entera. La expresión siguiente:

```
SUM Cantidad
```

desplegará la suma del campo Cantidad en la tabla actual, si TALK está activo. En los programas, usted utilizará comúnmente la sintaxis alterna:

```
SUM Cantidad TO CantSum
```

Para ver el valor colocado en la variable de memoria CantSum, puede teclear el signo "?" seguido del nombre de la variable de memoria, como sigue:

```
? CantSum
```

Sin embargo, usted utilizará la ventana Depuración para examinar el contenido de las variables de memoria o de los campos de tablas. Si oprime ALT-H (para Herramientas) y selecciona D para Depuración en el menú desplegable, puede teclear el nombre CantSum y examinar su valor. También puede utilizar una expresión de dominio:

```
SUM Cantidad TO CantSum FOR ESTADO = [DF]
```

COUNT TO NumRegs FOR ESTADO = [DF] trabaja en forma análoga. Total es un poco diferente, produce un registro con la misma estructura de la tabla actual, pero con campos numéricos totalizados en sus nombres de campo respectivos. Total se menciona para completar y puede ser útil cuando se realizan pruebas, pero el comando SQL Select, el cual se cubrirá con detalle en el capítulo 11, es una mejor forma de realizar lo mismo.

Set Filter

Las expresiones de dominio descritas con los comandos List, Display y Browse son un caso especial de los filtros de FoxPro. El comando:

```
SET FILTER TO expresion
```

provoca que FoxPro se comporte como si no existieran los registros que no coinciden con la expresión del filtro. Esto significa que usted puede especificar SET FILTER TO ESTADO = [DF], por ejemplo, ejecutar un informe que normalmente mostraría cada registro en la tabla, y ver sólo datos del Distrito Federal.

La filtración es un medio muy poderoso para analizar el contenido de una tabla. También simplifica infinitamente la programación, ya que usted puede simplemente dar a los usuarios la capacidad de establecer un filtro y luego ejecutar sus pantallas e informes normales con el filtro activo. En algunas aplicaciones, esta simple capacidad es la herramienta más importante de un usuario individual. No tiene que ser elaborada para ser justo lo que usted necesita.

Cómo moverse en una tabla de FoxPro

Cuando utiliza inicialmente una tabla de FoxPro, usted está ubicado en el primer registro de la tabla. Puede mover este puntero de registro imaginario alrededor de la tabla, en diversas formas:

Skip Mueve el puntero hacia adelante un registro.

Skip -1 Mueve el puntero hacia atrás un registro.

Go Top Mueve el puntero al primer registro en la tabla.

Go Bottom Mueve el puntero al último registro en la tabla.

Go 3 Mueve el puntero del registro al registro número 3.

Go (NumRegistro) Mueve el puntero al registro cuyo número sea el mismo contenido en la variable NumRegistro.

Existe también una cantidad de mecanismos de ciclos que puede utilizar para moverse a través de una tabla. El principal es Scan:

```
SCAN
        ... comandos...
ENDSCAN
```

Las líneas de código entre Scan y EndScan se ejecutan una vez por cada registro en la tabla. Usted puede utilizar Scan para moverse a través de una tabla buscando varias condiciones, imprimiendo registros, subtotalizando o introduciendo información en una tabla. También puede alterar y controlar el movimiento del puntero de registro en un ciclo Scan/EndScan de tres maneras:

- Scan seguido ya sea por For expresión o While expresión, de tal manera que el procesamiento termine cuando la expresión ya no se satisfaga. Scan Rest For expresión aplica la expresión de dominio sólo a los registros seguido de la ubicación actual del puntero de registro. Scan While expresión se detiene tan pronto como la condición ya no se satisfaga. La diferencia es sutil, pero cada uno tiene su propio uso.
- Si se ejecuta el comando Loop, el control regresa a Scan. Loop detiene el procesamiento y salta inmediatamente al siguiente registro. El control regresa al comando Scan. Si la expresión de dominio seguida por Scan For o Scan While evalúa en .F., el control pasa a la primera línea de código que sigue a EndScan. Si desea saltar ciertos registros pero la situación no encaja convenientemente en una expresión de dominio, utilice éste.
- Finalmente, use Exit para saltar a la primera línea de código seguida del comando EndScan.

¿Por qué necesita éstas tres formas? Porque existen situaciones en que cada una de éstas es la herramienta correcta. Exit le permite controlar el movimiento preciso del puntero de registro sin importar el tipo de mecanismo que usted utilice. Básicamente, si desea continuar moviéndose a través de la tabla, utilice Loop. Si quiere dejar de leer la tabla, utilice Exit.

Si quisiera poner en una tabla los nombres y teléfonos de las primeras 100 personas que no son de cierto estado; por ejemplo, las 100 primeras personas de Carolina del Norte o Carolina del Sur, quienes dieron a Jim y Tim Faye[1] al menos 100 dólares.

[1] Jim y Tammy Faye Bakker fueron dos criminales que robaron a la gente que creyó en ellos, usando un esquema de pirámide.

```
FUNCTION empezar
I = 1
DIMENSION Tontos(100)
SCAN FOR Cantidad >= 100.00
        IF ESTADO = [CN] OR ESTADO = [CS]
                LOOP
        ENDIF
        I = I + 1
        Tontos (I) = Nombre + [ ] + Telefono
        IF I = 100
                EXIT
        ENDIF
ENDSCAN
...
=empezar()
```

Las cosas son lo que son, probablemente me demande alguien que fue estafado por otro alguien que estaba usando este software. El software no roba a la gente, la gente roba a la gente a menudo utilizando software, desafortunadamente.

Índices

Las tablas nunca están en el orden en el que usted las quiere, y muchas tablas se utilizan mejor en muchas formas distintas. Por ejemplo, puede trabajar en listas de nombres en orden alfabético por apellido. En otra ocasión usted puede querer ir directo al primer nombre de Venezuela y pasar justo a través del último venezolano en la lista, sin mirar a ningún otro nombre.

Por suerte, FoxPro facilita este trabajo. Usted puede crear un índice en uno o más campos, o combinaciones de campos, y luego alternar entre ellos. Técnicamente, hay cuatro formas de crear un índice. En FoxPro 2.6, usted podía utilizar el comando:

```
INDEX ON ESTADO TO ESTADO COMPACT
```

el cual podría crear un archivo de índice llamado ESTADO.IDX. (Compact crea una estructura interna muy eficiente y es mucho más riguroso si utiliza el formato IDX.) Desafortunadamente, si alguna vez abre la tabla sin el índice y añade un registro, los punteros en el archivo de índice se deslizarán hacia abajo, y el índice tendrá que ser recreado. FoxPro trabaja mejor si se usa la sintaxis alterna siguiente:

```
INDEX ON ESTADO TAG ESTADO
```

lo cual no sólo crea automáticamente un índice compacto (los índices compactos que contienen cada campo individual en una tabla miden alrededor de la mitad del tamaño de la tabla original) sino que coloca cada etiqueta de índice en una tabla individual con la extensión .CDX (para índices compactos).

Si usted insiste, puede agregar el comando CDX NOMBRE DE ARCHIVO al final de su expresión de índice (como en INDEX ON ESTADO TAG ESTADO OF MICDX) y crear un archivo CDX con un nombre diferente al de su tabla de datos, pero no lo recomendamos. Si la tabla y el CDX tienen el mismo nombre, el CDX se conoce como CDX estructural y se abre automáticamente siempre que se abra la tabla.

Así que usted no tiene ni siquiera que recordarlo. De hecho FoxPro asigna un byte de bandera en el encabezado de la tabla para indicar que tiene un CDX estructural, mientras que no se incluye tal recordatorio del CDX relacionado si los nombres no concuerdan. Tal vez pueda crear un CDX con un nombre diferente para crear informes *adicionales*, luego destrúyalo después del informe; éste es el uso que tenemos para un CDX no estructural.

Los índices crean un archivo binario de los números de registro asociados con las claves de registro. Si usted crea un índice como éste:

```
INDEX ON UPPER (LEFT(Apellido,10) + LEFT(Nombre,10)) TO NOMBRE
```

FoxPro insertará 20 bytes más un número de registro para cada documento en la tabla, en un formato compacto y cifrado, alfabetizado por los valores del dato clave Apellido/Nombre. Cuando usted busca una clave (SEEK es el comando de FoxPro que se usa para buscar un registro o documento que tiene una clave de importancia particular), FoxPro busca en la primera clave del índice y después en la última. Si no está en ninguna de las dos busca en la clave intermedia. Así que si hay mil claves en el índice, buscará en la clave número 501. Si no concuerdan y el valor de la clave es menor que la clave 501, buscará en el registro 251; si el valor de la clave que está buscando es mayor que el valor de la clave 501, entonces busca en la clave 751. FoxPro continúa "dividiendo la diferencia" hasta que ambas se igualan o no puede subdividir más el contador de clave. Si usted es un matemático natural puede ver que el número máximo de búsquedas en una tabla de 1000 registros es el valor más grande de n, por el cual $2n$ excede a 1000, o 10 en este caso. Así que en lugar de tener que ver 1000 registros, FoxPro sólo tiene que ver 10. A esto se le llama *búsqueda binaria*, porque cada intervalo de búsqueda está dividido en dos en cada intento sucesivo.

Pero eso es sólo una parte de la historia. Tal vez habrá escuchado algo acerca de la tecnología Rushmore de FoxPro. Rushmore significa, más o menos, que los índices se cargan en la memoria a su máxima extensión. Cuando usted busca datos, usando especialmente el comando SELECT SQL, FoxPro lee el índice, no la tabla de datos. De tal manera que cuando usted hace una búsqueda, FoxPro no tiene que leer el disco; los datos ya están en la memoria. Las búsquedas en memoria son millones de veces más rápidas que las búsquedas en disco. De aquí es de donde vienen aquellos anuncios de búsquedas mucho más rápidas que en otros productos. Son verdaderas. No es magia, sólo una técnica inteligente.

Trucos para indexar

Existen algunos trucos para crear y usar índices. Primero use una conversión a mayúsculas para que pueda coincidir en la entrada simplemente convirtiendo la clave que su usuario introdujo en mayúsculas antes de buscar con el comando Seek. Segundo, no elimine espacios en formas que después no pueda reconstruir de manera correcta. Por ejemplo, si usted crea una clave que consiste en la concatenación de los diez primeros caracteres del nombre y los diez primeros caracteres del apellido como sigue:

```
INDEX ON LEFT(ALLTRIM(LEFT(NOMBRE,10)) ;
       + ALLTRIM(LEFT(APELLIDO,10)), 20) TAG NOMBRE
```

usted nunca encontrará el nombre Patricia Nixon si el usuario teclea Pat Nixon, debido a que podría estar buscando la clave de índice "PATRICIANIXON" usando la clave de usuario "PATNIXON". Es mejor usar apellido + nombre o piezas de igual tamaño del apellido y del nombre, y dejar los espacios dentro como sigue:

```
LEFT ( APELLIDO, 10 ) + LEFT ( NOMBRE, 10 )
```

Después buscar "NIXON PATRICIA " por lo menos lo pondrá en el campo de juego. Como corolario natural, el signo menos para la concatenación, el cual hace precisamente este tipo de manejo de los espacios en blanco que se dejan, es palabrería absoluta.

El tercer truco de índice es que usted pueda usar varios campos o partes de campos en una clave de índice y tal vez estará tentado a mezclar tipos. Por ejemplo: Apellido + Fecha de compra, listará primero las compras más recientes de cada cliente. De cualquier manera, FoxPro no le permitirá concatenar un campo de carácter y un campo de fecha en una expresión de índice. Para usar datos en sus expresiones de índice, utilice la función DTOC(Fecha,1) para

convertir fechas a expresiones de carácter de la forma AAAAMMDD. Por ejemplo, DTOC({01/31/95},1) se convierte en la cadena de caracteres "19953101", la cual puede ser concatenada legalmente al campo de nombre con esto:

```
INDEX ON UPPER + (LEFT(Apellido,10)) + DTOC(Fecha,1) TAG CLIENTE
```

Las cantidades pueden ser concatenadas de manera similar con nombres, siempre que primero se conviertan a forma de carácter. Esto lista primero a los clientes que han comprado más:

```
INDEX ON CODIGOCLIE + STR(Compra,10) DESCENDING TAG CLIENTE
```

Y esto lista primero a los clientes más grandes:

```
INDEX ON STR(Compra,10) + CODIGOCLIE DESCENDING TAG CLIENTE
```

Usted se puede volver creativo en este asunto. He aquí una lista alfabética de los distribuidores más grandes, primero las A, luego las B, y así sucesivamente:

```
INDEX ON LEFT(CODIGOCLIE,1) + STR(Compra,10) DESC TAG GASTOS
```

Normalización de tablas

La única y más importante técnica en el desarrollo de bases de datos es la capacidad de observar los datos y ver la base de la estructrura de los mismos. Si usted da a 100 programamdores de bases de datos los mismos documentos desordenados y les pide que estructuren las tablas necesarias para manejar la aplicación, los 100 programadores le darán exactamente las mismas tablas, los mismos espacios en cada tabla (aunque por supuesto no los mismos nombres de campo), ¡aun si ninguno de ellos ha tomado un sólo curso de diseño de bases de datos!

Este proceso se llama *normalización*. Es esencial para el trabajo en el desarrollo de bases de datos. Si sus tablas no están actualizadas, usted trabajará mucho y aun así su aplicación no funcionará. En seguida se muestra un ejemplo simple. Supongamos que su cliente quiere un diseño de sistema para facturar. Para ser útil, él diseñó una hoja de cálculo con los siguientes encabezados de columna:

```
# FACTURA
FECHA DE FACTURA
CODIGO DE CLIENTE
NOMBRE DEL CLIENTE
DIRECCION, CIUDAD, ESTADO, CODIGO POSTAL DEL CLIENTE
ARTICULO 1 #
ARTICULO 1 DESCRIPCION
ARTICULO 1 CANTIDAD
ARTICULO 1 PRECIO
ARTICULO 1 CREDITO
ARTICULO 2 #
ARTICULO 2 DESCRIPCION
ARTICULO 2 CANTIDAD
ARTICULO 2 PRECIO
ARTICULO 2 CREDITO
ARTICULO 3 #
ARTICULO 3 DESCRIPCION
ARTICULO 3 CANTIDAD
ARTICULO 3 PRECIO
ARTICULO 3 CREDITO
TOTAL FACTURA
```

A veces a esto se le conoce como primera forma normal. Para reducirlo a segunda forma normal tome los encabezados Nombre, Dirección, Ciudad, Estado y Código Postal del cliente y muévalos a una tabla Cliente. Es posible utilizar el código de cliente para encontrarlos. Así se puede llamar a la tabla Cliente, *tabla principal.*

La tercera forma normal es el siguiente paso, aquí es donde usted toma los encabezados Artículo #, Descripción, Cantidad, Precio, Crédito y los mueve a otra tabla. Coloque un prefijo a cada registro con # Factura para unirlos a la factura original. Esta tabla se llama *tabla hija,* ya que cada grupo de caracteres de línea tiene una sola factura "padre".

Ahora puede estructurar su aplicación como una tabla de facturas, una tabla principal de clientes y una tabla hija de líneas detalladas de facturas que pertenecen a los encabezados de factura en FACTURAS.DBF. En este formulario puede agregar facturas de clientes y líneas de detalle sin tener que cambiar ninguna de las estructuras de la tabla. No es complicado, y si no normaliza sus datos le pesará.

Vincule tablas

Ahora que tiene varias tablas en su aplicación, necesita relacionarlas. Por eso se le llama *base de datos relacional*. Usted puede utilizar Browse para ver registros relacionados. Primero cree la estructura de la tabla. En FoxPro puede usar el comando CREATE en la ventana Comandos. Teclee:

```
CREATE FACTURAS
```

y aparecerá la pantalla que se muestra en la figura 2-6. Teclee los campos que se muestran en la pantalla usando la tecla TAB para moverse de campo a campo. La tecla ENTER lo sacará de la ventana abruptamente, así que use TAB. Haga lo mismo para Cliente, como se muestra en la figura 2-7. Por último cree la tabla Detalle (ver figura 2-8) y después abra las tres tablas:

```
USE FACTURAS EXCLUSIVE IN A
USE CLIENTE EXCLUSIVE IN B
USE DETALLES EXCLUSIVE IN C
```

Agregue algunos datos:

```
SELECT FACTURAS
APPEND BLANK   && Inserte dos registros en blanco ahora
APPEND BLANK
BROWSE
```

Llene dos facturas como se muestra en la figura 2-9. Haga lo mismo con la tabla Cliente (ver figura 2-10). Por último llene algunas líneas de detalle para cada factura (ver figura 2-11). Ahora necesita los índices apropiados:

```
SELECT FACTURAS
INDEX ON NOFACTURA TAG NUMFACT

SELECT CLIENTE
INDEX ON CODIGOCLIE TAG CODIGOCLIE

SELECT DETALLES
INDEX ON NOFACTURA TAG DETALLES
```

Dése cuenta de que las banderas de índice no necesitan el mismo nombre como sus campos fuente principales. Si usted piensa en esto, los índices no sólo tienen que ser nombrados después de un campo, simplemente no es posible. De otra manera, ¿cómo llamaría a un índice construido concatenando dos campos? Déles nombres que describan lo que hacen.

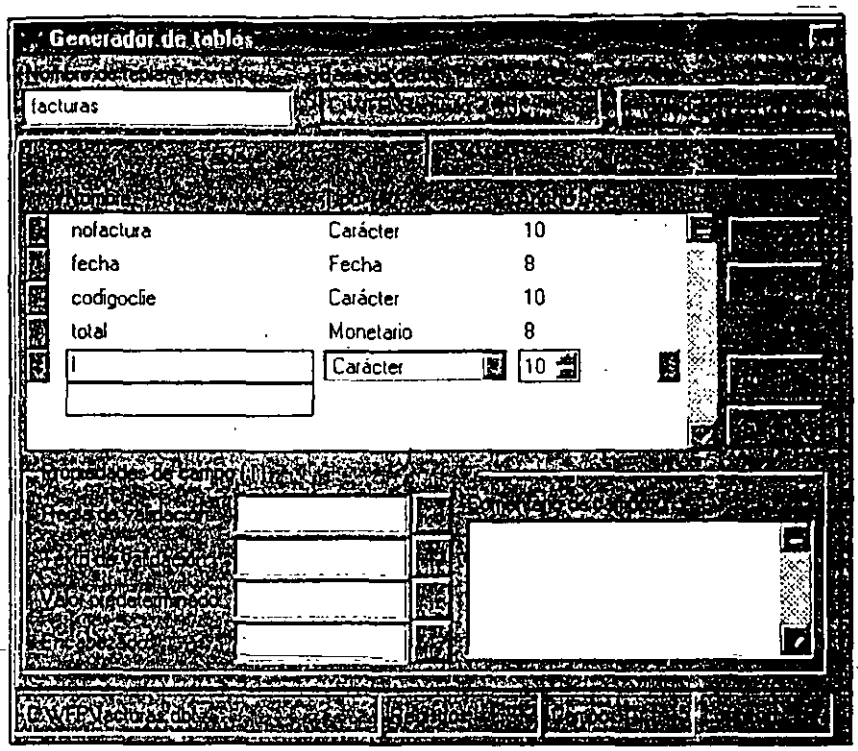


Figura 2-6

Cree la tabla Facturas.

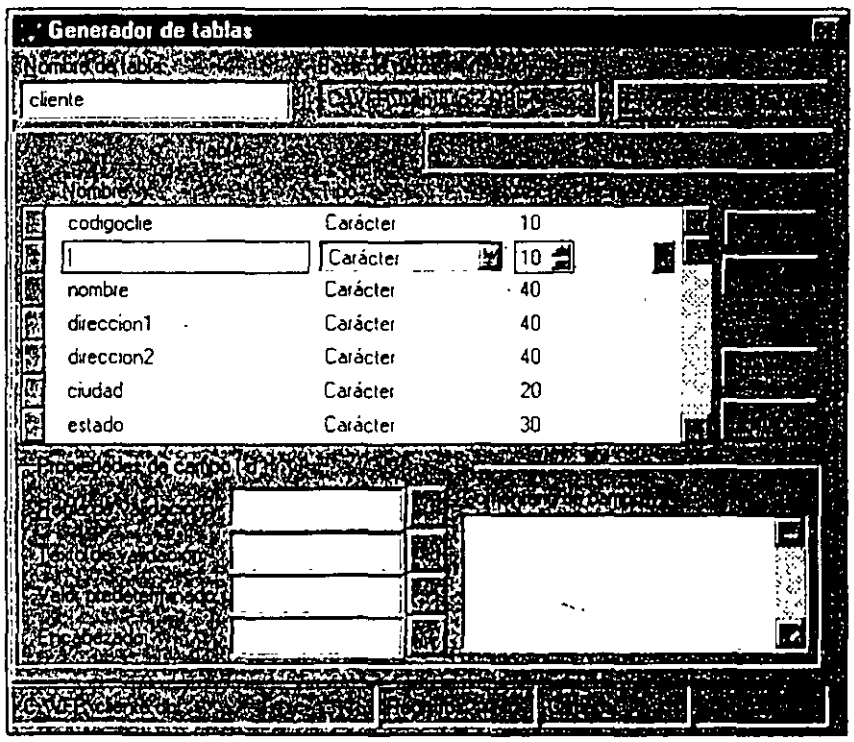
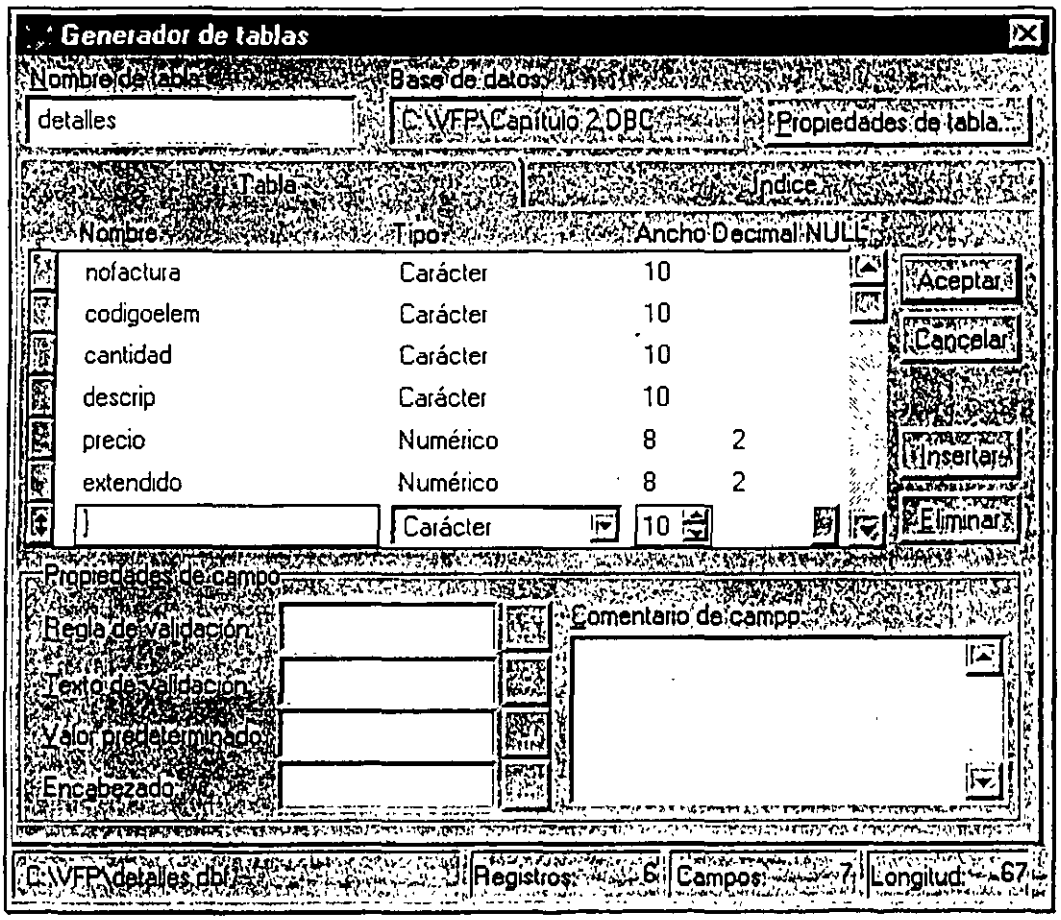


Figura 2-7

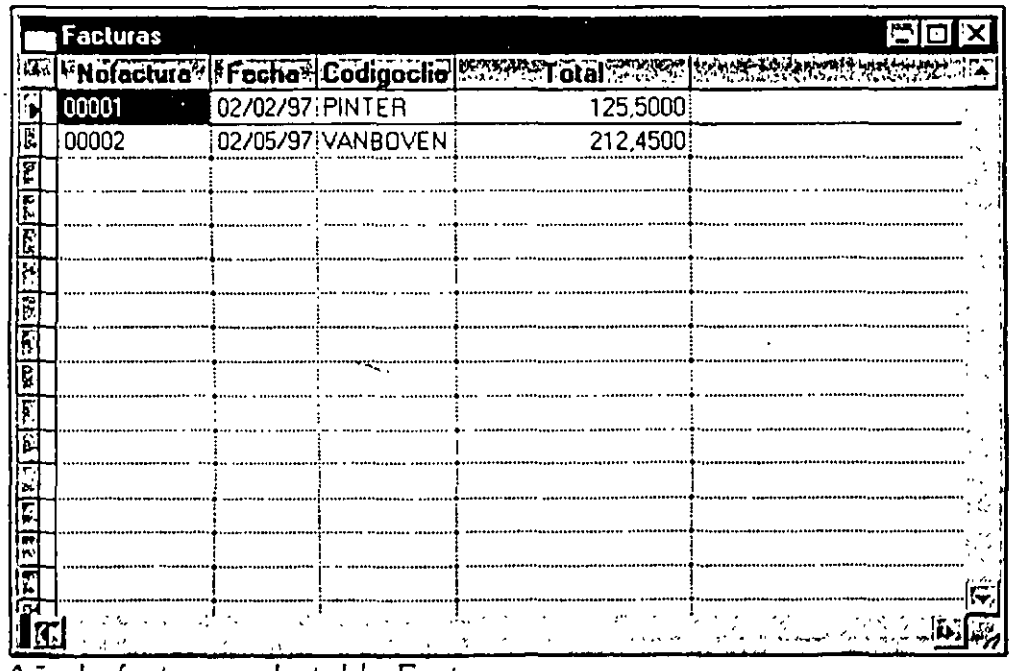
Cree la tabla Cliente.

igura 2-8



Cree la tabla Detalles.

igura 2-9



Añada facturas a la tabla Facturas.

Figura 2-10

Cliente

Codigoclie	Nombre	Dirección	Ciudad	Estado	Zip	
PINTER	Les Pinter	13213 Muhlebach Way	Truckee	CA	96161	1250.0000
VAN BOVEN	Greg Van Boven	2202 Wildflower Trail	Las Vegas	NV	82122	0.0000

Añada dos clientes a la tabla Cliente.

Figura 2-11

Detalles

Nofactura	Codigoitem	Cantidad	Descrip	Precio	Extendido
00001	1	1	Mesa	550.00	550.00
00001	3	2	Archivero	750.00	1500.00
00001	2	3	Silla	145.00	435.00
00002	1	2	Mesa	550.00	1100.00
00002	2	1	Silla	145.00	145.00
				0.00	0.00

Añada líneas de detalle a la tabla Detalles.

Ahora está listo para relacionar tablas. Un mecanismo es el comando Set Relation. Es un poco lento para algunos propósitos, pero para examinar múltiples tablas e informes, es lo único que hay. Teclee lo siguiente:

```
SELECT FACTURAS
SET RELATION TO CODIGOCLIE INTO CLIENTE
SET RELATION TO NOFACTURA INTO DETALLES ADDITIVE
SET SKIP TO DETALLES
```

SET RELATION TO CODIGOCLIE INTO CLIENTE indica a FoxPro mover el punto de registro en la tabla Cliente para apuntar al registro cuya clave coincide con el valor del campo CodigoClie en la tabla Facturas. Así que con cada nueva factura FoxPro encontrará la coincidencia con el registro de clientes. Esto se llama *relación uno a uno*, probablemente una de las que verá más a menudo.

El segundo Set Relation es diferente. Existen varias líneas de detalle para cada número de factura, así que este Set Relation siempre comenzará apuntando al primer registro de igualdad en la tabla Detalles relacionada. Por lo tanto, ¿cómo obtiene el resto de los registros de Detalles?

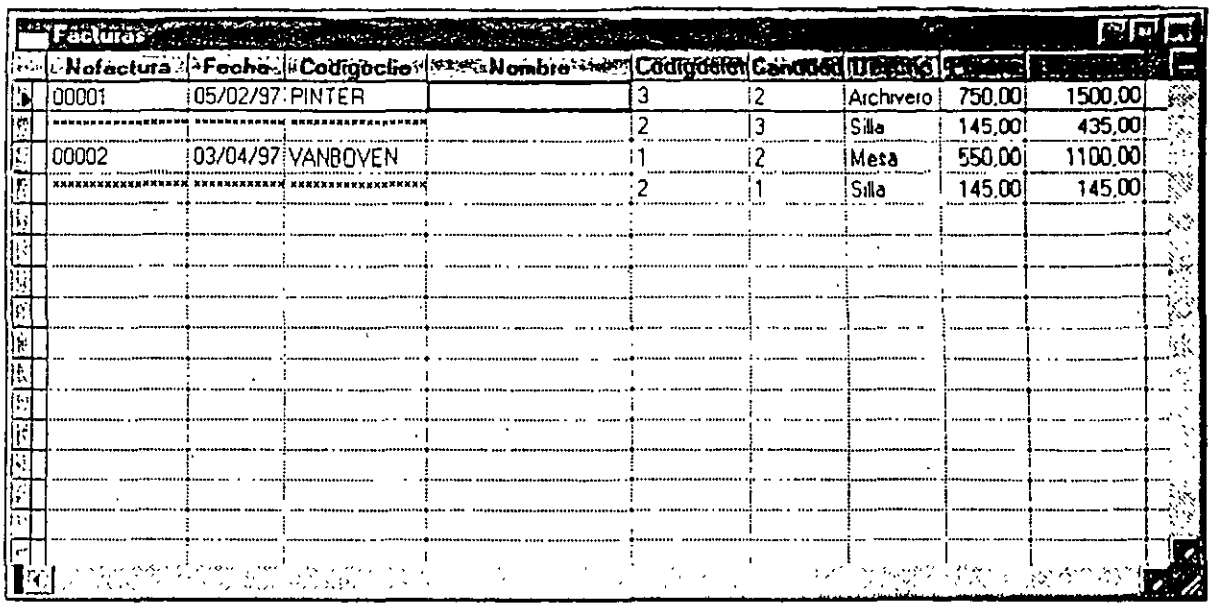
Aquí es donde entra Set Skip. Básicamente lo que éste indica es "cuando el comando Skip está dado, se salta un registro, no en la tabla Facturas sino en la tabla Detalles. Después cuando la clave NoFactura en Detalles cambia de tal forma que ya no es igual al campo NoFactura en Facturas, da un salto en la tabla Facturas". ¿Entendió? Vamos a ver los resultados en un modo visual. Teclee lo siguiente:

```
SELECT FACTURAS
BROWSE FIELDS                    ;
       FACTURAS.NOFACTURA,       ;
       FACTURAS.FECHA,           ;
       FACTURAS.CODIGOCLIE,      ;
       CLIENTE.NOMBRE,           ;
       DETALLES.CODIGOELEM,      ;
       DETALLES.CANTIDAD,        ;
       DETALLES.DESCRIPCION,     ;
       DETALLES.PRECIO,          ;
       DETALLES.EXTENDIDO,
```

El resultado de Browse aparece en la figura 2-12. Set Skip es útil en algunos tipos de procesamiento de registros hijo, pero con Browse es increíble. Los campos repetidos y sombreados muestran una característica agradable.

Esta pequeña excursión a través de la actualización de datos y datos relacionados es sólo una probada de lo que pude hacer con tablas relacionadas. Yo solía trabajar en el mundo de los mainframes, donde bases de datos gigantes almacenaban todos los campos y registros en una estructura amorfa que hacía que xBASE pareciera simple en una forma engañosa. Nada puede ir más allá de la verdad. Si usted necesita más de 65 535 tablas abiertas simultáneamente, FoxPro tal vez no dé el ancho, pero hay muchas aplicaciones pésimas en el mundo real que no necesitan más poder que el que FoxPro tiene bajo la manga.

Figura 2-12



Facturas

Nofactura	Fecha	Codigocli	Nombre	Codigoart	Cantidad	Titulo		
00001	05/02/97	PINTER		3	2	Archivero	750,00	1500,00
**********	**********	**********		2	3	Silla	145,00	435,00
00002	03/04/97	VANBOVEN		1	2	Mesa	550,00	1100,00
**********	**********	**********		2	1	Silla	145,00	145,00

Un Browse con Set Skip.

Ahora que ha visto algunos de los componentes de datos de una aplicación de FoxPro, vamos a ver las herramientas nuevas que FoxPro suministra, para hacer este proceso tan intuitivo como sea posible.

El contenedor de base de datos de Visual FoxPro

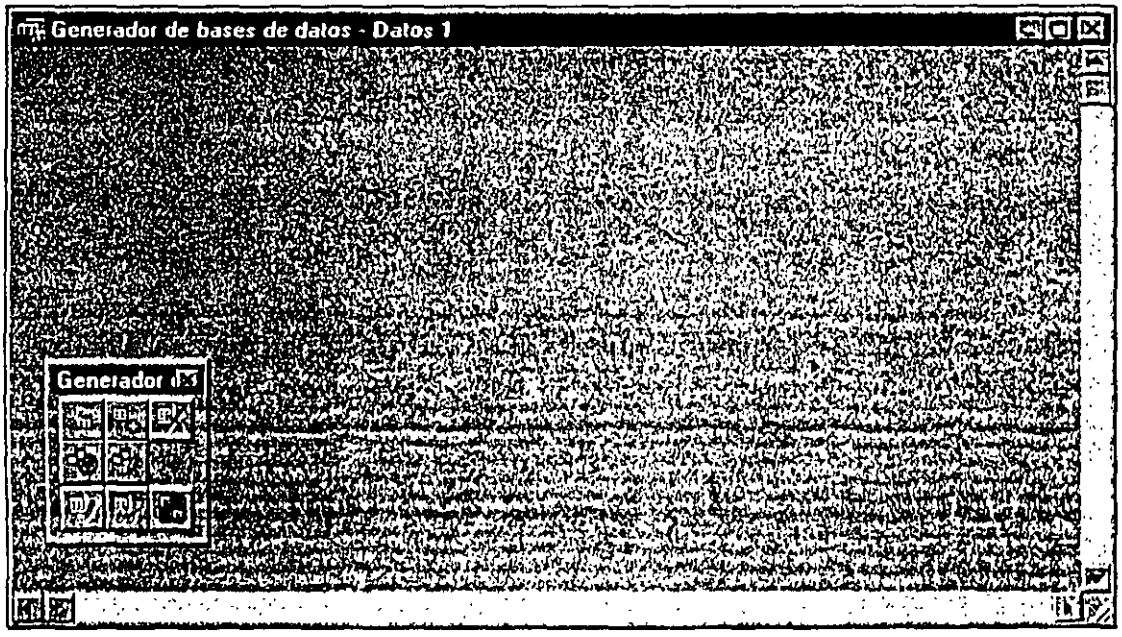
Una de las nuevas características más poderosas que distingue a Visual FoxPro de sus predecesores es el uso de un contenedor de base de datos para almacenar información sobre tablas, índices y otros temas relacionados. Un contenedor de base de datos es en realidad una tabla de FoxPro con la extensión .DBC. Sus campos memo relacionados están en una tabla del mismo nombre con la extensión .DCT.

Las DBC son esencialmente diccionarios de datos activos. Su representación visual de un modelo de datos de aplicación es especialmente bienvenida en la documentación y comunicación de las relaciones entre las tablas de sus aplicaciones. También almacenan los datos que usa FoxPro para implementar nuevas características, incluyendo desencadenantes y procedimientos almacenados de los que hablaremos después.

Para abrir una base de datos, haga clic en Archivo, Nuevo y seleccione Base de datos. La ventana Generador de bases de datos mostrada en la figura 2-13 se abrirá.

La ventana Generador de bases de datos es el centro de control para generar el modelo de datos que soporte su aplicación. En éste usted construye tablas visualmente, crea etiquetas de índice, establece relaciones entre tablas y proporciona otros componentes que ayudan en el proceso de diseño de una aplicación.

Figura 2-13



Generador de bases de datos.

El botón secundario del ratón

Hay un nuevo mecanismo de control en Visual FoxPro: el botón secundario del ratón (por lo general el derecho), el cual nunca había tenido un uso, es un tipo de visita guiada local a los componentes visuales.

En el Generador de bases de datos, si usted coloca el puntero del ratón en cualquier lugar de la ventana del Generador de bases de datos y hace clic con el botón secundario del ratón, se desplegará un menú contextual que contiene las siguientes opciones:

- Expandir todos
- Contraer todos
- Nueva tabla
- Nueva vista remota
- Nueva vista local
- Agregar tabla
- Procedimientos almacenados
- Integridad referencial
- Ayuda

Describiremos la mayoría de estas características en los siguientes capítulos. Comencemos el proceso de diseño.

Tablas

Como la primera tabla, construyamos una tabla Cliente. De nuevo, use el Archivador para asegurarse de que está en el subdirectorio que quiere estar; CAP02 estará bien.

Puede comenzar una tabla nueva de varias maneras. Una es teclear CREATE CLIENTE desde la ventana Comandos. Pero hay muchas más. Como ya se ha visto, puede seleccionar Archivo en la barra de menús de FoxPro, y luego las opciones Nuevo, Tabla. Puede usar el botón secundario para que aparezca el menú contextual local y hacer clic en Nueva tabla. O puede utilizar la nueva barra de herramientas Generador de bases de datos que debió aparecer debajo de la ventana del Generador de bases de datos. Si no sucedió así, seleccione Herramientas, Barra de herramientas, Generador de bases de datos, de la barra de menús de FoxPro. El icono Nueva tabla es el que aparece en el extremo superior izquierdo. Probablemente hay más formas de definir una tabla nueva, pero cuatro son suficientes por ahora.

El ahora familiar Generador de tablas debería estar en su pantalla. En este punto puede llenar los campos Nombre, Tipo y Ancho. Recuerde no presionar la tecla ENTER hasta que haya terminado. Si lo hace inadvertidamente puede teclear MODIFY STRUCTURE (asegúrese primero de usar la tabla que quiere modificar), elija el menú Ver y luego la opción Generador de tablas o entre al Generador de bases de datos y haga clic en el icono Modificar tabla, que se encuentra en la parte inferior derecha de la barra de herramientas del Generador de bases de datos. Nótese que debe ya sea examinar o editar una tabla antes de que el menú Ver agregue las opciones del Generador de base de datos y del Generador de tablas. Teclee lo siguiente en los campos de la tabla Cliente:

CODIGOCLIE	Carácter	10
NOMBRE	Carácter	40
DIRECCION1	Carácter	40
DIRECCION2	Carácter	40
CIUDAD	Carácter	20
ESTADO	Carácter	3
CODPOST	Carácter	10
BALANCE	Monetario	8
NOFACTURA	Carácter	8

Agregue un segundo archivo llamado Ordenes, como sigue:

NUMORDEN	Carácter	10
CODIGOCLIE	Carácter	10
FECHA	Fecha	8
CANTIDAD	Monetario	8
ENTREGA	Lógico	1

Un tercer archivo seguirá las líneas en cada orden:

NUMORDEN	Carácter	10
NUMELEM	Carácter	10
CANTIDAD	Numérico	3
PRECIO	Monetario	8
TOTAL	Monetario	8

Finalmente necesita un archivo maestro de los artículos que usted vende:

NUMELEM	Monetario	8
DESCRIPCION	Carácter	40
PRECIOUNIT	Monetario	8

Mientras usted está en la ventana Generador de bases de datos, dése cuenta de que si pone el puntero del ratón sobre una de las tablas y hace clic con el botón secundario del ratón obtendrá este menú contextual:

Examinar	invoca al comando BROWSE.
Eliminar	borra o elimina la tabla del DBC.
Contraer	minimiza la representación de la tabla a un símbolo pequeño, para ahorrar espacio.
Modificar	invoca al Generador de tablas.
Ayuda	invoca a LA AYUDA.

Índices

Usted puede crear etiquetas de índice desde la ventana Comandos con una expresión Index On, pero es más fácil hacerlo visualmente. En el Generador de tablas, usted verá dos fichas en la parte superior de la ventana: Índice y Tabla. Una vez que la tabla se ha creado, usted puede hacer clic sobre la ficha Índice y construir las etiquetas de índice en un entorno familiar. También puede crear índices filtrados que contienen apuntadores hacia registros que coinciden con la expresión del filtro de índice. Éste puede agregar una gran velocidad a ciertos tipos de aplicaciones.

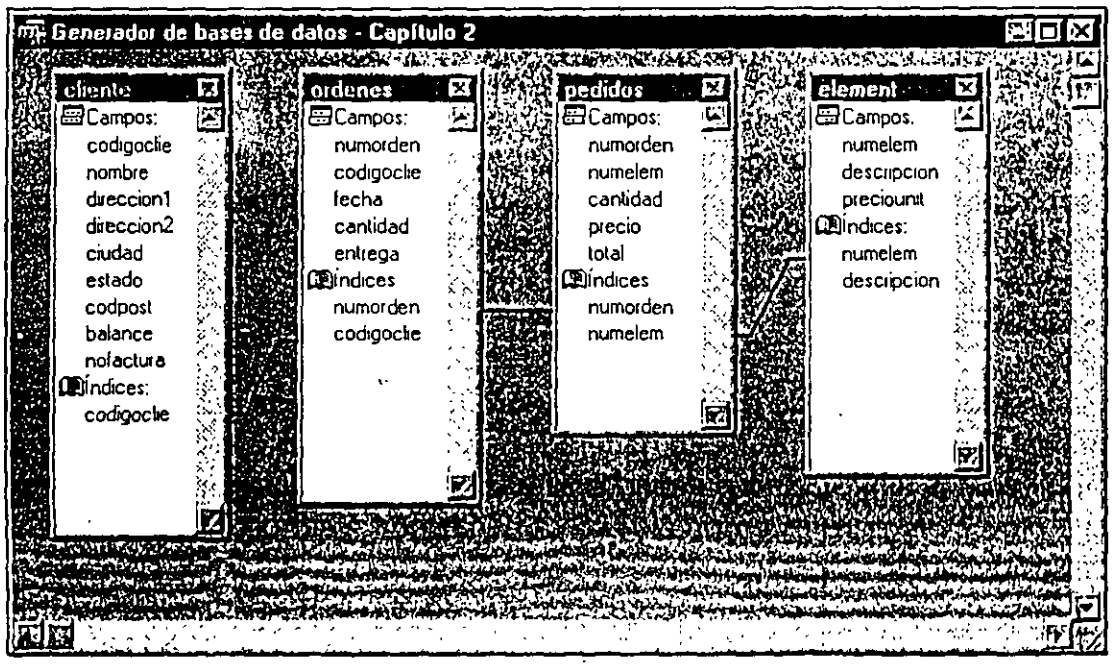
El Generador de bases de datos da un panorama visual a una nueva característica de Visual FoxPro. Para usar este generador para establecer una relación entre dos campos en dos tablas, ambos deben ser índices en sus tablas respectivas. Esto es nuevo para VFP, pero es un precio bajo a pagar por la simplicidad del Generador de bases de datos.

Vamos a crear etiquetas de índice para cada tabla. Haga clic en la tabla Cliente con el botón secundario del ratón para desplegar el menú contextual y seleccione Modificar para que aparezca de nuevo la ventana Generador de tablas. Después haga clic en la ficha Índice de la ventana. Teclee CODIGOCLIE como nombre de la etiqueta y CODIGOCLIE otra vez como expresión. (Nos saltaremos los diferentes tipos de índices por ahora.) Haga clic en Aceptar y habrá terminado. Las órdenes tendrán dos etiquetas de índice: CodigoClie para encontrar un índice relacionado y NumOrden para coincidir con las líneas de detalle en la orden. Las líneas de orden también tienen dos etiquetas de índice: NumOrden para encontrar el registro padre en cada orden (ya que cada orden puede tener varias líneas de detalle), y NumElem para localizar la descripción en el archivo de artículos. Por último, los artículos tienen dos etiquetas: NumElem, que facilita la búsqueda de artículos que necesita la descripción o precio del artículo, y Descripción, para poder desplegar las partes en orden alfabético en los menús desplegables. Ahora viene la parte divertida.

Relaciones

A veces una representación visual vale más que mil palabras. Para establecer una relación entre dos archivos en Visual FoxPro haga clic en la etiqueta de índice que representa la clave de búsqueda, arrástrela sobre el nombre de la etiqueta de índice en el archivo objetivo con el que quiere relacionarla y suéltelo. La línea que aparece indica que se ha establecido una relación. Ahora vamos a ver qué significan los archivos relacionados. La figura 2-14 muestra cómo deberá verse la ventana Generador de bases de datos cuando haya terminado.

Figura 2-14



Generador de bases de datos con relaciones establecidas.

¿Qué hará esto por usted? Si usted establece una relación entre las tablas A y B, FoxPro moverá el apuntador del registro en la tabla A cada vez que cambie el apuntador de registro en la tabla B, así es como cambia la clave. Por ejemplo, si tiene una tabla Ordenes, con un campo llamado CodigoClie y una tabla Cliente con el mismo nombre y *longitud de campo* (es esencial que las expresiones clave sean de la misma longitud), entonces los comandos:

```
USE ORDENES IN A ORDER NUMORDEN
USE CLIENTE IN B ORDER CLIENTE
SELECT ORDENES
SET RELATION TO CODIGOCLIE INTO CLIENTE
```

deberán especificar que el apuntador de registro en Cliente siempre apuntará al primer registro cuya clave CodigoClie coincida con el valor de Ordenes.CodigoClie.

Normalmente usted relaciona tablas de esta forma. En cualquier momento, si usted va a un registro dado en Ordenes sabe que el apuntador de registro de la tabla Cliente se moverá automáticamente a un registro cuyas claves CodigoClie coincidan con el valor de los campos Ordenes.CodigoClie. Recuérdelo de esta forma: SET RELATION está sobre un valor de campo dentro del nombre de una tabla.

La parte de la descripción previa "el primer registro cuya clave coincide" es crucial para entender la delicada peculiaridad de la sintaxis del comando que es única de FoxPro. Si tiene una relación de uno a muchos, Set Relation encontrará sólo la primera que coincide. Cuando se salte un registro en la tabla actual, FoxPro vuelve a colocar el apuntador de registro en la tabla relacionada usando el valor del campo clave de la tabla en uso.

De cualquier manera, puede indicar a FoxPro que salte usando los registros en la tabla relacionada mediante el comando Set Skip. El ejemplo siguiente ejecutará la función XYZ una vez para cada registro en la tabla Ordenes:

```
USE CLIENTE IN A
USE ORDENES IN B ORDER CODIGOCLIE
SELECT CLIENTE
SET RELATION TO CODIGOCLIE INTO ORDENES
SET SKIP TO ORDENES
SCAN
        =XYZ()  && función para imprimir un registro en ORDENES.DBF
ENDSCAN
```

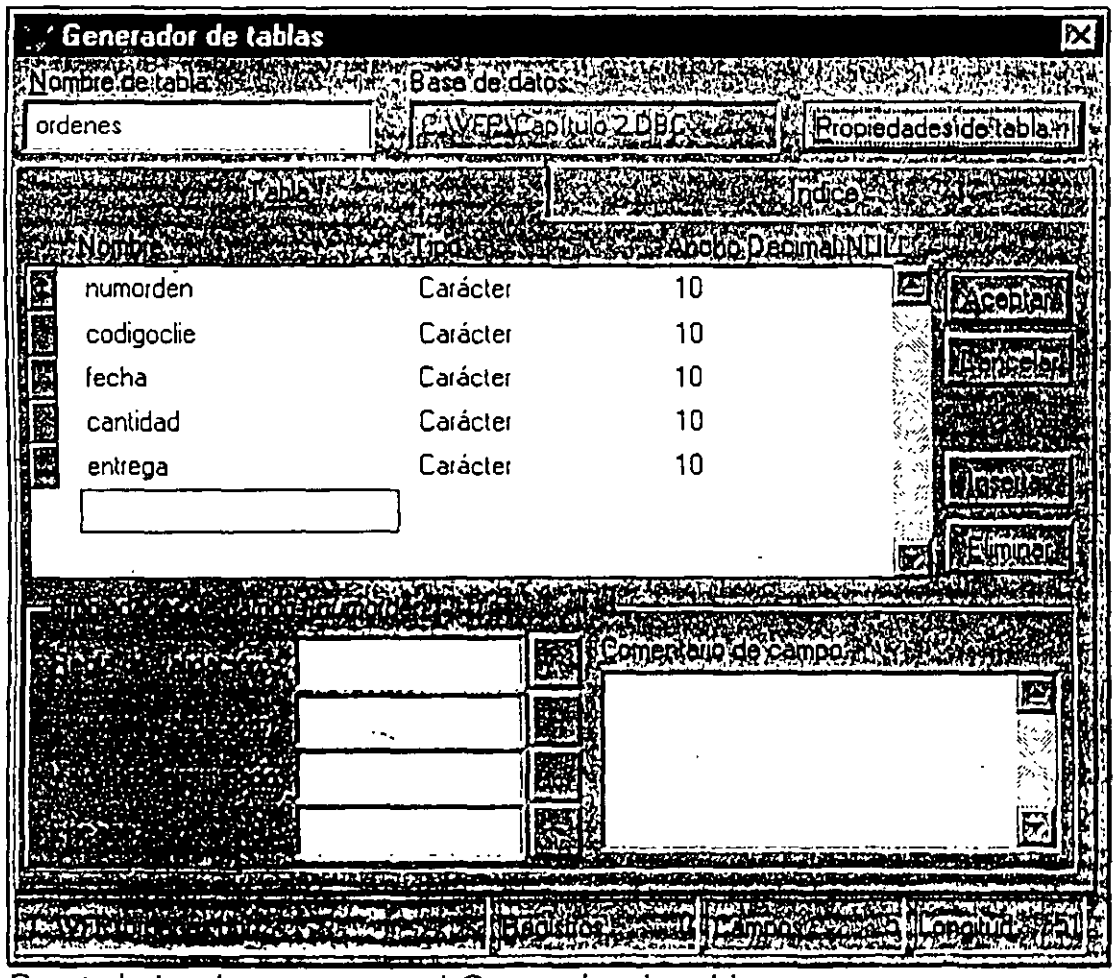
SET SKIP significa "con el comando Skip o cualquier comando que ordinariamente salte al siguiente registro en la tabla en uso al principio de la secuencia, salte un registro en la tabla relacionada".

En tablas visualmente relacionadas con SET RELATION TO (clave en A) INTO B se ve exactamente como SET RELATION TO (clave en B) INTO A. Tal vez tenga que experimentar un poco estableciendo relaciones y utilizar Browse para ver si usted hizo lo que deseaba. Generalmente, si tiene una relación uno a muchos, haga SET RELATION TO (clave en la tabla MUCHOS) INTO (la tabla UNO).

Validación y otras funciones de pantalla

En FoxPro 2.6, un programa llamado GENSCRN leía las tablas de la pantalla y generaba código con valores predeterminados, rutinas de validación y más. En VFP, el mecanismo que ejecuta un formulario tiene acceso al código en el contenedor de base de datos. Así puede almacenar la validación y otras propiedades de campo usadas previamente por GENSCRN. En la parte inferior del Generador de tablas, como se ve en la figura 2-15, usted puede introducir valores y código para la regla de validación, el texto a ser desplegado como un mensaje erróneo si las reglas de validación son violadas, un valor predeterminado para el campo y un encabezado para usarse en el formulario que se generó primero. (Si no se suministra un encabezado, se usará el nombre de campo.)

Figura 2-15



Propiedades de campos en el Generador de tablas.

Desencadenantes

Los desencadenantes son análogos a las funciones de validación pero éstos operan a nivel de registro. Si usted abre el Generador de tablas y hace clic sobre el botón Propiedades de tabla, verá la figura 2-16. Cuando usted inserta con Insert, actualiza (cambia) con Update o eleminia un registro, se invocan estos desencadenantes. Al igual que las claúsulas de validación, si regresan un valor de .F., la operación no está permitida.

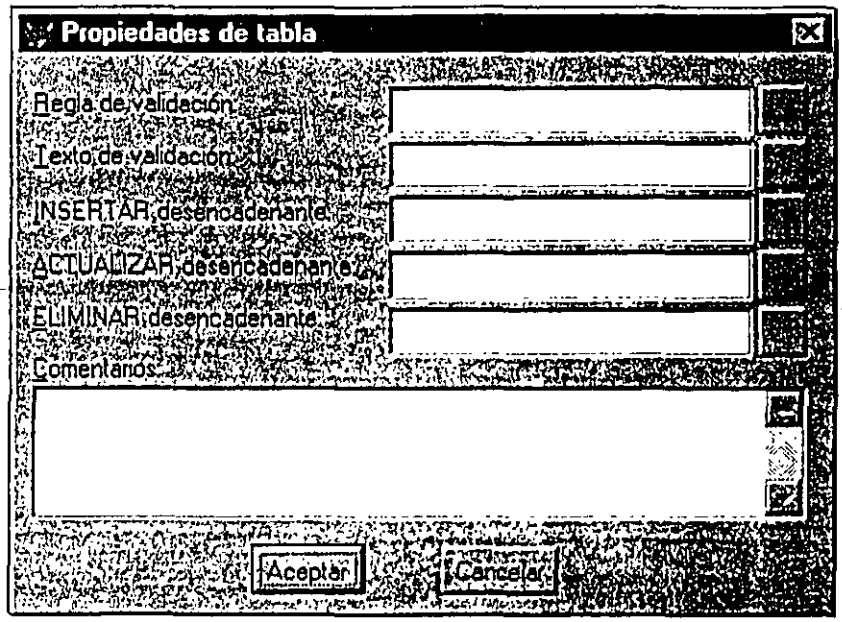


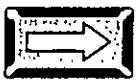
Figura 2-16

Propiedades de tabla en el Generador de tablas.

El lugar para almacenar estos procedimientos es en el código de procedimientos almacenados, lo que puede obtener haciendo clic con el botón secundario del ratón dentro del Generador de base de datos y seleccionando Procedimientos almacenados. Verá una ventana de edición como la que muestra el listado siguiente. He creado una función llamada Backup, la cual quiero ejecutar siempre que borro un registro desde cualquier archivo usando Respaldo( ) en ELIMINAR desencadenante. El código para Backup busca un archivo llamado Respaldo\, además de la tabla que esté seleccionada en ese momento en la pantalla, creándola si no la encontró. Después copia el registro actual.

Para usar Respaldo( ), incluya una llamada a éste en ELIMINAR desencadenante para respaldar cada tabla. Cada vez que su programa llame a la función ELIMINAR o un usuario use CTRL-T para borrar un registro, el registro se respaldará antes de que sea borrado de la tabla seleccionada.

```
PROCEDURE RESPALDO
IF NOT FOXTOOLS $ SET(LIBRARY)
        SET LIBRARY TO \VFP\FOXTOOLS  && Asume que el subdirectorio es VFP
ENDIF
NOMRESP = "RESPALDO\"+ JUSTEM ( DBF() )  && El directorio deberá existir
                                         && previamente.
IF NOT FILE ( NOMRESP )
        COPY STRUCTURE TO ( NOMRESP )
ENDIF
SCATTER MEMVAR
INSERT INTO ( NOMRESP ) FROM MEMVAR
ENDPROC
```

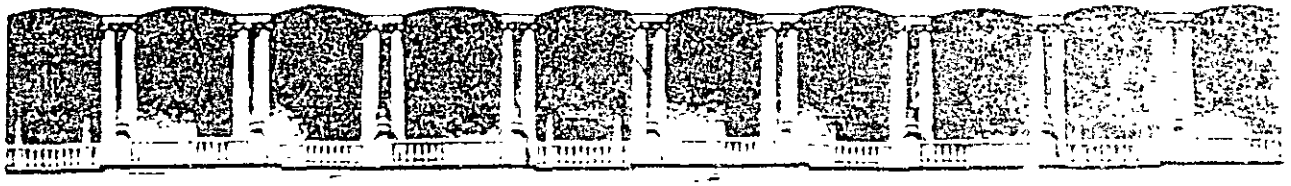


La sorpresa

Casi todo archivo en FoxPro -los formularios de pantalla, los informes e incluso las tablas del proyecto- son en sí mismos ¡tablas de FoxPro! La gente que creó FoxPro pensó que las tablas eran tan buena idea que las usaron para construir los elementos del mismo FoxPro.

En un capítulo posterior, cuando usted construya un formulario encontrará que el formulario se encuentra en una tabla. No tiene una extensión .DBF; utiliza en su lugar .SCX. Los campos memo se almacenan en una tabla con el mismo nombre pero con la extensión .SCT. Inclusive usted puede examinar la tabla del formulario; sólo especifique USE NOMBREFORMULARIO.SCX seguido de BROWSE.

En realidad, esto no debería ser una gran sorpresa. Las tablas son un excelente medio para organizar la información. Sólo observe cuánta información, especialmente información de referencia, es presentada en la forma de tablas en informes impresos. Además, sólo demuestra que FoxPro es tan buena herramienta que ¡puede ser usada para administrarse a sí misma!



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

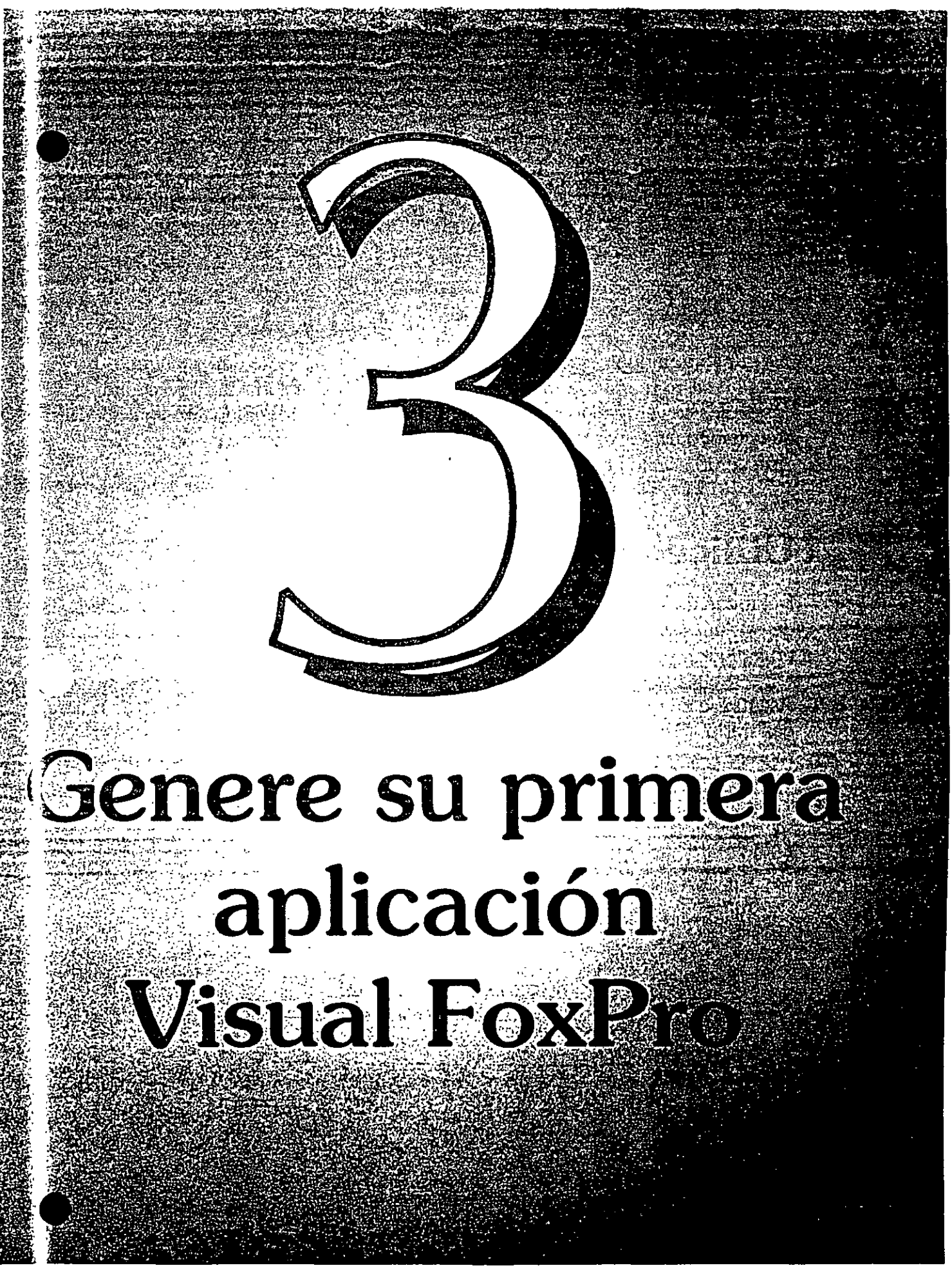
VISUAL FOX PRO VER 3.0

COMPLEMENTO

CAPITULOS: 3 y 4

OCTUBRE 1997

Palacio de Minería Calle de Tacuba
Teléfonos 512-89


3
Genere su primera aplicación Visual FoxPro

CAPITULO 3

HACE muchos años, había un programa de televisión acerca de un maestro que fue nombrado el mejor maestro de física a nivel preparatoria en Estados Unidos. (Él también fue mi maestro de física de preparatoria.) Su técnica era sencilla: Enseñaba el curso completo en las primeras dos semanas, luego pasaba el resto del año expandiéndose en los fundamentos que había presentado. Su teoría era que las cosas son más fáciles de aprender cuando usted las puede colocar en un marco de referencia. Obviamente, esto funcionaba muy bien. Nos gustó tanto la idea que decidimos usarla en este libro.

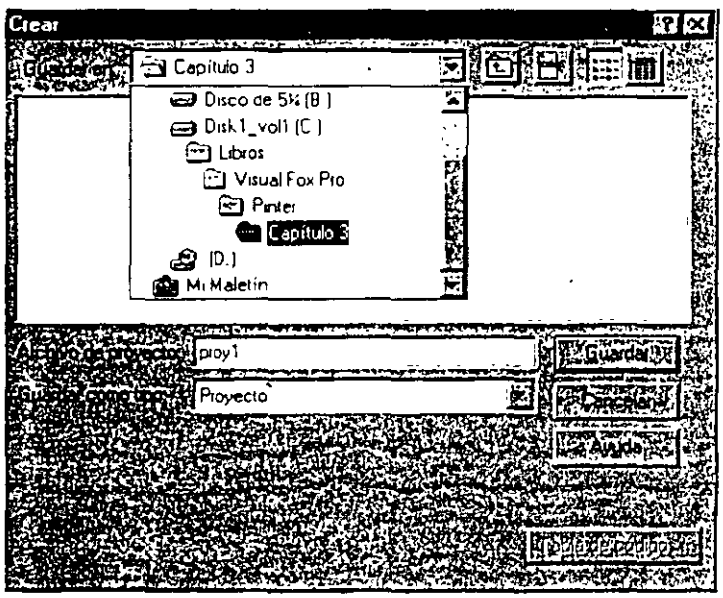
Así que, ahora que usted ha dado un vistazo a las tablas de datos y al entorno Visual FoxPro, comenzaremos por generar una aplicación sencilla. Algunos de los conceptos se verán algo rápido, pero deben proporcionarle un contexto, de forma que cuando usted vea los detalles en los capítulos subsecuentes podrá decir "Recuerdo dónde he visto esto".

Utilice un archivo de proyecto

Del menú de Visual FoxPro, seleccione Archivo, Nuevo. Escoja Proyecto de la lista de opciones disponibles. El cuadro de diálogo Crear aparecerá, como se muestra en la figura 3-1.

Asegúrese de que está viendo la unidad y el directorio del proyecto donde quiere generarlo. (Seleccione CAP03. Si no existe, utilice el comando Filer para crearlo y regrese a este paso.) Escriba el nombre del proyecto 1A-APLIC y oprima ENTER.

Figura 3-1



Cuadro de diálogo para crear un proyecto.

2

Elementos del archivo de proyecto

La ventana del Administrador de proyectos, en la figura 3-2, es la parte principal del escritorio de trabajo de un desarrollador de Visual FoxPro. Es aquí donde usted puede ver de un solo vistazo de qué se conforma su aplicación. Note las seis fichas en el marco de página del Administrador de proyectos:

Todos Muestra todos los componentes.

Datos Muestra sólo las bases de datos, tablas libres y consultas.

Documentos Muestra sólo los formularios, informes y etiquetas.

Clases Muestra sólo las bibliotecas de clases y clases.

Código Muestra sólo programas, bibliotecas API y aplicaciones.

Otros Muestra menús, archivos de texto y otros archivos.

Estas fichas corresponden a las principales clasificaciones dentro de un proyecto. Si hace clic en la ficha Todos, usted verá todo. Hacer clic en cualesquiera de las otras fichas le mostrará sólo la porción de esa ficha del proyecto subrayado.

El botón Control en la esquina superior derecha de la pantalla del Administrador de proyectos, el cual contiene una flecha hacia abajo, le permite minimizar o

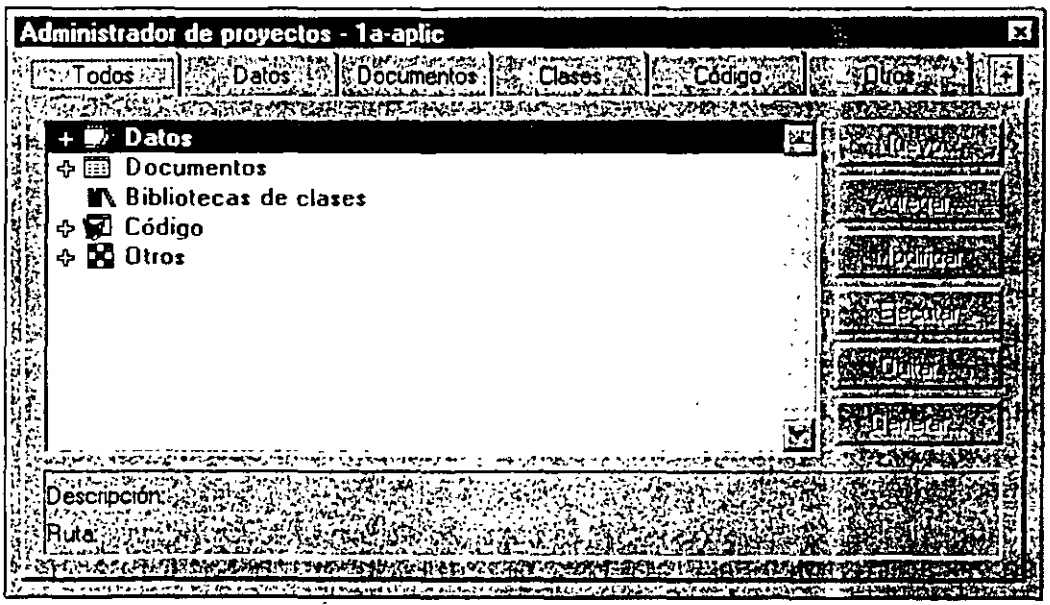


Figura 3-2

Administrador de proyectos.

maximizar la pantalla. Para quitar de la vista la mayor parte del marco de página del Administrador de proyectos, minimícelo. Cuando está minimizado, seleccionar alguna de las fichas le mostrará información sobre la ficha que seleccionó en una versión más pequeña de la ventana. Si necesita un despliegue más grande, haga clic otra vez en el botón (que ahora tiene una flecha hacia arriba) para maximizarlo. El pequeño control en la esquina superior izquierda es el botón Cerrar, como es común en las aplicaciones para Windows.

Los controles que están al lado derecho del Administrador de proyectos son para administrar los componentes del proyecto:

Nuevo Crea un nuevo componente.

Agregar Selecciona un componente existente para incluirlo en el proyecto.

Modificar Edita el componente resaltado. Es lo mismo que hacer doble-clic en un elemento de proyecto.

Abrir Visualiza el formato de informes y etiquetas. Puede ejecutar un informe o etiqueta en cualquier momento, sin acceder a los datos que normalmente utiliza.

Quitar Quita un componente del proyecto.

Generar Genera una aplicación, ya sea como un archivo .APP, .EXE o un .EXE aislado. Usted utilizará principalmente el formato .APP.

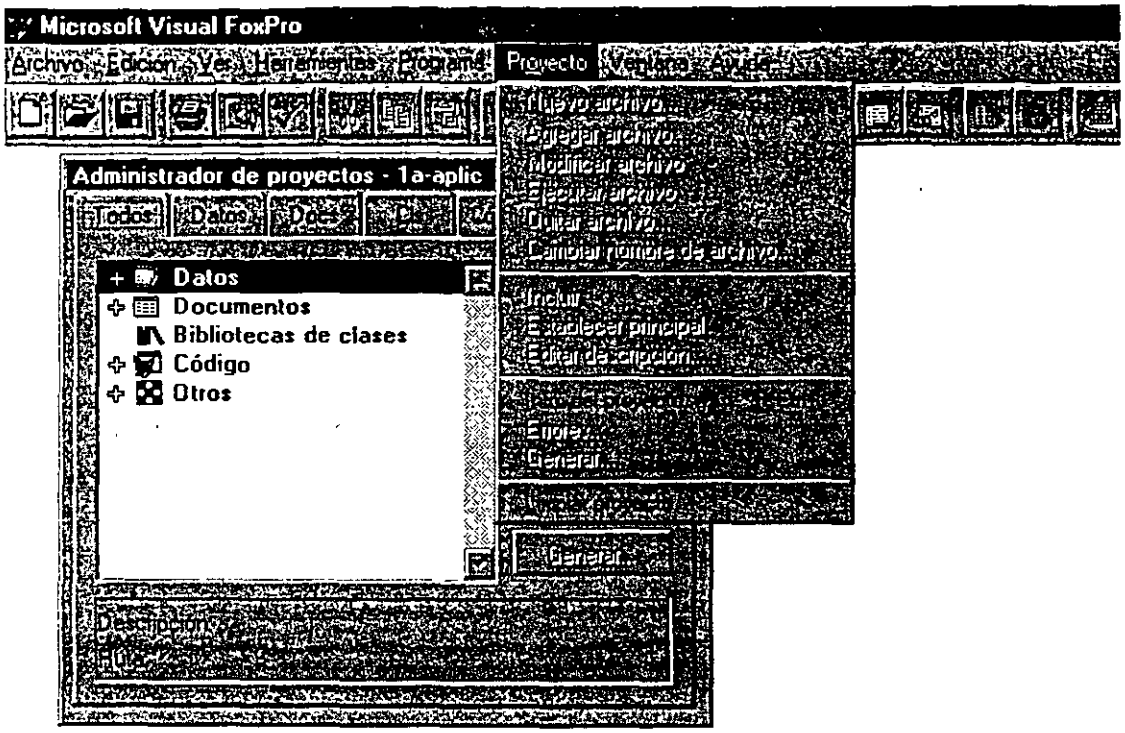
Puede expandir la altura o el largo de la ventana del Administrador de proyectos con el ratón mientras su proyecto crezca. Yo generalmente lo hago hasta la mitad del ancho estándar. Aun en un monitor de 17 pulgadas, los bienes raíces se encarecen rápido cuando usted tiene una docena de objetos abiertos.

El menú Proyecto

Podrá haber notado un nuevo elemento en el menú de Visual FoxPro: Proyecto. Es una práctica estándar en Visual FoxPro que cuando usted abre una nueva ventana, se agregue un elemento al menú que proporciona funciones aplicables a esa ventana. Haga clic en el menú Proyecto y verá el menú desplegable que se muestra en la figura 3-3.

También es característico de Visual FoxPro proporcionar muchas formas de realizar cualquier tarea. Note que los botones de control en la pantalla del

Figura 3-3



Menú desplegable Proyecto.

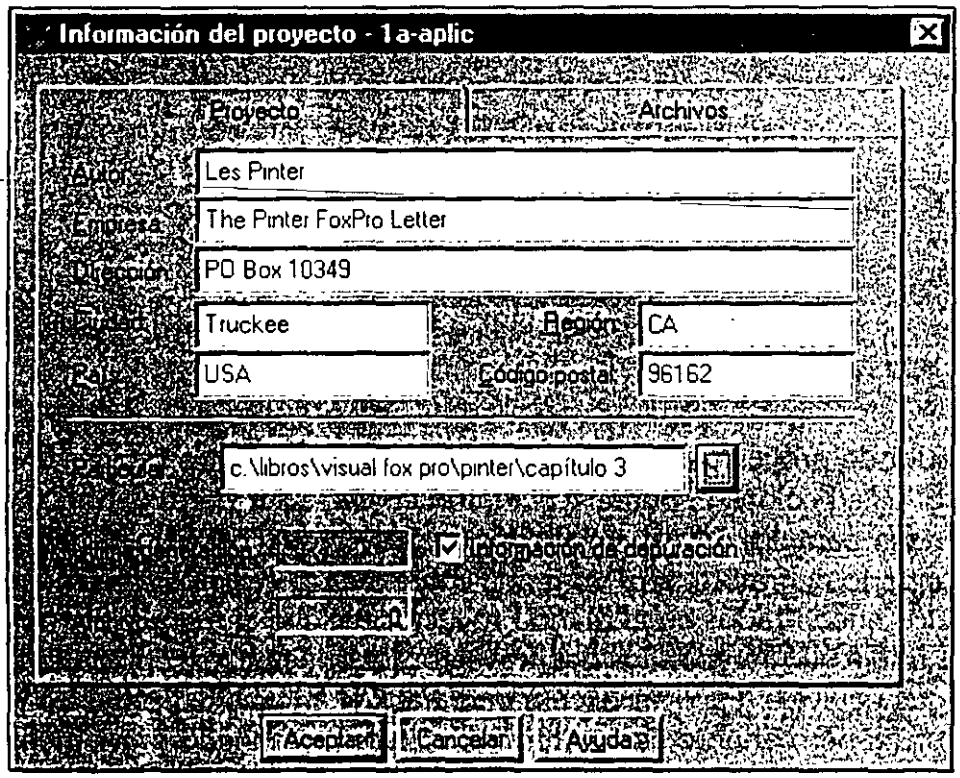
Administrador de proyectos contienen muchas de las funciones listadas en el menú Proyecto. Si usted coloca el puntero del ratón en cualquier parte dentro del área en blanco de la pantalla del Administrador de proyectos y hace un clic con el botón secundario del ratón, verá un menú contextual que consiste en muchos de los comandos encontrados en el menú Proyecto. Así que al menos hay dos formas de acceder a todos los comandos del proyecto. La principal excepción es Limpiar proyecto, que equivale a aplicar el comando Pack al archivo del proyecto para quitar en forma permanente elementos que usted ha eliminado previamente; esta opción se encuentra sólo en el menú principal Proyecto.

Info. del proyecto también está disponible en cualquier momento mientras el proyecto esté abierto con CTRL-J, que produce la pantalla mostrada en la figura 3-4. Aquí es donde usted introduce diversa información:

- Información sobre el desarrollador (nombre, dirección, etc.).
- El directorio donde se aloja el proyecto.
- Si se quiere o no incluir información de depuración en el archivo de aplicación. Esto aumenta un poco el tamaño de la aplicación, utilícelo a menos que tenga una buena razón para no hacerlo.

- Si quiere o no codificar su aplicación. Si la aplicación está codificada o cifrada, es extremadamente difícil para cualquiera aplicar ingeniería regresiva (decompilar) a su aplicación y reconstruir el código fuente. Yo personalmente nunca lo hago: hay mejores formas para asegurar la lealtad de los clientes.
- Qué icono utilizar cuando se minimiza su aplicación. Aquí es donde usted logra ser un artista. Debido a que mucha gente ha expresado interés en esto, Visual FoxPro viene con un editor de iconos incluido.

Figura 3-4



Pantalla Información del proyecto.

La segunda ficha, Archivos, lista todos los componentes del proyecto en un formato compacto (en forma subrayada) en el centro de la pantalla del Administrador de proyectos.

Usted puede ya sea expandir o contraer el control subrayado que lista los componentes del proyecto. Si un signo más aparece a la izquierda de un elemento, puede hacer clic en ese tópico para expandirlo. Si tiene un signo de menos en la izquierda, puede contraer ese componente haciendo clic en él. El menú contextual del Administrador de proyectos contiene una selección llamada Expandir todos, que muestra todos los componentes dentro del tópico subrayado.

6

Administrador de proyectos es único en el sentido de que también funciona como una barra de herramientas. Si usted toma la parte superior de la pantalla del Administrador de proyectos con el ratón y la mueve hasta abajo de la barra de herramientas estándar de Visual FoxPro, se convierte en una barra de herramientas él mismo, desde la cual puede seleccionar y expandir fichas individuales. Haga clic nuevamente en la ficha para cerrar la ventana de la ficha seleccionada. Para traer de regreso la ventana del Administrador de proyectos, haga clic cuidadosamente en el área posterior de cualquiera de las fichas y arrástrela fuera de su posición fija. Se volverá a expandir hasta su tamaño original.

Agregue elementos al proyecto

Una aplicación de Visual FoxPro necesita, por lo menos, un menú, un formulario y un programa principal. Sin que sea sorprendente, las tablas de datos usualmente también son parte de una aplicación. Nosotros agregaremos estos elementos uno a la vez, luego ejecutaremos el resultado. No debe tomar más de algunos cuantos minutos.

Agregue tablas al proyecto

Haga clic en el tópico subrayado Datos para expandirlo. Los componentes listados debajo del encabezado Datos son Bases de datos, Tablas libres y Consultas. ¿Por qué los Datos tienen tres categorías? Examinemos cómo Visual FoxPro presenta la información.

Primero, como se ha visto, la base de datos (contenedor de base de datos, o DBC) almacena información acerca de las tablas: sus índices, sus relaciones con otras tablas, cualquier desencadenante, insertar, actualizar o eliminar, y más. Es ahí donde se selecciona la base de datos y es además la opción más común.

Las Tablas libres son simples archivos DBF que no están incluidos en ningún DBC. Si tiene una tabla de búsqueda compartida por muchas bases de datos, hágala una tabla libre. Las tablas que están incluidas en un DBC pueden pertenecer a sólo una base de datos. De hecho, el DBC al que pertenece la tabla, si existe, se menciona en el encabezado del DBF. No hay espacio en el encabezado para nombrar más de uno. Además, con todo lo útil que puede ser el concepto de DBC, podría haber situaciones donde quiera estar libre de sus controles. Debido a la falta de un comando Set DataDict Off, utilice tablas libres.

Las Consultas están incluidas como fuentes de datos debido al soporte cliente-servidor de Visual FoxPro. Una consulta puede ir hacia un servidor de red, o incluso salir a través de un módem hacia una línea telefónica. Si usted hace su consulta, obtiene sus datos. Suena sencillo, y lo es. Sin embargo, como con Pablo Almunia, el gurú de Visual FoxPro en ambientes cliente-servidor de España, es bueno decir, "La recepción de datos en el acceso remoto puede ser más lenta que su peor pesadilla, a menos que lo planee cuidadosamente". Hablaremos más al respecto posteriormente.

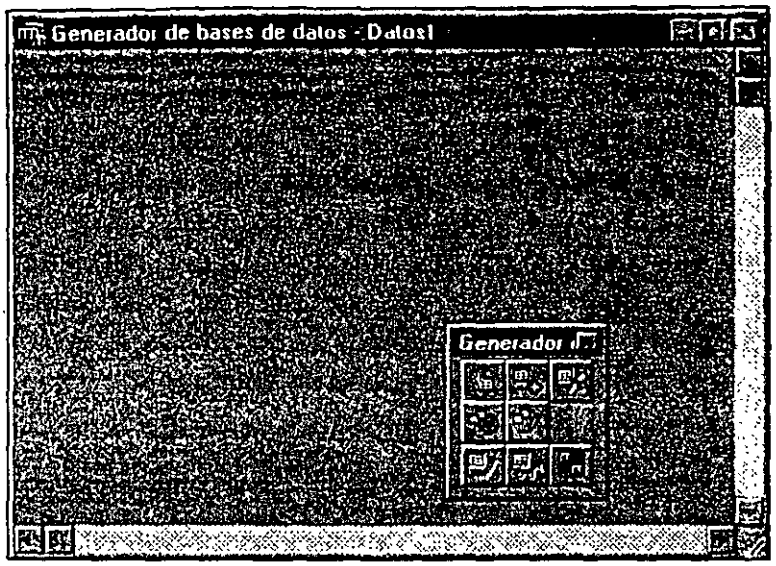
Para este primer proyecto, se utiliza una base de datos. Haga doble clic en Base de datos y obtendrá el cuadro de diálogo Crear base de datos, el cual sugiere el nombre DATOS1.DBC. Ese nombre está bien, así que haga clic en Crear para obtener la pantalla del familiar Generador de bases de datos, que se muestra en la figura 3-5. Las opciones disponibles en el nuevo menú Base de datos son las siguientes:

- Nueva tabla
- Agregar tabla
- Nueva vista remota
- Nueva vista local
- Modificar
- Examinar
- Quitar
- Volver a generar índices de tablas
- Quitar registros eliminados
- Editar relación
- Integridad referencial
- Editar procedimientos almacenados
- Limpiar bases de datos

Como lo hizo anteriormente, seleccione Nueva tabla para crear una tabla y Agregar tabla para agregar una tabla existente al DBC.

Es útil ver que este menú controla el contenido del contenedor de bases de datos abierto. Por ejemplo, el DBC contiene las expresiones de índice para las etiquetas de índice de las tablas incluidas. Cuando usted selecciona Volver a generar índices de tablas, Visual FoxPro vuelve a generar las etiquetas de índice como se desc...

Figura 3-5



Pantalla Generador de bases de datos, otra vez.

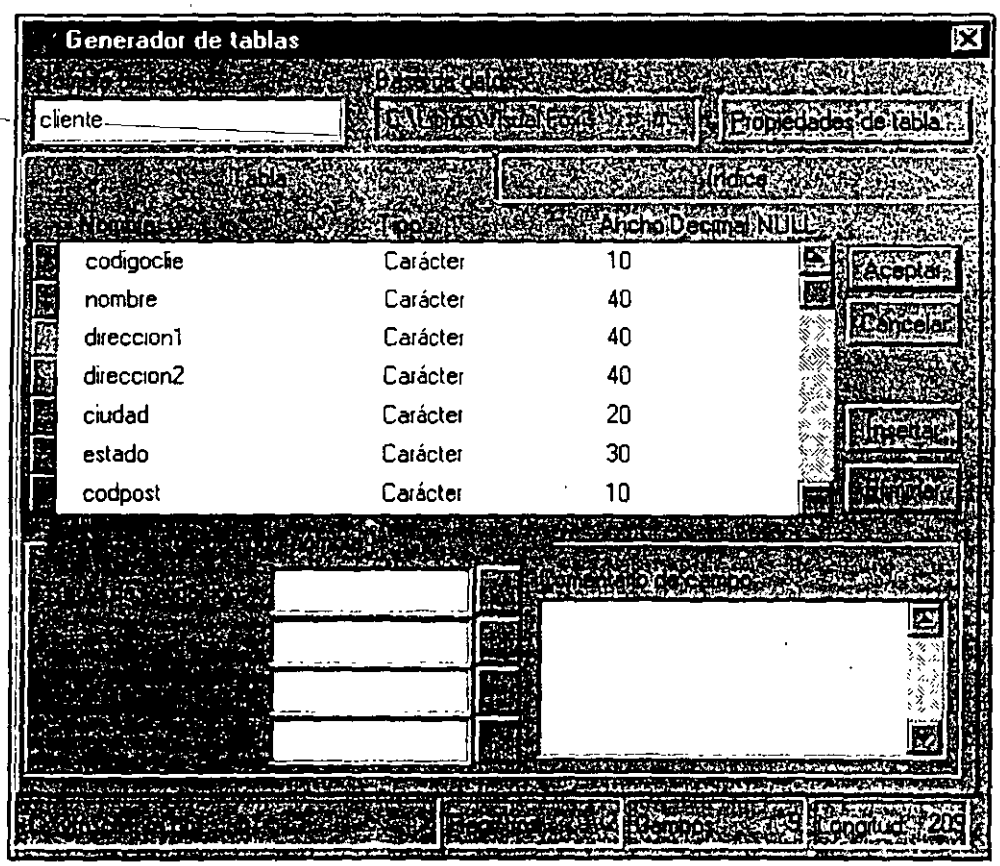
en los registros contenidos en el contenedor de bases de datos. ¿En qué tabla se va a trabajar? En aquella que está seleccionada cuando usted escoge la opción del menú. Es lo mismo quitar registros eliminados, o aplicar el comando Pack, *empacar,* como se le conoce en xBASE. Modificar despliega el Generador de blas para la tabla seleccionada; Examinar la examina, y Quitar quita del DBC la .eferencia a esa tabla. Limpiar base de datos borra los objetos que se han quitado del DBC (debido a que quitar una tabla del DBC sólo etiqueta el registro de la tabla y sus etiquetas de campos e índices para eliminarlos más tarde).

Además, si hace clic con el botón secundario del ratón, obtendrá una lista similar más pequeña, con dos opciones adicionales: Expandir todos y Contraer todos. Estos últimos comandos minimizan y maximizan las descripciones individuales de las tablas. Así que agreguemos una tabla; Cliente será un agradable punto de inicio. Usted tiene cuatro opciones:

- Hacer clic en el elemento de la parte superior izquierda de la barra de herramientas Generador de bases de datos.
- Hacer clic en el menú Base de datos, luego seleccionar Nueva tabla. (Note que con el fin de que esté disponible el menú Base de datos, el Generador de bases de datos debe ser la ventana seleccionada en ese momento.)
- Haga clic con el botón secundario del ratón en la ventana del Generador de bases de datos y seleccione Nueva tabla.
- Teclee CREATE CLIENTE en la ventana Comandos.

Aparecerá el Generador de tablas. Llene los nombres de campos como se muestra en la figura 3-6. Una vez que se introducen los campos, haga clic en la ficha Índice y seleccione CODIGOCLIE como ambos, el nombre y la expresión. Note que puede hacer clic con el botón secundario del ratón en cualquier lugar de la representación gráfica de la tabla para traer el menú contextual con la opción Modificar, la cual lo llevará otra vez al Generador de tablas. La opción Modificar en el menú Base de datos hará lo mismo, pero la tabla con la que usted desea trabajar debe ser seleccionada (por ejemplo, su título de ventana debe estar resaltado, por lo general oscurecido) con el fin de activar las selecciones correspondientes en el menú desplegable Base de datos.

Figura 3-6



Cree la tabla Cliente en el Generador de tablas.

Genere un formulario con el Asistente para formularios

Aunque usted puede generar su formulario desde cero, los buenos amigos de Microsoft han alcanzado grandes latitudes para hacer que esta tarea particular no

10

ólo sea fácil, sino divertida. Estoy hablando acerca del Asistente para formularios, y a usted le va a gustar mucho.

Un *asistente* es una serie de pantallas relacionadas que lo guían a través de un proceso. El Asistente para formularios, selecciona el archivo o los archivos cuyos campos aparecerán en un formulario, le permite escoger de una variedad de presentaciones visuales y luego agrega botones de control al formulario. Usted puede hacer estos pasos solo, pero los asistentes lo hacen tan sencillo que no debe pasarlo por alto.

Haga clic en la ficha Documentos del Administrador de proyectos, luego haga clic en Formularios. Una vez que Formularios está resaltado, haga clic en Nuevo. Verá la pantalla mostrada en la figura 3-7. Haga clic en Asistente para formularios. Esto producirá inmediatamente la pantalla mostrada en la figura 3-8. Escoja la opción Asistente para formularios (en lugar de uno a varios) y haga clic en Aceptar. Aparecerá la pantalla que se muestra en la figura 3-9.

La pantalla proporcionada también especifica Datos1 para bases de datos y Cliente para tablas, haga clic en el control con doble flecha hacia la derecha para mover todos los campos de Cliente a la lista de selección del lado derecho de la pantalla del asistente.

Los campos se copian en su orden original. Si usted quiere regresar los campos a cualquier otro orden, tiene que mover los campos con el ratón y hacer clic en Siguiente.

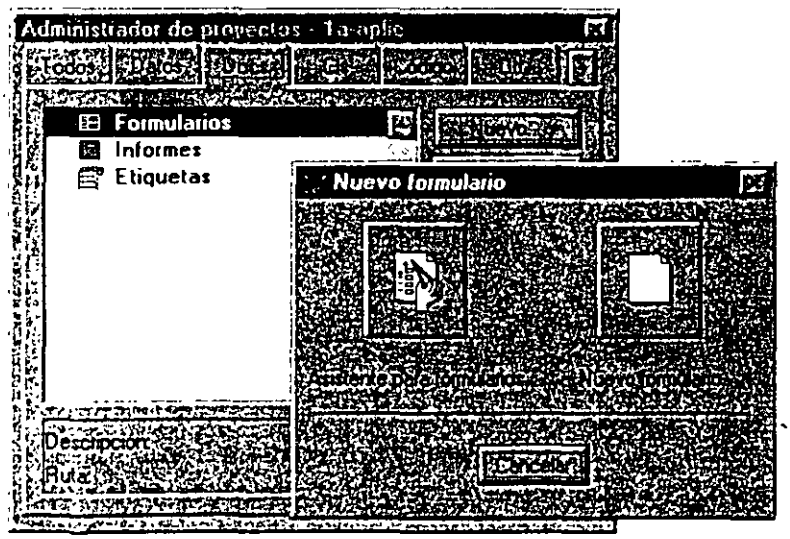
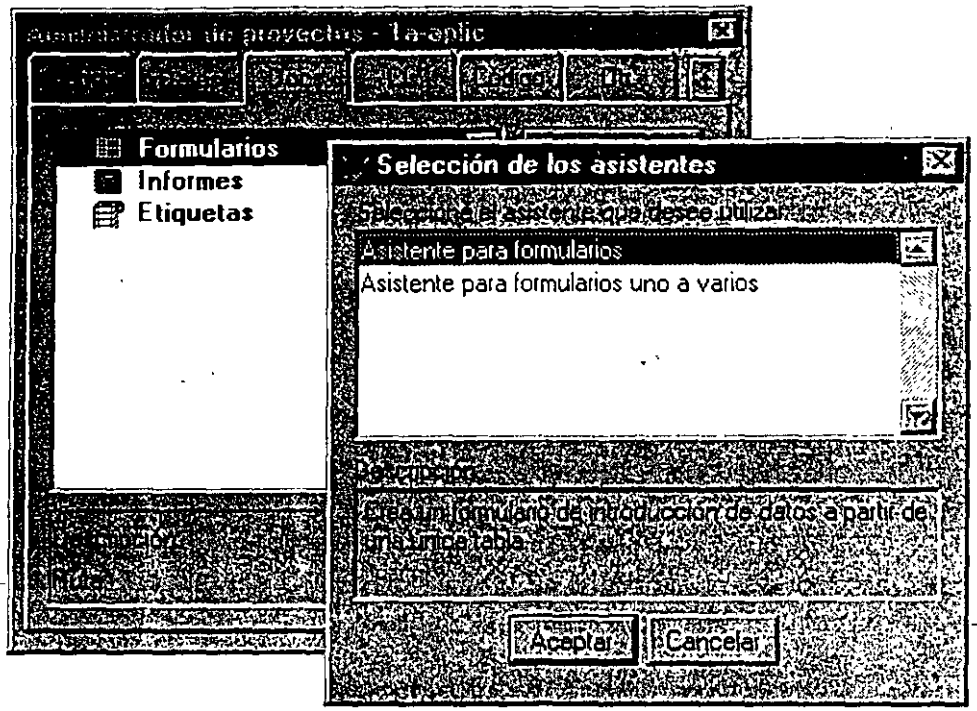


Figura 3-7

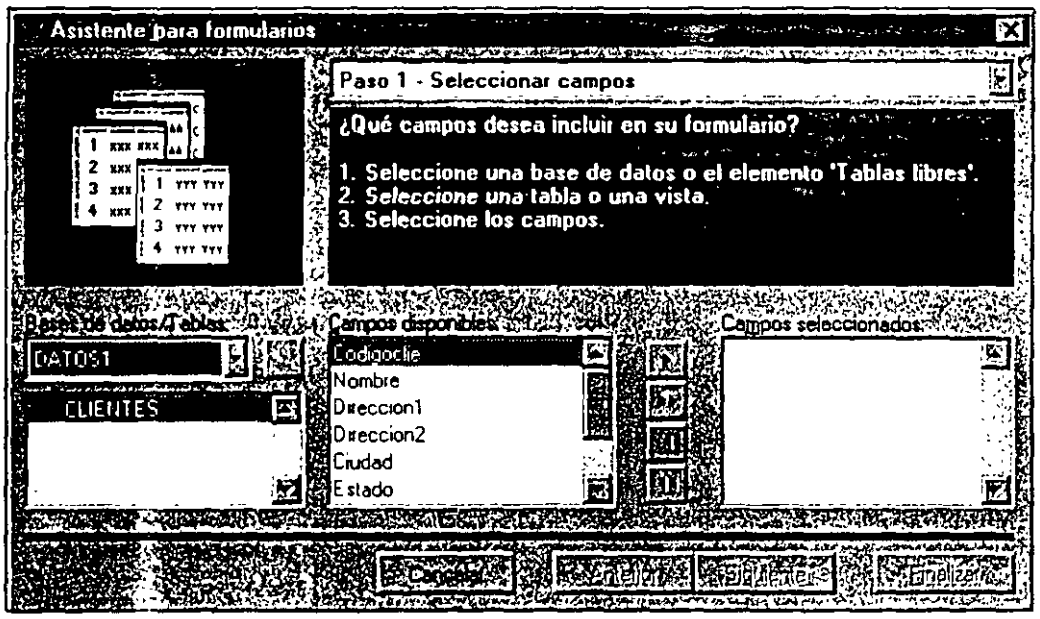
Menú Proyecto.

Figura 3-8



Cuadro de diálogo para seleccionar el Asistente para formularios.

Figura 3-9



Asistente para formularios, paso 1.

12

La siguiente pantalla en el Asistente para formularios, mostrada en la figura 3-10, le proporciona cinco opciones para dar formato a los campos de entrada de datos en el formulario, así como tres opciones para los botones de navegación: Botones de texto, Botones con imagen y Sin botones. Usted utiliza los botones de navegación para moverse hacia adelante y atrás a través de un archivo, eliminar el registro en la pantalla y cerrar la pantalla y salir. Los controles de oprimir botones de Visual FoxPro se utilizan como botones de navegación en este tipo de pantalla, y pueden desplegar ya sea una cadena de texto o un icono .BMP. Yo nunca recuerdo a qué se refiere un icono, así que prefiero texto. Para los campos de entrada seleccione la tercera opción, Sombreado, y luego haga clic en Siguiente.

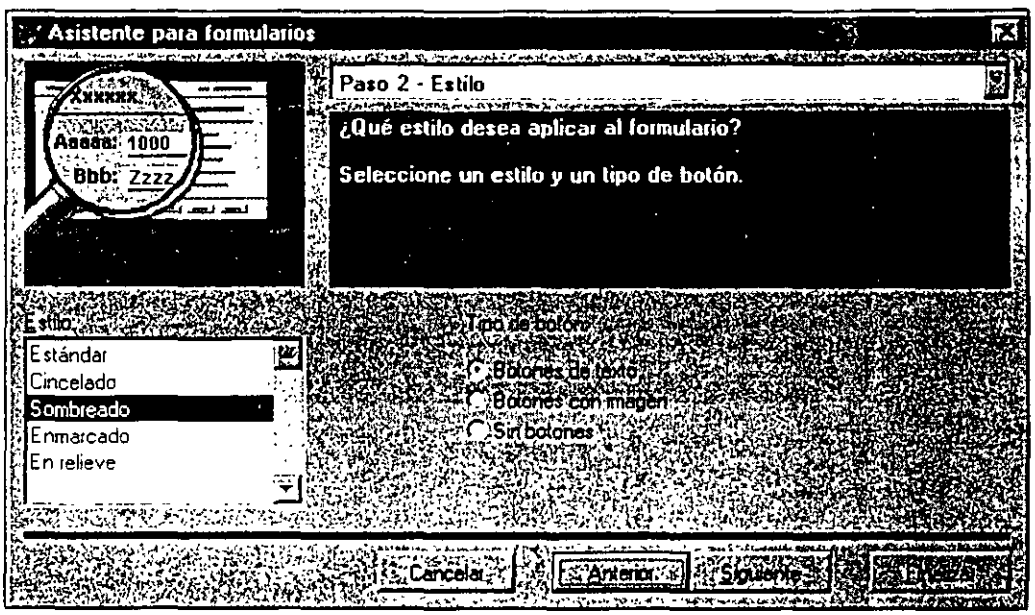


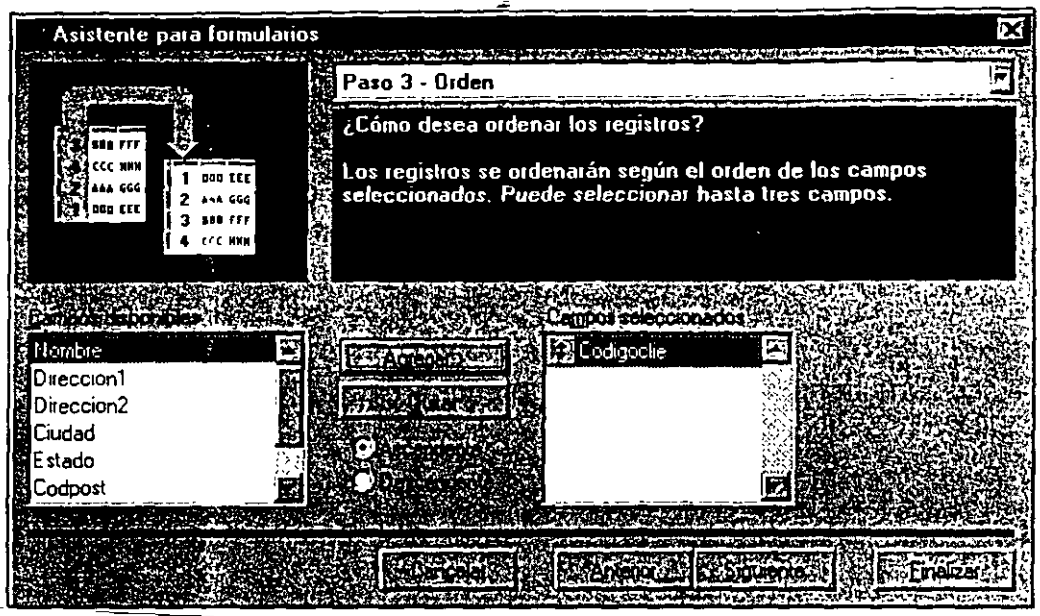
Figura 3-10

Asistente para formularios, paso 2.

La tercera pantalla del asistente, que se exhibe en la figura 3-11, determina el orden del índice cuando el archivo está abierto. Sólo tiene una etiqueta de índice, CodigoClie, así que selecciónela y haga clic en Finalizar.

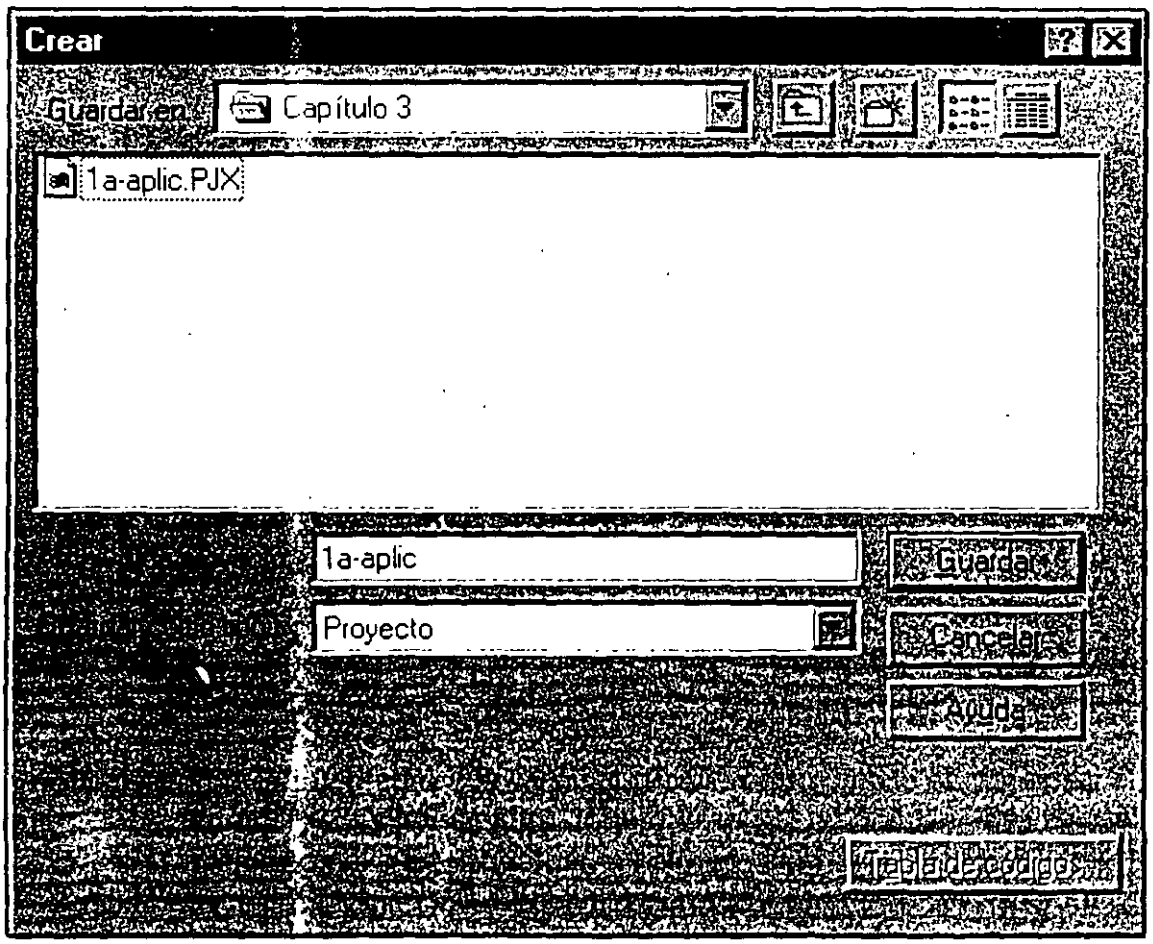
La pantalla del cuadro de diálogo final le pregunta si quiere ver los resultados o no. El formulario completo se muestra en la figura 3-12. No sólo están todos los campos presentes, completos con sus descripciones justificadas correctamente hacia la izquierda de cada campo, sino que un conjunto de botones de navegación completo ya está incluido en la parte inferior de la pantalla, hay botones para permitirle al usuario ir al Primer, Siguiente, Anterior o al Último registro. Hay un botón para Buscar y uno para Imprimir. Hay controles para Agregar, Modificar y Eliminar un registro, y hay un botón para Salir. Nada mal para unos cuantos minutos de trabajo.

Figura 3-11



Asistente para formularios, paso 3.

Figura 3-12



Formulario Cliente generado.

Agregue un menú

Veremos más de cerca a los formularios en breve. Pero primero, hay otra pieza faltante en el rompecabezas que usted puede colocar fácilmente. En una aplicación, especialmente de Windows, se utiliza un menú para desplegar pantallas. Así que usted necesita un menú para que inicie la aplicación Cliente. Haga clic en la esquina superior izquierda del formulario para cerrarlo y regresar a la pantalla del Administrador de proyectos, haga clic en la última opción, Otros, luego resalte Menús y haga clic en el control Nuevo del lado derecho de la pantalla del Administrador de proyectos. Aparecerá el Generador de menús mostrado en la figura 3-13.

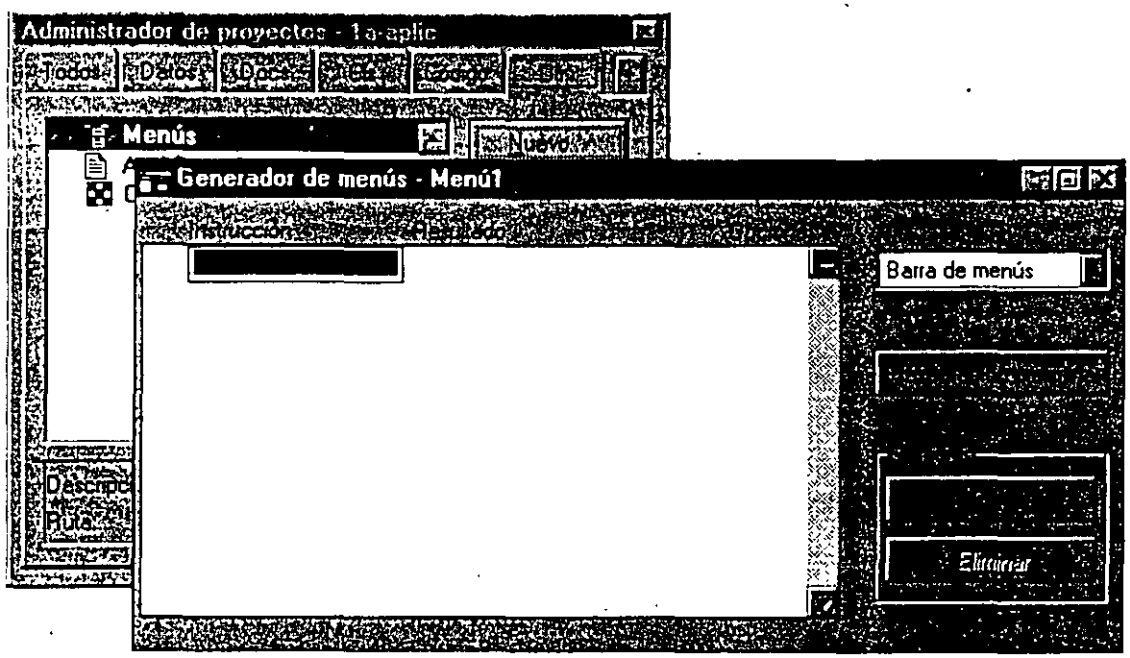
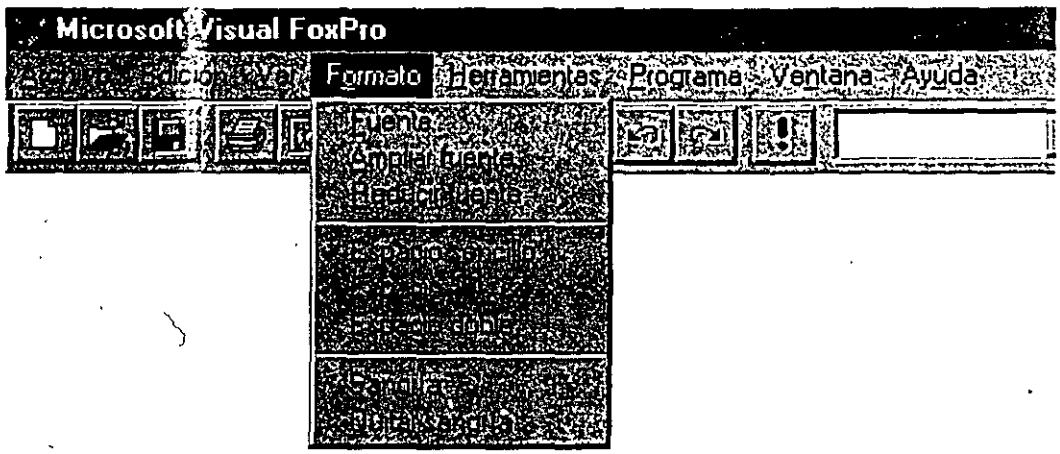


Figura 3-13

Generador de menús.

Primero, pongámonos de acuerdo en cuanto a terminología. Las palabras clave en el menú que aparecen a lo largo de la parte superior de la pantalla son llamadas *menús principales*. Cuando un submenú se crea como resultado de seleccionar un menú principal, su forma física se llama *menú desplegable* y las opciones individuales dentro de él se llaman *opciones de menú*. En el ejemplo mostrado en la figura 3-14, Archivo, Edición, y el resto de las palabras claves en la línea superior del menú son menús principales, mientras que Cerrar en el menú desplegable Archivo es una opción de menú.

Figura 3-14



Menú desplegable.

Por ahora usted necesita sólo dos menús principales en el menú. Primero, agregue un menú principal llamado Cliente, utilizando la columna Resultado para indicar qué tipo de selección de menú desea. Hay cuatro opciones.

- Comandos
- Título de menú
- Submenú
- Procedimiento

Usted quiere un comando, una sola línea de código, así que seleccione Comandos y luego teclee:

```
DO FORM CLIENTE
```

Luego cree un segundo menú principal, llamado Salir, e incluya el sencillo comando:

```
CLEAR EVENTS
```

Para ver cómo se ve el menú, haga clic en Presentación preliminar. El menú en la parte superior de la pantalla será reemplazado con el menú que acaba de generar. Debe tener exactamente dos opciones: Cliente y Salir. La C en Cliente y la S en Salir están subrayadas, lo que significa que son *teclas clave*; una vez que el menú esté activado, ya sea con la tecla ALT o F10 y oprimiendo C, se ejecutará el comando Do Form Cliente, mientras que S ejecutará el comando Clear Events. Usted verá lo que significa Clear Events en un momento.

Para guardar el menú y salir, seleccione ya sea Archivo, Cerrar del menú de Visual FoxPro o haga clic en el botón Cerrar en la esquina superior izquierda del Generador de menús. Cuando el cuadro de diálogo le pregunta por el nombre del menú (con la extensión .MNX), teclee MENU. Así, el trabajo que usted introdujo se guardará en un archivo llamado MENU.MNX y su archivo memo asociado MENU.MNT. Esto lo regresará al Administrador de proyectos.

Agregue un programa principal

Usted está casi listo para ejecutar un ejemplo. Sólo necesita un programa principal para controlar el flujo del proyecto. Así que haga clic en Código, luego en Programas y después en el control Nuevo. Obtendrá una ventana como la de la figura 3-15. Por el momento, el título de la ventana será Programa1. Teclee las siguientes líneas de código:

```
OPEN DATABASE DATOS1
CLEAR SCREEN
DO MENU.MPR
READ EVENTS
CLOSE DATABASES
SET SYSMENU TO DEFAULT
```

Utilice ya sea Archivo, Cerrar o haga doble clic en el botón Cerrar ventana, y teclee el nombre PRINCIP. La extensión .PGR se agrega automáticamente y usted regresa al Administrador de proyectos.

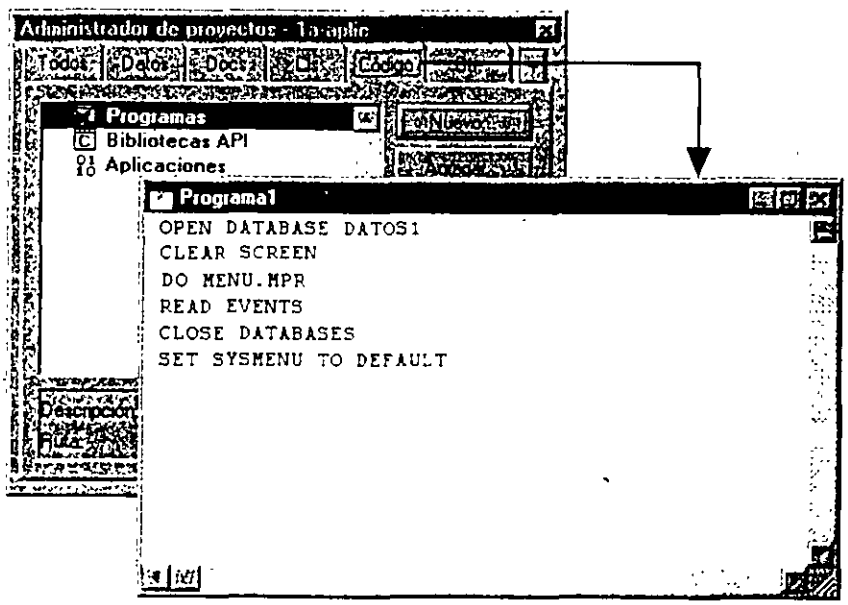


Figura 3-15

Ventana de código para introducir código de programa.

17

Note el punto negro y grande al lado del formulario Cliente. Éste le indica a Visual FoxPro que inicie su aplicación con el código en el formulario Cliente. En lugar de eso usted querrá iniciar con PRINCIP.PRG, así que haga clic con el botón secundario del ratón en Princip, en la sección Código, para resaltar Principal y desplegar el menú contextual Administrador de proyectos. Haga clic en Establecer principal. El punto se moverá hacia la izquierda de Principal. Ahora su aplicación hará lo siguiente:

1. Abrir la base de datos Datos1, logrando con esto que estén disponibles el archivo Cliente, otros archivos y relaciones.
2. Reemplazar el menú estándar de Visual FoxPro con el suyo. Note que el menú que se menciona tiene la extensión de su código generado, por ejemplo, MENU.MPR. Desde este punto en adelante, el menú que usted generó será el único menú dentro de la aplicación hasta que usted especifique SET SYSMENU TO DEFAULT.
3. Iniciar el comando READ EVENTS. Esto le permite a su formulario y a otros objetos estar activos. El comando está activo hasta que el usuario selecciona Salir, el cual inicia el comando Clear Events. En ese instante, el control pasa a la instrucción que está inmediatamente después del comando Read Events. Es del tipo del ciclo Do...EndDo, y Clear Events es como Salir.
4. Quitar los formularios que permanezcan en la pantalla.
5. Cerrar todos los archivos del usuario.
6. Restablecer el menú original de Visual FoxPro.

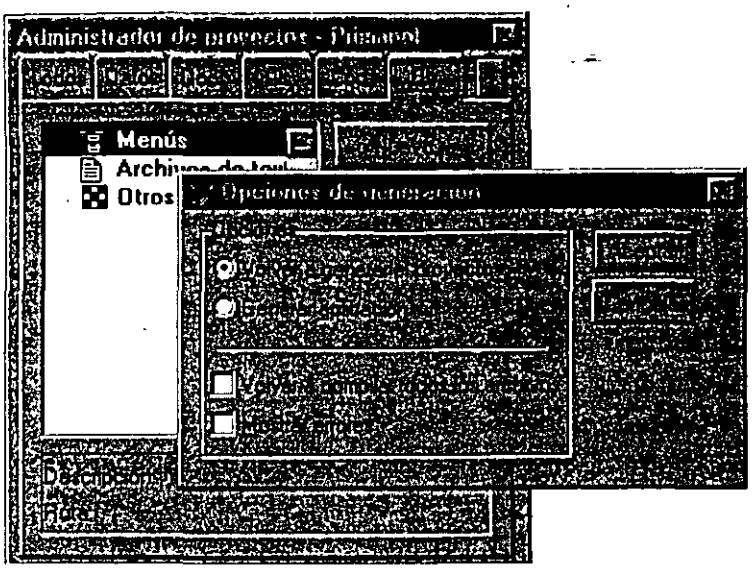
Genere la aplicación

Lo último que debe hacer antes de ejecutar su aplicación es generarla. Haga clic en Generar en los controles del lado derecho de la pantalla del Administrador de proyectos. Verá el cuadro de diálogo mostrado en la figura 3-16, incluyendo tres opciones mutuamente excluyentes (el mecanismo usado para esto dentro de Visual FoxPro son los grupos de opciones):

- Volver a generar el proyecto
- Generar aplicación
- Generar ejecutable

18

Figura 3-16



Opciones para generar una aplicación.

La opción Volver a generar el proyecto, realmente lee a través de los componentes de su proyecto y crea el archivo del proyecto, agregando elementos a los que se alude en sus pantallas, programas y menús. Usted puede introducir PRINCIP.PRG e indicarle a Visual FoxPro Volver a generar el proyecto, y encontrará el resto de los componentes. Esto es muy útil cuando usted tira a la basura su archivo de proyecto. Pero desde luego, eso nunca le pasará...

Generar aplicación, genera un archivo de salida de Visual FoxPro con la extensión .APP, el cual contiene todos los elementos nombrados en su archivo de proyecto, en formato compilado. Para puristas está realmente sacralizado. Se escribe en una forma compacta, la cual el cargador de Visual FoxPro puede leer y trabajar. Queda también en una forma en la que los usuarios no pueden desenredarlo fácilmente, de esta manera proporciona un nivel de seguridad.

Años atrás, algunos corazones rotos se enamoraron de Clipper debido a que producía archivos .EXE. Visual FoxPro también produce archivos .EXE, pero éstos no son sacralizados. La diferencia entre ellos y el formato .APP es que cuando usted ejecuta un .APP, necesita hacerlo directamente utilizando Visual FoxPro (VFP nombredeprog). Usted puede distribuir sus programas como archivos .EXE para usuarios que no tienen una copia de Visual FoxPro. Si usted genera un .EXE, la sintaxis para ejecutar el programa es simplemente *nombredeprog,* por ejemplo si el nombre es MIAPLI.EXE, usted utiliza MIAPLI.

Una aplicación compilada en formato .EXE reconoce internamente que tiene que enlazar a los archivos en tiempo de ejecución de Visual FoxPro. (Desde luego, usted

debe proporcionar a sus usuarios copias de los archivos en tiempo de ejecución, disponibles en la edición profesional de Visual FoxPro.) Sin embargo, como nosotros no estamos generando una aplicación en tiempo de ejecución, seleccionemos la segunda opción, Generar aplicación.

Ahora vea las dos casillas de verificación en la parte inferior de la pantalla. Utilice Volver a compilar todos los archivos, para asegurarse de que el archivo del proyecto está completamente generado con todos los elementos incluidos. Esto requiere una explicación. Cuando usted modifica un archivo, el sistema operativo cambia su etiqueta de fecha/hora. El Administrador de proyectos guarda la etiqueta de fecha/hora de cada elemento que está en la estructura de su archivo. Cuando usted selecciona Generar aplicación, el Administrador de proyectos compara la fecha en el archivo de proyecto con la fecha que está en el directorio para cada elemento del proyecto. Todo lo que esté con una etiqueta de fecha/hora más reciente se vuelve a compilar. Así que si usted obtiene una copia revisada de código fuente de un programa que otro desarrollador ha modificado y su reloj no está sincronizado con el de usted, el Administrador de proyectos podría pensar que no es necesario volver a compilarlo.

Normalmente, Visual FoxPro coloca cualquier mensaje acerca de errores que se encontraron durante la compilación dentro de un archivo con el mismo nombre de su aplicación y con la extensión .ERR. Si usted selecciona Mostrar errores, una ventana de edición aparecerá al final del paso generado para mostrarle sus errores. Si usted lo prefiere, puede dejar esta casilla sin seleccionar. Si no está seleccionada, entonces la única indicación de que hubo errores es el hecho de que la barra de errores en el menú Proyecto estará activa. Si usted selecciona Errores, la ventana Errores se abrirá y le mostrará sus errores de compilación.

Tal vez no sea necesario decirlo, pero no duele repetirlo. El mero hecho de que su programa se compile sin errores, de ninguna forma indica que se ejecutará sin ningún error. Ni si quiera existe una correlación. Visual FoxPro es un lenguaje muy complaciente. Usted se puede referir a campos que no existen, lo cual es muy extraño entre los lenguajes de programación. Esto es por lo que Visual FoxPro no revisa en el momento de la compilación qué nombres de campos existen en los archivos que usted abre en el tiempo de ejecución. De hecho, ni siquiera los revisa cuando usted los abre. Sólo importa que el campo exista en el tiempo en que usted se acerque al momento de referirse a él. Este espíritu de hacer las cosas fáciles se congracia consigo mismo para muchos programadores principiantes debido a que usted puede, después de todo, conseguir que el programa se ejecute, aun si estalla cien veces.

Hecho esto como es debido, la generación ha finalizado, y no parece haber errores. Así que démosle una oportunidad.

Ejecute el programa

Para ejecutar el programa, haga clic en Programa, Ejecutar, del menú y seleccione 1A-APLIC.APP, o vaya a la ventana Comandos y escriba:

```
DO 1A-APLIC
```

Todo lo que sucede es que el menú cambia del actual menú de Visual FoxPro a uno nuevo, con las dos opciones Cliente y Salir. Sin embargo, haga clic en Cliente y verá su bella pantalla, como se muestra en la figura 3-17.

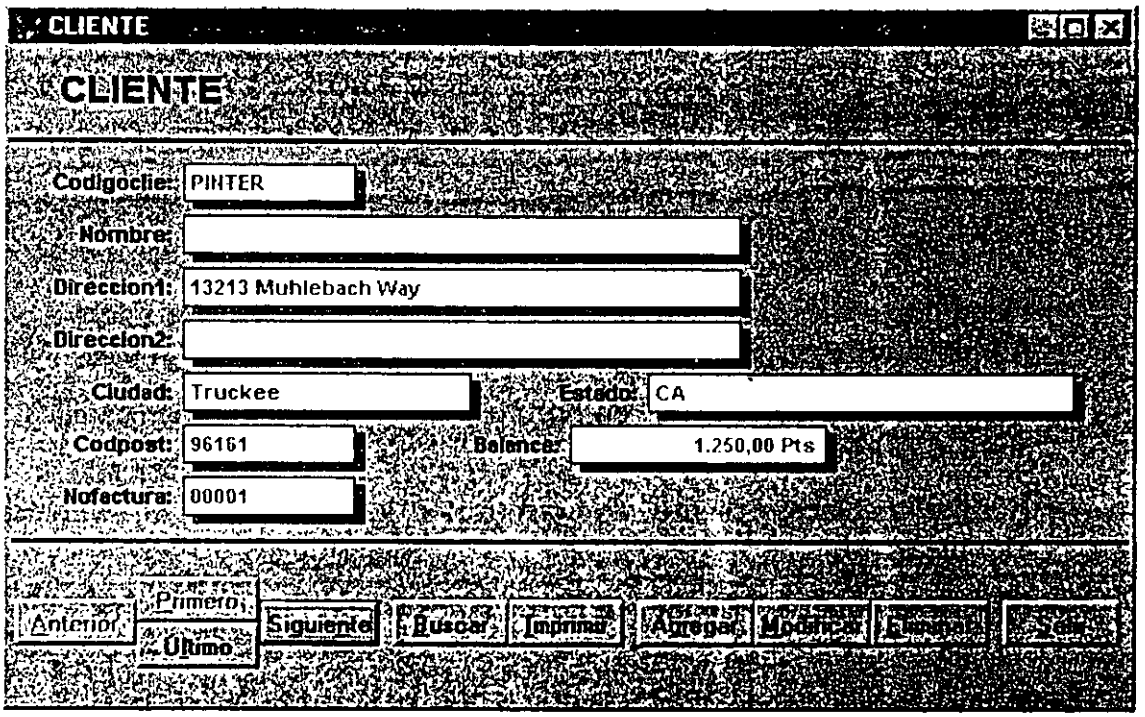


Figura 3-17

Pantalla Cliente.

Intentemos algunas cosas. El cursor probablemente se encuentre en el campo CodigoClie. Aunque intente escribir, no pasará nada. Los campos de entrada están desactivados. Haga clic en el botón Agregar. Usted puede ahora llenar una pantalla de información. Haga clic en Guardar para guardar la información. Agregue otro registro, y estará listo para probar su programa.

Cuando usted haga clic en Primero, irá al primer registro en orden alfabético de la clave CodigoClie. Haga clic en Siguiente, y saltará al siguiente registro. Haga clic

2/

en Siguiente de nuevo, y los botones Siguiente y Último se desactivarán. Usted no puede ir más allá del final de su archivo; si hace clic en el botón Siguiente, nada pasará.

Ésta es una buena característica para integrar dentro del software. Los usuarios están acostumbrados a ser advertidos con alarmas y mensajes desagradables cuando intentan ir más allá del final de los datos. Ocultar y desactivar opciones no disponibles es un gran paso en el aspecto de hacer que el software no sea hostil para los usuarios.

Haga clic en el botón Salir y la pantalla Cliente desaparecerá. Haga clic en el menú principal Salir, y volverá a la ventana Comandos con el menú original de Visual FoxPro. ¿Cinco minutos para diseñar un programa que funciona? Bienvenido a Visual FoxPro.

Adentro del formulario generado

Demos un vistazo más de cerca a cómo funcionan los formularios. Hay cuatro formas de abrir el formulario:

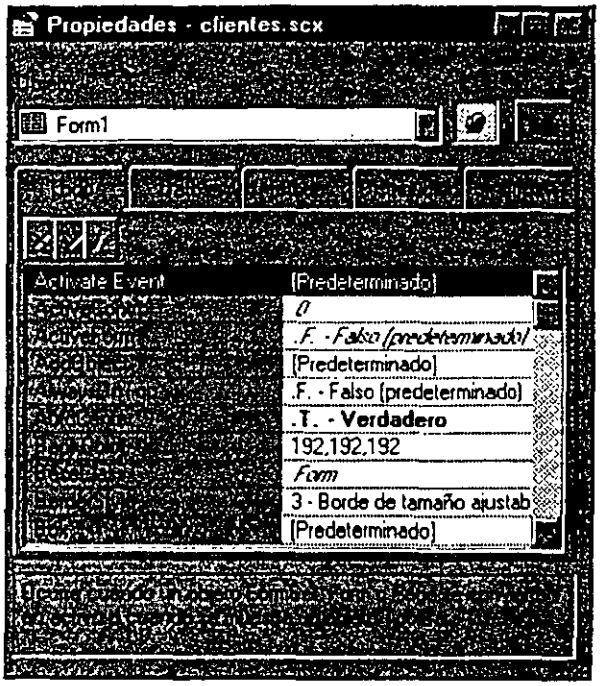
- Haga doble clic en la palabra "cliente" en Formularios dentro del Administrador de proyectos.
- Resalte la palabra "cliente" en Formularios, luego seleccione Modificar archivo del menú de Visual FoxPro.
- Resalte la palabra "cliente" en Formularios, luego haga clic en el botón Modificar.
- Resalte la palabra "cliente" en Formularios, luego presione ENTER.

Una vez que usted está viendo el formulario, haga clic en cualquier parte dentro del área del campo CodigoClie. Verá una serie de marcas cuadradas alrededor del título y el área de entrada de datos del campo. Todo el grupo, que consiste en un título, un campo de entrada y el reflejo de una sombra, es un *objeto*. Fue generado por el Asistente para formularios utilizando la biblioteca de clases WizStyle que se encuentra en el subdirectorio \VFP\WIZARDS. Tendremos más al respecto de esto posteriormente.

Ahora usted puede ver las propiedades del objeto seleccionado. Como es usual, hay muchas maneras para desplegar la ventana Propiedades. Puede hacer clic con el botón secundario del ratón en cualquier parte del objeto para producir el menú

contextual Objeto y luego hacer clic en Propiedades. O puede seleccionar Ver, Propiedades, del menú de Visual FoxPro. Usted verá la ventana Propiedades que se muestra en la figura 3-18, en el lado derecho de la pantalla.

Figura 3-18



Ventana Propiedades.

Propiedades y métodos

Las propiedades y los métodos es donde toma lugar la mayoría de la acciones dentro de la interfaz de Visual FoxPro. Veamos estos dos nuevos aspectos de Visual FoxPro.

Propiedades

La ventana Propiedades consiste de un marco de página de control con cinco fichas:

Todo Muestra todos los tipos de propiedades.

Datos Muestra sólo la fuente de los datos del objeto. La máscara de entrada (cláusula InputMask), si el campo es originalmente de sólo lectura, y si hacer clic en el objeto termina la lectura o no.

Distribución Muestra sólo el color, la visibilidad, la fuente, el estilo y algunas otras propiedades relacionadas con la apariencia del objeto.

Métodos Muestra sólo métodos. Los métodos son funciones pegadas al objeto. Hay un método para cada evento pegado a un objeto, y los objetos pueden tener cualquier número de eventos, incluyendo Load, Click, DoubleClick, DragDrop y GotFocus. Usted puede agregar métodos (pero no eventos) a cualquier objeto.

Otros Muestra solamente información sobre clases, modo de arrastrar, estado activo o inactivo, y algunos otros.

La lista de opciones en cada página de cada marco de página depende del tipo de objeto seleccionado. Usted selecciona un objeto haciendo clic sobre él o seleccionándolo de la lista de objetos. Por ejemplo, haga clic en cualquier parte del fondo del formulario y el nombre del formulario, Form1, aparecerá en el cuadro del nombre del objeto (propiedad name).

Mientras un objeto está seleccionado, usted puede cambiar sus propiedades haciendo clic en la propiedad, luego ya sea escribiendo un valor o haciendo clic repetidamente en la propiedad para pasar a través de los valores que puede tomar. Si usted intenta escribir un valor inválido, FoxPro se lo hará saber.

En el caso de objetos que son a su vez colecciones de otros objetos, haga clic en el grupo con el botón secundario del ratón, luego seleccione Edición. Una elegante caja azul-verde rodea el grupo. Hasta que usted la deje de seleccionar, cualquier cambio que se haga a una propiedad se aplica a todos los miembros del grupo.

La programación con objetos consiste en gran parte en qué propiedades existen, qué valores pueden asumir y qué efecto se produce tomando como base el valor que usted seleccione. Por ejemplo, los objetos transparentes permiten que se pueda ver a través de ellos el color de fondo, mientras que los objetos opacos tienen sus propios colores de fondo. Mediante sólo cambiar los fondos de los objetos en un formulario de opaco a transparente, usted puede cambiar fácilmente la apariencia y el carácter de un formulario.

Los métodos consisten de eventos y otros métodos, Los métodos de eventos son una lista arreglada de eventos que FoxPro colecta en una forma que es análoga a interrupciones. En el instante en que ocurre un evento, el método (código) correspondiente a ese evento se ejecuta. Además, usted puede ejecutar un método en cualquier momento refiriéndose a su nombre. Esta sintaxis será explorada con profundidad posteriormente, pero como un ejemplo rápido, si el botón Salir tiene un código Click, usted puede ejecutarlo durante el proceso del botón Anterior al incluir la línea:

```
THISFORM.SALIR.Click
```

el código del método del evento Click del botón Anterior. Esto especifica ejecuta el código del método.click para el botón Salir en el formulario que está activo justo ahora". Bajo las manos del antiguo FoxPro esto se vería algo así como DO (Salir.Click) IN (THISFORM), si se nos permitiera inventar la sintaxis.

Hay mucho más acerca de este asunto de las clases, eventos y métodos, pero lo mantendremos al margen por ahora. La cosa importante que hay que recordar aquí es que, en muchos casos, en lugar de escribir código para hacer que se realice algo, usted sólo cambia las propiedades para un objeto. Y cuando está escribiendo código, usted puede simplemente asignar un nuevo valor a una propiedad de esta forma:

```
THISFORM.Backcolor = RGB(150,150,150)
```

Ahora veamos algo acerca de estos objetos y de que es lo qué hace su código. Haga doble clic en el botón Primero del fondo de la pantalla, y se abre una ventana de código con el título BUTTONSET1.Move. ¿Y qué es lo que ve? Nada, eso es. En la parte superior hay cuadros combinados mostrando que el objeto es ButtonSet1 y el procedimiento Move. Haga clic en la flecha hacia abajo en la parte derecha del cuadro de lista desplegable Objeto, y verá la lista que se muestra en la figura 3-19. Haga clic en el cuadro de lista desplegable Procedimiento, y se mostrará la lista de la figura 3-20. Si usted cambia el nombre de la selección en el cuadro combinado Objeto, la lista de procedimientos para la selección del objeto cambiará.

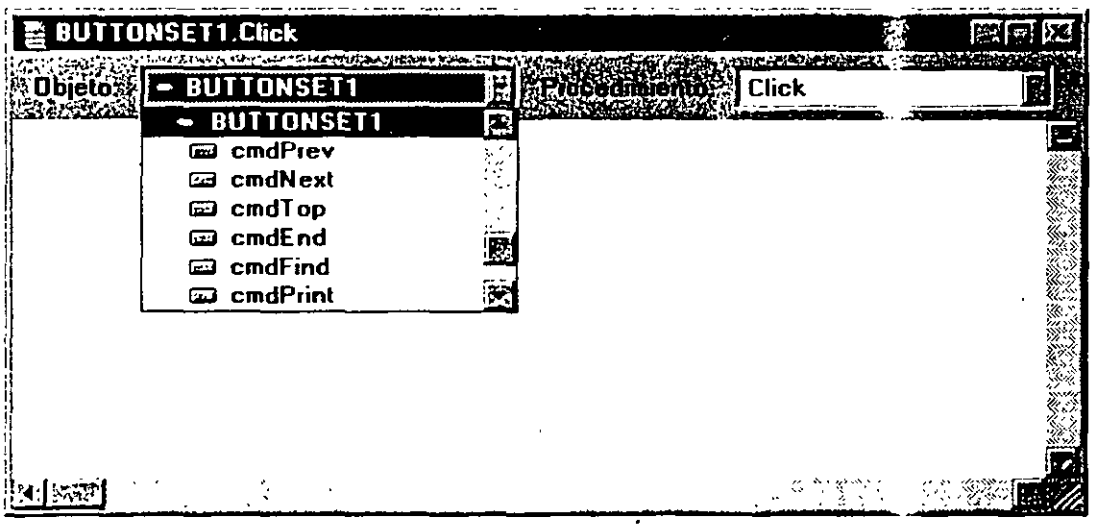
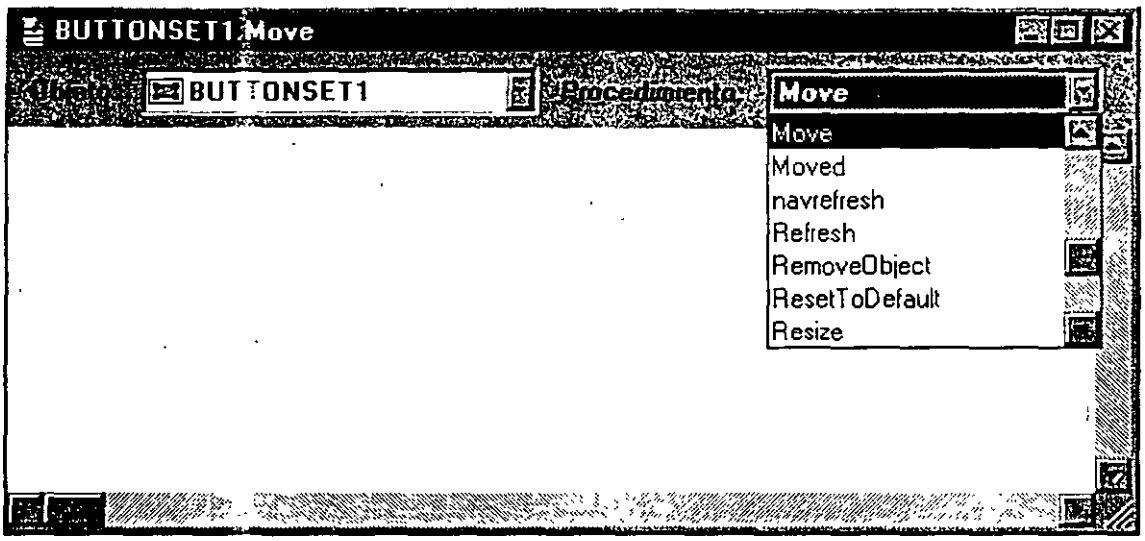


Figura 3-19

Cuadro de lista desplegable Objeto.

Figura 3-20



Cuadro de lista desplegable Procedimiento.

Los miembros de la lista de objetos tienen nombres como cmdNext, cmdTop y cmdBottom, así que ciertamente se ven como si estuvieran relacionados con los botones de la parte inferior de la pantalla. Pero, para parafrasear aquella maravillosa frase en el comercial de hamburguesas, ¿dónde está el código? La respuesta es que los controles en esta pantalla vienen de una biblioteca de clases, y el código está aún en la biblioteca de clases. Para poder verlo, usted tiene que abrir la clase.

¿Pero qué biblioteca de clases y qué clase? Con los botones seleccionados, haga clic en la ficha Otros de la ventana Propiedades. La biblioteca de clases es WIZSTYLE.VCX, y el nombre de la clase es TxtBtns. Como muchas de las nuevas características de Visual FoxPro, es sólo una rutina saber dónde buscar.

Así que volvamos al Administrador de proyectos, haga clic en Clases, luego haga clic en Agregar. Encuentre el archivo WIZSTYLE.VCX en el subdirectorio \VFP\WIZARDS y haga clic en él. La ventana del Administrador de proyectos reflejará la adición. Ahora resalte WizStyle y haga doble clic sobre él. O utilice cualquiera de los otros mecanismos para editar un archivo, tal como lo hizo para editar un formulario. Cuando usted está en el Administrador de proyectos, cada control sabe qué hacer.

Ahora usted puede hacer clic en Clase para mostrar su contenido (la clase sola WizStyle), haga clic en WizStyle para extender su lista de objetos, luego haga clic en TxtBtns. Esto produce la pantalla mostrada en la figura 3-21.

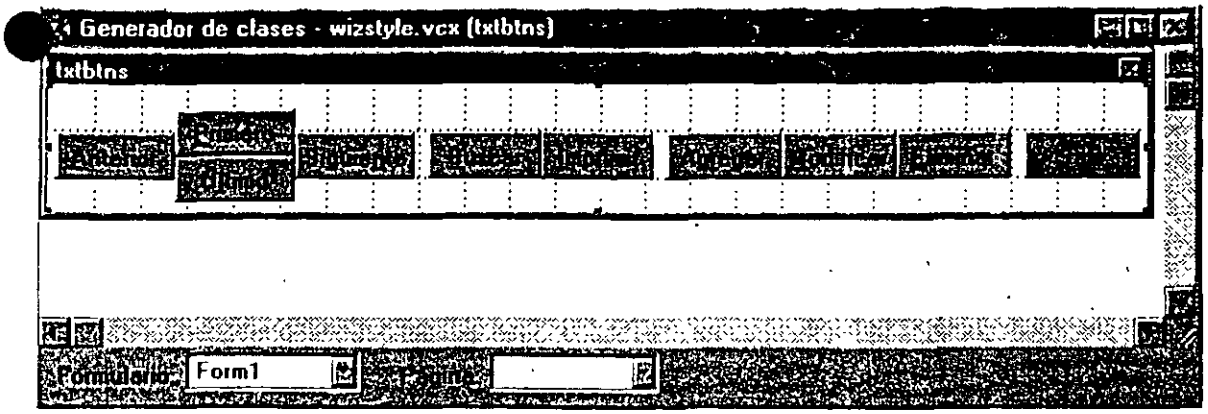


Figura 3-21

Clase TxtBtns.

Ahora cuando usted haga clic en algún lugar en el espacio en blanco entre el botón Último y Buscar, obtendrá una ventana de código como la que se muestra en la figura 3-22. Asegúrese de que TxtBtns se muestra en la ventana Objeto. Si no es así, haga clic en el cuadro de lista y busque TxtBtns. Esto es necesario debido a que las rutinas para propósitos generales para la clase están asociadas con el nombre de la clase, no con los nombres de sus objetos componentes. Si cmd selecciona, digamos, cmdNext, no podrá ver los procedimientos de propósitos generales en el cuadro de lista Procedimiento. (Para el antiguo manejo de FoxPro, estoy tentado a llamar a esto el máximo nivel de código para limpiar pantallas de métodos y métodos para cada uno de los botones con el código de cláusula válida para Gets individuales, pero es demasiado extenso.)

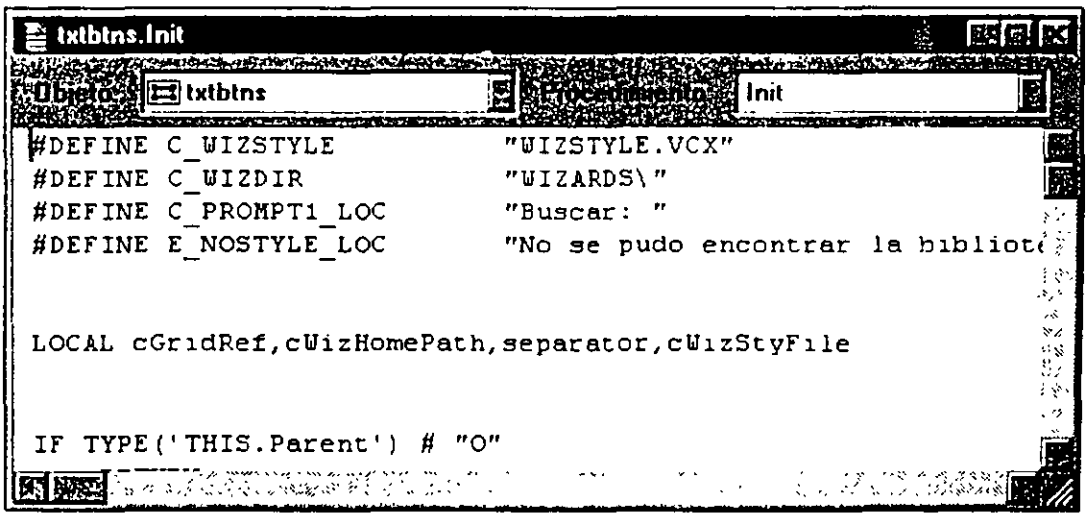


Figura 3-22

Ventana de código en la clase TxtBtns.

Eventos y métodos

Cada uno de los objetos de Visual FoxPro tiene ciertos eventos integrados. Usted puede agregar sus propios métodos, pero no puede cambiar el juego de eventos que vienen con él. Los eventos tienen métodos asociados a ellos. El evento Click tiene una ventana de código en la cual usted escribe el código del método click. Cuando usted hace clic en un control, su evento click es disparado y se ejecuta el código de su método click.

Los eventos en Visual FoxPro mejor deberían ser llamados *ganchos,* ellos le dan un lugar en donde enganchar su código. Para formularios, los métodos Load y Unload son particularmente interesantes, debido a que son llamados justo después de que un formulario ha sido creado y liberado, respectivamente. En el grupo de botones de la pantalla generada, el evento/método Init es la primera cosa que pasa.

El evento Init

Haga clic en el cuadro de lista Procedimiento para ir hacia abajo, y seleccione Init. Init es el primer evento de una clase que es ejecutada. Éste tiene exactamente dos líneas de código, las cuales se muestran cerca del final de la figura 3-22. Son:

```
THIS.InitVars()
THIS.ButtonRefresh()
```

Aunque los métodos de Visual FoxPro no requieren de paréntesis, los métodos proporcionados por el usuario pueden terminar con paréntesis, y de hecho los tienen, cuando han pasado parámetros o regresado un valor. En las dos líneas previas de código, THIS significa "busca este método en la clase a la que pertenece el objeto actual". ¿Por qué éstos son referidos como métodos en lugar de funciones?

- Son invocados de forma diferente.
- Viendo la forma en que son nombrados, sabemos lo que son.
- Bill quería que fuera de esta forma.

Así que de cualquier manera, ¿qué es lo que hacen InitVars y ButtonRefresh? InitVars establece valores para las variables locales de la clase TxtBtns. Los comentarios siguientes son nuestros, no del código generado.

28

```
* Crea arreglos locales y contadores
LOCAL aTablesUsed,nTablesUsed,i
DIMENSION aTablesUsed[1]

* Crea variables locales para controlar el dimensionamiento de los botones
* de navegación
THIS.TopFile = .F.
THIS.EndFile = .F.
THIS.AddMode = .F.

* Recuerde qué área antes de que el botón de navegación fuera seleccionado.
THIS.nWorkArea = SELECT()

* Estas propiedades no deberán ser usadas, son de uso reservado
* para el botón preliminar del Asistente para formularios
THIS.PreviewMode = IIF(TYPE("THIS.PreviewMode")#"L",.F.,THIS.PreviewMode)
* Determina el ambiente. La función IIF es muy
* útil para esta clase de cosas. Su sintaxis es
* x = IFF (condición, ValorSiesVerdadero, ValorSiesFalso)
* La función TYPE ("nombredevariable") regresa el tipo de datos
* o "U" para indefinido. Así que si las tres variables nombradas
* no existen aún, se establecen a .F.
* Si están ejecutando la pantalla en modo "preview", sale Init
THIS.PreviewInit = IIF(TYPE("THIS.PreviewInit")#"L",.T.,THIS.PreviewInit)

* Verificar el entorno de datos
DO CASE
CASE TYPE("THISFORM.DataEnvironment") = "O"
    THIS.UseDataEnv = .T.
    nTotMem = AMEMBERS(aMems,THISFORM.DataEnvironment,2)
    cDataEnvRef = "THISFORM.DataEnvironment"
CASE TYPE("THISFORMSET.DataEnvironment") = "O"
    THIS.UseDataEnv = .T.
    nTotMem = AMEMBERS(aMems,THISFORMSET.DataEnvironment,2)
    cDataEnvRef = "THISFORMSET.DataEnvironment"
    * también establezcamos el
    IF TYPE("THISFORM")="O" AND !THISFORM.VISIBLE
         THISFORM.VISIBLE = .T.
    ENDIF
OTHERWISE
    THIS.UseDataEnv = .F.
ENDCASE

IF THIS.UseDataEnv
WITH EVAL(m.cDataEnvRef)
* Verifiquemos la relación
FOR i = 1 TO m.nTotMem
IF UPPER(EVAL("."+aMems[m.i]+".BaseClass")) = "RELATION"
         THIS.oDataRelation = m.cDataEnvRef+"."+aMems[m.i]
    EXIT
ENDIF
ENDFOR
```

```
* Verificar las vistas
FOR i = 1 TO m.nTotMem
    IF UPPER(EVAL(".+aMems[m.i]+".BaseClass")) = "CURSOR"
    WITH EVAL("."+aMems[m.i])
IF CURSORGETPROP("SourceType",.Alias)#3
* Verifica si se requiere actualizar la consulta (sql) para registros
* eliminados.
* - Nota: vistas parametrizadas no se requieren ya que los registros serán
* enviados.
* Para remediar esta situación se puede colocar el comando SET DELETED ON
* en el evento
* BeforeOpenTables del entorno de datos.
IF THIS.oldSetDelete = "OFF" AND ATC("?",CURSORGETPROP("SQL",.Alias))=0
        =REQUERY(.Alias)
ENDIF
* Verifica si se realizaron las actualizaciones.
IF !CURSORGETPROP("SendUpdates",.Alias) AND !m.lShowedMess

        =MESSAGEBOX(C_NOUPDATEVIEW_LOC)
        lShowedMess = .T.
    ENDIF
ENDIF
ENDWITH
ENDIF
ENDFOR
ENDWITH
ENDIF

THIS.EditMode = IIF(TYPE("THIS.EditMode")#"L",.F.,THIS.EditMode)

IF ISREADONLY()
    WAIT WINDOW C_READONLY_LOC TIMEOUT 2
    THIS.EditMode = .F.
ENDIF

* Deshabilita ciertos botones
THIS.cmdAdd.Enabled = !ISREADONLY()
THIS.cmdEdit.Enabled = !ISREADONLY()
THIS.cmdDelete.Enabled = !ISREADONLY()

IF THIS.PreviewMode
    RETURN
ENDIF

THIS.GetGridRef()

IF THIS.UseDataEnv
    RETURN
ENDIF
```

```
* Guarda los parámetro de algunas variables del entorno, luego vuelve a
* establecerlas como se vaya necesitando.
THIS.oldSetFields = SET("FIELDS")
SET FIELDS OFF
THIS.oldSetDelete = SET("DELETED")
SET DELETED ON
THIS.oldReprocess = SET("REPROCESS")
SET REPROCESS TO 0
THIS.oldMultiLocks = SET("MULTILOCKS")
SET MULTILOCKS ON
THIS.oldRefresh = SET("REFRESH")
SET REFRESH TO 5

* Un valor optimista para el búfer ( propiedad '5' ) es lo mejor para
* condiciones normales
IF !EMPTY(ALIAS()) && "!" significa "NOT"
    THIS.oldBuffering=CursorGetProp("buffering")
* Carga una lista de alias de tablas abiertas
    m.nTablesUsed = AUSED(aTablesUsed)
    FOR i = 1 TO m.nTablesUsed
        IF CursorGetProp('sourcetype',aTablesUsed[m.i,1])#3
* Salta el "views"; sólo se colocan tablas en el búfer
        =CursorSetProp("buffering",5,aTablesUsed[m.i,1])
* Establece la propiedad buffering de la tabla a un optimista 5.
        ENDIF
    ENDFOR
ENDIF
```

Ésta es una precipitación muy drástica dentro del código de FoxPro; pienso que yo conozco al tipo que escribió esto, y él está tan lejos del resto de nosotros que ni siquiera es gracioso. Pero usted todavía puede decir qué hace el código. El método InitVars revisa si la pantalla está siendo ejecutada en modo preliminar, en dicho caso la pantalla de todos modos no operará con datos debido a que el modo preliminar sólo le permite ver cómo se verá su pantalla cuando sea ejecutada. Luego crea algunos apagadores lógicos locales. Las variables lógicas pueden tener sólo los valores .T. o .F., y pueden ser utilizadas para hacer algunas pruebas de condiciones muy legibles, por ejemplo:

```
IF Testing
    =PrintTest()
ENDIF
```

Finalmente, el programa asegura que todos los archivos utilizan un búfer optimista para tablas, lo cual significa que los registros están bloqueados sólo cuando se está escribiendo en ellos y, si un registro en blanco es agregado para datos y el usuario cancela la adición, no se deja un registro en blanco no utilizado dentro del archivo. FoxPro, de hecho, sólo agrega un registro en blanco cuando usted lo guarda. Verá en unos momentos qué es lo que significa guardar.

El otro método llamado en Init es ButtonRefresh. Los comentarios son del código generado originalmente, y son muy claros. El código se muestra aquí:

```
* Ésta es una rutina general para refrescar los botones.
* Asegúrese de que tiene seleccionada la tabla correcta.

IF SELECT()# THIS.nWorkArea
    SELECT (THIS.nWorkArea)
ENDIF

THIS.SetAllProp()
THIS.cmdFind.Enabled = !THIS.EditMode
THIS.cmdPrint.Enabled =  !THIS.EditMode
THIS.cmdExit.Enabled =  !THIS.EditMode
THIS.cmdDelete.Enabled =  !THIS.EditMode AND !ISREADONLY()
THIS.SetCaption()


* Aquí mostramos el código de SetAllProp()

LPARAMETER oContainer
LOCAL i,oControlParent,nCtrlCount

IF PARAMETERS() = 0
   m.oControlParent = THISFORM
ELSE
   m.oControlParent = m.oContainer
ENDIF

IF ATC("Pageframe",m.oControlParent.BaseClass)#0
   nCtrlCount = oControlParent.PageCount
ELSE
   nCtrlCount = oControlParent.ControlCount
ENDIF

FOR i = 1 TO m.nCtrlCount
   DO CASE
   CASE ATC("Pageframe",m.oControlParent.BaseClass)#0
        THIS.SetAllProp(m.oControlParent.Pages[m.i])

   CASE ATC("Container",m.oControlParent.Controls[m.i].BaseClass) # 0 OR;
        ATC("Page",m.oControlParent.Controls[m.i].BaseClass) # 0
        THIS.SetAllProp(m.oControlParent.Controls[m.i])

   CASE ATC(m.oControlParent.Controls[m.i].BaseClass,"CheckBox,TextBox, ;
   OleBoundControl")# 0
        m.oControlParent.Controls[m.i].Enabled = THIS.EditMode

   CASE ATC(m.oControlParent.Controls[m.i].BaseClass,"EditBox") # 0
        m.oControlParent.Controls[m.i].ReadOnly = !THIS.EditMode
```

```
        IF !THIS.HasMemo
            WITH m.oControlParent.Controls[m.i]
                THIS.EditForeColor = .ForeColor
                THIS.EditDisForeColor =  .DisabledForeColor
                THIS.EditBackColor = .BackColor
                THIS.EditDisBackColor =  .DisabledBackColor
                THIS.HasMemo = .T.
            ENDWITH
        ENDIF
        m.oControlParent.Controls[m.i].ForeColor =       ;
IIF(THIS.EditMode,THIS.EditForeColor,THIS.EditDisForeColor)
        m.oControlParent.Controls[m.i].BackColor =       ;
IIF(THIS.EditMode,THIS.EditBackColor,THIS.EditDisBackColor)

    CASE ATC(m.oControlParent.Controls[m.i].BaseClass,"Grid") # 0
        m.oControlParent.Controls[m.i].ReadOnly = !THIS.EditMode
        m.oControlParent.Controls[m.i].DeleteMark = THIS.EditMode
        ENDCASE
ENDFOR

* Aquí mostramos el código de This.SetCaption()
* Definición de cadenas misceláneas de texto:
#DEFINE ADD_CAPTION_LOC       "Ag\<regar"
#DEFINE EDIT_CAPTION_LOC      "\<Modificar"
#DEFINE REV_CAPTION_LOC       "Re\<vertir"
#DEFINE SAVE_CAPTION_LOC      "\<Guardar"

IF THIS.EditMode
    THIS.cmdAdd.Caption = SAVE_CAPTION_LOC
    THIS.cmdEdit.Caption = REV_CAPTION_LOC
ELSE
    THIS.cmdAdd.Caption = ADD_CAPTION_LOC
    THIS.cmdEdit.Caption = EDIT_CAPTION_LOC
ENDIF

**** Manejo del botón de navegación ****
LOCAL OldLockScreen,KeyValue,cFiltExpr
m.OldLockScreen = THISFORM.LockScreen
THISFORM.LockScreen = .T.

IF SELECT()#THIS.nWorkArea
    SELECT (THIS.nWorkArea)
ENDIF

IF !THIS.EditMode

    * Verifica el final del archivo
    THIS.EndFile = EOF() OR THIS.EndFile

    * Revisa para ver si estamos en el último registro
    IF !THIS.EndFile
        SKIP
```

```
            THIS.EndFile = EOF()
            SKIP -1
        ELSE
            GO BOTTOM
        ENDIF

        * Verifica el principio del archivo
        THIS.TopFile = BOF() OR EOF() OR THIS.TopFile

        * Revisa para ver si estamos en el primer archivo
        IF !THIS.TopFile
            SKIP -1
            THIS.TopFile = BOF()
            IF !THIS.TopFile
                SKIP
            ENDIF
        ENDIF

        IF THIS.TopFile
            GO TOP
        ENDIF

    ENDIF

    * Aquí mostramos el código de navRefresh()
    THIS.cmdTop.Enabled = !THIS.TopFile AND !THIS.EditMode
    THIS.cmdPrev.Enabled = !THIS.TopFile AND !THIS.EditMode
    THIS.cmdNext.Enabled = !THIS.EndFile AND !THIS.EditMode
    THIS.cmdEnd.Enabled = !THIS.EndFile AND !THIS.EditMode

    * Verifica si hay registros en la consulta
    DO CASE
    CASE THIS.PreviewMode OR ISREADONLY()
        * nada
    CASE THIS.EditMode
        THIS.cmdEdit.Enabled = .T.
    CASE RECCOUNT()=0 OR BOF() OR EOF()
        THIS.cmdEdit.Enabled = .F.
        THIS.cmdDelete.Enabled = .F.
    CASE !THIS.cmdEdit.Enabled
        THIS.cmdEdit.Enabled = .T.
        THIS.cmdDelete.Enabled = .T.
    ENDCASE

    * Actualiza cuadrícula para las vistas
    IF !THIS.EditMode AND !EMPTY(THIS.ViewKey)
        KeyValue = EVAL(THIS.ParentKey)
        DO CASE
        CASE TYPE(THIS.ParentKey) = "C"
            cFiltExpr = THIS.ViewKey + "=" + "["+m.KeyValue+"]"
        CASE TYPE(THIS.ParentKey) = "L"
            cFiltExpr = THIS.ViewKey
        CASE TYPE(THIS.ParentKey) = "D"
```

```
        cFiltExpr = THIS.ViewKey + "=" + "{"+DTOC(m.KeyValue)+"}"
    CASE TYPE(THIS.ParentKey) = "T"
        cFiltExpr = THIS.ViewKey + "=" + "{"+TTOC(m.KeyValue)+"}"
    OTHERWISE
        * Numérico
        cFiltExpr = THIS.ViewKey + "=" + ;
        ALLTRIM(STR(m.KeyValue,20,18))
    ENDCASE

    SELECT (THIS.GridAlias)
    DO CASE
    CASE .F. && consulta parametrizada
        * Establece parámetro aquí
        * =requery()
    CASE THIS.ViewType = 1    && vistas locales
        SET FILTER TO &cFiltExpr
    CASE THIS.ViewType = 2    && vistas remotas
    ENDCASE
    SELECT (THIS.nWorkArea)
ENDIF

THISFORM.Refresh()
THISFORM.LockScreen = m.OldLockScreen
```

Aquí es donde los botones son activados o desactivados cada vez que el puntero de registros se mueve de modo que los botones que representan opciones no disponibles puedan ser desactivados. Note que todo lo que se toma para desactivar un botón es el siguiente código:

```
THISFORM.nombredeboton.Enabled = .F.
```

FoxPro sabe a qué formulario pertenece cada botón –THISFORM y el instante en que la propiedad activada se establece en .F., el objeto se muestra como desactivado, lo cual significa que su texto se ha reducido y no puede ser seleccionado. Aquí es donde nosotros estábamos acostumbrados a usar SHOW GET variable de memoria DISABLE en FoxPro 2.6. Además, note que el método THIS.SetCaption es llamado a mitad del camino a través del código. Como los títulos pueden ser cambiados fácilmente –el título es una propiedad de la mayoría de los objetos– la rutina SetCaption tiene un código justo como esto:

```
IF THIS.EditMode
    THIS.cmdAdd.Caption = SAVE_CAPTION_LOC
    THIS.cmdEdit.Caption = REV_CAPTION_LOC
ELSE
    THIS.cmdAdd.Caption = ADD_CAPTION_LOC
    THIS.cmdEdit.Caption = EDIT_CAPTION_LOC
ENDIF
```

Estas últimas siete líneas de código significan que los botones Agregar y Modificar pueden volver a establecerse a Guardar y Revertir, respectivamente. Usted verá más tarde que el código Click para estos dos botones tiene que estar dispuesto a negociar con cualquier caso; por ejemplo, si usted está editando o agregando, el título del botón Agregar será Guardar y necesita hacer todo lo necesario para salvar, agregar o editar información.

Siguiente

¿Qué hay acerca de los botones de navegación actuales? Aquí está el código del evento Click para Siguiente:

```
SELECT (THIS.parent.nWorkArea)
SKIP 1
THIS.Parent.EndFile = .F.
THIS.Parent.TopFile = .F.
THIS.Parent.ButtonRefresh()
```

Esto significa "asegúrese de que está en el área de trabajo correcta" (¿recuerda dónde fue dado el valor correcto a nWorkArea en InitVars?), luego salta al siguiente registro, establece las dos variables locales TopFile y EndFile, y llama a navRefresh para volver a establecer lo que sea necesario.

Anterior

El código del evento Click para cmdPrev es idéntico, excepto que en lugar de Skip 1, tiene Skip -1. Si usted lo establece correctamente, es muy sencillo:

```
SELECT (THIS.parent.nWorkArea)
SKIP -1
THIS.Parent.TopFile = .F.
THIS.Parent.EndFile = .F.
THIS.Parent.ButtonRefresh()
```

Primero y Último

Estos dos métodos son prácticamente iguales a Siguiente y Anterior, excepto por el manejo del botón. Lo siguiente es el código del evento Click para cmdTop en TxtBtns:

```
SELECT (THIS.parent.nWorkArea)
GO TOP
THIS.Parent.TopFile = .T.
THIS.Parent.EndFile = .F.
THIS.Parent.ButtonRefresh()
```

y éste es el código del evento Click para cmdBottom en TxtBtns:

```
SELECT (THIS.parent.nWorkArea)
GO BOTTOM
THIS.Parent.TopFile = .F.
THIS.Parent.EndFile = .T.
THIS.Parent.ButtonRefresh()
```

Buscar

Buscar algún registro en particular implica un poco más de trabajo. A continuación se muestra el código para el evento Click de cmdFind:

```
LOCAL oSearchDlog

oSearchDlog = CREATE("searchform")
oSearchDlog.SHOW()

* Reestablecer con respecto al anterior
THIS.Parent.TopFile = .F.
THIS.Parent.EndFile = .F.
THIS.Parent.ButtonRefresh()
```

Este código crea una instancia de la clase SearchForm en la biblioteca de clase WizStyle. El código de diálogo LocFile se utiliza en caso de que Visual FoxPro tenga problemas para encontrar la biblioteca de clase WizStyle cuando se ejecute. Se ha añadido WizStyle al proyecto, así que los formularios pueden encontrarlo y utilizar SearchForm siempre que se genere algo. Pero el código generado no asume que WIZSTYLE.VCX estará disponible, debido a que usted puede correr un formulario independiente. Así que incluye código para encontrar la biblioteca de clase. Ésta es la razón por la cual es tan difícil de leer. La clase SearchForm es un objeto genérico, usted puede ejecutarlo desde cualquier pantalla. El código que lee:

```
CASE ATC(C_WIZSTYLE,SET("BIBLCLASE")) # 0
```

rara vez es confuso. Quiere decir "Si la biblioteca de clase cuyo nombre está contenido en la variable de memoria c_WizStyle se encuentra en la lista de bibliotecas de clase adjuntas devueltas por la función Set...". La función ATC regresa un número si encuentra el objeto de su búsqueda, y un cero si no lo

encuentra. Y # significa Not Equal; generalmente yo utlizo < >. ATC y su compañero AT son funciones útiles, pero no es fácil seguirlas cuando usted está tratando de leer código fuente.

Agregar

Agregar un registro es muy sencillo. Sólo tenga cuidado con eso, si el usuario selecciona previamente Agregar, el título del botón Agregar será ahora Guardar, y su método para salvar los datos es en ese caso utilizar el método UpdateRows( ). Si usted no está familiarizado con la sintaxis, ! significa NOT (No):

```
#DEFINE OPT_CANCEL                  0
#DEFINE OPT_ADD_PARENT              1
#DEFINE OPT_ADD_CHILD               2
#DEFINE OPT_ADD_BOTH                3
#DEFINE MB_Q_YESNO                  36
#DEFINE MB_A_YES                    6
#DEFINE C_KEYFLDNOUPDATE_LOC        "El campo que relaciona la vista de la ;
cuadrícula con el origen de datos primario no es actualizable. "+   "¿Desea ;
simplemente agregar un nuevo registro a la tabla primaria?"

#DEFINE C_BADCHILDKEY_LOC           "Los campos que relacionan las tablas ;
primaria y secundaria no tienen el mismo tipo de datos. "+  "¿Desea ;
simplemente agregar un nuevo registro a la tabla primaria?"

#DEFINE C_NOCHILDUPDATE_LOC         "El origen de datos secundario es una ;
vista y no envía actualizaciones. "+  "¿Desea agregar un nuevo registro a la ;
tabla primaria?"

#DEFINE C_NOOBJ_LOC                 "Falló la creación de la clase de ;
formulario Agregar registro. Verifique o vuelva a instalar el archivo ;
WIZSTYLE.VCX."
#DEFINE C_NOUPDATE_LOC              "No puede agregar un nuevo registro ;
porque la vista o las vistas seleccionadas no envían actualizaciones."
#DEFINE C_NOUPDATE2_LOC             "No puede agregar un nuevo registro ;
porque la vista o las vistas seleccionadas no envían actualizaciones y el ;
origen de datos secundario tiene una clave principal."

LOCAL oSearchDlog,oAddRec,cChildAlias,cPapaAlias,i,lPrimeKey
LOCAL cPapaKey,cChildKey,nSaveSess,oRel,cTagName,lBadViewKey,nSaveRec,nSaveRec2
LOCAL lBadChildKey,lUpdatableParentKey,lNoSendParentUpdates,lNoSendChildUpdates

DO CASE
CASE THIS.Parent.EditMode

** Código para agregar un registro
    THIS.Parent.UpdateRows()
```

```
CASE EMPTY(THIS.Parent.GridRef)                  && no utiliza cuadrícula

    ** Código para adicionar un registro
    THIS.Parent.OldAlias = ALIAS()    && guardar el alias en caso de revertir
    THIS.Parent.OldRec = RECNO()      && salvar el registro en caso de revertir
    IF CURSORGETPROP("SourceType")#3 AND !CURSORGETPROP("SendUpdates")
        =MESSAGEBOX(C_NOUPDATE_LOC)
        RETURN
    ENDIF
    APPEND BLANK

OTHERWISE

    ** código para adicionar el código
    THIS.Parent.OldAlias = ALIAS()    & salvar el alias en caso de revertir
    THIS.Parent.OldRec = RECNO()      && salvar el registro en caso de revertir
    lPrimeKey = .F.                   && es la llave del índice primario

    IF !EMPTY(THIS.Parent.oDataRelation)
        oRel = EVAL(THIS.Parent.oDataRelation)
    ENDIF

    DO CASE
    CASE TYPE("m.oRel") = "O" && entorno de datos
        WITH oRel
            cPapaAlias = .ParentAlias
            cPapaKey = .RelationalExpr
            cChildAlias = .ChildAlias
            cChildKey = .ChildOrder
        ENDWITH
            oRel = .NULL.    && reiniciar
    CASE !EMPTY(THIS.Parent.ViewKey)  && utilizando vistas
            cPapaAlias = ALIAS()
            cPapaKey = THIS.Parent.ParentKey
            cChildAlias = THIS.Parent.GridAlias
            cChildKey = THIS.Parent.ViewKey

    * Necesita saber si el campo clave es actualizable
    IF CURSORGETPROP("SourceType",m.cChildAlias)#3 AND ;

    ATC(",",+m.cChildKey+",",",",",+CURSORGETPROP("UpdatableFieldList", ;
    m.cChildAlias)+",")=0
    lBadViewKey = .T.
    ENDIF
    OTHERWISE
    cPapaAlias = ALIAS()
    cPapaKey = RELATION(1)
    cChildAlias = THIS.Parent.GridAlias
    cChildKey = ORDER(m.cChildAlias)
    ENDCASE
```

```
* Revisa para ver si ambas claves son del mismo tipo de datos
SELECT (THIS.Parent.GridAlias)
cGridKeyType = ""

IF EMPTY(THIS.Parent.ViewKey) AND !EMPTY(m.cChildKey) && Tabla regular
                                                      && usada
* Obtiene la clave de campo hija ya que sólo la etiqueta de índice
* está aquí
* Si ésta es una expresión, poner por omisión un espacio en blanco.
cTagName = ""
FOR i = 1 TO TagCount("")
     IF UPPER(TAG(m.i)) == UPPER(m.cChildKey)
          cTagName = KEY(m.i)
          lPrimeKey = PRIMARY(m.i)
          EXIT
     ENDIF
ENDFOR

     * Revisa si tenemos una expresión de índice aquí.
     IF ATC("(",m.cTagName)#0 OR ATC("+",m.cTagName)#0
          cChildKey = ""
     ELSE
          cChildKey = m.cTagName
          cGridKeyType = TYPE(m.cChildKey)
     ENDIF
ENDIF

IF CURSORGETPROP("SourceType",m.cPapaAlias)#3 AND !CURSORGETPROP("Send ;
Updates",m.cPapaAlias)
     lNoSendParentUpdates = .T.
ENDIF
IF CURSORGETPROP("SourceType",m.cChildAlias)#3 AND !CURSORGETPROP("Send ;
Updates",m.cChildAlias)
     lNoSendChildUpdates = .T.
ENDIF

* Revisa si hay dos vistas y no envia actualizaciones
IF m.lNoSendParentUpdates AND m.lNoSendChildUpdates
     =MESSAGEBOX(C_NOUPDATE_LOC)
     RETURN
ENDIF

SELECT (THIS.Parent.OldAlias )
IF !EMPTY(m.cGridKeyType) AND m.cGridKeyType # TYPE(m.cPapaKey)
     lBadChildKey = .T.
ENDIF

DO CASE
CASE m.lNoSendParentUpdates AND (m.lBadViewKey OR m.lBadChildKey)
     =MESSAGEBOX(C_NOUPDATE_LOC)
     RETURN
CASE m.lPrimeKey AND m.lNoSendParentUpdates
     =MESSAGEBOX(C_NOUPDATE2_LOC)
```


40

```
        RETURN
    CASE m.lNoSendChildUpdates    && la vista de uno a varios en
                                  && varios no envíe las actualizaciones
        IF MESSAGEBOX(C_NOCHILDUPDATE_LOC,MB_Q_YESNO) # MB_A_YES
            RETURN
        ENDIF
        APPEND BLANK
    CASE m.lBadViewKey    && la llave de la vista de varios no es
                          && actualizable
        IF MESSAGEBOX(C_KEYFLDNOUPDATE_LOC,MB_Q_YESNO) # MB_A_YES
            RETURN
        ENDIF
        APPEND BLANK
    CASE m.lBadChildKey       && mala llave hija - datos diferentes de la
                              && tabla padre
        IF MESSAGEBOX(C_BADCHILDKEY_LOC,MB_Q_YESNO) # MB_A_YES
        RETURN
        ENDIF
        APPEND BLANK
    OTHERWISE
    * Verifica si el campo llave es actualizable
    lUpdatableParentKey = .T.
    IF CURSORGETPROP("SourceType",m.cPapaAlias)#3 AND ATC(","+m.cPapaKey ;
    +",",","+CURSORGETPROP("UpdatableFieldList",m.cPapaAlias)+",")=0
    * cPapaKey = ""
    lUpdatableParentKey = .F.
    ENDIF

* Verifica la expresión en cPapaKey entonces no actualizar el valor de la llave
IF ATC("(",m.cPapaKey)#0 OR ATC("+",m.cPapaKey)#0
    cPapaKey = ""
ENDIF

nSaveSess = SET("DATASESSION")
oAddRec = CREATE("GridAddRec")
IF TYPE("m.oAddRec") # "O"
    =MESSAGEBOX(C_NOOBJ_LOC)
    RETURN
ENDIF
oAddRec.ChildPrimaryKey = m.lPrimeKey
oAddRec.UpdatableParentKey = m.lUpdatableParentKey
oAddRec.NoSendUpdates = m.lNoSendParentUpdates
oAddRec.KeyField = m.cPapaKey
oAddRec.KeyValue =
IF(!EMPTY(m.cPapaKey),EVAL(m.cPapaKey),"")
oAddRec.RunAddForm()
IF oAddRec.AddOption = OPT_CANCEL
    RETURN
ENDIF

SET DATASESSION TO nSaveSess
```

```
SELECT (THIS.Parent.OldAlias)

IF TYPE("oAddRec.KeyValue") = "C"
    oAddRec.KeyValue = TRIM(oAddRec.KeyValue)
ENDIF

* Añade un registro a la tabla padre
nSaveRec = RECNO()
IF INLIST(oAddRec.AddOption,OPT_ADD_PARENT,OPT_ADD_BOTH)
    IF EMPTY(m.cPapaKey)
    APPEND BLANK IN (m.cPapaAlias)
    ELSE
     INSERT INTO (m.cPapaAlias) ((oAddRec.KeyField)) VALUES(oAddRec.KeyValue)
    ENDIF
ENDIF
nSaveRec2 = RECNO()

    * Añade un regristro hijo
    IF INLIST(oAddRec.AddOption,OPT_ADD_CHILD,OPT_ADD_BOTH)
         * Necesita verificarse
         GO m.nSaveRec
         IF EMPTY(m.cChildKey) OR TYPE(m.cChildKey)#TYPE('oAddRec.KeyValue')
              APPEND BLANK IN (m.cChildAlias)
              ELSE
         INSERT INTO (m.cChildAlias) ((m.cChildKey)) VALUES(oAddRec.KeyValue)
         ENDIF
         GO m.nSaveRec2
    ENDIF
    ENDCASE
ENDCASE

THIS.Parent.EditMode = !THIS.Parent.EditMode
THIS.Parent.AddMode = THIS.Parent.EditMode
THIS.Parent.TopFile = .F.
THISFORM.LockScreen = .T.
THIS.Parent.ButtonRefresh()
THIS.Parent.NavRefresh()
THISFORM.LockScreen = .F.
```

Modificar

Modificar es muy similar a Agregar. Si el título del botón Modificar ha sido cambiado a Revertir (cancelar), el programa no deberá guardar ninguna modificación a los datos que estaban en la pantalla, ya que datos de más de una tabla pueden estar en pantalla, todas las tablas utilizadas deberán ser revertidas.

```
#DEFINE C_NOUPDATE_LOC "No puede modificar porque la vista o las vistas ;
seleccionadas no envían actualizaciones."
```


42

```
LOCAL lNoSendParentUpdates,lNoSendChildUpdates
LOCAL aTablesUsed,nTablesUsed,i

** Revirtiendo el registro
IF THIS.Parent.EditMode

    IF THIS.Parent.UseDataEnv
         SELECT (THIS.Parent.OldAlias)
         =TableRevert(.T.)
         IF !EMPTY(THIS.Parent.GridAlias)
         SELECT (THIS.Parent.GridAlias)
         =TableRevert(.T.)
         ENDIF
    ELSE
         DIMENSION aTablesUsed[1]
         m.nTablesUsed = AUSED(aTablesUsed)
         FOR i = 1 TO m.nTablesUsed
              =TableRevert(.T.,aTablesUsed[m.i,1])
         ENDFOR
    ENDIF

    * Va atrás al lugar original
    SELECT (THIS.Parent.OldAlias)
    IF RECCOUNT() < THIS.Parent.OldRec       && adicionar el registro al
                                             && final
         GO TOP
    ELSE
         GO THIS.Parent.OldRec
    ENDIF
ELSE

    * Verifica si las vistas pueden ser modificadas
    IF CURSORGETPROP("SourceType")#3 AND !CURSORGETPROP("SendUpdates")
         lNoSendParentUpdates = .T.
    ENDIF
    IF
         !EMPTY(THIS.Parent.GridAlias) AND ;
         CURSORGETPROP("SourceType",THIS.Parent.GridAlias)#3 AND ;
         !CURSORGETPROP("SendUpdates",THIS.Parent.GridAlias)
              lNoSendChildUpdates= .T.
    ENDIF
         IF (m.lNoSendChildUpdates AND m.lNoSendParentUpdates) OR ;
         (EMPTY(THIS.Parent.GridAlias) AND m.lNoSendParentUpdates)
         =MESSAGEBOX(C_NOUPDATE_LOC)
         RETURN
    ENDIF

    THIS.Parent.OldAlias = ALIAS()     && salvar el alias en caso de
                                       && revertir
    THIS.Parent.OldRec = RECNO()       && salvar el registro en caso de
                                       && revertir
ENDIF
```

```
** Editando registro
THIS.Parent.EditMode = !THIS.Parent.EditMode
THIS.Parent.AddMode = .F.
THISFORM.LockScreen = .T.
THIS.Parent.ButtonRefresh()
THIS.Parent.NavRefresh()
THISFORM.LockScreen = .F.
```

Guardar

Si ha estado agregando o editando, quizás usted requiera guardar o revertir (cancelar). El título del botón Agregar cambia a Guardar cuando se está agregando y modificando, y la variable EditMode se establece a verdadera (.T.). Así que es llamado el método UpdateRows:

```
#DEFINE OPT_CANCEL                  0
#DEFINE OPT_ADD_PARENT              1
#DEFINE OPT_ADD_CHILD               2
#DEFINE OPT_ADD_BOTH                3
#DEFINE MB_Q_YESNO                  36
#DEFINE MB_A_YES                    6

#DEFINE C_KEYFLDNOUPDATE_LOC        "El campo que relaciona la vista de la ;
cuadrícula con el origen de datos primario no es actualizable. "+    ;
"¿Desea simplemente agregar un nuevo registro a la tabla primaria?"

#DEFINE C_BADCHILDKEY_LOC           "Los campos que relacionan las tablas  ;
primaria y secundaria no tienen el mismo tipo de datos. "+  "¿Desea   ;
simplemente agregar un nuevo registro a la tabla primaria?"

#DEFINE C_NOCHILDUPDATE_LOC         "El origen de datos secundario es una  ;
vista y no envía actualizaciones. "+  "¿Desea agregar un nuevo registro ;
a la tabla primaria?"

#DEFINE C_NOOBJ_LOC                 "Falló la creación de la clase de ;
formulario Agregar registro. Verifique o vuelva a instalar el archivo ;
WIZSTYLE.VCX."
#DEFINE C_NOUPDATE_LOC              "No puede agregar un nuevo registro ;
porque la vista o las vistas seleccionadas no envían actualizaciones."
#DEFINE C_NOUPDATE2_LOC      "No puede agregar un nuevo registro porque ;
la vista o las vistas seleccionadas no envían actualizaciones y el origen ;
de datos secundario tiene una clave principal."

LOCAL oSearchDlog,oAddRec,cChildAlias,cPapaAlias,i,lPrimeKey
LOCAL cPapaKey,cChildKey,nSaveSess,oRel,cTagName,lBadViewKey,nSaveRec,nSaveRec2
LOCAL lBadChildKey,lUpdatableParentKey,lNoSendParentUpdates,lNoSendChildUpdates
```

```
DO CASE
CASE THIS.Parent.EditMode

    ** Código para salvar registro
    THIS.Parent.UpdateRows()

CASE EMPTY(THIS.Parent.GridRef)              && no utilizando cuadrícula

    **Código para adicionar registro
    THIS.Parent.OldAlias = ALIAS()    && salvar alias en caso de revertir
    THIS.Parent.OldRec = RECNO()      && salvar registro en caso de adicionar
    IF CURSORGETPROP("SourceType")#3 AND !CURSORGETPROP("SendUpdates")
        =MESSAGEBOX(C_NOUPDATE_LOC)
        RETURN
    ENDIF
    APPEND BLANK

OTHERWISE

    ** Código para adicionar registro
    THIS.Parent.OldAlias = ALIAS() && salvar alias en caso de revertir
    THIS.Parent.OldRec = RECNO()   && salvar registro en caso de revertir
    lPrimeKey = .F.                && es la clave primaria de la tabla hijo
                                   && índice

    IF !EMPTY(THIS.Parent.oDataRelation)
        oRel = EVAL(THIS.Parent.oDataRelation)
    ENDIF

DO CASE
CASE TYPE("m.oRel") = "O"     && entorno de datos
    WITH oRel
        cPapaAlias = .ParentAlias
        cPapaKey = .RelationalExpr
        cChildAlias = .ChildAlias
        cChildKey = .ChildOrder
    ENDWITH
    oRel = .NULL.  && inicializar
CASE !EMPTY(THIS.Parent.ViewKey)      && utilizando vistas
    cPapaAlias = ALIAS()
    cPapaKey = THIS.Parent.ParentKey
    cChildAlias = THIS.Parent.GridAlias
    cChildKey = THIS.Parent.ViewKey

    * Necesita verificar si el campo clave es actualizable
    IF CURSORGETPROP("SourceType",m.cChildAlias)#3 AND ATC(","+m.cChildKey ;
    +",","+CURSORGETPROP("UpdatableFieldList",m.cChildAlias)+",")=0
        lBadViewKey = .T.
    ENDIF
OTHERWISE
    cPapaAlias = ALIAS()
    cPapaKey = RELATION(1)
    cChildAlias = THIS.Parent.GridAlias
```

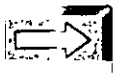
```
        cChildKey = ORDER(m.cChildAlias)
ENDCASE

* Probar para verificar que ambas llaves son iguales
SELECT (THIS.Parent.GridAlias)
cGridKeyType = ""

    IF EMPTY(THIS.Parent.ViewKey) AND !EMPTY(m.cChildKey)    && tabla regular
                                                             && utillizada
* Conseguir el campo llave de la tabla hijo ya que sólo la llave del índice
* se encuentra
* Si está es una expresión entonces asignar por omisión blanco.
cTagName = ""
FOR i = 1 TO TagCount("")
    IF UPPER(TAG(m.i)) == UPPER(m.cChildKey)
         cTagName = KEY(m.i)
         lPrimeKey = PRIMARY(m.i)
         EXIT
    ENDIF
    ENDFOR

    * Verifica si hemos tenido una expresión de índice aquí.
    IF ATC("(",m.cTagName)#0 OR ATC("+",m.cTagName)#0
         cChildKey = ""
    ELSE
         cChildKey = m.cTagName
         cGridKeyType = TYPE(m.cChildKey)
    ENDIF
ENDIF

    IF CURSORGETPROP("SourceType",m.cPapaAlias)#3 AND ;
!CURSORGETPROP("SendUpdates",m.cPapaAlias)
         lNoSendParentUpdates = .T.
    ENDIF
    IF CURSORGETPROP("SourceType",m.cChildAlias)#3 AND ;
!CURSORGETPROP("SendUpdates",m.cChildAlias)
         lNoSendChildUpdates = .T.
    ENDIF

    * Verifica si dos vistas no envían actualizaciones
    IF m.lNoSendParentUpdates AND m.lNoSendChildUpdates
         =MESSAGEBOX(C_NOUPDATE_LOC)
         RETURN
    ENDIF

    SELECT (THIS.Parent.OldAlias )
    IF !EMPTY(m.cGridKeyType) AND m.cGridKeyType # TYPE(m.cPapaKey)
         lBadChildKey = .T.
    ENDIF

    DO CASE
    CASE m.lNoSendParentUpdates AND (m.lBadViewKey OR m.lBadChildKey)
         =MESSAGEBOX(C_NOUPDATE_LOC)
```

```
        RETURN
CASE m.lPrimeKey AND m.lNoSendParentUpdates
        =MESSAGEBOX(C_NOUPDATE2_LOC)
        RETURN
CASE m.lNoSendChildUpdates    && la vista hijo no envía actualizaciones
        IF MESSAGEBOX(C_NOCHILDUPDATE_LOC,MB_Q_YESNO) # MB_A_YES
                RETURN
        ENDIF
        APPEND BLANK
CASE m.lBadViewKey &&la llave no es actualizable
        IF MESSAGEBOX(C_KEYFLDNOUPDATE_LOC,MB_Q_YESNO) # MB_A_YES
                RETURN
        ENDIF
        APPEND BLANK
CASE m.lBadChildKey         && llave del hijo errónea - diferente tipo de
                            && dato que el padre
        IF MESSAGEBOX(C_BADCHILDKEY_LOC,MB_Q_YESNO) # MB_A_YES
                RETURN
        ENDIF
        APPEND BLANK
OTHERWISE
        * Revisa si el campo llave es actualizable
        lUpdatableParentKey = .T.
        IF CURSORGETPROP("SourceType",m.cPapaAlias)#3 AND ;
        ATC(","+m.cPapaKey+",",","+CURSORGETPROP("UpdatableFieldList", ;
        m.cPapaAlias)+",")=0
                * cPapaKey = ""
                lUpdatableParentKey = .F.
        ENDIF

        * Revisa expresión en cPapaKey, si es verdadero entonces no
        * actualizar el registro con keyvalue
        IF ATC("(",m.cPapaKey)#0 OR ATC("+",m.cPapaKey)#0
                cPapaKey = ""
        ENDIF

        nSaveSess = SET("DATASESSION")
        oAddRec = CREATE("GridAddRec")
        IF TYPE("m.oAddRec") # "O"
                =MESSAGEBOX(C_NOOBJ_LOC)
                RETURN
        ENDIF
        oAddRec.ChildPrimaryKey = m.lPrimeKey
        oAddRec.UpdatableParentKey = m.lUpdatableParentKey
        oAddRec.NoSendUpdates = m.lNoSendParentUpdates
        oAddRec.KeyField = m.cPapaKey
        oAddRec.KeyValue = IIF(!EMPTY(m.cPapaKey),EVAL(m.cPapaKey),"")
        oAddRec.RunAddForm()
        IF oAddRec.AddOption = OPT_CANCEL
                RETURN
        ENDIF
```

```
            SET DATASESSION TO nSaveSess
            SELECT (THIS.Parent.OldAlias)

            IF TYPE("oAddRec.KeyValue") = "C"
                oAddRec.KeyValue = TRIM(oAddRec.KeyValue)
            ENDIF

            * Adicionar registro a tabla padre
            nSaveRec = RECNO()
        IF INLIST(oAddRec.AddOption,OPT_ADD_PARENT,OPT_ADD_BOTH)
            IF EMPTY(m.cPapaKey)
                APPEND BLANK IN (m.cPapaAlias)
            ELSE
                INSERT INTO (m.cPapaAlias)
                ((oAddRec.KeyField))VALUES(oAddRec.KeyValue)
            ENDIF
        ENDIF
        nSaveRec2 = RECNO()

        * Adicionar registro hijo
        IF INLIST(oAddRec.AddOption,OPT_ADD_CHILD,OPT_ADD_BOTH)
            * Es necesario verificar
            GO m.nSaveRec
            IF EMPTY(m.cChildKey) OR TYPE(m.cChildKey)#TYPE('oAddRec.KeyValue')
                APPEND BLANK IN (m.cChildAlias)
            ELSE
                INSERT INTO (m.cChildAlias) ((m.cChildKey)) ;
                VALUES(oAddRec.KeyValue)
                ENDIF
                GO m.nSaveRec2
            ENDIF
        ENDCASE
    ENDCASE

    THIS.Parent.EditMode = !THIS.Parent.EditMode
    THIS.Parent.AddMode = THIS.Parent.EditMode
    THIS.Parent.TopFile = .F.
    THISFORM.LockScreen = .T.
    THIS.Parent.ButtonRefresh()
    THIS.Parent.NavRefresh()
    THISFORM.LockScreen = .F.
```

Este código es demasiado complejo para tratar de explicarlo línea por línea en este momento. Sin embargo, al final de este libro habremos cubierto todo lo mostrado aquí. Básicamente si usted selecciona Guardar, el registro que usted ha introducido se convierte en un miembro permanente de la tabla. En versiones anteriores de FoxPro, usted tenía que copiar valores de campos a variables de memoria, y luego colocarlas de nuevo dentro de los campos si el usuario seleccionaba Guardar. Además, usted no especificaba APPEND BLANK hasta que Guardar era seleccionado debido a que era relativamente complicado borrar un registro en

blanco que había sido añadido con el único propósito de proporcionar campos de entrada en blanco a la pantalla, especialmente en un ambiente multiusuario. Así que este código es difícil de leer, pero es fácil vivir con él.



Salir

Salir es el botón que cierra la pantalla y regresa el control al evento cíclico. Este código es muy simple:

```
RELEASE THISFORM
```

Eso es todo. Como el evento cíclico controlado por Read Events continúa, usted no tiene que hacer nada más. Sin embargo, el comando Release disparará el código del evento Destroy, de manera que veremos lo siguiente:

```
* Restaurar las variables del entorno
LOCAL nTablesUsed,aTablesUsed,i,nDECursors,aDECursors,cDataEnvRef
DIMENSION aTablesUsed[1]

IF TYPE('THIS.Parent') # "O"
    RETURN
ENDIF

IF TYPE("THIS.oldTalk") = "C" AND THIS.oldTalk="ON"
    SET TALK ON
ENDIF

* Los servidores OLE todavía pueden enviar información a los campos
* generales, aunque no se encuentren en modo de edición. Necesitamos
* reinicializar el búfer a 1 para que éste no se actualice por el
* servidor OLE.
* También es posible que se salgan mientras están editando.
IF THIS.UseDataEnv
    DIMENSION aDECursors[1]
    DO CASE
    CASE TYPE("THISFORM.DataEnvironment") = "O"
         nDECursors = AMEMBERS(aDECursors,THISFORM.DataEnvironment,2)
         cDataEnvRef = "THISFORM.DataEnvironment"
    CASE TYPE("THISFORMSET.DataEnvironment") = "O"
         nDECursors = AMEMBERS(aDECursors,THISFORMSET.DataEnvironment,2)
         cDataEnvRef = "THISFORMSET.DataEnvironment"
    ENDCASE
    FOR i = 1 TO m.nDECursors
         WITH EVAL(m.cDataEnvRef + "." + aDECursors[m.i])
              IF USED(.ALIAS) AND ATC("CURSOR",.BaseClass)!0 AND ;
              CursorGetProp("sourcetype",.ALIAS)=3 AND ;
              CursorGetProp("buffering",.ALIAS)>1
```

```
                =TableRevert(.T.,.ALIAS)
                =CursorSetProp("buffering",1,.ALIAS) && Buffering de
                                                     && tabla optimista
            ENDIF
        ENDWITH
    ENDFOR
ENDIF

* Saltar si se está utilizando pervisualizar
IF THIS.PreviewMode
    RETURN
ENDIF

IF THIS.oldSetDelete = "OFF"
    SET DELETED OFF
ENDIF
SET REPROCESS TO THIS.oldReprocess
SET MESSAGE TO
SELECT (THIS.nWorkArea)

IF THIS.UseDataEnv
    RETURN
ENDIF

* El código es para soportar formas sin entorno de datos
m.nTablesUsed = AUSED(aTablesUsed)
FOR i = 1 TO m.nTablesUsed
    IF USED(aTablesUsed[m.i,1]) AND ATC(".TMP",DBF(aTablesUsed[m.i,1]))=0
                                                   && Salta las vistas
=CursorSetProp("buffering",THIS.oldBuffering,aTablesUsed[m.i,1]) && buffering
                                                                 && de tabla
                                                                 && optimista
    ENDIF
ENDFOR

IF THIS.oldMultiLocks = "OFF"
    SET MULTILOCKS OFF
ENDIF

IF THIS.oldSetFields = "ON"
    SET FIELDS ON
ENDIF

SET REFRESH TO THIS.oldRefresh
```

El método Destroy es el inverso de Init. Es llamado cuando el formulario se elimina de la ventana. Mire la próxima sección para que vea qué es lo que quieren los amigos de Microsoft que se haga cuando desactivamos un formulario.

Esto sólo coloca las cosas de regreso en donde las encontró. Usted guarda los valores Set cuando carga el formulario, y restablece los parámetros guardados cuando sale de

él. Algunas veces es necesario en las aplicaciones de bases de datos con múltiples bases de datos y donde un formulario podría tener diferentes parámetros que otro. Los desarrolladores de MS/Fox son bastante cuidadosos, y aunque a mí me gustaría decir que esto es exagerado, usualmente saben de lo que están hablando.

Resumen

Es una vista muy bien detallada del ejemplo más simple de aplicación que usted puede generar con FoxPro. No es que el código del Asistente sea simple, pues no lo es. Pero la aplicación, un archivo de pantalla simple y llano llamado por una sencilla entrada de menú, no hace que sea más sencillo. Le dará algunas cosas en que pensar mientras aprende más acerca de FoxPro. Para el momento en que usted esté a prueba otra vez en el capítulo 9, todo esto le parecerá más familiar.

En el próximo capítulo, le daremos una visión general del entorno de trabajo completo de Visual FoxPro. Verá dónde están localizadas más herramientas, y empezará a desarrollar instintos para solucionar problemas con el equipo de trabajo de Visual FoxPro.

4

El menú de Visual FoxPro

CAPITULO 4

LA herramienta principal que usted utiliza para moverse alrededor de Visual FoxPro es el menú de sistema de FoxPro. El menú consiste en menús principales a lo largo de la parte superior de la pantalla, el menú desplegable aparece cuando selecciona el menú principal correspondiente y los métodos abreviados asignados a las barras de menús. En este capítulo usted verá todas las opciones disponibles que hay en el menú de FoxPro. Al principio no necesitará usar todas ellas, pero eventualmente, le serán de gran utilidad. Los menús principales más importantes son:

- Archivo
- Edición
- Ver
- Formato
- Herramientas
- Programa
- Ventana
- Ayuda

Durante ciertas operaciones, como la creación de un formulario o un informe, aparecerán menús principales adicionales. De hecho, el menú Formato estará presente debido a que la ventana Comandos está abierta en la pantalla. Seleccione Ventana, Ocultar del menú principal y verá a qué me refiero. (Utilice CTRL-F2 para obtener la ventana Comandos de nuevo.)

Debajo de cada menú principal hay un menú desplegable, consistente en una o más opciones de menú. Algunas de las opciones, cuando usted las selecciona, desempeñan una sola operación. Otras, como Compilar e Imprimir, producen un cuadro de diálogo que requiere datos adicionales. Las opciones de menú que producen cuadros de diálogo son seguidas de tres puntos, como éstos ...

En algunos casos, una opción de menú tendrá en sí misma un número de opciones posibles. Si es el caso, tendrá una flecha que apunta hacia la derecha. La opción de menú Asistentes, del menú principal Herramientas, es un ejemplo. Los menús que usted genere para interactuar con sus propios programas funcionan igual que el menú de Visual FoxPro. De hecho, como usted verá, estos menús son el menú de Visual FoxPro.

Menús principales y barras de herramientas eventuales

Además, cuando usted activa algunos generadores, otros menús principales y sus menús desplegables asociados se insertarán dentro de la barra de menús en medio de otros dos menús principales existentes. Por ejemplo, cuando el Generador de bases de datos está activo, el menú principal Bases de datos se añade entre los menús principales Formato y Herramientas.

Las barras de herramientas están íntimamente relacionadas con el menú de sistema. Inmediatamente debajo de los menús principales de FoxPro, verá una serie de iconos: carpeta de archivos, discos flexibles, impresora, etc. Estos iconos pertenecen a la barra de herramientas estándar de Visual FoxPro. Hay once barras de herramientas:

- Paleta de colores
- Generador de bases de datos
- Controles de formularios
- Generador de formularios
- Distribución
- Presentación preliminar
- Generador de consultas
- Controles de informes
- Generador de informes
- Estándar
- Generador de vistas

Cada una de estas barras de herramientas contiene varios iconos que, en general, representan opciones que usted puede seleccionar desde uno de los menús principales de Visual FoxPro. Las barras de herramientas son otra forma de tomar atajos. Por ejemplo, en la barra de herramientas estándar que aparece cuando usted inicia Visual FoxPro, los tres primeros iconos son Nuevo, Abrir y Guardar, como usted podrá ver si coloca el puntero sobre cualquiera de ellos y lo deja ahí. El pequeño cuadro amarillo que aparece con la descripción después de un pequeño lapso se llama consejo de ayuda. Estas tres opciones, así como las dos siguientes, Imprimir y Presentación preliminar, vienen directamente del menú

principal Archivo de Visual FoxPro. El siguiente, Ortografía, viene del menú principal Herramientas, y los siguientes cinco son del menú principal Edición. Ejecutar (el signo de admiración), el cual puede utilizar para ejecutar programas externos, no se encuentra en ningún lugar dentro del menú de sistema de Visual FoxPro. Los dos siguientes son los iconos para llegar a la ventana Comandos o para abrir la ventana Ver (las últimas dos opciones de menú del menú principal Ventana), seguidos de cuatro opciones de menú asistentes (Formulario, Informes, Asistente para Autoformularios, Asistente para Autoinformes) y Ayuda.

En el capítulo 14, verá cómo crear barras de herramientas para sus propias aplicaciones. Por ahora, es un alivio saber que puede usar un solo clic del ratón, en lugar de dos golpes de tecla, para lograr muchas operaciones que utiliza con frecuencia.

Administre el menú de Visual FoxPro

Internamente, el menú se llama también SysMenu o _MSysMenu, dependiendo del comando. Todos los elementos del menú tienen nombres, los cuales están dados en el apéndice A. Por ejemplo, el menú principal Edición es llamado _MEdit. El menú de Visual FoxPro está siempre activo cuando usted inicia Visual FoxPro. Para ocultarlo al usuario, use el comando siguiente:

```
SET SYSMENU OFF
```

Para dejarlo disponible nuevamente, utilice:

```
SET SYSMENU ON
```

Para mostrar sólo ciertas opciones, emplee:

```
SET SYSMENU TO nombredelmenu1 nombredelmenu2...
```

Puede definir su propio sistema de menú con sólo sus menús principales y opciones de menú o con una combinación de los suyos y los de FoxPro. Cuando crea un sistema de menú, es una práctica estándar guardar el menú en uso con:

```
PUSH MENU _MSYSMENU
```

Luego defina su propio sistema de menú. Cuando termine, restablezca el menú anterior con:

```
POP MENU _MSYSMENU
```

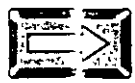
Para lograr que su propio menú actúe como el menú normal de Visual FoxPro utilice:

```
SET SYSMENU AUTOMATIC
```

Este comando se ejecuta automáticamente durante la generación de menús cuando FoxPro escribe el código fuente basado en su menú "definido por el usuario", de modo que usted no tendrá que utilizarlo. Y si pierde el menú o lo vuelve a definir en alguna forma que las opciones sean inútiles, aquí está su salida de emergencia de programador:

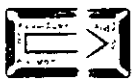
```
SET SYSMENU TO DEFAULT
```

Este comando lo regresa al menú que tenía cuando inició Visual FoxPro.



Ventanas modales y el menú de Visual FoxPro

Algo que se mencionará aquí, sólo para completar, es el concepto de ventana modal. Usted necesita responder a algunas ventanas, especialmente cuadros de diálogo, antes de que otra cosa pueda suceder. Éstas son llamadas *ventanas modales*, en oposición a las *ventanas sin modo*. Si usted establece la propiedad Window Type como Modal, dentro de Otros en el marco de página Propiedades, el menú no será accesible mientras esa ventana esté activa.



Active el menú de Visual FoxPro

Puede activar el menú de Visual FoxPro mediante cualquiera de estas cuatro maneras:

- La tecla ALT
- La tecla F10
- Haciendo doble clic con el botón secundario del ratón
- El comando Activate Menú

Haga una selección desde el menú de Visual FoxPro

Usted también puede tener acceso a los menús principales en cualquiera de cuatro formas. Puede oprimir la tecla de acceso, la letra subrayada (usualmente la primera) en el título del menú principal. Puede asignar una tecla de acceso en sus propios menús mediante preceder la letra con una diagonal invertida y el símbolo menor que. Por ejemplo:

```
Número de Tarjeta de \<Crédito
```

producirá este menú principal:

```
Número de Tarjeta de Crédito
```

Puede hacer clic en el menú deseado con el botón primario del ratón, usar las teclas de flecha arriba y abajo para resaltar una opción de menú en el menú que aparece, y luego oprimir la tecla ENTER. También puede presionar el método abreviado asignado a la opción de menú, si existe uno. El método abreviado es único en el sentido de que el menú no tiene que ser desplegado para que funcione.

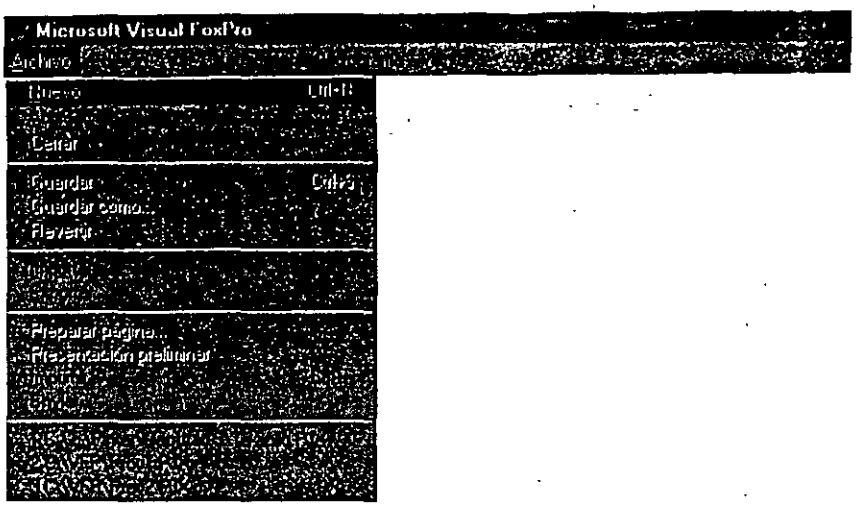
Ahora explore el menú de sistema de Visual FoxPro y vea el mundo de herramientas disponibles para el desarrollo de aplicaciones.

El menú Archivo

El menú Archivo es el punto de inicio para muchas operaciones. Además, maneja la comunicación con el mundo exterior, especialmente importando y exportando archivos e imprimiendo. Finalmente, le recuerda en qué ha estado trabajando. El menú desplegable Archivo se muestra en la figura 4-1.

Nuevo y Abrir forman una pareja. Nuevo significa que usted quiere crear un archivo y Abrir significa que quiere usar un archivo existente. Si selecciona Nuevo e intenta usar el nombre de un archivo existente, Visual FoxPro le ofrecerá borrarlo y empezar otro. Es un error que usted cometería solamente una vez.

Figura 4-1



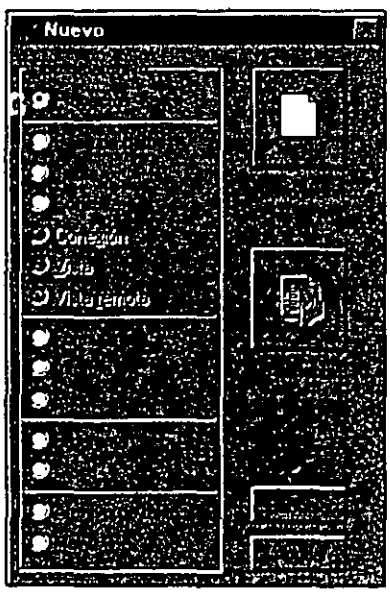
El menú desplegable Archivo.

Archivo, Nuevo

Si usted selecciona Nuevo, obtendrá el cuadro de diálogo que se muestra en la figura 4-2. Haga clic en las diferentes opciones sucesivamente, y notará que el botón Asistente algunas veces está activado y otras desactivado. Como ha visto, los asistentes son programas que lo guían paso a paso a través de un proceso particular, como generar una tabla o un formulario. Si un asistente existe, el botón se activa.

Si el botón Asistente no se activa, se debe ya sea a que el asistente no ha sido proporcionado o a que la tarea es tan sencilla que usted no podrá fallar. CTRL-N (oprimir la tecla CTRL y presionar N) es el método abreviado para el cuadro de diálogo Nuevo. No necesita desplegar el menú para que este método abreviado funcione.

Figura 4-2



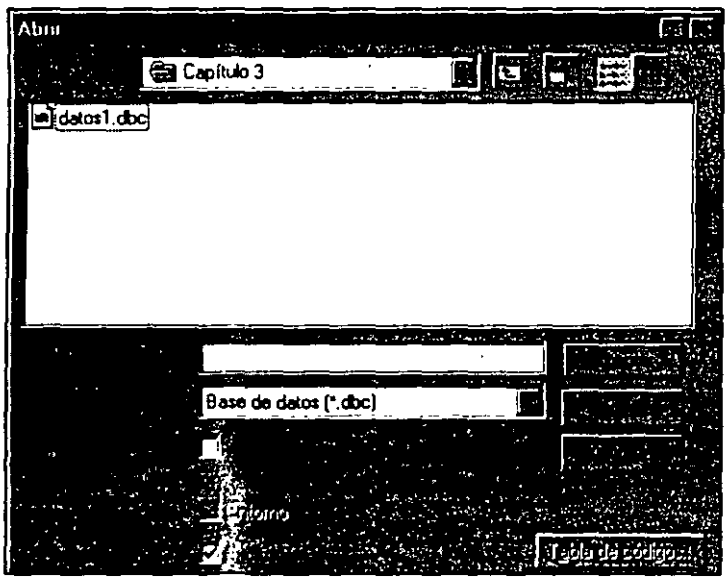
Cuadro de diálogo Archivo, Nuevo.

Los métodos abreviados están activados todo el tiempo. También puede definir métodos abreviados en sus propios menús. De hecho, en algunas aplicaciones, los usuarios se basan en métodos abreviados que ellos usan como una extensión para no usar el menú con mucha frecuencia.

Archivo, Abrir

Si usted selecciona Abrir, Visual FoxPro asume que desea usar un archivo existente y activa el cuadro de diálogo que se muestra en la figura 4-3. Note que puede utilizar el cuadro de lista desplegable dentro del cuadro de diálogo para localizar el archivo, si es que no está en el directorio en uso.

Figura 4-3



Cuadro de diálogo Archivo, Abrir.

FoxPro usa extensiones de archivo (las tres letras que siguen al nombre de archivo) para distinguir entre varios tipos de archivos. Cuando selecciona Abrir, el cuadro de diálogo que le sigue ofrece mostrarle todos los archivos del tipo requerido dentro del directorio en uso. FoxPro sabe cuáles archivos son del tipo requerido por medio de su extensión. Baje al final del cuadro de diálogo, verá una sección para:

```
Archivos de tipo:
```

seguida de un cuadro de lista que le permite seleccionar un tipo de archivo. Al final de la descripción de cada tipo de archivo, dentro de paréntesis, verá el tipo de archivo correspondiente. Muchos archivos llevan un tipo de archivo compañero donde están almacenados campos memo. Los archivos memo se usan en muchos, pero muchos elementos de Visual FoxPro. Los distintos tipos de archivos junto con sus extensiones son:

.DBC Contenedor de bases de datos.

.TXT Archivo, el cual no es muy descriptivo; los archivos .TXT usualmente mantienen la salida de texto desde ya sea los comandos de Visual FoxPro o texto importado.

.SCX Formularios que por lo general se llaman pantallas; sus campos memo tienen extensiones .SCT. El código generado se almacena dentro de un archivo con la extensión .SPR, la cual, cuando se compila, produce archivos .SPX. ¿Entendió eso?

.IDX y .CDX Archivos de índices. En general, los archivos .CDX se usan para los índices de producción, mientras que los archivos .IDX, creados con la palabra clave COMPACT del comando Index, se usan para índices temporales. Esto no es un requerimiento, sólo es una práctica usual.

.LBX y .LBL Archivos de etiqueta. Los campos memo correspondientes se almacenan en archivos con la extensión .LBT.

.MNX Archivos de menú. Los archivos memo correspondientes se almacenan en archivos con la extensión .MNT.

.PRG, .MPR, .SPR y .QPR Archivos de programa que contienen código fuente. Las extensiones de estos cuatro tipos de archivos son .FXP, .MPX, .SPX y .QPX. .PRG es la extensión dada para el código que usted introduce en el segmento de código del Administrador de proyectos. .MPR es dado para el código de menús generado; es por esto que, cuando usted diseñó un menú llamado MENU.MNX en el ejemplo anterior, lo instaló tecleando DO MENU.MPR dentro del programa PRINCIPAL.

.PJX, .FPC, y .CAT Archivos de proyectos. Los campos memo correspondientes se almacenan en archivos con la extensión .PJT.

.QPR Programas de consulta, generados usualmente por el Asistente de SQL.

.FRX Informes. Los campos memo correspondientes se almacenan en archivos con la extensión .FRT.

.DBF Tablas. Los campos memo correspondientes se almacenan en archivos con la extensión .FPT.

.VUE Vistas, las cuales son colecciones de tablas con las configuraciones relevantes Order y Set Relation intactas. Usted puede usarlas para guardar y

restablecer una vista particular de un grupo de tablas después de que una operación que interviene ha cerrado uno o más archivos, índices o el comando Set Relation.

.VCX Biblioteca de clases visuales, la cual es el corazón de la programación orientada a objetos. Los campos memo correspondientes se almacenan en archivos con la extensión .VCT.

Hay cinco controles, del tipo botón para oprimir, a la derecha del cuadro de diálogo Abrir. Éstos son:

Aceptar Toma la selección y trata de abrirla.

Cancelar No pasa nada.

Tabla de códigos Se aplica sólo a tablas y determina cómo se ordenan distintos alfabetos. Si está trabajando en inglés, probablemente no le interese. El ruso es 866, como dolorosamente he aprendido.

Red Esto hace aparecer el cuadro de diálogo Red, que le permite llegar a archivos que usen cualquier convención de nomenclatura de red que usted haya adoptado.

Ayuda Éste le lleva directamente a la ayuda por temas de Visual FoxPro relacionada con el cuadro de diálogo en uso.

Algunos de estos cuadros de diálogo vienen de recursos de Windows, los cuales saben si está conectado a una red o no. Nosotros estamos escribiendo en un ambiente de red Windows NT 3.5, así que nuestros cuadros de diálogo deben tener más opciones que las suyas. NT 3.5 es la plataforma más estable que hay, y es agradable tener una red cuando tres personas tratan de escribir un libro grande en español, inglés y ruso al mismo tiempo. Las tres casillas de verificación tienen usos especializados.

Entorno Está disponible si usted va a abrir una forma de Informe o Etiqueta, debido a que estas dos deben ser generadas con una colección de tablas abiertas en particular. Si no es aplicable, la casilla Entorno no estará disponible.

Exclusivo Esto significa que nadie más puede tener acceso a la tabla mientras usted la esté usando. En un ambiente de red, esto puede ser importante. Si trabaja solo, Exclusivo es el modo de operación normal, y no importa si la casilla está marcada o no. No puede cambiar la estructura de una tabla a menos que usted tenga uso exclusivo.

ólo lectura Esto significa exactamente lo que dice. Si usted no tiene acceso xclusivo, automáticamente tiene acceso de sólo lectura. Puede usar una tabla en forma exclusiva, y luego limitar el acceso a sólo lectura. Ciertamente, la razón para hacerlo es otra cuestión.

Si usted especifica SET STATUS BAR ON, el mensaje en la barra de estado que se encuentra en la parte inferior de la pantalla afirmará este hecho. Exclusivo significa exclusivo, y un registro no bloqueado quiere decir que no es exclusivo. Si quiere asegurar el acceso exclusivo a tablas, puede hacerlo con el comando SET EXCLUSIVE ON.

CTRL-O es el método abreviado para el cuadro de diálogo de la operación Archivo, Abrir. El menú no necesita ser desplegado para que este método abreviado funcione.

Archivo, Guardar

Seleccione Archivo, Guardar para guardar la selección en uso desde la memoria o un archivo temporal de respaldo a su lugar de origen. Esto asume que el archivo que será guardado ya tiene un nombre. Si no es así, FoxPro le pedirá un nombre. o necesita agregar una extensión; Visual FoxPro automáticamente le añade la orrecta. Si usted mismo agrega una extensión, asegúrese de que sea la correcta.

CTRL-S es el método abreviado para la operación Archivo, Guardar. El menú no necesita ser desplegado para que este método abreviado funcione.

Archivo, Guardar como

Utilice la barra Archivo, Guardar como, para guardar la selección en uso con un nombre distinto al que tiene en ese momento, por ejemplo, para guardar en un disco flexible el trabajo que está haciendo o para cambiar un nombre de archivo. Hay otras formas para hacer esto, pero ésta funcionará.

Archivo, Revertir

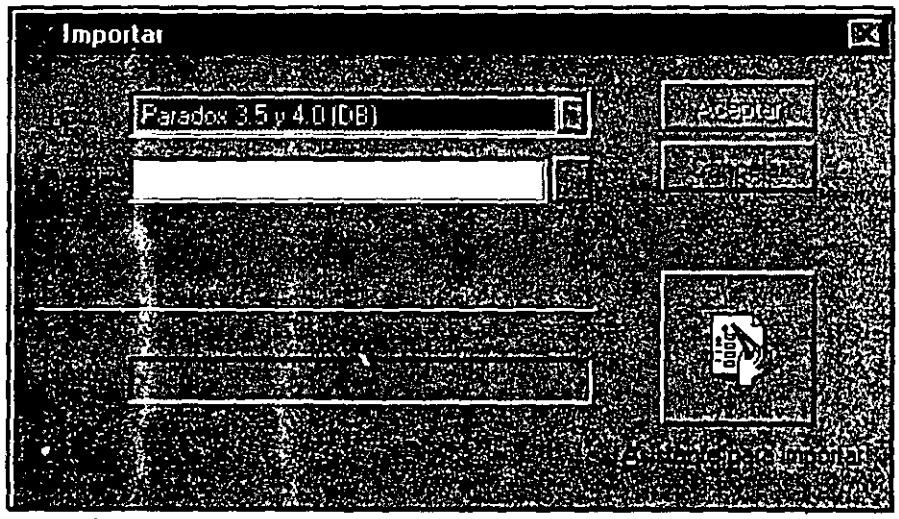
La selección Archivo, Revertir es el inverso de Archivo, Guardar. Si ha hecho algunos cambios y luego decide no guardarlos, presionar ESCAPE lo lleva fuera del formulario, entonces seleccione Revertir, los cambios desaparecerán y el formulario regresará al estado en que se encontraba antes de que hiciera los cambios.

Archivo, Importar

El comando Archivo, Importar le puede hacer la vida mucho más fácil si usted está convirtiendo una aplicación a Visual FoxPro desde otro lenguaje o plataforma. El nuevo Asistente para importar es ridículamente fácil de usar, y cubre una amplia variedad de requerimientos. Se verá esto más adelante, así que aquí sólo se tocarán sus principales puntos.

Si usted hace clic en Archivo, Importar, obtendrá el cuadro de diálogo que se muestra en la figura 4-4. Escoja Asistente para importar, y verá el cuadro de diálogo mostrado en la figura 4-5.

Figura 4-4



Cuadro de diálogo Archivo, Importar.

El Asistente para importar, le pregunta el tipo de archivo que desea importar. Si es un archivo de alguno de los productos comerciales listados en el cuadro de lista (Excel, MultiPlan, Lotus 123, Symphony, Paradox o RapidFile), el asistente averiguará dónde van las cosas y qué tipo de datos son representados. Por otro lado, si es un archivo de texto, sólo hay dos opciones. Puede ser tanto un archivo delimitado, con caracteres de separación entre campos, o un archivo de longitud fija, registros delimitados por Return.

Si es un archivo delimitado, debe proporcionar sólo los nombres de los campos y alguna información sobre las propiedades de campos. Si el archivo tiene registros de longitud fija, debe hacer un clic en el final de cada columna (debido a que FoxPro no puede decirle que constituye un campo). ¡Es todo lo que se necesita para hacerlo! En la figura 4-6 se muestran las Opciones de campos, el segundo paso en el Asistente para importar.

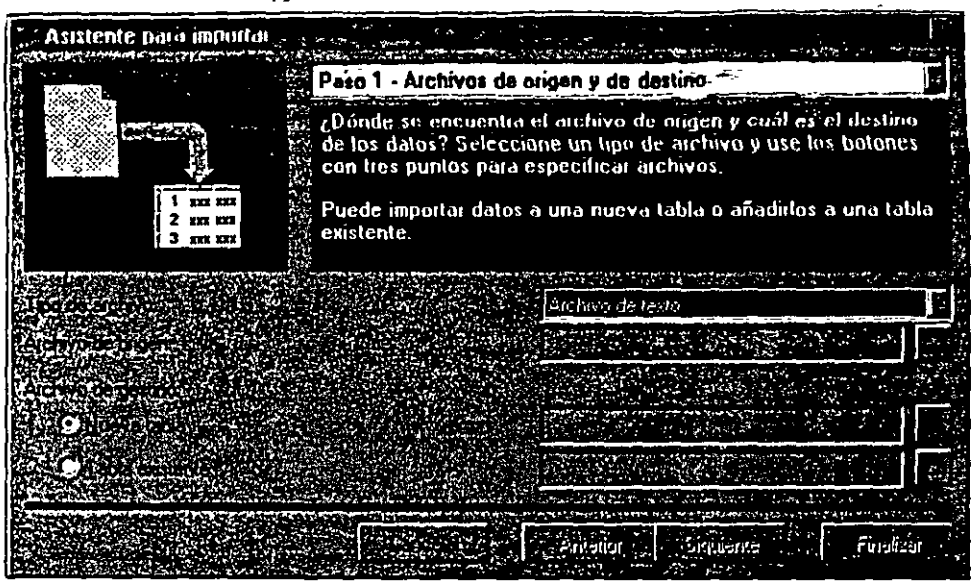


Figura 4-5

Cuadro de diálogo Asistente para importar.

Archivo, Exportar

El comando Archivo, Exportar es aún más simple. Nombre la tabla (.DBF) de la cual exportará, el nombre y tipo de la tabla del archivo de destino (ya sea uno nuevo o uno existente), y haga clic en Aceptar.

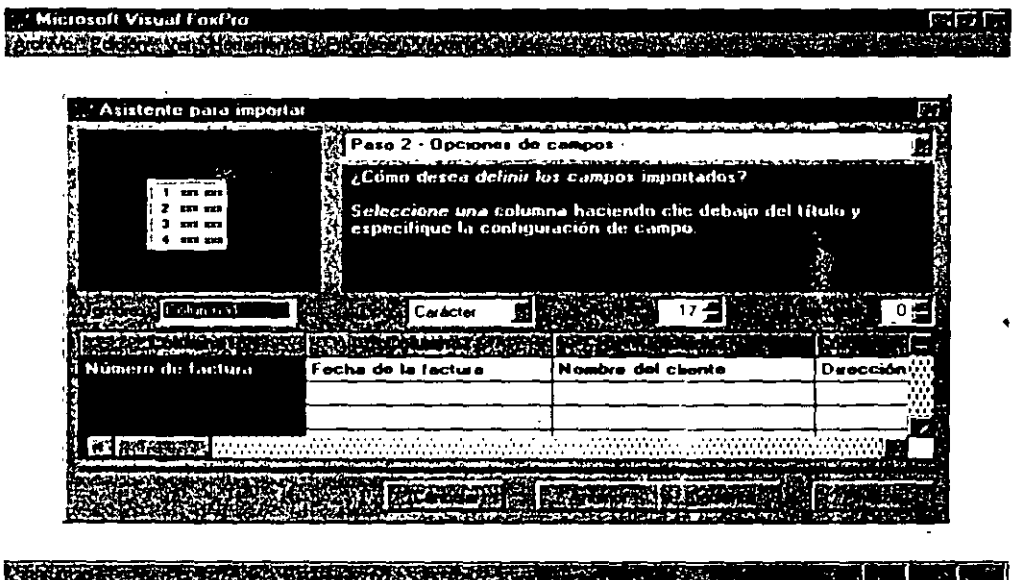


Figura 4-6

Opciones de campos del cuadro de diálogo Asistente para importar.

Preparar página

Archivo, Preparar página está disponible sólo cuando se está utilizando ya sea el Generador de informes o el Generador de etiquetas. Preparar página es una herramienta fabulosa para hacer informes, especialmente informes de dos o tres columnas, de abajo hacia arriba. Para demostrar esta característica, voy a crear una tabla de nombre y dirección llamada Cliente. Teclee:

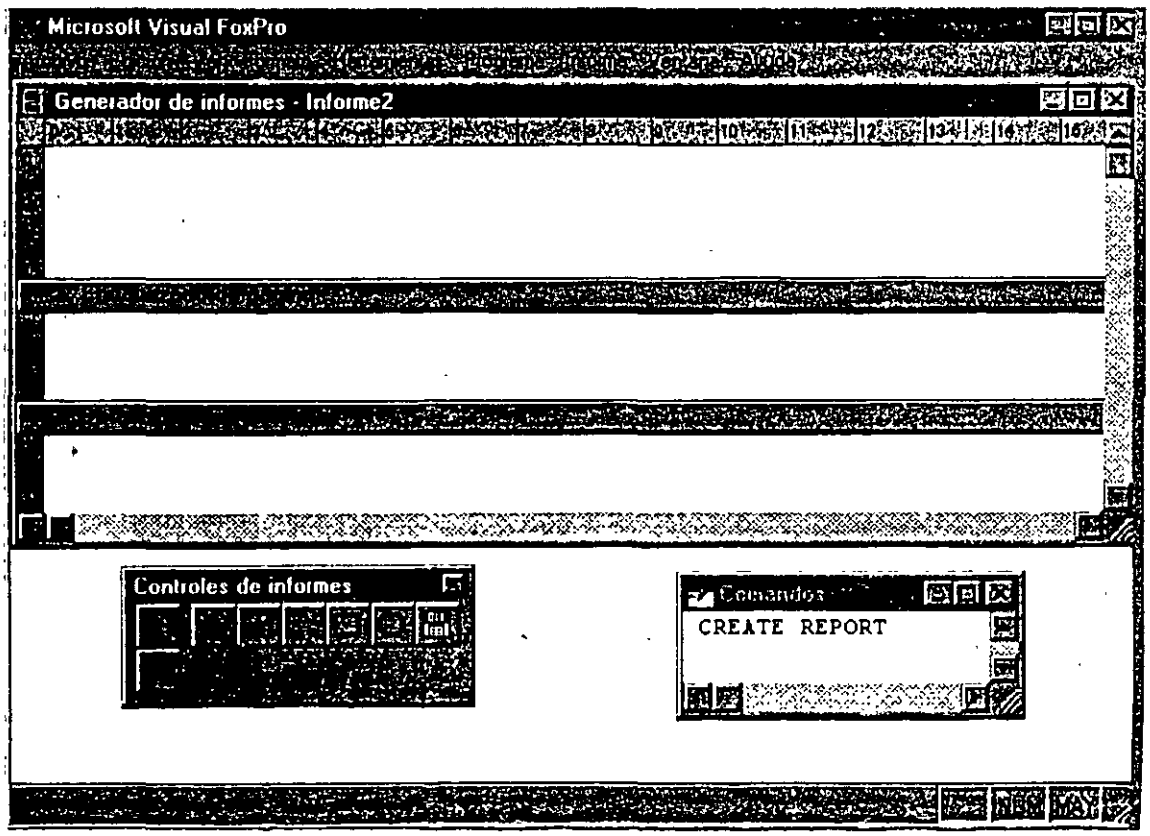
```
CREATE CLIENTE
```

y luego introduzca los campos Nombre, Domicilio, Ciudad, Estado y Cod. Post. Use el comando Browse para introducir una media docena de nombres, y luego teclee:

```
CREATE REPORT CLIENTE
```

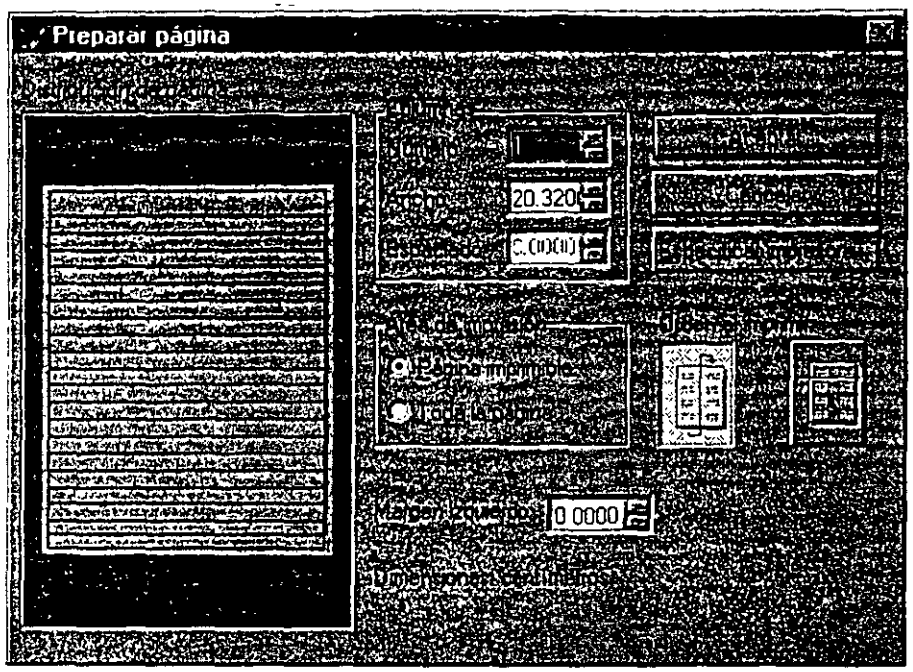
Obtendrá un cuadro de diálogo como el que se muestra en la figura 4-7. Ahora haga clic en Archivo, Preparar página para obtener el cuadro de diálogo mostrado en la figura 4-8.

Figura 4-7



Cuadro de diálogo Generador de informes.

Figura 4-8



Cuadro de diálogo Preparar página.

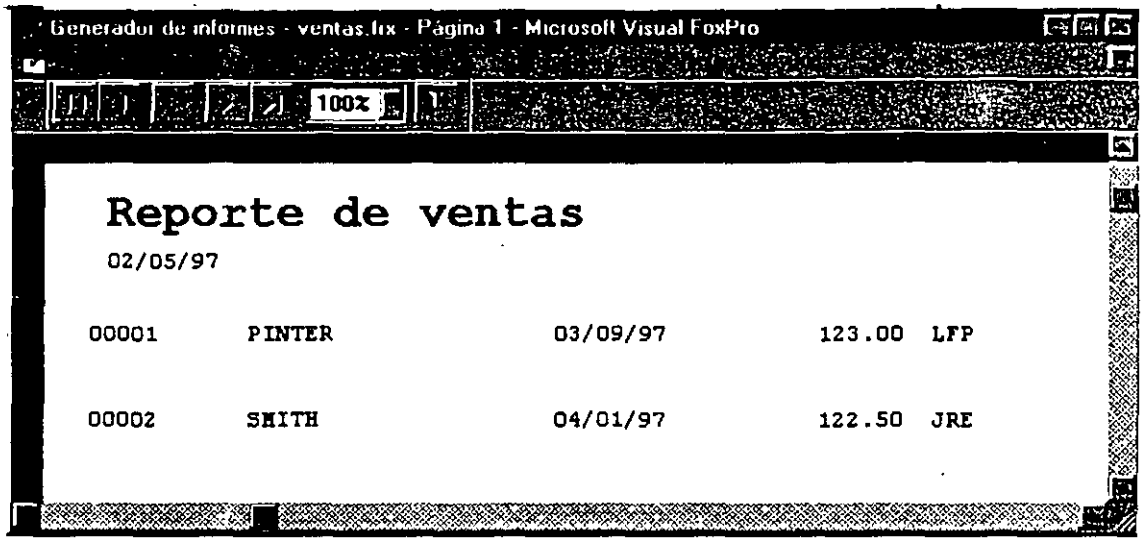
Haga clic en el control numérico para el Número de columnas, luego haga clic en Aceptar y mire la apariencia del informe. El área Encabezado de página se mantiene cubriendo el ancho completo de la página, pero el área Detalle es sólo un tercio del ancho de la página. Haga clic en el nuevo menú principal Informe, el cual está presente sólo cuando usted está en el Generador de informes, y seleccione Informe rápido. En el cuadro de diálogo generado, haga clic en el control de distribución de campos más alejado hacia la derecha, para que los campos en el archivo Cliente estén apilados uno arriba de otro, en lugar de lado a lado. Haga clic en la casilla de verificación Títulos, para dejar la opción desactivada y luego haga clic en Aceptar. Verá los campos en los campos de su nuevo archivo Cliente en el área Detalle. Use el ratón, para volver a arreglarlos, si así lo desea, y para volver a dar tamaño al área Detalle. Si lo desea, introduzca un título en el área del Encabezado. Ahora está listo para ir a la siguiente opción, Archivo, Presentación preliminar.

Presentación preliminar

Haga clic en Archivo, Presentación preliminar. El resultado aparece en la figura 4-9. Se añadieron algunos detalles exóticos, pero el resultado final es el mismo. Usted terminó un listado de empleados de una compañía en menos de un minuto.

Desde luego, aún no está impreso en papel, lo cual nos lleva a la siguiente opción:

Figura 4-9



Cuadro de diálogo Presentación preliminar.

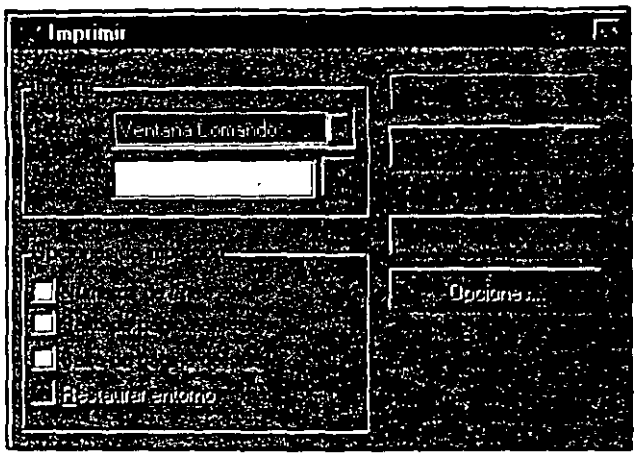
Imprimir

La opción Imprimir, a la cual usted siempre puede tener acceso mediante el método abreviado CTRL-P, produce el cuadro de diálogo mostrado en la figura 4-10. Imprimir enviará alguno de varios tipos de salida hacia la impresora:

- Informes
- Etiquetas
- Ventana Comandos (programas)
- Archivos
- Archivos ASCII
- El contenido del Portapapeles

En algunos tipos de archivos se imprimen números de línea a la izquierda de cada línea de salida, como listados de programas, pero no en informes y etiquetas. Además, puede forzar un salto de página (*return*) antes o después de imprimir la mayoría de los tipos de salida, pero sólo antes de informes o etiquetas. Visual FoxPro asume que los archivos ASCII incluyen todo lo que necesitan, así que no se permiten opciones. Cualquier cosa que usted copie al Portapapeles resaltando y oprimiendo CTRL-C, también puede ser impresa. Y puede poner cualquier cosa en el Portapapeles colocándolo dentro de la variable de memoria del sistema _cliptext. Una vez que se encuentra en el Portapapeles, puede usar CTRL-V para pegarla.

Figura 4-10



Cuadro de diálogo Imprimir.

El botón Especificar impresora, que muchas aplicaciones de Windows incluyen como una opción separada del menú Archivo, es accesible en Visual FoxPro sólo a través de la opción Archivo, Imprimir. Especificar impresora, proporciona control para seleccionar la impresora, selección de la forma (tamaño del papel), orientación e impresión por los dos lados. El botón Propiedades maneja opciones avanzadas de control de impresión, y Red le permite utilizar una impresora de la red por su nombre.

Sin importar el tipo de archivo que va a ser impreso, el cuadro de diálogo Configuración de impresión le permite seleccionar varias características de la impresora. Primero, si tiene muchos controladores de impresoras instalados, puede seleccionar una impresora, así como el tipo de forma (tamaño de papel) y la orientación. Y si está en una red, puede escoger una impresora de la red mediante un clic en el botón Red.

Haga clic en el botón Propiedades, para activar otras opciones basadas en la impresora que seleccionó. Por ejemplo, una HP LaserJet 4 puede imprimir a una resolución de 150, 300 o 600 dpi (puntos por pulgada), así que ésa es una opción presentada por el botón Propiedades.

En el caso de informes y etiquetas, hay otras dos opciones disponibles. Primero, si usted escoge informe o etiqueta, el botón Opciones se activa. Opciones incluye lo siguiente:

Alcance Todos, Siguientes *n*, Registro *n* o Restantes.

For Utilice el Generador de expresiones o teclee directamente para introducir una expresión que filtre el archivo; todos los registros que pasen por el filtro serán impresos en forma de un informe o una etiqueta.

While Utilice el Generador de expresiones, o teclee directamente para introducir una expresión que filtre el archivo; todos los registros que pasen por el filtro, empezando por el registro abierto, serán impresos en forma de informe o etiqueta. La manera más rápida de producir un informe o grupos de etiquetas basados en un subgrupo dentro de una tabla, es en archivos indexados en los cuales usted ya ha usado ya sea Browse, Seek o Locate para encontrar un registro en particular y en el cual la expresión de índice tiene agrupados todos los registros que quiere, siguiendo al que está localizado en el puntero del registro en uso.

Puede salvar informes y etiquetas con instrucciones para abrir los archivos necesarios para el informe, y establecer índices y relaciones. Si usted marca Restaurar entorno, estos pasos son tomados automáticamente. Si usted ya tiene las cosas configuradas de la manera que desea, podría declinar este ofrecimiento de ayuda. La opción Enviar lo conecta a MS Mail, si está instalado. Para usuarios en red, esto proporciona una forma de comunicación. Para Microsoft ésta es una oportunidad de anunciar MS Mail. Realmente es muy bueno.

Cualesquier proyectos en los que esté trabajando actualmente son listados entre las opciones Enviar y Salir. Son precedidos por una numeración subrayada que empieza con 1, de manera que puede presionar 1 para abrir el primer proyecto listado, 2 para abrir el segundo, etc. Puede activar o desactivar esta opción usando la página principal Herramientas, Opciones, Proyecto, comentada más adelante en este capítulo.

La opción Salir es una de las formas de salir de Visual FoxPro. Es equivalente a teclear Quit en la ventana Comandos. También puede presionar ALT-F4 para salir de FoxPro.

El menú Edición

El menú Edición es tan útil que muchos desarrolladores incluyen varias de sus opciones en sus propias aplicaciones.

Deshacer y Rehacer

Deshacer cancela la última operación que realizó, mientras que Rehacer cancela el efecto del último Deshacer. Lo que, dependiendo del contexto, constituye la "última operación". Por ejemplo, si usted está escribiendo en la ventana Comandos, cada vez que presione la barra espaciadora, en cuanto concierne a Deshacer, usted completa una operación. Teclee el zorro veloz, presione CTRL-Z (el método abreviado para Deshacer), y verá desaparecer las tres palabras cada

ıa a la vez. Presione tres veces CTRL-C (el método abreviado para Rehacer), y las palabras aparecerán una por una.

Pero si está escribiendo en código de programa, el retorno al final de cada línea es, para Deshacer y Rehacer, el suceso relevante. Si usted está en el Generador de formularios o en el Generador de informes, es la acción de agregar un control nuevo al formulario. Es realmente justo lo que esperaría de cada contexto.

Cortar, Copiar y Pegar

Éstos son los tres elementos siguientes en el menú Edición. Usted marque algo, luego seleccione Copiar o Cortar para poner la selección marcada en el portapapeles. Hay muchas maneras de marçar texto. Si mantiene presionada la tecla MAYÚS y hace clic en un párrafo, todo el texto desde el cursor hasta el final del párrafo quedará marcado. Si mueve el puntero del ratón hacia el principio de una palabra, haga clic y mantenga oprimido el botón principal del ratón, luego arrastre el ratón a la derecha o izquierda, el texto que usted marque aparecerá en forma de video inverso.

El texto que ha sido marcado puede ser cortado (quitado y copiado al ortapapeles) o copiado (al portapapeles) para pegarlo posteriormente. La opción egar se activa sólo cuando algo ha sido copiado o cortado.

En los varios generadores (Formularios, Informes y Clases), ya que no puede marcar el principio y final de un texto, usted marca un objeto para cortar o copiar haciendo un clic en él una sola vez. Cuando un objeto es seleccionado, un grupo de ocho pequeños cuadrados negros lo encierran en sus cuatro esquinas y en la mitad de las líneas que los conectan. Una vez seleccionado, puede cortarlo o copiarlo de la misma forma en que corta y pega texto.

Cortar y pegar es menos importante en Visual FoxPro de lo que fue en versiones previas, pero sigue siendo de gran utilidad. Y usted no está limitado a trabajar con un solo formulario o informe; puede copiar de un formulario, luego abrir otro y pegar dentro de éste. Sin embargo, no deje que la facilidad de las operaciones cortar y pegar lo distraigan del uso de las características orientadas a objetos, que son el corazón de Visual FoxPro.

Pegado especial se asocia con objetos OLE (*Object Linking and Embedding*: vinculación e incrustación de objetos), y se comentará con detalle cuando se hable acerca de OLE.

Borrar

Utilice esta opción para borrar un elemento o texto seleccionado. En general, si usted resalta una sección de texto y luego presiona, por decir, la letra X, su texto resaltado será reemplazado por la letra X. Borrar reemplaza con nada lo que esté seleccionado.

Seleccionar todo

Seleccionar todo es una forma cómoda de marcar todos los elementos de una pantalla para moverlos un poco hacia abajo. Desafortunadamente, no puede seleccionar, por ejemplo, todas las etiquetas en una pantalla y cambiar sus propiedades de fuente. Para eso necesita la utilería Property Changer, la cual está disponible a través de nuestro foro en CompuServe.

Buscar y Reemplazar

Utilice Buscar para encontrar una cadena de texto en particular. Funciona en programas y ventanas de código dentro de formularios. Cuando selecciona Buscar, aparece el cuadro de diálogo Buscar, que se muestra en la figura 4-11. Teclee lo que quiera encontrar y oprima ENTER.

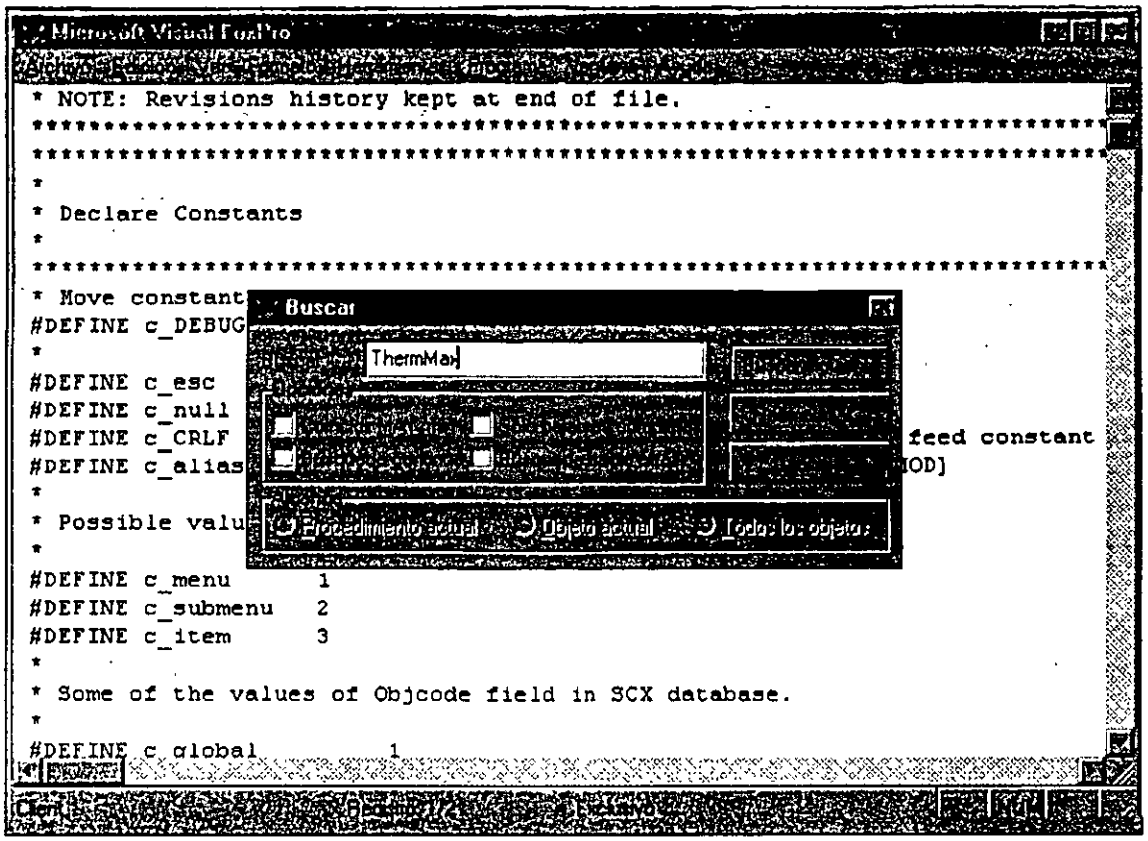
Si hace clic en Reemplazar, el cuadro de diálogo cambia para incluir el cuadro de texto Reemplazar con, como se muestra en la figura 4-12. Si introduce una cadena dentro del cuadro de texto Reemplazar con, luego hace clic en Reemplazar, la primera cadena que coincida, resaltada, se reemplaza con el texto de la segunda cadena.

Puede llevarse usted mismo a un mundo de sufrimiento con el botón Reemplazar todo. Si reemplaza todas las instancias *cabo* con *vaya*, la palabra *trascabo* también será reemplazada con *trasvaya*. Así que salve su trabajo antes de reemplazar y hágalo a menudo. Significa no tener que decir nunca que lo siente.

Ir a la línea

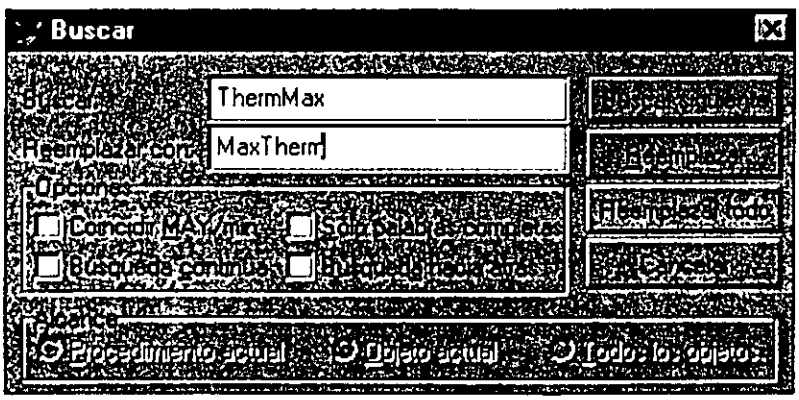
Esto es útil para depurar código de programa. El cuadro de diálogo al que llama le permite introducir un número de línea particular para saltar directamente hacia el código que está editando. Desde la aparición de números de línea en el compilador de Visual FoxPro e informes con mensajes de error, puede ir directo a la fuente del problema reportado.

Figura 4-11



Cuadro de diálogo Buscar.

Figura 4-12



Cuadro de diálogo Buscar, con la opción Reemplazar con.

Insertar objeto, Objeto y Vínculos

Insertar objeto despliega un cuadro de diálogo que le permite insertar un objeto OLE en un campo general, esto será comentado con detalle cuando se hable acerca de OLE. Objeto activa un objeto OLE y Vínculos despliega el cuadro de diálogo Editar vínculo, de manera que puede actualizar un vínculo hacia un objeto.

Aquí hay algunas palabras sobre un consejo no solicitado: Si usted viene de otro lenguaje o entorno, podría verse tentado a usar su propio editor, o al menos sentir que da un paso hacia atrás en el mundo por usar el editor FoxPro. Por favor dé al entorno Visual FoxPro, incluyendo al editor, una oportunidad. Está hecho para trabajar junto con el resto de las herramientas de Visual FoxPro, y realiza su trabajo muy, pero muy bien.

El menú Ver

El menú principal Ver consiste inicialmente de sólo una opción de menú, Barras de herramientas. Si usted tiene algunas tablas abiertas, precederá a la opción Barras de herramientas una opción para Examinar (nombre de la tabla), para cada una de las tablas abiertas.

La opción Barras de herramientas será todo lo que verá hasta que abra alguno de los generadores, y después Ver se convierte en su mejor amigo. Aparece un número de nuevas opciones, dependiendo de donde esté, y Barras de herramientas se vuelve la opción inferior del menú.

El menú Ver cuando se genera un formulario

Abra una ventana Administrador de proyectos, usando Archivo, Nuevo. Por ahora está bien usar el nombre por omisión. Luego abra un formulario nuevo haciendo clic en Documentos, y resaltando Formulario, y despues haciendo clic en el botón Nuevo. Haga clic en Ver, y verá las siguientes opciones:

Diseño Éste es el principal modo de generar un formulario.

Orden de tabulación Utilice ésta para volver a ordenar los campos de entrada y las opciones controles de un formulario.

Entorno de datos Le permite configurar o cambiar el entorno de datos del formulario.

Propiedades Despliega el cuadro de diálogo Propiedades, donde se hacen los ajustes más finos a un formulario.

Código Aquí es donde se ve el código del método para su formulario, informe o entorno de datos.

rra de herramientas Controles de formularios Muestra u oculta la barra de herramientas Controles de formularios.

Barra de herramientas Distribución Muestra u oculta la barra de herramientas Distribución.

Barra de herramientas Paleta de colores Muestra u oculta la barra de herramientas Paleta de colores.

Líneas de cuadrícula Cuando está marcada, coloca líneas de cuadrícula a intervalos regulares sobre la superficie del generador.

Mostrar posición Esta opción hace aparecer la barra de estado en la parte inferior de la pantalla que despliega la posición actual del cursor y el tamaño, en pixeles, del elemento.

El menú Ver cuando se genera un informe

Si hace clic en Informes y luego en Nuevo, las primeras dos opciones del menú Ver, distintas a las de los examinadores para cualquier tabla que pueda estar ıierta, son Diseño y Presentación preliminar. La Presentación preliminar de un ınforme le muestra cómo se verá, hasta ese momento, el informe que ha diseñado. No utiliza datos, de forma que puede ver la presentación preliminar aun cuando su informe se encuentra en un estado muy primitivo.

También se añade el entorno de datos al menú Ver, debido a que los informes requieren de datos. Y las opciones para desplegar las tres barras de herramientas aplicables se convierten en las tres opciones siguientes:

- Barra de herramientas Controles de informes
- Barra de herramientas Distribución
- Barra de herramientas Paleta de colores

Finalmente, dos opciones de menú adicionales le dan control sobre la posición:

Líneas de cuadrícula Despliega líneas de cuadrícula tanto verticales como horizontales en un informe, para ayudar visualmente a la colocación de objetos.

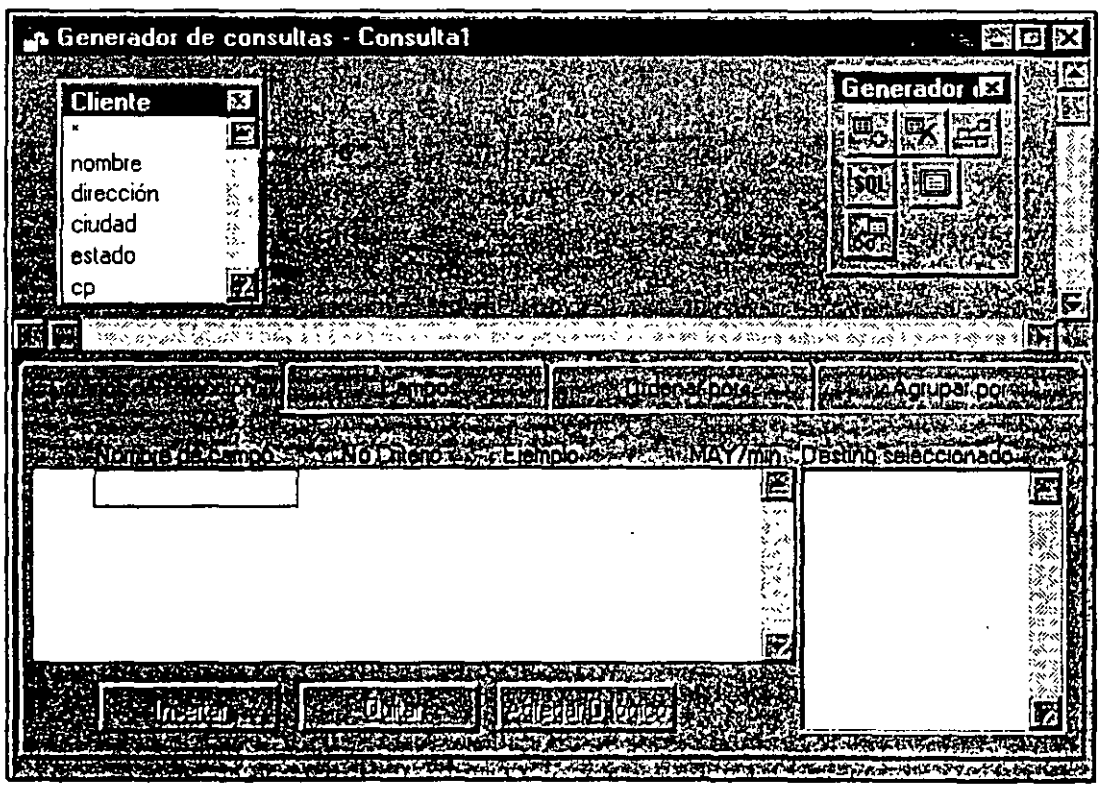
Mostrar posición Muestra la posición del elemento, que está actualmente eleccionado en la barra de estado.

El menú Ver cuando se utiliza el Generador de consultas

Si usted hace clic en Datos, Consulta, Nuevo en la ventana del Administrador de proyectos, se abre el Generador de consultas mostrado en la figura 4-13. El menú Ver contiene ahora una nueva opción, Maximizar panel superior. El Generador de consultas que se muestra en la figura 4-14, aparece con el panel superior maximizado. Compárelo con la figura 4-13 en la que el panel superior está minimizado. El panel superior, el cual es como el Generador de datos, muestra cuáles tablas, índices y relaciones están involucrados en la consulta, puede ser maximizado para que pueda ver sus tablas con mayor detalle.

El menú Ver cambia mientras el generador cambia. Aunque generalmente hay muchas maneras para hacer todo dentro de cada generador, el menú de Visual FoxPro podrá convertirse en su método preferido de acceso, especialmente si está aprendiendo a usarlo.

Figura 4-13



Cuadro de diálogo Generador de consultas.

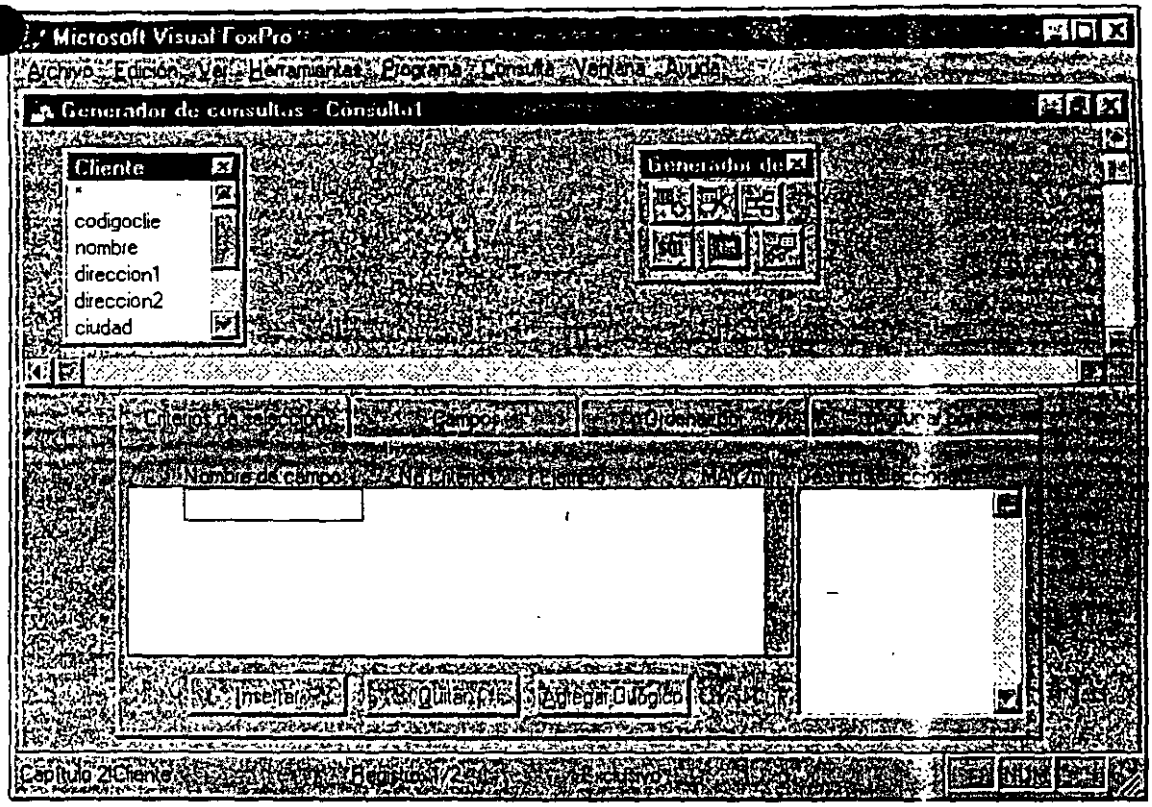


Figura 4-14

Cuadro de diálogo Generador de consultas con el panel superior maximizado.

El menú Formato

Utilice el menú Formato para cambiar la apariencia y otras características de texto en una ventana seleccionada. Generalmente, usted utiliza esta opción para configurar sus ventanas de código. Las opciones caen dentro de tres grupos:

Tipo y tamaño de fuentes

Fuente Despliega el cuadro de diálogo Fuente. Puede seleccionar cualquiera de las fuentes que ha instalado en Windows, aunque la fuente Courier de 12 puntos es un estándar para programar.

Ampliar fuente Aumenta en puntos el tamaño del texto dentro de la ventana que se está utilizando.

Reducir fuente Disminuye en puntos el tamaño del texto dentro de la ventana en uso.

Espaciado de línea

Puede establecer el espacio entre líneas a espacio sencillo, uno y medio espacios o doble espacio.

Sangrado

Sangría Da sangría al texto resaltado por medio de un tabulador. Use el ratón o mantenga oprimida la tecla MAYÚS y utilice las teclas de movimiento del cursor para resaltar texto, luego haga clic en Sangría.

Quitar sangría Deshace el efecto de la selección anterior.

El menú Herramientas

El menú Herramientas es uno de los más importantes para los programadores. Es usado para llamar a los nuevos asistentes de Visual FoxPro, los cuales están muy mejorados comparados con los de versiones previas, y hacen mucho del trabajo que normalmente se tenía que hacer; despliega las ventanas Seguimiento y Depuración, toda la ayuda para depurar que necesitará y para activar el marco de página Opciones, la cual controla la mayoría del entorno Visual FoxPro.

Asistentes

El menú Herramientas es el puente de acceso para llegar a los asistentes de Visual FoxPro:

- Tabla
- Formulario
- Consulta
- Informe
- Etiqueta
- Combinar correspondencia
- Tabla dinámica
- Importar
- Documentación

77

- Instalación (el cual se proporciona para la distribución de aplicaciones)
- Upsizing

La revisión ortográfica es quizá menos útil para programadores, debido a que nosotros tenemos maravillosa ortografía y porque la revisión ortográfica no se presta para código de programas. Pero es una herramienta, y éste es el menú Herramientas, así que es aquí donde reside.

Seguimiento y Depuración

Puede usar el menú Herramientas para llamar muchas herramientas que son muy valiosas durante la depuración:

Seguimiento Lo guía paso a paso a través de la ejecución de un programa mientras despliega la línea de código que está siendo ejecutada.

Depuración Le permite mostrar el contenido actual de los campos de una tabla, variables de memoria, funciones, propiedades y para establecer cortes de programa para suspender la ejecución de un programa mientras corre una aplicación.

La ventana Seguimiento lo guía paso a paso y línea por línea a través de su código de programa; puede usarla para establecer cortes de programa, ya sea en un número de línea particular o para evaluar si es verdadera una condición que especificó en una línea de la ventana Depuración. Por ejemplo, si no sabe dónde ha sido desactivada una variable intercambiable llamada Listo, sólo teclee not listo dentro de la ventana Depuración, haga clic en la barra vertical de la parte derecha para establecer el corte de programa, y ejecute su programa. La inmediata Listo es falsa, su programa se detiene y la ventana Seguimiento aparece mostrándole la línea del código de programa que hizo la condición Not Listo verdadera.

El marco de página Opciones

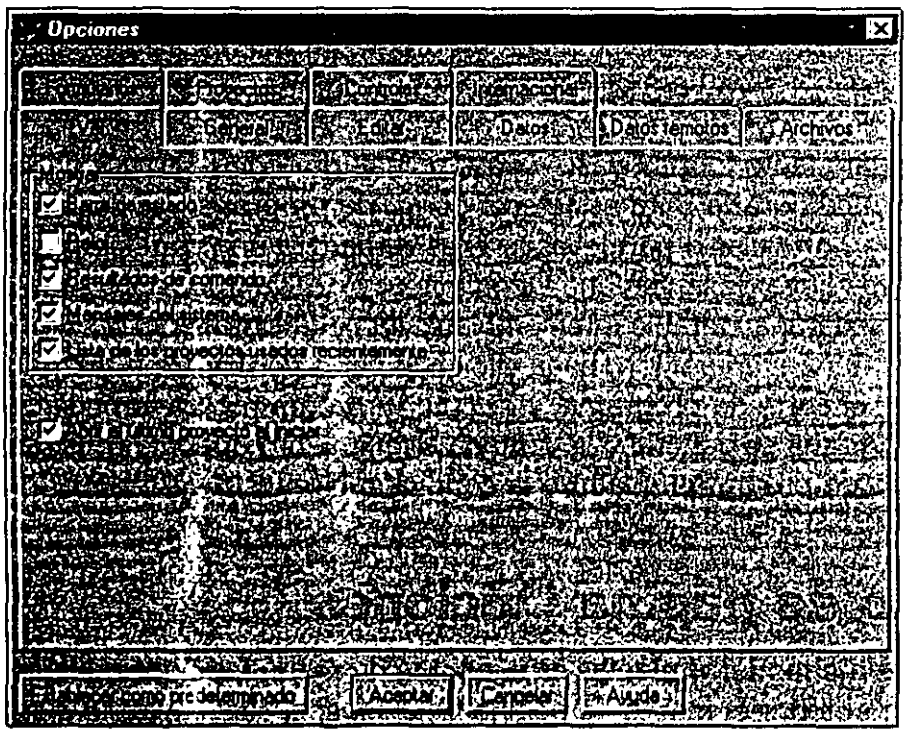
El principio del menú Herramientas es la opción de menú Opciones. Haga clic en Opciones para obtener el marco de página Opciones, el cual contiene la mayoría de las opciones que usted controla con los comandos Set en FoxPro 2.6. Además, aquí puede especificar la conducta precisa de muchas de las nuevas características de Visual FoxPro. Dentro de el marco de página Opciones hay dos filas, indicadas por fichas. Si selecciona una ficha de la fila superior, las dos filas de fichas intercambiarán lugares antes de que la ficha seleccionada sea traída al frente.

La ficha Ver

La ficha Ver mostrada en la figura 4-15 contiene casillas de verificación que se asocian con la barra de estado (la barra en la parte inferior de la pantalla) y el reloj (el cual puede activar o desactivar en cualquier momento). Los últimos dos corresponden a Set Status Bar On/Off y Set Clock On/Off. También contiene la casilla de verificación para mostrar Resultados del comando, que es lo mismo que Set Talk On/Off. Cuando Talk está en On, si usted escribe X=3 en la ventana Comandos, el valor asignado 3 se imprime en la pantalla. Mostrar Mensajes del sistema, al igual que su compañero, Set Notify On, despliega mensajes como Do Canceled.

Si la opción Lista de los proyectos usados recientemente está marcada, su menú Archivo tendrá una lista de unos cuantos archivos que ha abierto. Si está marcada la casilla Abrir el último proyecto al iniciar, la primera pantalla que verá siempre que abra Visual FoxPro será el Administrador de proyectos, con el último proyecto abierto en el que estuvo trabajando cuando salió del programa.

Figura 4-15



Herramientas, Opciones, ficha Ver.

La ficha General La ficha General se muestra en la figura 4-16. Muchos aspectos de FoxPro son controlados aquí:

Sonido de advertencia Esto es lo que se llamaba Set Beep. Usted puede activarlo o desactivarlo, o proporcionar un nombre de un archivo (.WAV) para musicalizarlo.

Cancelar programas con ESC Esta casilla es lo que era Set Escape On/Off; el nombre actual hace que, efectivamente, el significado sea más claro.

Mostrar la ventana Seguimiento Esto se llamaba antes Set Step On.

Registrar errores de compilación Esto coloca los errores de compilación en un archivo llamado appname.ERR.

Set Development Ésta es una casilla de verificación que usted no debe necesitar. Controla la recompilación de archivos .PRG, los cuales tienen una asignación date/time más antigua que la que corresponde a .FXP. Es necesaria sólo si utilizó un editor distinto que el editor FoxPro, y trabajó fuera del área de trabajo del Administrador de proyectos. Note que:

- Set Compatible dBASE es lo mismo que Compatibilidad con dBASE.
- Confirmar el reemplazo de archivos es como Set Safety On/Off.

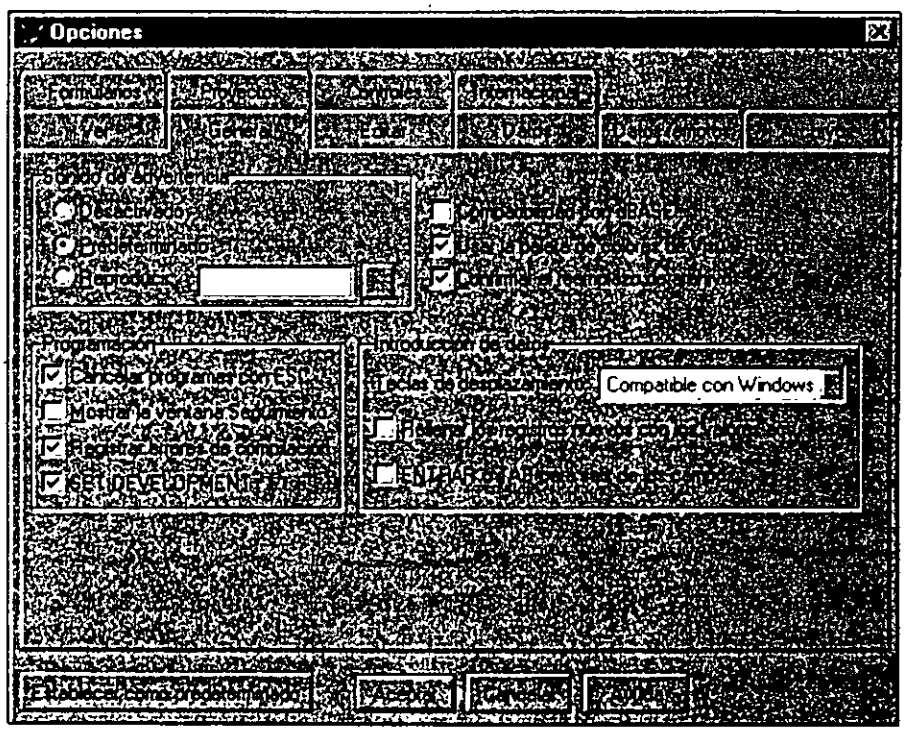


Figura 4-16

Herramientas, Opciones, ficha General.

- Para especificar si las teclas de desplazamiento son de tipo DOS o Windows, se configura Set KeyComp to DOS, Windows o Mac. Consulte la Ayuda para más información en el tema Set KeyComp. Si está trabajando con Windows, probablemente debe usar los golpes de tecla compatibles con Windows.
- Usar ENTRAR o TAB para salir de los campos es lo mismo que Set Confirm On/Off. A los usuarios nuevos les gusta establecer Set Confirm On, mientras que los capturistas experimentados prefieren Set Confirm Off. Probablemente no es una mala idea incluir esta opción en sus menús de usuario, y permitirles activar o desactivar esta característica. Mientras que una casilla de verificación es una buena manera de activar o desactivar una opción dentro de un formulario, dentro de un menú usted tiene que usar algo llamado una *marca*, la cual se conoce en Windows como marca de verificación al lado de una opción seleccionada. Se ensañará cómo usar el comando Set Mark para manejar opciones de activación dentro de menús en el capítulo 8.

La ficha Editar

La ficha Editar se muestra en la figura 4-17. Si hace clic en Editar cuando no está editando un programa, ninguna de las opciones se activa. Usted puede aplicar todas las opciones ya sea a un archivo solo o a todas sus sesiones de edición. Las opciones son:

Edición con Arrastrar y colocar Cuando esta opción está marcada, usted puede seleccionar texto y moverlo a otra parte del programa. Como ésta no es una conducta estándar de edición de texto en Windows, es seleccionable aquí. Pruébela, le gustará.

Ajuste de línea automático Esta opción está bien para escribir cartas, pero yo no la uso cuando estoy escribiendo código de programas.

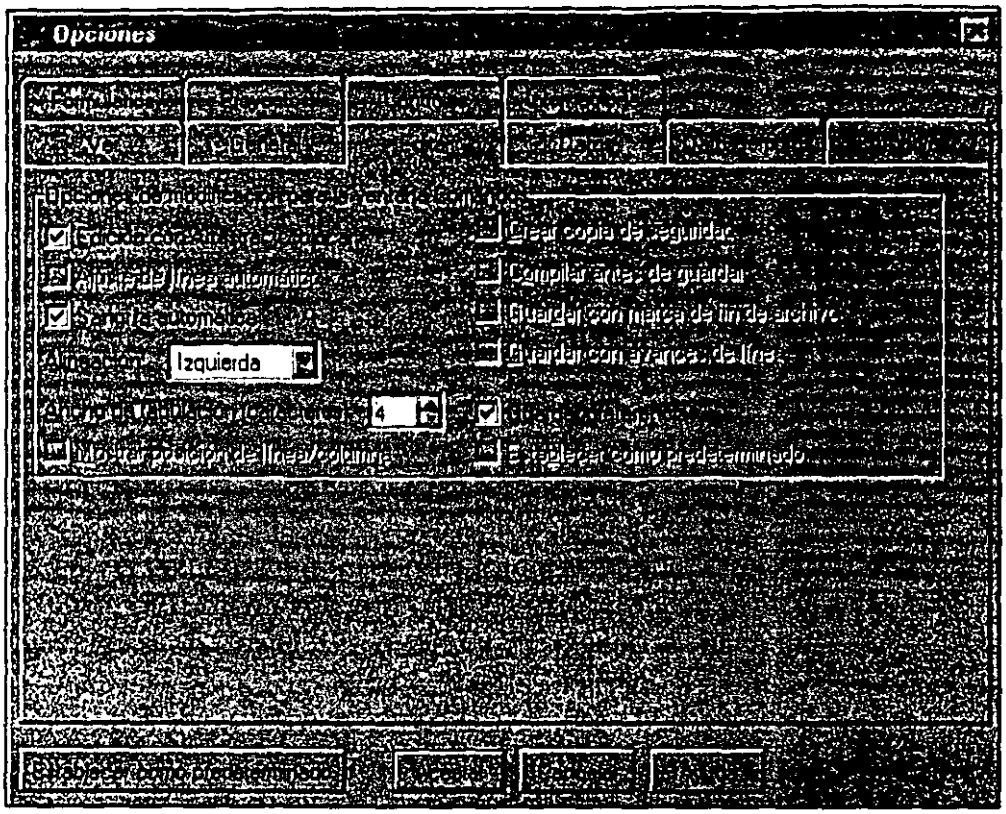
Sangría automática Cada línea sucesiva se alinea con la anterior, lo que usualmente usted desea que se haga.

Alineación Está establecida a la izquierda, y usted rara vez la cambiará.

Ancho de la tabulación (caracteres) Por omisión, esta opción es 4 caracteres, pero puede cambiarla. A mí me gusta en 3, debido a que la primera palabra después de los comandos If y Do se alinea con el código sangrado debajo de ellos

81

Figura 4-17



Herramientas, Opciones, ficha Editar.

Mostrar posición de línea/columna Activa la barra de estado en la parte inferior de la pantalla y continuamente despliega la posición en donde se encuentra su cursor.

Crear copia de seguridad Si está marcada, cada vez que edite un programa, Visual FoxPro hará una copia con la extensión .BAK. Si usted hace algo terriblemente estúpido, esto le ayuda a recuperarse. Utilice esta opción, el espacio de disco es barato.

Compilar antes de guardar Le ayuda a encontrar errores de sintaxis si también seleccionó la casilla de verificación Registrar errores de compilación, en la ficha General.

Guardar con marca de fin de archivo Esta opción es para dar compatibilidad con programas anteriores.

Guardar con avances de línea Esto es muy requerido si quiere ver su código en otro entorno.

Guardar preferencias Esta opción es la forma en que se aplica la configuración de preferencias de esta pantalla al programa que está editando. Sabe que usted está editando un programa debido a que no podría establecer nada en esta página a menos que lo esté haciendo.

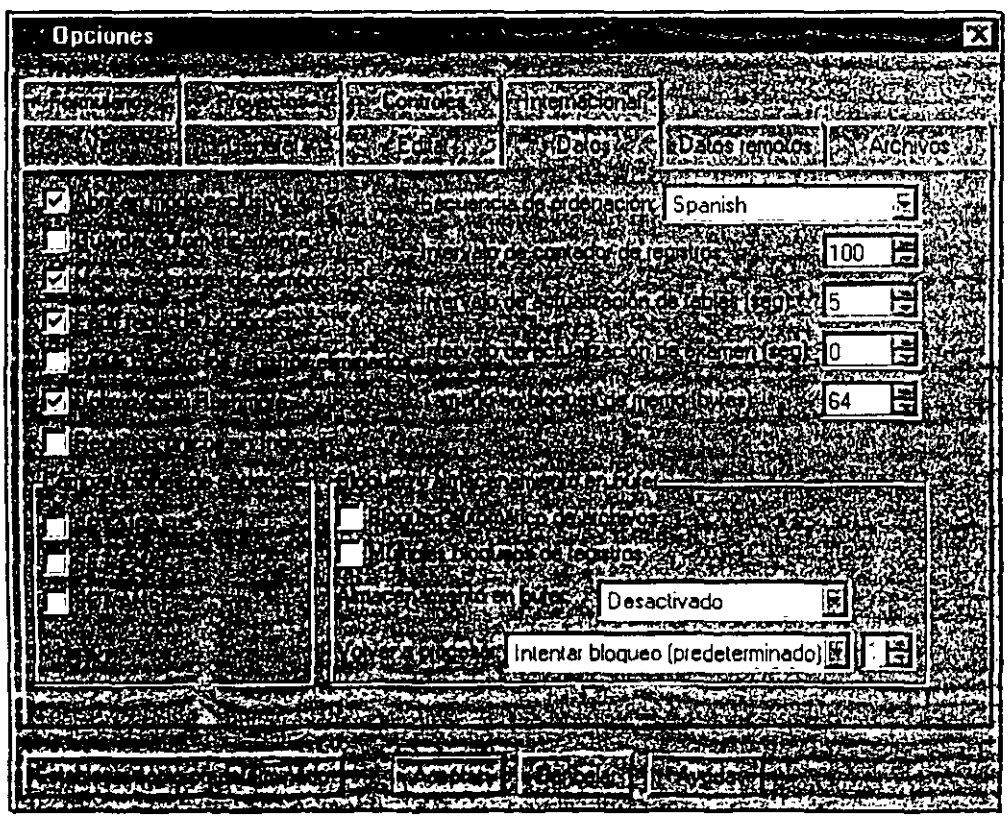
Establecer como predeterminado Se aplica a todos sus programas. Es recomendable ser consistente en la codificación.

La ficha Datos

La ficha Datos se muestra en la figura 4-18, y contiene las siguientes opciones para manipular datos y tablas:

Abrir en modo exclusivo Cuando esta opción está marcada, las tablas son abiertas exclusivamente, lo cual es normal para entornos de un solo usuario pero ocasional si alguna vez utiliza aplicaciones multiusuario. El uso de modo exclusivo es requerido si usted cambia la estructura de una tabla, borra los registros eliminados, les aplica el comando zap o recrea etiquetas de índices usando etiquetas existentes con el comando Reindex. Es similar a Set Exclusive.

igura 4-18



Herramientas, Opciones, ficha Datos.

Guardar automáticamente Ésta es lo mismo que Set Autosave.

Mostrar nombres de campos Determina si los nombres de campos aparecen como encabezados de columnas en el comando List y algunos otros. Es igual a Headings On/Off.

Pedir tabla de códigos Cuando esta opción está marcada, FoxPro pregunta si desea pegar una página de códigos cuando abre tablas que aún no tienen una. Las opciones de página de códigos controlan el orden en grupos de caracteres de un idioma que no sea inglés. Es igual a Set CPDialog.

Pasar por alto los registros eliminados Determinà si sus programas pueden o no "ver" registros que han sido eliminados. Es igual a Set Deleted.

Optimización Rushmore Activa la optimización Rushmore. Rushmore es la tecnología que permite a FoxPro leer datos indexados en la memoria residente, si su consulta usa datos que ya están en el índice. Algunas veces es mejor desactivar esta opción, y para eso sirve esta casilla. Es igual a Set Optimize.

Registros únicos en índices Utilice esta opción para producir sólo un índice de una sola entrada para cada valor clave único. Es similar a Set Unique.

Secuencia de ordenación Utilice esta opción para proporcionar una secuencia de ordenación alternativa, comúnmente para alfabetos distintos al inglés. Es igual a Set Collate, sin embargo las páginas de códigos resuelven muchos aspectos relacionados.

Intervalo de contador de registros Cuando usted crea etiquetas de índices, copia registros o utiliza alguno de los muchos otros comandos, Visual FoxPro reporta su progreso cada *n* registros. Es igual a Set Odometer.

Tamaño en bloques de memo (bytes) El contenido de los campos memo está almacenado en bloques con un incremento de tamaño específico, el cual se determina aquí. Es como Set BlockSize.

Intervalo de actualización de examen (seg) Una examinación puede ser ejecutada para volver a leer su(s) tabla(s) fuente y volver a desplegar los datos cada ciertos segundos usando esta opción. Es similar a Set Refresh.

Intervalo de actualización de tablas (seg) Esta opción es igual a Intervalo de actualización de examen, excepto que se usa cuando se comparten archivos en una red. Es similar a Set Refresh.

Las siguientes son opciones del grupo Comparaciones de cadenas:

Set Near Cuando esta casilla está marcada, si usted realiza el comando Seek, y no se encuentran coincidencias exactas, Visual FoxPro coloca el puntero de registros en el primer registro después del lugar en que el registro debería estar.

Set Exact Esta casilla causa que el operador de comparación = actúe como el operador ==. Normalmente yo especifico SET EXACT OFF y luego uso = en los casos en los que quiero comparar la cadena sólo hasta el largo de la expresión del lado derecho del signo igual que. Note que a SQL no le importa de qué lado del signo igual está la cadena más corta, así que experimente antes de asumir algo acerca de cómo actuarán sus comparaciones.

Set ANSI Ésta es la versión SQL de Set Exact.

Éstas opciones están dentro del grupo Bloqueo y almacenamiento en búfer:

Bloqueo automático de archivos Activa el bloqueo automático de archivos. Corresponde al comando Set Lock.

Múltiples bloqueos de registros Especifica si usted puede o no bloquear múltiples registros. Corresponde al comando Set MultiLocks.

Almacenamiento en búfer Especifica qué actualización y mantenimiento de datos están protegidos en entornos multiusuario. Esta opción está disponible sólo si usted marcó la opción Múltiples bloqueos de registros. Corresponde a la función CursorSetProp( ).

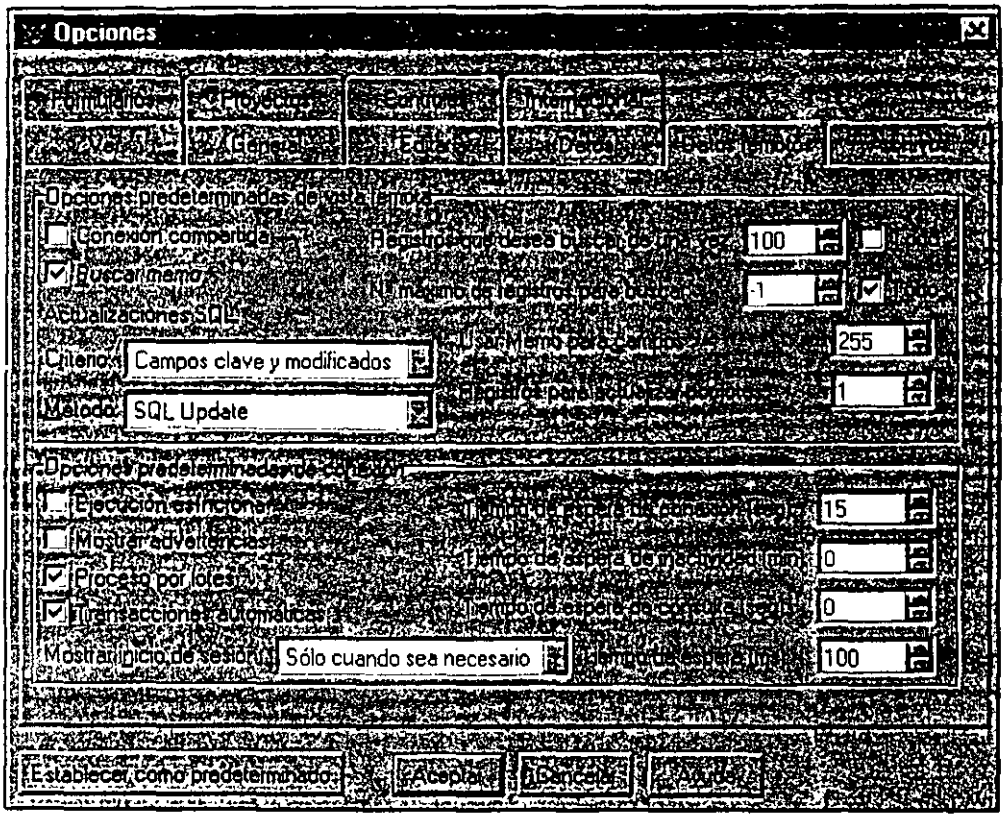
Volver a procesar Controla el número de veces y la duración de los intentos que hace Visual FoxPro para bloquear un archivo o registro después de un intento fallido de bloquear. Es como Set Process.

La ficha Datos remotos

La ficha Datos remotos, mostrada en la figura 4-19, le permite manipular vistas o conexiones a través de ODBC (conectividad abierta en bases de datos). Las opciones siguientes pertenecen al grupo Opciones predeterminadas de vista remota:

85

Figura 4-1



Herramientas, Opciones, ficha Datos remotos.

Conexión compartida Marque esta casilla si desea utilizar la conexión en uso para nuevas vistas.

Buscar memo En lugar de traer campos memo automáticamente a través de una línea telefónica y un módem, esta opción especifica sólo recuperar datos memo cuando el usuario marca una opción que abre el campo memo. Ésta es una idea mucho mejor.

Actualizaciones SQL Esta opción determina si desea o no sobrescribir cambios hechos por otro usuario desde que obtuvo su vista o copia de los datos. La actualización fallará si, cuando intenta enviar actualizaciones de regreso, los criterios seleccionados han sido cambiados en la tabla remota. Los criterios son:

- Sólo campos clave
- Campos clave y campos actualizables
- Campos clave y campos modificables
- Campos clave y marca de hora

Método SQL Update Esta opción determina cuál de los dos comandos SQL disponibles, Update o Delete e Insert, actualizan la información en el servidor. También puede borrar todo con su cursor y reemplazarlo, o encontrar registros que fueron agregados, cambiados y eliminados, y negociar con cada caso específico. Reemplazar todo lo que aparezca con todo lo que está enviando es lo mejor en muchos casos, así que úselo a menos que el desarrollo dicte que usted deba tratar algo con más cuidado.

Registros que desea buscar de una vez Especifica cuántos registros desea enviar a la fuente de datos remotos por vez. Esta opción se usa con ejecución asíncrona, en la cual, por omisión, Visual FoxPro envía 100 registros a la vez. Usted puede examinar los datos que han sido recuperados de vez en cuando e informar a sus usuarios el avance, y los usuarios pueden cancelar si, en su humilde opinión, se está tomando demasiado tiempo.

Nº máximo de registros para buscar Establece un límite superior al número total de registros enviados por una vista. Si tiene una enorme tabla siendo accedida a través de un módem, esto mantendrá a los usuarios lejos de hacer algo que podrían lamentar.

Usar Memo para campos >= Debido a que las tablas de FoxPro tienen diferentes estructuras que la mayoría de las SQL, lo que se conoce como campos de carácter son demasiado pequeños. Si su tabla remota tiene campos de carácter de 500 caracteres de longitud, éstos serán truncados a la longitud máxima para campos de caracteres de Visual FoxPro, que es 254 caracteres, a menos que le indique a FoxPro que convierta los campos de caracteres largos a campos memo durante la salida de su vista.

Registros para actualizar por lotes Especifica el número de registros que se incluyen en un lote cuando actualiza una tabla remota.

Éstos son parte del grupo Opciones predeterminadas de conexión:

Ejecución asíncrona Activa o desactiva el procesamiento asíncrono. Procesamiento asíncrono significa que FoxPro trae de regreso o envía datos de acompañamiento. Su programa tiene que usar el comando SQLEXEC( ) para consultar de vez en cuando si el proceso ha terminado, y para cancelarlo si es que así lo desea el usuario. Es una idea muy buena para tablas remotas grandes.

Mostrar advertencias Esta opción activa o desactiva el despliegue de mensajes de advertencia.

roceso por lotes Activa o desactiva el procesamiento por lotes. Si está marcado, Visual FoxPro no envía resultados a una llamada de SQLExec( ) hasta que todos los grupos de resultados individuales han sido recuperados.

Transacciones automáticas Cada instrucción es considerada por el servidor como una transacción. Si a usted no le gusta esto, puede usar la característica Begin Transaction...End Transaction.

Mostrar inicio de sesión Tiene tres opciones: Sólo cuando sea necesario, Siempre y Nunca.

Tiempo de espera de conexión (seg) Especifica el número de segundos de espera para una conexión con el servidor remoto. Si la conexión no se puede establecer en el tiempo especificado, Visual FoxPro enviará un mensaje de error.

Tiempo de espera de inactividad (min) Debido a que los usuarios pueden retirarse o ir a almorzar y usted puede estar en una llamada de larga distancia, esta opción le permite colgar si no está ocurriendo nada. Pero para el usuario es casi transparente debido a que Visual FoxPro vuelve a intentar conectarse automáticamente si se hace una solicitud al servidor después de que el tiempo de espera ha terminado.

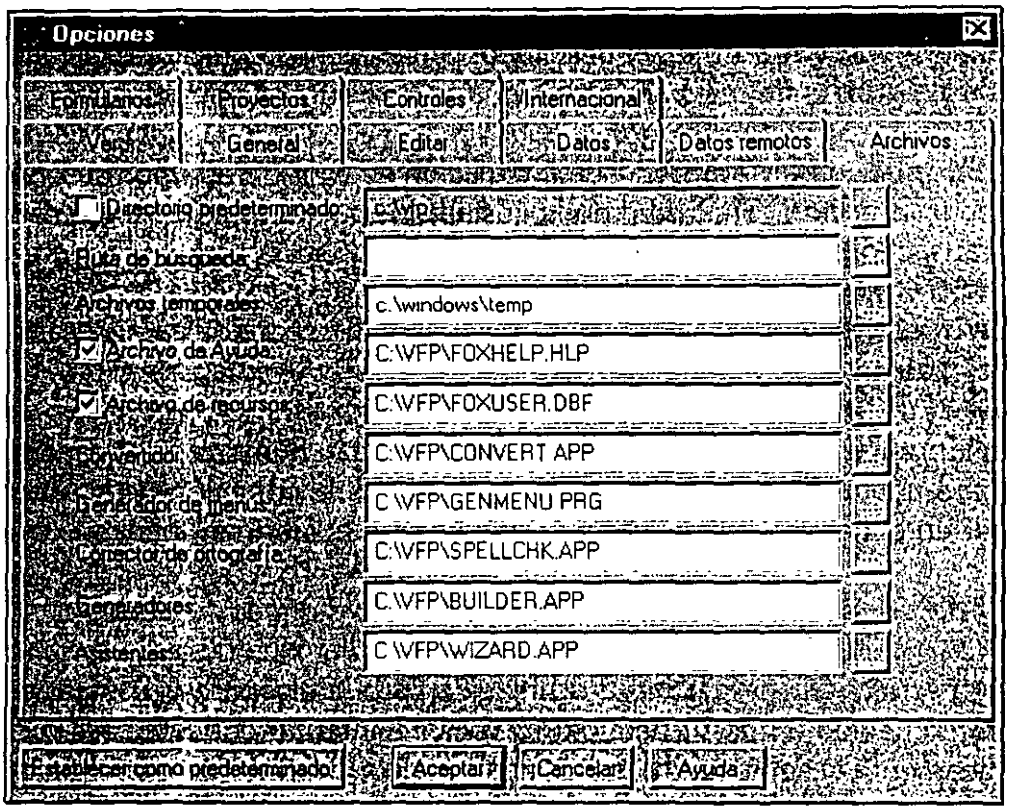
Tiempo de espera de consulta (seg) Algunas veces el servidor está muy ocupado como para poder manejar las solicitudes en un lapso de tiempo normal. Si el servidor toma un tiempo mayor que *n* segundos para procesar la consulta, Visual FoxPro enviará una condición de error.

Tiempo de espera (ms) Puede indicar a FoxPro qué tan a menudo debe revisar si una solicitud ha sido completada o no. Revisar más a menudo provocará más lentitud, pero si se espera que la solicitud sea rápida, esto optimizará los resultados de obtención.

La ficha Archivos

La figura 4-20 muestra la ficha Archivos. FoxPro necesita saber dónde colocó sus generadores, archivo de Ayuda de FoxPro, archivo de recursos y el código generador de menús GENMENU.PRG. Además, FoxPro utiliza archivos de trabajo temporales mientras está siendo ejecutado; usted puede recordar que tiene que borrar unos cientos de archivos del disco duro de un usuario que inocentemente presionó CTRL-ALT-SUPR cuando las cosas no estaban saliendo bien. Aquí es donde usted le dice a FoxPro dónde ponerlos. Si usted los pone en un disco RAM, ellos desaparecerán automáticamente, y algunos aspectos de FoxPro podrían ser

Figura 4-20



Herramientas, Opciones, ficha Archivos.

mesuradamente más rápidos. Sin embargo, esté atento, esas cosas, por ejemplo volver a indizar y ordenar, pueden usar archivos temporales demasiado grandes, así que usar un disco RAM no es bueno para todos.

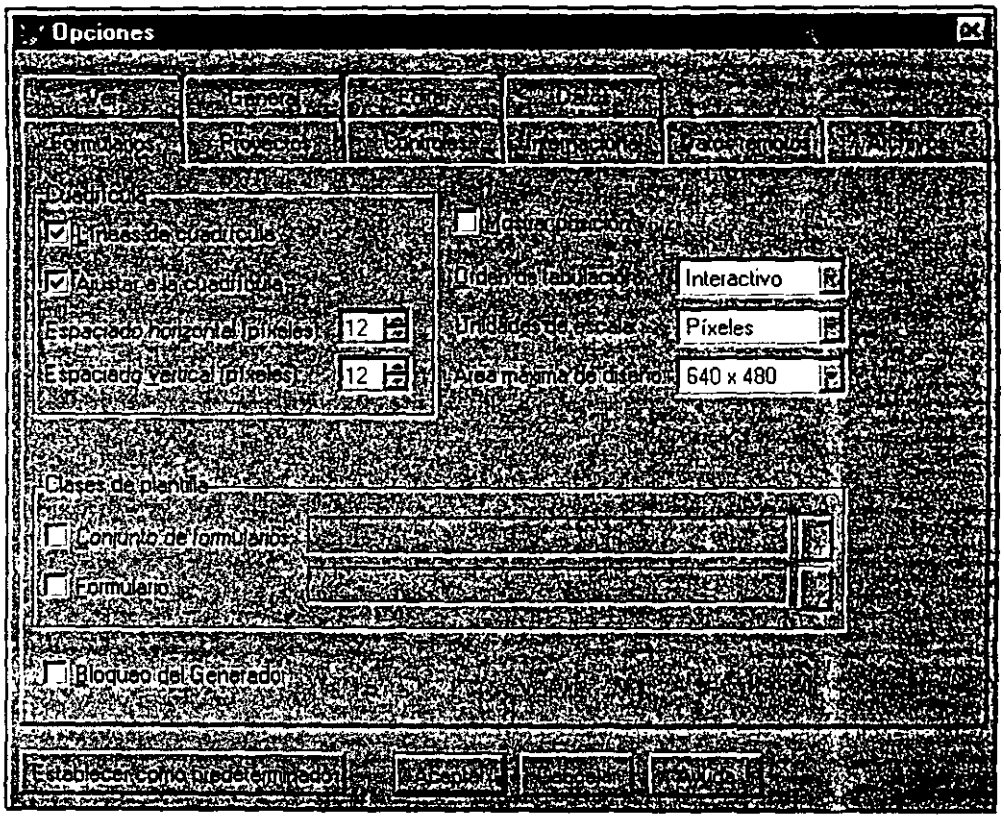
La ficha Formularios

La ficha Formularios mostrada en la figura 4-21, controla las siguientes opciones del Generador de formularios (éstas se encuentran en el grupo Cuadrícula):

Líneas de cuadrícula Despliega líneas de cuadrícula en el Generador de formularios. Las líneas de cuadrícula ayudan visualmente en cuanto al alineamiento de elementos dentro del formulario o informe.

Ajustar a la cuadrícula Activa o desactiva la colocación automática de objetos en las líneas de cuadrícula. Cuando la casilla Ajustar a la cuadrícula está activada. FoxPro coloca cosas en los límites de la cuadrícula más cercana al lugar exacto en donde usted las puso. Si a usted no le gusta cómo trabaja esto, la barra de herramientas Distribución le permite alinear las cosas muy fácilmente.

Figura 4-21



Herramientas, Opciones, ficha Formularios.

Espaciado horizontal Especifica el ancho del espacio entre líneas de cuadrícula horizontales. El ancho está basado en la opción Unidades de escala.

Espaciado vertical Especifica la altura del espacio entre líneas de cuadrícula verticales, en pixeles o caracteres.

Mostrar posición Activa o desactiva la característica que despliega, en la barra de estado, la localización y el tamaño de su elemento en uso.

Orden de tabulación Hay dos maneras de volver a ordenar los campos de entrada en un formulario FoxPro, y ambas son mejores que la forma en que se tenía que hacer con FoxPro 2.6 para DOS. Una utiliza una numeración en pequeños cuadros desplegables, y la otra le permite volver a ordenar la lista de todos los Gets.

Unidades de escala Establece el modo de escala predeterminado, pixeles o fóxeles, para los generadores de formularios y clases.

Área máxima de diseño Si es nuevo en Windows, debería intentar hacer su primera aplicación en una resolución de 1280 x 1024, de esta forma usted tendrá suficiente espacio para poner todas sus herramientas en la pantalla y hacer su formulario tan grande como pueda y después poner sus herramientas fuera de la pantalla. Si lo hace y, su primer usuario tiene una computadora portátil con una resolución máxima de 600 x 480, será visible un poco más de un cuarto de su formulario. Utilice la resolución 600 x 480 para el área máxima de diseño, a menos que esté absolutamente seguro de que cada uno de los usuarios tendrá un monitor de alta resolución. Por cierto, si usted piensa que es el único que cometió ese error en su primera aplicación Windows, sepa que yo fui el idiota que dio el buen ejemplo descrito previamente.

Puede especificar una biblioteca de clases y la clase en la que basará sus nuevos formularios y grupos de formularios. Si no está seleccionada ninguna, FoxPro usa la clase base. Especificar las clases hace que sea más fácil darle a sus formularios una apariencia y sensibilidad consistentes. Las opciones siguientes pertenecen al grupo Clases de plantilla:

Conjunto de formularios La lista desplegable Bibliotecas registradas contiene bibliotecas de clases que usted ha registrado. Puede registrar bibliotecas de clases en la ficha Controles, del marco de página Opciones. La lista Nombres de clases contiene clases de formularios de la biblioteca de clases seleccionada.

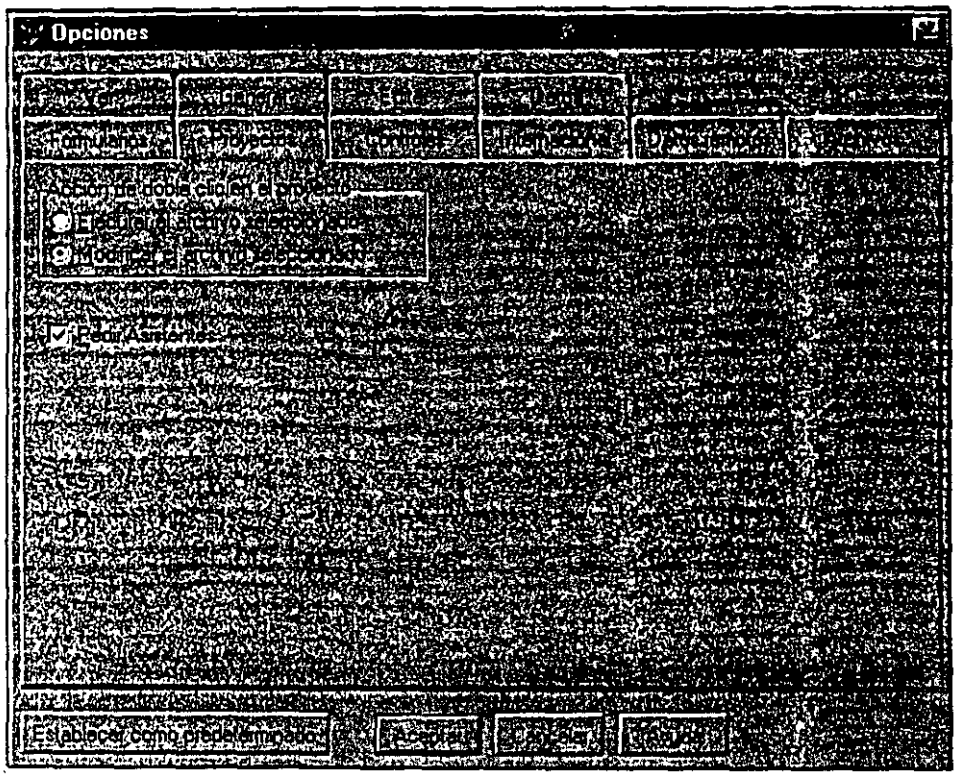
Formulario La lista desplegable Bibliotecas registradas contiene bibliotecas de clases que usted ha registrado. La lista Nombres de clases contiene clases de formularios en la biblioteca registrada seleccionada.

Bloqueo del Generador Cuando esta opción está activada, los generadores se despliegan automáticamente donde es apropiado.

La ficha Proyectos

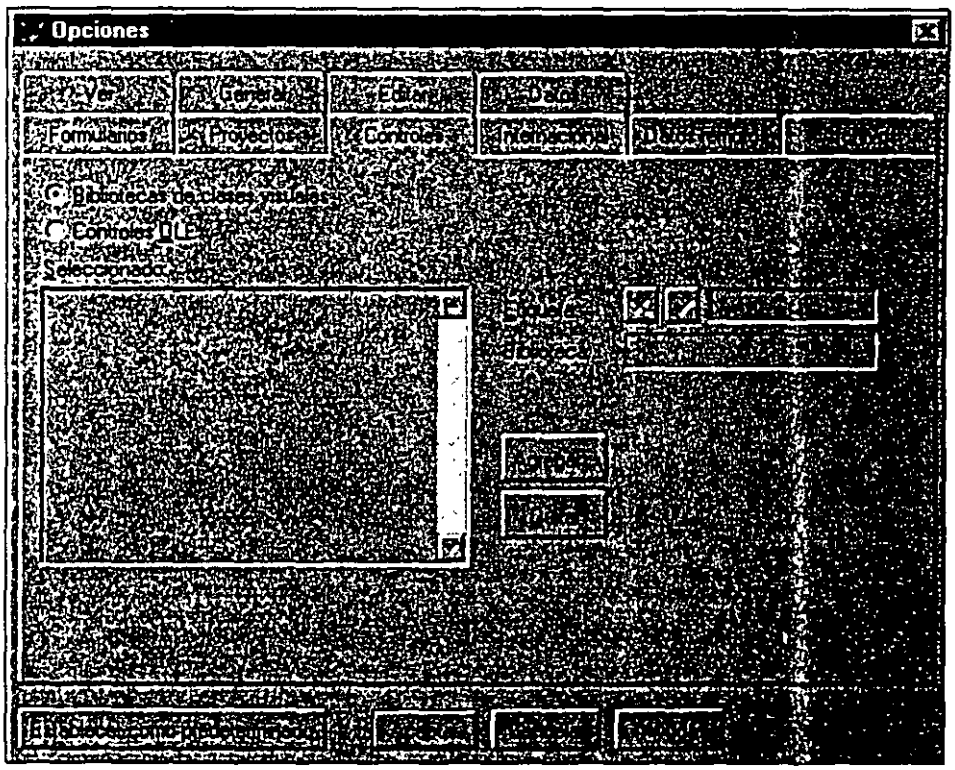
La ficha Proyectos, que se muestra en la figura 4-22, tiene sólo dos opciones: Acción de doble clic en el proyecto, y Pedir Asistentes. Para la primera, cuando usted hace doble clic en un programa, formulario o informe que está resaltado, sus opciones son ya sea ejecutar o modificar el archivo seleccionado. En el ejemplo está marcado Modificar el archivo seleccionado. Para la segunda selección, normalmente lo mejor es seleccionar Pedir Asistentes.

Figura 4-22



Herramientas, Opciones, ficha Proyectos.

Figura 4-23



Herramientas, Opciones, ficha Controles.

La ficha Controles

La ficha Controles, mostrada en la figura 4-23, registra las bibliotecas de clases visuales y controles OLE.

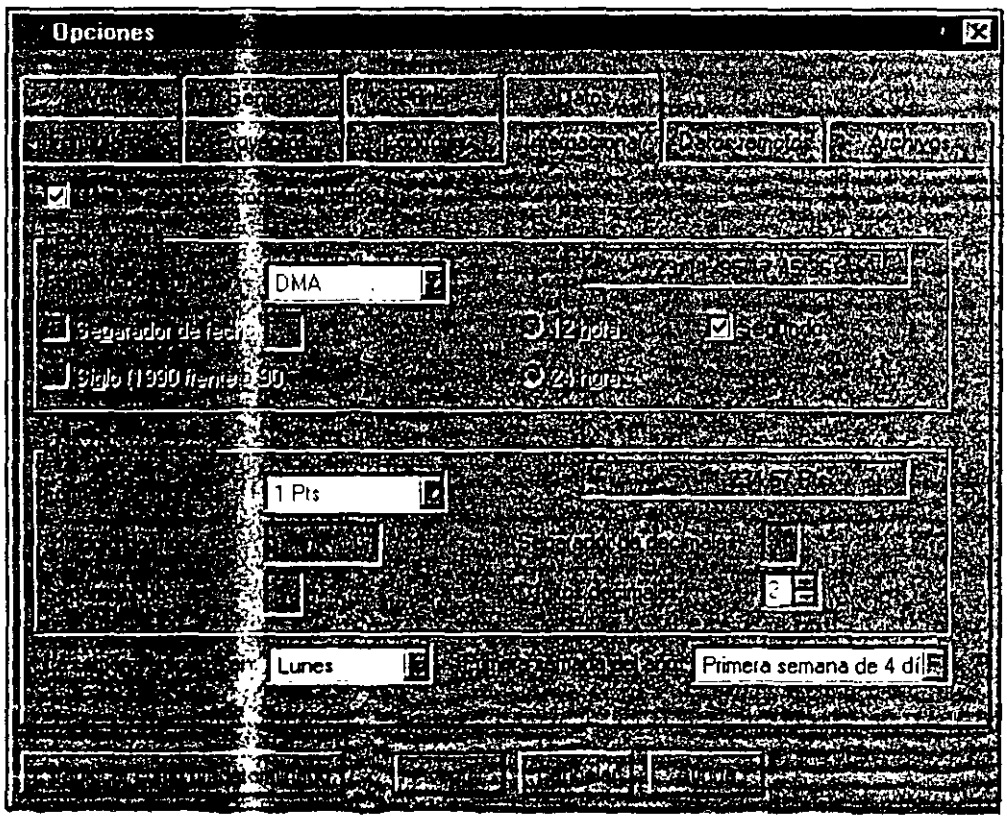
Las bibliotecas de clases visuales, las cuales tienen la extensión .VCX, pueden ser ya sea hechas por usted o las que vienen con Visual FoxPro. Como se ha dicho, hay docenas de VCX que vienen con Visual FoxPro, y es seguro que vendrán más. Otros proveedores para desarrolladores y publicaciones estarán produciendo dichas bibliotecas tan rápido como pueden.

Los Controles OLE vienen en dos sabores: objetos que pueden ser insertados, como clip art, imágenes de Paintbrush, documentos MS Word 6.0, paquetes multimedia y sonido, y Controles, por ejemplo el control Outline.

La ficha Internacional

La ficha Internacional, que se exhibe en la figura 4-24, es de interés especial para desarrolladores y usuarios fuera de los Estados Unidos. Ya que otros idiomas y

Figura 4-24



Herramienas, Opciones, ficha Internacional.

93

países tienen diferentes maneras de representar fechas, moneda y números, este cuadro de diálogo le permite controlar su despliegue. Además, puede usar la casilla de verificación Usar la configuración del sistema, para copiar la mayoría de estas opciones de la configuración del Panel de control de Windows. Hay opciones separadas para todas las distintas formas en que se puede dar formato a la fecha, incluyendo opciones separadas para Inglaterra y Francia, aunque estos países usan precisamente el mismo formato.

Familiarícese con las distintas opciones. No es poco común gastar mucho tiempo planeando cómo trabajar alrededor de un comportamiento de FoxPro, sólo para descubrir que el comportamiento deseado estuvo disponible con el simple oprimir de un botón.

El menú Programa

El menú principal Programa es útil durante la depuración y pruebas. Tiene cinco opciones de menú, como se muestra en la figura 4-25.

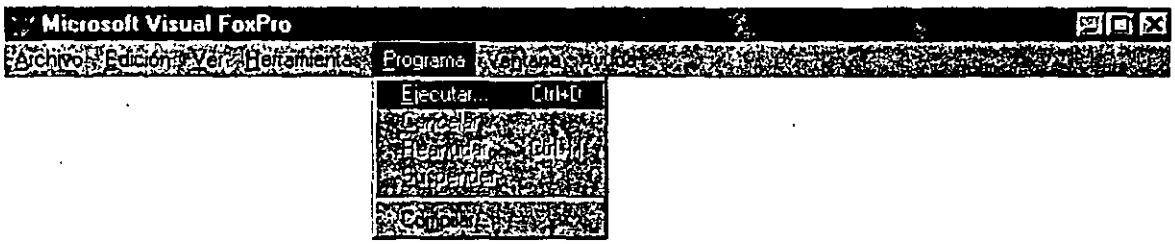


Figura 4-25

Menú principal Programa.

Ejecutar Despliega un cuadro de diálogo para buscar un programa y ejecutarlo. Si usted no ha hecho esto antes, haga clic en Programa, Ejecutar, y luego encuentre su archivo y ejecútelo:

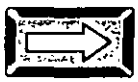
```
VFP\EXAMPLES\CONTROLS\CONTROL.APP
```

Cancelar Cancela la ejecución de un programa que está activo.

Suspender Suspende la ejecución de un programa que está activo; puede abrir las ventanas Depuración o Seguimiento, curiosear por ahí y luego escoger Reanudar, para continuar.

Reanudar Reanuda la ejecución de un programa suspendido.

Compilar Compila un archivo con la extensión .PRG o .MPR. Se producen archivos compilados con las extensiones .FXP y .MPX respectivamente. La compilación se hace para corregir sintaxis, o si quiere distribuir parte de su aplicación como archivos separados. El código de compilación normalmente se almacena en el archivo de proyecto. De hecho, ése es uno de sus principales propósitos.



El menú Ventana

El menú desplegable Ventana se muestra en la figura 4-26. Tiene las siguientes opciones:

Organizar todo Cuando hago clic aquí, me toma un minuto y medio llevar mi ventana Comandos de regreso a la forma en que me gusta. Estoy seguro de que existe un uso adecuado para esta opción, pero yo no lo he descubierto.

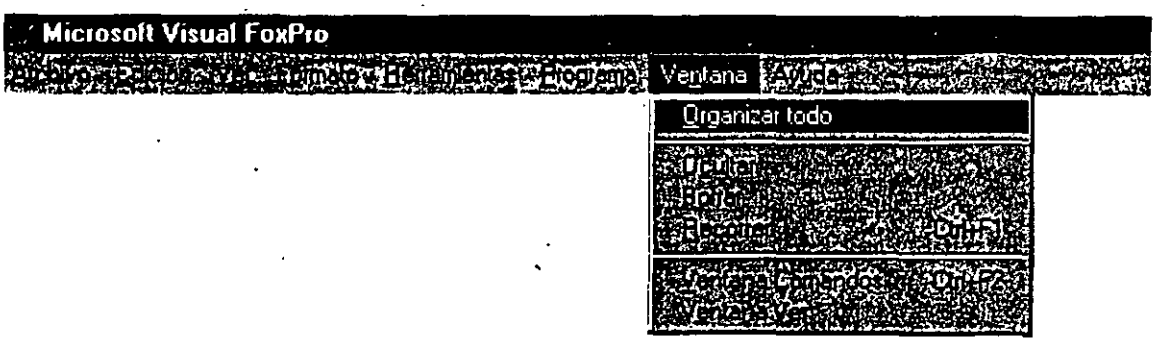
Ocultar Oculta la ventana seleccionada.

Borrar Borra la ventana seleccionada.

Recorrer Para moverse exitosamente de ventana en ventana, puede ya sea hacer clic en esta opción o presionar CTRL-F1.

Ventana Comandos Para saltar directamente a la ventana Comandos de FoxPro, presione el método abreviado CTRL-F2.

Figura 4-26



El menú desplegable Ventana.

Ventana Ver Esta opción de menú le da una visión de sus tablas abiertas y la relación que hay entre ellas. Es como el Generador de bases de datos, pero puede usarlo sin construir un proyecto o diseñar una pantalla.

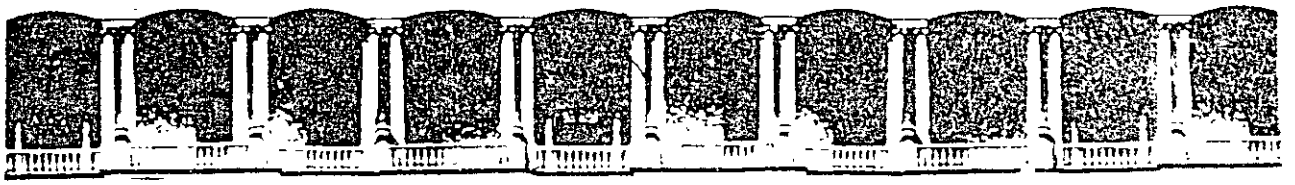
El menú Ayuda

Ayuda despliega el sistema de Ayuda de FoxPro. Es muy parecido al sistema de Ayuda de Windows, debido a que está escrito como una aplicación de Ayuda Windows. De hecho, para aquellos que lo prefieren, hay un sistema basado en DBF que es paralelo a éste. Usted decide si hace un clic en Ayuda y pasea por sus ventanas hasta que se sienta en casa.

Cuando usted desarrolla sus propios sistemas de ayuda, puede usar ya sea el estilo de ayuda DBF o el estilo de ayuda Windows. Existe un excelente programa shareware desarrollador de ayuda Windows.

Esto concluye su visita guiada alrededor del sistema de menús de Visual FoxPro. Estoy seguro de que muchas de las opciones tendrán más sentido cuando las use en un entorno, pero al menos usted tiene una idea general de la profundidad del producto.

En el próximo capítulo, escribirá unos cuantos programas para familiarizarse con la sintaxis, los comandos y las funciones del lenguaje de programación de FoxPro. Aunque las propiedades y métodos eliminan mucho de lo que usted estaba acostumbrado a hacer, aún tiene que escribir un pequeño código. Creo que encontrará interesante el capítulo de programación. Después de todo, aún somos programadores.



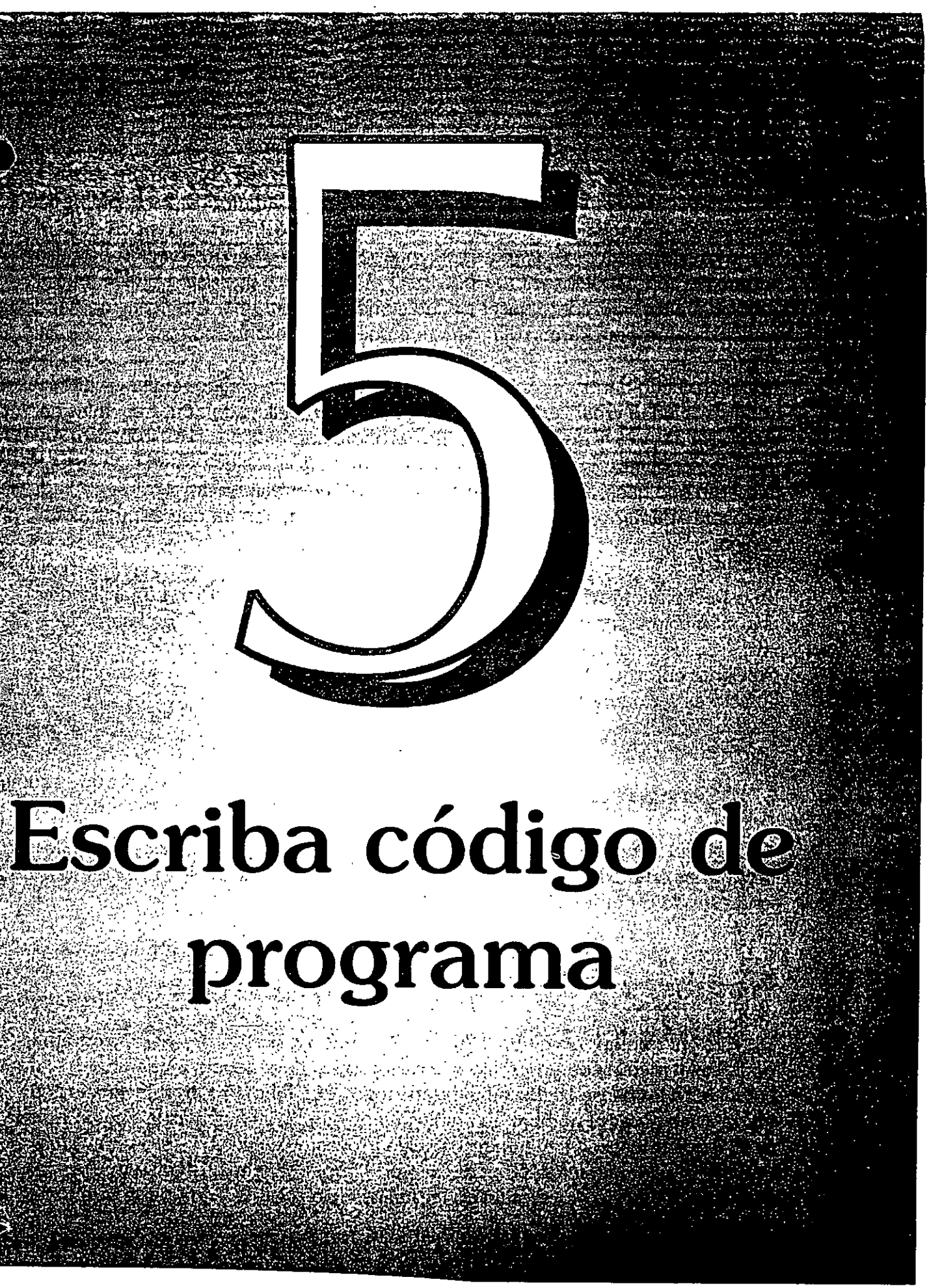
FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

COMPLEMENTO DE MATERIAL DIDACTICO

VISUAL FOXPRO

OCTUBRE, 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico, D.F. APDO. Postal M-2285
Telefonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26



5

Escriba código de programa

CAPITULO 5

HASTA ahora, si usted tiene experiencia en programación con versiones anteriores de FoxPro podría estar poniéndose un poco nervioso. Si todo esto le parece nuevo y de alguna forma intimidante, estará contento de saber que, con excepción de la sintaxis utilizada en propiedades y métodos, ha habido pocos cambios en el lenguaje de programación de Visual FoxPro. Hay unos cuantos cientos de comandos para sustentar el nuevo entorno y la construcción de clases, pero éstos son paralelos al entorno visual; esto significa que usted puede tanto apuntar y hacer clic como hacer declaraciones de programa. Las cosas no han ido tan lejos como para cambiar la manera en que usted escribe código.

Aunque mucho dentro de Visual FoxPro no requiere de código, el día en que los usuarios puedan crear sus propias aplicaciones no ha llegado aún. Las hojas de propiedades para formularios y objetos proporcionan formas para controlar muchas conductas de los objetos, pero siempre hay cosas que usted y sus usuarios desearán que haga el programa y que no se pueden realizar simplemente mediante apuntar y hacer clic. Probablemente alguien tendrá que escribir algún código.

En este capítulo, revisaremos los elementos del lenguaje FoxPro. La facilidad con la que principiantes pueden conseguir que algo "funcione bien" fue la atracción inicial de los lenguajes xBASE. Yo creo que usted estará de acuerdo en que es un excelente entorno de aprendizaje. De hecho, algunos sienten que su mayor defecto es ser demasiado complaciente. Sea usted el juez.

Programación básica

Las siguientes secciones explicarán los componentes básicos de la escritura de programas en FoxPro, sin embargo mucha de la información se aplica a todos los lenguajes de programación.

Nombres y descripciones de programas

En versiones anteriores de FoxPro, era una práctica común preceder programas, pantallas y menús con un prefijo que denotaba su tipo, debido a que el Administrador de proyectos alfabetizaba todo. Visual FoxPro arregla eso, así que ahora puede usar el nombre de archivo completo. Existe además la opción Editar descripción, dentro del menú desplegable Proyecto, la cual le permite introducir una descripción de una línea para cada programa. Mientras usted se mueve de un elemento a otro, la descripción que usted escribió aparece en la parte inferior de la

pantalla del Administrador de proyectos. Esto puede ser muy útil para otros p amadores que intentan imaginar qué programa hacé que cosa.

Ejecute programas

Para ejecutar código de programas, puede basarse en varias sintaxis:

```
DO nombredeprograma
= nombredeprograma()
Objectname.MethodName
```

Además, puede incluir código en un método asociado a un evento. Cuando el evento se arranca, su código se ejecuta. El evento Click de un control es un ejemplo de un lugar para incluir código de programa.

Funciones vs. procedimientos

Yo siempre he ignorado la diferencia entre funciones y procedimientos debido a que FoxPro así lo hace, pero tradicionalmente las funciones regresan valores, por (iplo, terminar con la última línea:

```
RETURN ( valor )
```

mientras que los procedimientos no. A FoxPro esto no le interesa en lo absoluto. Si usted es obsesivo en cuanto a seguir la reglas, ahora ya las conoce. En código de métodos, usted no utiliza la función o procedimiento en la primera línea, como se hace en código .PRG.

Dónde viven los programas

Los programas son escritos dentro de archivos con la extensión .PRG, en ventanas de código de procedimientos en el Generador de menús, en ventanas de código de métodos en el Generador de formularios, o en el Generador de clases.

Para iniciar un programa nuevo puede, ya sea hacer un clic en CTRL-N y Programa, usar el menú para seleccionar Archivo, Nuevo, Programa, o escribir modify command en la ventana Comandos. Si usted está en el Administrador de proyectos, resalte Programas y luego haga clic en Nuevo. Cuando lo guarde y ı, le tendrá que dar un nombre DOS válido. Para editar un programa

3

existente, vaya al Administrador de proyectos y haga ya sea un doble clic en su nombre o resalte el nombre del programa y presione ENTER.

Otra forma de escribir código es agregar un nuevo método a un formulario o clase. Dentro de los Generadores de formularios o clases, haga clic en el menú Formulario (el menú Clase en el Generador de clases), luego haga clic en Nuevo método. Una vez que ha proporcionado un nombre, puede hacer doble clic en cualquier lugar dentro de la pantalla del formulario o clase; asegúrese de que ha seleccionado el mismo formulario o clase en la lista Objeto (es el primer elemento), luego escoja el nombre de su nuevo método de la lista Métodos y escriba ahí su código.

Dentro de un programa, usted puede llamar a otros programas. Estos programas pueden localizarse en sus propios archivos .PRG, pero es más común juntar todos los programas llamados por uno solo y ponerlos dentro de un mismo archivo .PRG bajo el código que los llama. Ésta es una estructura de programa típica:

```
PRINCIPAL.PRG
   ...
FUNCTION Primera
   ...
FUNCTION Segunda
   ...
PROCEDURE MAS
   ...
PROCEDURE Denuevo
   ...
FUNCTION Loquesea
   ...
```

Si tiene muchas funciones que varios programas podrían llamar, puede ponerlas a todas dentro de un solo archivo .PRG e incluirlo en su proyecto. Puede poner todo ese tipo de funciones dentro de una sola biblioteca de procedimientos o en varias bibliotecas. Entonces incluya algo como esto:

```
SET PROCEDURE TO BIBLIOTECA1
SET PROCEDURE TO BIBPROC2 ADDITIVE
```

en su programa Principal. Visual FoxPro encontrará sus funciones cuando sus programas las necesiten. El mero hecho de que una biblioteca de procedimientos esté ligada dentro de su proyecto, no le da a Visual FoxPro suficiente información para encontrar las funciones que están incluidas ahí. Con el propósito de que FoxPro registre la existencia de funciones incluidas, el archivo que las contiene debe ser ejecutado en algún lugar dentro del proyecto, es decir, debe haber una línea de código, ya sea DO NombreProg, o =NombreProg( ), las dos formas de

llamar a un programa. Esto causará que el Generador de proyectos evalúe el contenido de NombreProg y registre todas las funciones ahí incluidas.

Haga que el código sea legible

Visual FoxPro entenderá su código siempre y cuando sea correcto en cuanto a sintaxis, aunque éste pueda verse como si no fuera nada. Pero eso no significa que un código sea mejor que otro.

* Espacio en blanco

Hay muchas formas de hacer que su código sea fácil de leer; el espacio en blanco es una de ellas. Espacios en blanco significan áreas vacías dentro de código de programa, también espacios, tabulaciones o líneas vacías. Un estudio sobre la productividad de programadores encontró que dos factores que se correlacionan en gran medida con las habilidades de un programador fueron la cantidad de espacio utilizado entre bloques de código y el grado en el cual el código fue alineado de manera que visualmente no fuera complicado. Esto no afecta el tamaño del programa compilado, así que no hay vuelta de hoja.

* Sangrado

Sangrar es la mejor manera de optimizar su código. Es una práctica común sangrar dentro de comandos pares anidados, por ejemplo:

```
DO...ENDDO
IF...ENDIF
FOR...NEXT
SCAN...ENDSCAN
DO CASE...ENDCASE
WITH...ENDWITH
```

El editor Visual FoxPro automáticamente hará el sangrado por usted. La página Editar, de Herramientas, Opciones, le permite activar el sangrado automático (el valor que está como predeterminado) y establecer el número de espacios del sangrado para cada tabulador. El tabulador inserta caracteres de tabulación en lugar de espacios. Si usted prefiere usar espacios -CH(32)-, está en su derecho.

* Usar mayúsculas y minúsculas

Muchos programadores utilizan mayúsculas y minúsculas para distinguir entre los elementos del programa. Yo prefiero mayúsculas para los comandos de Visual FoxPro, minúsculas para variables de memoria y una combinación de mayúsculas seguidas de minúsculas para funciones.

* Comentarios

Los comentarios pueden existir en una línea, precedidos por un asterisco:

```
* Éste es un comentario.
```

También pueden estar después de una línea de código, si están separados por dos signos & consecutivos, de este modo:

```
FOR I = 1 TO NumeroCampos              && Uno para cada campo en la lista.
```

* Continuar líneas largas de código

Si una línea de código es tan larga que no cabe dentro de una sola línea, puede permitirle simplemente pasar a la siguiente línea. Sin embargo, a mí no me gusta hacer eso debido a que se ve desalineado y es difícil de leer. Yo publico un artículo periódicamente en el cual hacer que el código sea legible es una parte principal de mi trabajo. Además, la lógica de un programa puede ser más fácil de seguir si cosas similares están alineadas una debajo de otra.

Usted puede continuar líneas largas de código en la siguiente línea insertando un punto y coma al final de la línea, como sigue:

```
DO WHILE NOT EOF()      ;
   AND contador < 10    ;
   AND nombre <> [PERALTA]
```

A mí me gusta sangrar código que está dentro de construcciones que se abren y cierran, como For...EndFor, Scan...EndScan, If...EndIf y With...EndWith. Verá que esto hace que el código sea más fácil de leer.

* Incluir archivos

Visual FoxPro le permite usar archivos que contengan instrucciones DEFINE para hacer el código más legible. En particular, hay un archivo en el directorio \VFP llamado FOXPRO.H, el cual contiene instrucciones DEFINE que dan nombres en inglés a muchos de los parámetros requeridos por las funciones de FoxPro que tienen un argumento. Una expresión como IF TYPE(T_DATE) es más fácil de leer que IF TYPE("D").

Ejecute programas

Para ejecutar un programa, utilice, ya sea DO NombreProg o =NombreProg( ). Si el código que desea ejecutar es un método definido por el usuario o está contenido en una ventana de código de un evento en un objeto, simplemente llámelo por su nombre: THISFORM.MiMetodo o THISFORM.BOTON.Click. Los programas pueden llamar a otros programas explícitamente con estas dos sintaxis.

Además, como Visual FoxPro soporta eventos, el código de programa se ejecuta con frecuencia debido a que un usuario hace clic en un elemento o porque algún otro evento integrado arranca la ejecución del código correspondiente a ese evento. Usted no puede agregar eventos al juego de eventos de FoxPro, pero hay tantos de ellos que es difícil imaginar el caso en el que usted necesite hacerlo. Daremos un vistazo a los eventos en el capítulo 7.,

Compilación

Cuando ejecuta una aplicación Visual FoxPro, usted ejecuta código compilado, una especie de traducción de lo que realmente escribió. FoxPro en realidad no npila su código; en otras palabras, no lo traduce a lenguaje máquina. En vez de eso lo simboliza, o lo traduce a un formato interno que es más compacto y fácil de manipular. Si usted distribuye aplicaciones Visual FoxPro en formato de tiempo de ejecución (runtime), sólo el código compilado puede ejecutarse. El entorno de desarrollo no se envía con el código, y el entorno de desarrollo contiene el compilador.

Todo el código de un proyecto se compila y almacena en un archivo de proyecto. Si FoxPro encuentra un archivo .PRG, primero tiene que compilarlo. Como el compilador revisa los errores de sintaxis, es una buena idea marcar la casilla de verificación Compilar antes de salvar, en la ficha Editar del menú Herramientas, Opciones. Entonces, cuando usted cierra un archivo .PRG después de editarlo, inmediatamente sabe si su código tiene algún error de tipografía u otro problema de sintaxis. La mayoría del código que usted escriba no será almacenado en archivos .PRG. Un formulario o clase de Visual FoxPro puede contener docenas de métodos, cada uno de los cuales es realmente un campo memo dentro de un archivo .SCT o .VCT, para el formulario o clase. Sin embargo, el código de los métodos es código FoxPro, y se compila y ejecuta de acuerdo con las mismas reglas. Por ello, no se sorprenda si se descubre a sí mismo recortando el camino de regreso con los archivos .PRG. En muchas aplicaciones creadas con Visual FoxPro, el único archivo .PRG es el programa Principal.

Macros y Evaluate

Uno de los aspectos verdaderamente distinguidos de Visual FoxPro es su capacidad para compilar y ejecutar sobre la marcha líneas individuales de un programa con la expansión de macros. Si usted teclea:

```
x = [? "Hola" ]
&x
```

Visual FoxPro imprimirá "Hola" en la pantalla. Usted puede construir comandos a partir de pedazos y fragmentos de código, para luego ejecutarlos con esta sintaxis. Yo no sé si esto es una bendición o una maldición, debido a que crean muy malos hábitos de programación. Es del tipo de Go To de xBASE. Aun así, puede convertirse en una valiosa ayuda. La función Evaluate hace lo mismo, y se sabe que trabaja más rápido.

Funciones y procedimientos

Aunque un programa que sea el único habitante de un archivo .PRG no necesita ser precedido por nada, a mí me gusta colocar un programa de encabezado, el cual es una serie de comentarios:

```
* ID-de-programa       : nombre
* Autor                : Perezgrovas, Alberto
* Propósito            : Dibujar cuadros en la pantalla
* Llamado por          : MuchosArchivos.PRG
* ************************************************************
```

No tengo una regla rígida acerca de lo que va aquí, sólo quiero tener una idea de lo que estoy viendo. Y si usted envía un listado del programa a su impresora y su software no imprime un encabezado con el nombre de archivo en él, espero que tenga buena suerte tratando de averiguar qué es lo que se ha impreso.

Instrucciones de compilación

El código se puede mejorar considerablemente con instrucciones de compilación. La instrucción Define es una de ellas. Las instrucciones de compilación son precedidas por el signo (#). Son soportadas tres, al momento de esta publicación: #Define/#Undefine, #If...#Else...#Endif e #Include.

Es una buena costumbre de programación asignar constantes a variables de memoria al principio de su programa, así usted sólo tendrá que cambiarlas en un solo lugar. Es todavía una mejor costumbre usar la instrucción #Define, la cual permite no sólo definir constantes en un lugar, sino que evita tener que usar variables de memoria para ellas. El preprocesador realmente va a través de su código y reemplaza, con la constante, todas las coincidencias con el elemento definido. Así que además ahorra espacio en la aplicación compilada. El código:

```
#DEFINE FechCadu DATE() + 90  && Deles tres meses
...
...
IF DATE() > FechCadu          && ¡Evaluado cuando se compila!
   WAIT WINDOW ;
      [Su software ha expirado. Envíenos un cheque]
      TIME 2
      QUIT
ENDIF
```

proporciona un método primitivo pero eficaz para evitar que las copias de pruebas de su software se salgan de control. #Define también encontrará cadenas coincidentes en sus comentarios.

Usted puede construir un archivo de instrucciones #Define y agregarlo a cualquiera de sus archivos .PRG con un solo comando, usando #Include. Si se da por hecho que usted tiene un archivo llamado ENCABEZA.TXT, con todas sus instrucciones #Define dentro de él, sólo escriba esta línea en la parte superior de cada archivo .PRG:

```
#INCLUDE Encabeza.TXT
```

La instrucción #IF...#ELSE...#ENDIF se usa para incluir código condicionalmente. Por ejemplo:

```
#IF _DOS
    Declaraciones que se incluyen si se compila bajo DOS
   #ELSE
    Declaraciones que se incluyen si no se compila bajo DOS
#ENDIF
```

Los formularios y las clases también permiten agregar un archivo Include. Abra un formulario, luego haga clic en la opción Incluir archivo, del menú Archivo. Usted estará en la posición de escribir el nombre de su archivo a incluir (con frecuencia llamado un archivo de encabezado debido a su uso, dentro de los programas C, pues proporciona definiciones de función requeridas en el principio de un archivo de código fuente).

Por ahora, usted ha visto la mayoría de las bases. Vayamos al fondo de los detalles del lenguaje de programación. ¿Me acompaña?

Dónde residen sus datos

En los programas Visual FoxPro, los datos son almacenados en tres lugares:

- Campos en tablas
- Variables de memoria, normalmente llamadas *memvars*
- Propiedades de formularios que usted define

Si una tabla está en uso pero no está seleccionada, se puede referir a sus campos con ALIAS.NombreCampo. Por ejemplo, para imprimir el contenido del campo Nombre en la tabla Cliente, usted puede escribir:

```
? CLIENTE.Nombre
```

Si el campo al que usted quiere referirse está dentro del archivo actualmente seleccionado, no tiene que incluir el alias; esto hace que su código no sea ambiguo, una cualidad admirable en código de programa. Las variables de memoria pueden tener muchos de los tipos de datos como los que hay en campos de tablas, excepto datos de tipo General:

- Carácter
- Monetario
- Numérico
- Flotante
- Fecha
- Fecha-hora
- Doble
- Lógico
- Memo
- Carácter (binario)
- Memo (binario)

10

Las variables de memoria, al igual que los campos, pueden tener nombres muy largos. Creo que el límite es 254 caracteres, pero usted puede ignorar esto, a menos que su sistema de valores sea muy extraño. Lo que sería agradable que tuvieran los nombres largos de variables de memoria son espacios, sin embargo, no están permitidos.

Las variables de memoria en Visual FoxPro, a diferencia de cualquier otro lenguaje de programación, pueden cambiar su tipo. Por ejemplo, lo siguiente es perfectamente legal:

```
Name = 3
Name = "Perez"
Name = .F.
```

Por motivos de organización, algunos programadores utilizan la notación húngara, inventada por uno de los primeros programadores de Microsoft. Usted da un prefijo a todas las variables de memoria con dos letras: la primera indica una variable local o global, y la segunda representa el tipo de variable de memoria. Por ejemplo, lcNombre es una variable local de tipo carácter. Conocí al tipo que inventó la notación húngara. Él es un compañero muy agradable, pero creo que la notación húngara es completamente opcional dentro de FoxPro, ¡y yo no soy húngaro! Simplemente utilice la notación que tenga sentido para usted. Las variables de memoria son creadas principalmente en las formas siguientes:

- Asignación, poniendo su nombre a la izquierda del operador de asignación =, por ejemplo, X = 3. Una sintaxis alternativa es STORE 3 TO X.
- Utilizando el comando Scatter MemVar, el cual crea una variable de memoria para cada nombre de campo en la tabla seleccionada, por ejemplo:

```
SELECT CLIENTE
SCATTER MEMVAR
```

- Nombrando variables de memoria después de una instrucción Public, Private o Local, por ejemplo, PUBLIC a,b,c.
- Nombrando arreglos, que también son variables de memoria, después de una instrucción Dimension o Declare, por ejemplo, DIMENSION nombres(20).
- Ejecutando un comando o una función, tal como Select...Into Array, Copy Structure Extended, Copy to Array y Adir( ), la cual produce arreglos como sus datos de salida.

- La instrucción FOR I = 1 TO 10, la cual también asigna un valor a I.
- Creando una propiedad en un formulario o clase.

Incluí este último elemento debido a que las propiedades definidas por el usuario actúan como variables locales unidas a su objeto propietario. Usted puede almacenar valores dentro de propiedades, igual a como lo puede hacer en variables locales.

Si está en el Generador de formularios o clases, el menú Formulario o Clase tendrá una opción de menú llamada Nueva propiedad, si hace clic en ella, se convertirá en una de las propiedades que se pueden configurar de ese formulario o clase. Aparece en la página Otros del marco de página Propiedades.

Las variables de memoria, como su nombre lo implica, residen en la memoria. Cuando su programa ha terminado, desaparecen. Las variables de memoria sirven para los propósitos de los programas. Hay una forma de salvarlas en el disco y recuperarlas (los comandos Save y Restore), pero el tiempo y la experiencia han desacreditado a estos comandos. Las tablas son una mejor forma de almacenar sus datos.

Las variables de memoria solían ser una parte importante de la generación de pantallas FoxPro, pero ahora su uso como campos de entrada no sólo no es necesario, sino que es un error. Una nueva característica, llamada *row buffering*, ocasiona que Visual FoxPro considere los nombres de campos dentro de los campos de formularios exactamente igual a como consideraba las variables de memoria. Esto se menciona debido a que usar variables de memoria en pantallas de entrada solía ser un acto de fe, y eso ha cambiado. Además, en el contexto de las convenciones de nombramiento, había buenas razones para asegurar que campos similares en tablas diferentes tuvieran nombres ligeramente distintos, como NomClien y NomTien en lugar de Nombre.Cliente y Nombre.Tienda. Ahora esto no importa en lo absoluto. A usted le va a encantar esto.

Comandos y funciones

El lenguaje de Visual FoxPro consiste en comandos y funciones. Primero se presentarán algunos de los comandos principales, luego se describirán las 40 funciones que la mayoría de los programadores utiliza frecuentemente. Si usted cuenta el número de comandos en FoxPro, el resultado es abrumador. Pero probablemente usará sólo un subgrupo de ellos en sus programas.

12

Comandos de asignación

El primer tipo de comando que querrá conocer es uno de asignación, al cual nos referimos en la sección previa. Para crear variables de memoria y darles valores al mismo tiempo, escriba lo siguiente:

```
X = 3
STORE "Lunes" TO DiaEntrega
=ADIR(MYDBFS,"DBF")
```

Para solamente crearlas, utilice alguna de estas opciones:

```
PUBLIC A,B,C
PRIVATE X,Y,Z
LOCAL CHICO, TRABAJA, BIEN
DIMENSION NAMES(3)
```

Si usted crea una variable de memoria y no le asigna un valor, su valor predeterminado es un valor de tipo lógico falso (.F.).

Las variables de memoria normalmente se declaran en el programa Principal, aquel que llama a todos los demás programas. Si declara una variable de memoria en el programa Principal y luego cambia su valor en el programa B, el cual es llamado por el programa Principal, cuando usted regrese al programa Principal, la variable de memoria se mantendrá cambiada. Eso es lo que usted esperaría. Pero, si el programa Principal llama a los programas B y C, una variable de memoria que usted cree en el programa B no existirá, por lo menos en lo que respecta al programa Principal. Incluso será inexistente también para el programa B. Por ello, usted querrá declarar todas sus variables globales en el programa Principal.

La excepción es cuando usted quiere tener su código principal limpio. Puede declarar variables de memoria públicas en un programa nombrado como B y ellas se comportarán como si hubieran sido creadas en el programa Principal. Sin embargo, no intente hacer pública una variable de memoria que ya existe, FoxPro estallará. Si usted quiere dejar las declaraciones públicas fuera de su programa Principal, hágalo al principio de su programa.

Operadores

Los operadores + y - son, por supuesto, usados para añadir y substraer números, pero también se pueden usar en campos de caracteres. El operador más eslabona s expresiones de caracteres, campos o variables de memoria, de forma que

Fred + Smith producirá Fred Smith. El operador menos quita espacios rezagados, así que Fred - Smith produce FredSmith. TRIM(Nombre) + " " + TRIM(Apellido) hace exactamente lo que quiero, así que no uso el operador menos para texto. Los otros operadores, / (división), * (multiplicación), ^ (exponencial) y % (módulo), funcionan como usted lo espera.

Operaciones cíclicas y ramificaciones

Las operaciones cíclicas y de ramificación son los mecanismos para crear la lógica de un programa. Veamos los principales comandos disponibles en FoxPro. El siguiente fragmento de código imprime los números del 1 al 10:

```
FOR I = 1 TO 10
   ? I
ENDFOR
```

Esto lee un archivo desde el inicio hasta el final:

```
SELECT CLIENTE
SCAN
   ? CLIENTE.Nombre
ENDSCAN
```

y éste hace la misma cosa:

```
SELECT CLIENTE
GO TOP
DO WHILE NOT EOF()
   ? CLIENTE.Nombre
   SKIP
ENDDO
```

Aunque Scan...EndScan se usa exclusivamente con tablas y no requiere de los comandos Go Top, While Not EOF( ) y SKIP, DO WHILE no necesita estar asociado a una tabla. Por ejemplo, DO WHILE .T. o DO WHILE I <= 10 son utilizados con frecuencia sin ninguna referencia a una tabla en particular. FOR I = 1 TO 10 es diferente de DO WHILE I <= 10, ya que automáticamente asigna un valor de 1 a I. Si usted no hubiera inicializado I para 1, DO WHILE I <= 10 causaría un error. Las diferencias son sutiles, pero cada construcción tiene su máxima utilidad propia. El progreso del comando Scan...EndScan puede ser alterado utilizando una expresión For, la cual quiere decir "procesa sólo los registros en esta tabla para los cuales la expresión es evaluada a .T.", y la expresión While, la cual significa "procesa sólo los registros en la tablas en que la expresión es evaluada a.T., empezando desde el registro actual".

14

Usted puede salir de estas construcciones cíclicas en cualquier momento mediante el comando Exit. Pero para eso, se necesita comentar el comando If, el cual controla la ramificación. La ramificación en Visual FoxPro es controlada con la construcción If...EndIf. Por ejemplo, puede escribir:

```
IF I > 10
   ? [El contador excedió 10]
ENDIF
```

Los contadores se utilizan a menudo para salir de operaciones cíclicas, como aquí:

```
I = 1
DO WHILE I <= LEN (NombredeLista)
* Una vez para cada nombre en el arreglo NombredeLista
  IF [SMITH] $ NombredeLista(I)
* El signo $ significa 'está contenido en la cadena'
     EXIT
  ENDIF
ENDDO
```

También puede usar la ramificación para saltar procesos subsecuentes dentro de un proceso cíclico:

```
I = 1
DO WHILE I <= LEN (NombredeLista)
* Una vez para cada nombre en el arreglo NombredeLista
   IF [SMITH] $ NombredeLista(I)
      SEEK NombredeLista(I)
         IF Nombre = [John]
         DO ImpNombr
         LOOP          && Regresa al principio
         ENDIF
   ENDIF
ENDDO
```

Operaciones con tablas

Una de las características más interesantes del lenguaje de bases de datos es que usted puede realizar operaciones entre tablas con un solo comando. Considere los dos comandos Count y Append From. Aquí está un pequeño programa para leer el texto de un archivo que contiene todas las oraciones de un libro, y le dice cuántas oraciones tienen una palabra en particular dentro de ellas:

```
CREATE TABLE TEXTO ( ORACION M(10) )
SET TALK ON
APPEND FROM LIBRO.TXT SDF
```

```
SET TALK OFF
DO WHILE .T.
   CLEAR
   Palabra = SPACE(20)
   @ 24, 0 SAY [Palabra que hay que buscar:] ;
      GET Palabra PICTURE "!" SIZE 1,20 FUNCTION "!"
   READ
   IF EMPTY ( Palabra )
      EXIT
   ENDIF
   COUNT TO CUANTAS FOR TRIM(Palabra) $ UPPER ( ORACION )
   @ 24, 45 SAY ALLTRIM(STR(CUANTAS)) + [ que contienen ] ;
                                      + TRIM(Palabra)
ENDDO
CLOSE ALL
DELETE FILE TEXTO.TXT
```

Hay muchas cosas como ésta que se pueden hacer con sólo unos cuantos comandos.

Busque un registro en particular

Los comandos Locate y Seek le permiten buscar un registro en particular. Locate funciona en cualquier campo dentro de una tabla, ya sea que esté indexada o no; Seek da por hecho que la clave que está buscando encaja con el índice actual. Suponga que tiene un archivo que contiene los campos Código, Descripción y Departamento. Si usted sólo tiene una etiqueta Código, puede usar:

```
SET ORDER TO CODIGO
SEEK THISFORM.PARTNUM.Value
```

o puede usar:

```
LOCATE FOR DEPARTAMENTO = [0014]
```

La pregunta es ¿qué tanto es más lento Locate? La respuesta podría sorprenderle. Si tiene un índice que contiene el campo Departamento, la respuesta es que la velocidad es idéntica.

Este resultado sorprendente se debe a la tecnología Rushmore de FoxPro. Si los datos que busca están en la etiqueta de índice, ¡FoxPro ni siquiera busca en la tabla! Así que el tiempo de búsqueda es el mismo.

Si lo que está buscando se encuentra dentro de la etiqueta de un índice, usted podría usar Locate sin cambiar etiquetas y buscar el registro que desee igual de

16

rápido. Por otro lado, si usted quiere buscar y procesar una serie de registros asados en una clave que no forma parte de ningún índice, podría ser más rápido crear sobre la marcha un índice temporal, especialmente si utiliza un índice condicional. Por ejemplo, para buscar todos los nombres que parezcan ser de origen irlandés en los campos de nombres que contengan nombres como Jack O'Malley, usted podría especificar:

```
INDEX ON SUBSTR (NOMBRE, SUBSTR(AT([O'],NOMBRE)) TO IRLANDES COMPACT ;
   FOR AT ([O'],NOMBRE) > 0
```

Esto le dará nombres alfabetizados que empiecen con O', y se podría especificar SEEK O'TOOLE (debido a que la expresión de índice no es la misma que el campo en que la expresión de índice estuvo basada).

Datos de salida

Hay un buen número de formas para poner datos en su pantalla o impresora:

- ?/??
- List/Display
- Report Form/Label Form
- Set Alternate
- Set TextMerge
- Comandos de bajo nivel I/O

Los operadores ? y ?? (con o sin un avance de línea previo, respectivamente) los usan generalmente los programadores para obtener una mirada rápida a algo en caso de que no deseen usar la ventana Depuración del menú Herramientas. List y Display son usados casi para los mismos propósitos. Report Form y Label Form crean datos de salida muy bien formateados, por lo que generalmente se les prefiere. Set Alternate To nombredearchivodetexto dirige todos los datos a un archivo sin detalles superfluos. TextMerge y los comandos relacionados \ y \\, con expresiones encerradas entre dos paréntesis angulares (como <<expr>>) se usan en el programa GenMenu para escribir código de programa generado y proporcionan un control muy preciso sobre los datos de salida.

Sin embargo, para el completo control de salida y entrada de datos, nada vence a los comandos de bajo nivel, de Visual FoxPro I/O:

- FCHSize
- FClose
- FCreate
- FEOF
- FFlush
- FGets
- FOpen
- FPuts
- FRead
- FSeek
- FWrite

Al usar estas funciones, usted puede leer o escribir cualquier cosa, incluso formatos de archivos que de otra manera no son soportados por Visual FoxPro. Yo escribo archivos que los programas COBOL pueden leer.

Otros comandos

Hay cientos de comandos de programación en Visual FoxPro, pero la tabla 5-1 es una colección de los comandos que realmente usamos en el código. Si piensa que debe conocer cada uno de los comandos de FoxPro para escribir programas, no esté tan seguro. La lista de la tabla 5-1 incluye sólo 64 comandos, y pueden ser todos los que usted usará en su código de programa. La mayoría de los comandos de Visual FoxPro están hechos para completar, de forma que usted puede hacer las mismas cosas en programas vía el menú de FoxPro, pero es poco probable que utilice más del 20 o 40 por ciento de ellos.

Tabla 5-1 **Colección de los comandos más comunes dentro de Visual FoxPro**

Comando	Acción
@ *fila,columna* Get *variable de memoria/campo*	Entrada de datos en pantalla
@ *fila,columna* Say *variable de memoria/campo*	Salida de datos en pantalla
\ \ \ \	Salida de texto combinado

18

Continuación Tabla 5-1

Comando	Acción
? *expr*	Imprimir
Activate PopUP	Despliega un menú en la pantalla
Append Blank	Añade un registro a una tabla
Browse	Llama al comando Browse de FoxPro
Cancel	Termina la ejecución de un programa
CD o ChDir	Cambia el directorio predeterminado
Clear	Limpia la ventana activa
Clear Events	Termina Read Events
Compile	Compila código fuente o formularios
Continue	Va a la siguiente coincidencia localizada
Close	Cierra tablas y bases de datos
Define PopUp	Empieza una definición de menú desplegable
Define Window	Empieza una definición de ventana
Delete Next 1	Borra el registro actual
Do Case...EndCase	Ejecución condicional
Do Form	Ejecuta un formulario
Do While...EndDo	Procesos cíclicos
Export	Envía datos de una tabla a otro formato
Exit	Sale de la estructura cíclica actual
For...EndFor	Para controlar procesos cíclicos
Gather MemVar	Mueve las variables de memoria hacia campos
GetEnv	Lee un valor de una variable de configuración de DOS
GoTo	Va al Inicio, al Final o a un número de registro guardado
Hide Menu	Oculta el menú
If...EndIf	Ramificación (condición)
Insert (SQL)	Agrega un registro y asigna valores a los campos
Keyboard	Almacena datos de tipo carácter dentro del búfer de teclado
List	Lista el contenido de una tabla
Locate	Busca una coincidencia en una tabla que no está indexada

Tabla 5-1 **Continuación**

Comando	Acción
MD o MkDir	Crea un directorio
Modify Command	Edita un archivo .PRG o de texto
Modify Memo	Edita un campo memo
On Error	Configura una trampa para errores
On Key Label	Vuelve a definir una tecla
On Selection PopUp	Controla acciones de menú desplegable
On ShutDown	Proceso de finalización
Pack	Borra registros eliminados de tablas
Parameters	Define parámetros
Pop Key/Push Key	Usado para volver a definir teclas
Pop Menu/Push Menu	Utilizado para volver a definir menús
Quit	Sale de FoxPro
Read Events	Inicia el proceso cíclico eventual
Recall Next 1	Restablece un registro eliminado
Release PopUp	Libera un menú desplegable de la memoria
Rename	Vuelve a nombrar un archivo
Replace	Coloca datos dentro de un campo en una tabla
Resume	Continúa la ejecución después de Suspend
Retry	Intenta un comando por segunda vez
Return	Abandona un programa
Run	Ejecuta un programa externo
Scan...EndScan	Procesa los registros de una tabla
Scatter MemVar	Mueve datos de campos hacia variables de memoria
Seek	Busca en una tabla indexada
Select	Cambia áreas de trabajo/alias
Set (todos)	Comandos de entornos misceláneos
Suspend	Suspende una ejecución
Unlock	Desbloquea el registro en uso
Use	Abre una tabla
Wait Window	Coloca un mensaje en la pantalla
With...EndWith	Define propiedades de clase
Zap	Borra los registros de una tabla

unciones

FoxPro contiene una gran cantidad de funciones. Y, a diferencia de los comandos de FoxPro, muchas funciones realmente son usadas en programación todos los días. Están agrupadas confusamente dentro de seis categorías:

Tipo de datos Carácter, fecha, tiempo, conversión de datos y funciones numéricas.

Bases de datos Manipulación de bases de datos, manipulación de campos, índices, manipulación de registros, relaciones y manipulación de tablas.

Entorno Administración de entornos y archivos.

Multiusuarios Bloqueo de archivos y registros.

I/O Formateo de datos, entrada de datos por teclado/ratón, menús, impresión, informes y etiquetas, y ventanas.

Programación Arreglos, manejo de errores y depuración, controladores de eventos, archivos I/O de bajo nivel, manipulación de variables de memoria, programación orientada a objetos, ejecución de programas y programación estructurada.

Si usted hace un clic en Ayuda, Contenido, y selecciona Referencia del lenguaje, encontrará que la última opción, Comandos y funciones por categoría, lista cada uno de los comandos y funciones de FoxPro, y para qué son usados. La mayoría de los programadores en FoxPro han usado, en un momento u otro, casi todos los comandos y funciones. En lugar de tratar de describirlos aquí se presentará una serie de fragmentos de códigos que demuestran cómo desarrollar varias tareas típicas dentro de Visual FoxPro.

Ejemplos de código

En las siguientes secciones encontrará un número de ejemplos de programas. No están en ningún orden en particular, pero demuestran el tipo de cosas que puede hacer con código Visual FoxPro.

Un programa Principal

El código principal para iniciar muchas aplicaciones FoxPro es el mismo:

```
Princip.PRG
OPEN DATABASE XYZ
DO MENU.MPR
READ EVENTS
CLOSE DATABASES
SET SYSMENU TO DEFAULT
```

Este programa da por hecho que sus archivos están en un contenedor de bases de datos llamado XYZ y usted ha generado un menú llamado MENU. Aclare la pantalla para comenzar con el área de trabajo limpia.

Abrir archivos

El siguiente programa, ABREARCH.PRG, da por hecho que usted tiene una lista maestra de tablas en una tabla llamada ArchiMae, en la que cada registro consiste en un campo llamado Tabla que contiene el nombre de una tabla a abrirse:

```
PROCEDURE AbreArch
LOCAL ArchivoFaltante
ArchivoFaltante = .F.  && No es esencial debido a que .F. es el valor
                       && predeterminado
* Asegúrese de que no hay tablas abiertas
CLOSE TABLES ALL
* Cree la variable de memoria DirDatos si es que no existe aún
IF TYPE ( "DirDatos" ) = "U"
   DirDatos = [DATOS]
ENDIF

* Cree el subdirectorio si aún no existe
ON ERROR a=1
MD &DirDatos
ON ERROR
USE ARCHIMAE IN 99
SELECT ARCHIMAE

* Abre cada tabla dentro de ARCHIMAE en su propia área de trabajo
SCAN
   Tabla = ( DirDatos + [/] + TRIM(ARCHIMAE.Tabla) )
   IF NOT FILE ( Tabla )
      WAIT WINDOW [ Table ] + Tabla + [ no se encontró ] TIME 1
        ArchivoFaltante = .T.
      ELSE
```

```
      USE ( Tabla ) IN 0
   ENDIF
ENDSCAN

* Cierra el archivo de la lista de tablas
USE IN ARCHIMAE

IF ArchivoFaltante
  WAIT WINDOW ;
      "  Uno o más archivos de datos no se encuentran " + CHR(13)          ;
   +  " Ejecute Instalar para empezar con archivos vacíos " + CHR(13)      ;
   +  " De otra forma, restablezca sus archivos de datos desde un respaldo " ;
   +  " (Presione Enter)   "
ENDIF
```

Si una variable de memoria no ha sido definida, escriba ("variabledememoria") y se obtendrá "U" (que indica que no está definida). Así que si una variable de memoria DirDatos existe, es usada como el directorio de datos del sistema. Si no existe, el valor "DATOS" es asignado al nombre de directorio.

A continuación, trate de crear un subdirectorio que tenga su nombre grabado en DirDatos. El signo & en frente de DirDatos le dice a FoxPro que expanda el contenido de la variable de memoria y luego ejecute la línea, por lo que la línea es xpandida para leer MD DATOS.

Normalmente si usted intenta crear un subdirectorio que ya existe, Visual FoxPro se rendirá y producirá un mensaje de error. Y no hay forma directa de probar la existencia de un nombre de subdirectorio como sí la hay para un nombre de archivo (la función FILE( )). Así que simplemente cambie a desactivado el detector de errores usando ON ERROR a=1. Si el directorio ya existe, &DirDatos invocará la rutina de error. Una vez que asignó un 1 dentro de a, simplemente continúa. Entonces usted puede restablecer la detección de errores y continuar.

En la aplicación de ejemplo si un archivo falta, quiere decir que la rutina inicial de poner en marcha no fue ejecutada o que un archivo ha sido borrado sin advertencia. Así que usted querrá prevenir a los usuarios de este hecho y darles la oportunidad de restablecer el archivo desde una cinta. Por otro lado, podría haber una razón por la cual el archivo faltara y ellos podrían continuar trabajando sin él. De esta forma, cuando Scan dé vueltas tratando de abrir cada archivo, si un archivo falta, un mensaje de un segundo aparecerá en la esquina superior izquierda de la pantalla, y se establecerá a .T. la variable ArchivoFaltante. Entonces, al final, se les dice a los usuarios que al menos hubo un archivo faltante. El CHR(13) coloca la parte restante del texto del mensaje en la pantalla Wait Window en la línea siguiente.

Use In ArchiMae cierra el archivo ARCHIMAE. Note que usted puede abrir y cerrar archivos dentro de áreas de trabajo sin seleccionar el área de trabajo.

Un programa de alarma

Aquí hay una función que toca una campana que suena como un teléfono (CAMPANA.PRG):

```
FUNCTION Campana
FOR I = 1 TO 3
   SET BELL TO 800,1
   ?? CHR(7)
   SET BELL TO 1100,1
   ?? CHR(7)
ENDFOR
```

Para usarla, incluya =Campana( ) en su código, o escríbala como un método Campana en su formulario (menos el encabezado de la función) y llámela con THISFORM.Campana.

Cree una tabla libre

Refiérase al siguiente programa, HACERTBL.PRG:

```
PROCEDURE HacerTbl
CREATE TABLE TRANSACT (  ;
   TipoTran  C( 1 ),     ;
   Transano  C( 12),     ;
   Codigo    C( 13),     ;
   Descripc  C( 25),     ;
   Orden     N(  3),     ;
   Recibida  N(  3)      )
```

Cuando usted cree una tabla, ésta será creada en el área de trabajo siguiente disponible.

Cree, actualice y quite una barra de termómetro

Refiérase al siguiente programa, INITERMO.PRG:

24

```
*PROCEDURE IniTermo
  *Propósito....: Llamada de inicialización de la rutina Termómetro
PARAMETERS vTitulo
SET TALK OFF

* Si no le proporcionaron un título, haga que aparezca uno
IF PARAMETERS() = 0
   TitlVen = [  Avance ]
  ELSE
   TitlVen = vTitulo
ENDIF

* If MaxTermo no tiene un valor, utilice RECCOUNT() de la tabla
* en uso. Si no existe una DBF, no tenemos forma de adivinar.
IF TYPE ( [MaxTermo] ) = [U]
   IF NOT EMPTY ( DBF() )
      PUBLIC MaxTermo
      MaxTermo = RECCOUNT()


     ELSE
      WAIT WINDOW ;
         [ No ha sido proporcionado un contador para la rutina del termómetro ] ;
      + [ (presione Enter) ]
      RETURN
   ENDIF
ENDIF

* Inicializar el contador
PUBLIC ContTermo
ContTermo = 0
DEFINE WINDOW TERMO   ;
    FROM 18, 8 TO 21, 72   ;
    DOUBLE SHADOW          ;
    TITLE (TitlVen)

* Asegúrese que el despliegue es direccionado hacia la pantalla
GuardarVentana = WONTOP()
GuardarDispositivo = SET ('DEVICE')
GuardarImprimir = SET ('PRINT')
SET DEVICE TO SCREEN
SET PRINT OFF

ACTIVATE WINDOW TERMO
@ 0, 1 SAY '...10...20...30...40...50...60...70...80...90...100'

SET DEVICE TO &GuardarDispositivo
SET PRINT     &GuardarVentana
```

Lo siguiente es MOSTTERMO.PRG:

```
* Propósito.....: Desplegar el termómetro a avance de monitor
* Requiere......: Llamada previa a INITERMO

PARAMETERS contador
* Puede ser llamado con o sin un valor para el contador.
* Si no es pasado ninguno, utilice su propio contador, "conttermo".
IF PARAMETERS() < 1
   ContTermo = ContTermo + 1
  ELSE
   ContTermo = contador
ENDIF
GuardarVentana = WONTOP()
GuardarDispositivo = SET ('DEVICE')
GuardarImprimir = SET ('PRINT')
SET DEVICE TO SCREEN
SET PRINT OFF
ACTIVATE WINDOW TERMO
@ 1, 1 SAY  ;
   REPLICATE (CHR(245), (ContTermo/MaxTermo)*60)
SET DEVICE TO &GuardarDispositivo
SET PRINT     &GuardarImprimir
```

Éste es QUITERMO.PRG:

```
* Propósito.....: Borrar la ventana termómetro
*
IF WEXIST('TERMO')
   RELEASE WINDOW TERMO
ENDIF
IF TYPE ( 'ContTermo' ) <> [U]
   RELEASE ContTermo
ENDIF
IF TYPE ( 'MaxTermo' ) <> [U]
   RELEASE MaxTermo
ENDIF
```

Aquí hay un ejemplo de uso del programa:

```
* California finalmente se separó y formó un país con tres estados
USE CLIENTE
COUNT TO MaxTermo FOR ESTADO = [CA]
=IniTermo()
SCAN FOR ESTADO = [CA]
     =MostTermo()
     REPLACE NEXT 1 Estado WITH ;
     IIF ( BETWEEN ( CP [91001],  [93499] ) , [C1], ;
     IIF ( BETWEEN ( CP [93500],  [95999] ) , [C2], ;
```

```
      IIF ( BETWEEN ( CP [96000],  [98999] ) , [C3], [ER] )))
   ENDSCAN
   =QuitTermo ( )
```

Note el uso anidado de instrucciones IIF. En la mayoría de los casos, usted utilizará instrucciones Do Case, y ésta es una de las pocas ocasiones en que no se les usará.

Utilice Browse como una lista para escoger

Refiérase al siguiente programa, ESCOGERLISTA.PRG:

```
PROCEDURE  EscogerLista
GuardaAlias = ALIAS()
SELECT PORDERS
SET ORDER TO PORDERS
GO TOP
DEFINE    WINDOW BROWSER FROM 2, 7 TO 23, 73 DOUBLE SHADOW ;
          TITLE [ Presionar ENTER para seleccionar, Esc para cancelar ]
ACTIVATE WINDOWS BROWSER
ON KEY LABEL ENTER KEYBOARD CHR(23)
ON KEY LABEL LEFTMOUSE KEYBOARD CHR(23)
   OWSE WINDOWS EXAMINADOR    ;
   MENU NOEDIT NOAPPEND       ;
FLELDS                        ;
    TRASNO     :H=[P/O]       , ;
    Vendedor                  , ;
    Orden                     , ;
    ListaPrecio               , ;
    PrecDesc                  , ;
    PrecioNeto                , ;
ON KEY LABEL ENTER
ON KEY LABEL LEFTMOUSE
RELEASE WINDOW BROWSER
IF NOT EMPTY ( GuardarAlias )
   SELECT ALIAS ( GuardarAlias )
ENDIF
```

Primero guarde el alias del área de trabajo en uso, luego seleccione el archivo que será examinado. Defina una ventana, aunque debido a que Browse la activará, usted no tiene que hacerlo.

Luego asigne el carácter CTRL-W a la tecla ENTER con el comando Keyboard. Como CTRL-W cierra un Browse y usted desea que los usuarios apunten a lo que quieren y luego oprima ENTER para regresar a la pantalla previa con el puntero de registros en el registro seleccionado, ésta es la manera fácil de hacerlo. La misma ignación es hecha para el botón izquierdo (principal) del ratón.

Browse permite muchas opciones para formateo de campos (:## le permite establecer la longitud del campo; :H[encabezado] le permite cambiar el encabezado de columna, y :V=[###.##] proporciona el formateo). Hay otras funciones de ese tipo, pero en listas para escoger, esto es probablemente todo lo que necesitará.

Las opciones NoEdit y NoAppend aseguran que sus usuarios no hagan cambios cuando se supone que todo lo que deben hacer es seleccionar algo.

El sorprendente Joe Gotthelf escribió un agregado de búsquedas incrementales para Browse llamado JKEY.PBL. Lo puede encontrar en nuestro foro CompuServe.

Agregue archivos

Muchos sistemas suben archivos de tiendas subsidiarias todas las noches, luego los consolidan. Este ejemplo atraviesa una serie de subdirectorios que han adquirido copias de archivos subidos en la noche. Los archivos son colocados en directorios de almacenamiento de forma que usted puede decir de un vistazo cuáles no han sido subidos aún. Una vez que todos están presentes, querrá copiar sus contenidos a una tabla de historial de ventas con exactamente la misma estructura de tabla, y borrar los archivos subidos. El sistema sabe cuántas tiendas están en (NumTiendas). Refiérase al programa HISTVENT.PRG:

```
PROCEDURE HistVent
* Propósito.......: Subir archivos de historiales de ventas de tiendas.
FOR I = 0 TO NumTiendas
    IDTienda = TRANSFORM ( I , "@L ##" )
    Dir      = [SUBIDOS] + [\] + IDTienda + [\]
    NombredeArchivo = Dir + [HISTVENT.DBF]
    IF NOT FILE ( NombredeArchivo )  && Nada para subir
      LOOP
    ENDIF
    WAIT WINDOW ;
     [ Añadiendo ] + NombredeArchivo + [ a historial de ventas ] NOWAIT
    SELECT HISTVENT
    APPEND FROM ( NombredeArchivo )
    DELETE FILE ( NombredeArchivo )
    WAIT CLEAR
ENDFOR
```

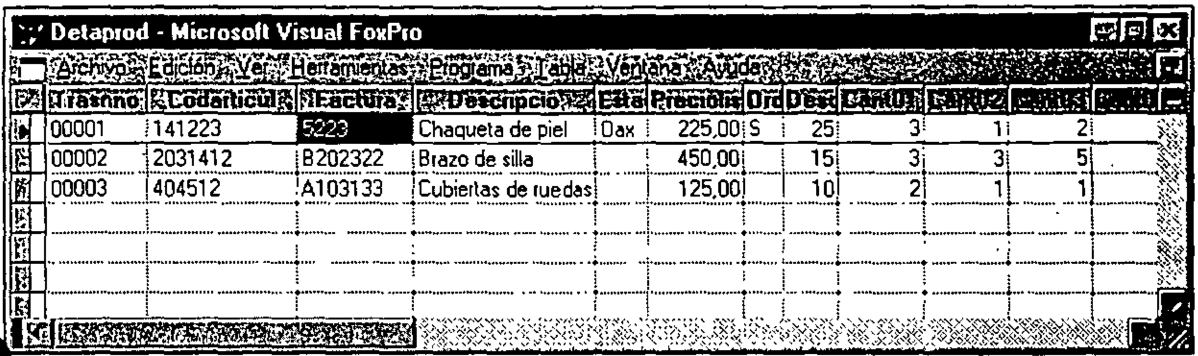
Este ejemplo busca archivos llamados SUBIR\01\HISTVENT.DBF, SUBIR\02\HISTVENT.DBF, etc. y los añade a una tabla abierta, también llamada HistVent. Note como el comando Transform crea una cadena de dos dígitos

formada con ceros (01,02...) para el ciclo For... contador I. Después de añadirlos. ...rra cada uno usando el comando Delete File. Los paréntesis funcionan como el signo &, ocasionando que el comando elimine el archivo nombrado en la variable de memoria.

Cree subjuegos de un archivo

Algunos sistemas de compras por pedido utilizan Browse para ver en pantalla las cantidades existentes de un artículo dado para cada una de varias tiendas, como las mostradas en la figura 5-1.

Figura 5-1



Ventana para visualización múltiple de órdenes de compra.

De esta forma, un agente de compras puede planear ordenar diez para la tienda 1, cinco para la tienda 2 y así sucesivamente. Sin embargo, en algún momento usted querrá convertir esta tabla a una para cada tienda, ver un encabezado repentinamente, y enviar órdenes de compra para cada tienda a su proveedor. Aquí hay un programa para crear tablas separadas, con cantidades para cada tienda en la columna Cantidades:

```
PROCEDURE SalidadeCorte
* Cree TIENDA01.DBF, TIENDA02.DBF, ETC.
* Archivo fuente: DETAPROD.DBF
* Nombre de campo  Tipo         Longitud
*  1 TRASNNO       Carácter     12
*  2 CODARTICULO   Carácter     13
*  3 FACTURA       Carácter     15
*  4 DESCRIPCION   Carácter     32
*  5 ESTADO        Carácter     3
*  6 PRECIOLISTA   Numérico     8,2
*  7 ORDEN         Carácter     1
*  8 DESCUENTO     Numérico     2
*  9 CANT01        Numérico     3           Existencia Tienda 1
*  10 CANT02       Numérico     3           Existencia Tienda 2
```


29

```
*  11 CANT03        Numérico      3           Existencia Tienda 3
*  12 CANT04        Numérico      3           Existencia Tienda 4
*  13 CANT05        Numérico      3           Existencia Tienda 5

FOR I = 1 TO NumTiendas       && 5, en este caso.
   Tienda = TRAN ( I, "@L ##" )
   WAIT WINDOW [ Extrayendo datos de tienda ] + Tienda + [DATA];
   NOWAIT
   SELECT                      ;
     Transno                  ,;
     m.Tienda AS Tienda       ,;
     Codarticulo               ,;
     Cant&Tienda AS Cantidad  ,;
     Descripcion              ,;
     Preciolista              ,;
     Descuento                ,;
     Orden                    ,;
   FROM DETAPROD               ;
   HAVING Cantidad > 0         ;
   INTO TABLE TIENDA&TIENDA
ENDFOR
WAIT CLEAR
```

Este programa usa el comando de SELECT SQL para extraer los datos que necesita, un paso por tienda. Los registros cuya columna Tienda## contienen un cero no son incluidos. La expansión de la macro Tienda&Tienda crea los nombres de tabla TIENDA01.DBF, TIENDA02.DBF, etcétera.

Calcule un dígito de verificación

En operaciones de entrada de datos, usted intenta asegurarse de que sólo los datos buenos lleguen a sus tablas. Un truco que ha estado por ahí desde hace algún tiempo es el llamado dígito de verificación. Usted diseña sus números de código de tal forma que los primeros, digamos, nueve números calculan el décimo. Luego, cuando un código es introducido, usted toma los primeros nueve dígitos, calcula el décimo, y posteriormente lo compara con el décimo número actual. Si el dígito calculado no coincide con el actual, algo no está funcionando. El siguiente programa, VerifDgt, se presenta debido a que muestra muchos trucos de manipulación de cadenas:

```
FUNCTION verifdgt
* Propósito      : Calcular el último de diez dígitos de un código
PARAMETERS Codigo
PRIVATE N, K, verifdigito, digito
K = 0
VerifDgt = 0
FOR K = 1 TO 9
```

```
   digito      = SUBSTR ( Codigo , K , 1 )
   valdigito   = IIF ( digito = 'X' , 10 , VAL ( digit )
   verifdigito = verifdigito + valdigito * ( 11 - K )
ENDDO
j = 0
DO WHILE MOD ( verifdigito + j , 11 ) <> 0
   j = j + 1
ENDDO
RETURN Codigo + IIF ( j = 10 , [X] , STR ( j , 1 ) )
```

Establezca parámetros de comunicación

Si tiene un formulario con un campo donde los usuarios establecen parámetros de comunicación, usted no querrá que ellos los pongan en el lugar que se les antoje; los parámetros de comunicación están dentro de los tipos de datos de entrada más sensibles, y los usuarios, no pueden adivinar. ¿Cómo presentar a ellos las opciones?

La solución es lo que antes se llamaba una *cláusula válida*. Cuando un campo pierde su destino, usted puede revisar si lo que hay en el campo (si es que ya está ahí o fue cambiado por el usuario) es correcto o no. A mí no me gusta forzar a los suarios a ir a través de una lista desplegable cuando ellos ya han introducido un valor correcto. Aquí hay un enfoque:

```
* FUNCTION LineaPar
DIMENSION LineaPar(17)
LineaPar( 1) = [   300, E, 7, 1]
LineaPar( 2) = [  1200, E, 7, 1]
LineaPar( 3) = [  2400, E, 7, 1]
LineaPar( 4) = [  9600, E, 7, 1]
LineaPar( 5) = [ 14400, E, 7, 1]
LineaPar( 6) = [\]
LineaPar( 7) = [   300, N, 8, 1]
LineaPar( 8) = [  1200, N, 8, 1]
LineaPar( 9) = [  2400, N, 8, 1]
LineaPar(10) = [  9600, N, 7, 1]
LineaPar(11) = [ 14400, N, 7, 1]
LineaPar(12) = [\-]
LineaPar(13) = [   300, O, 7, 1]
LineaPar(14) = [  1200, O, 7, 1]
LineaPar(15) = [  2400, O, 7, 1]
LineaPar(16) = [  9600, O, 7, 1]
LineaPar(17) = [ 14400, O, 7, 1]
FOR i = 1 TO 17
   IF LineaPar ( i ) = THISFORM.LineaPar.Value
      RETURN
   ENDIF
```

```
ENDFOR
DEFINE POPUP LineaPar FROM 2, 24 SHADOW MARGIN
FOR I = 1 TO 17
   Txt = LineaPar(I)
   DEFINE BAR I OF LineaPar PROMPT Txt
ENDFOR
ON SELECTION POPUP LineaPar DEACTIVATE POPUP LineaPar
ACTIVATE POPUP LineaPar
mPrompt = PROMPT()
RELEASE POPUP LineaPar
THISFORM.LineaPar.Value=IFF(EMPTY(mPrompt),SPACE(11),mPrompt)
```

Los 17 valores válidos pudieron haber sido cargados en PRINCIP.PRG; se puso el código aquí para mostrarle todo el panorama. Primero se dimensiona un arreglo suficientemente grande para tener todos los valores posibles y luego los valores son guardados. Posteriormente usted confirma que cada entrada en el campo sea validada con cada uno de los valores en el arreglo; si alguno coincide, usted puede salir del código de evento. Este código va en la ventana de código del evento Valid del objeto LineaPar en el formulario. Note cómo se comparan los valores en el arreglo con la propiedad valor del formulario de entrada, y no directamente con el valor del campo en la tabla. Aquí hay una buena razón para ello.

Esto se está saliendo un poco del tema, pero aquí hay un pequeño resumen. El campo LineaPar en el formulario apunta al campo LineaPar en una tabla, pero cuando usted escribe algo dentro del campo en el formulario no se han tocado los datos en el campo dentro de una tabla. Esto se debe al búfer en filas. Usted escribe en la tabla sólo cuando mueve el puntero de registros lejos del registro actual. Así que asigne el nuevo valor de la lista desplegable ThisForm.LineaPar.Value y deje que Visual FoxPro piense cuándo debe poner el nuevo valor dentro de una tabla, si es que hay uno. Recuerde que usted puede llamar a TableRevert( ) (cancelar) o TableUpdate( ) (escribir los datos en la tabla). Esto es por lo que usted asigna un valor nuevo a la propiedad valor del nuevo objeto. Por cierto, usted no tiene que nombrar el objeto LineaPar; sólo dígale que Cliente.LineaPar es su fuente de datos. Yo uso los mismos nombres, de esta forma no me perderé más de lo que normalmente lo estoy. También puede usar este valor en cláusulas válidas.

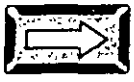
Busque registros coincidentes

La búsqueda es uno de los requerimientos más comunes en los sistemas de bases de datos. Un enfoque consiste en asumir que sus usuarios tienen alguna idea de lo que quieren. Por ejemplo, si alguien escribe SMITK, este alguien probablemente desea Smith. (Para los lectores que no hablan inglés, Smith es un apellido

extremadamente común.) Usted necesita mostrarles todo lo que coincida, si es que hay más de una coincidencia, o todo lo que casi coincida, si nada coincide exactamente.

¿Qué significa que algo "casi coincida"? Descartar caracteres rezagados hasta que los caracteres restantes coincidan parcialmente (recuerde que Visual FoxPro sólo compara hasta el último carácter al lado derecho de la expresión) luego les muestra todo lo que coincida parcialmente con la clave. Refiérase a BUSCAR2.PRG:

```
PROCEDURE Buscar
PARAMETERS Clave, NombArch, NombEti, CampoEti,
   * Todos excepto el primer parámetro son cadenas de caracteres, por
   * ejemplo
   * =SEARCH(CLIENTE.ESTADO, "ESTADOS", "CODIGODEESTADO",
   * "ESTADOS.CODIGO")
LOCAL ClaveLocal
ClaveLocal = TRIM(Clave)
GuardarAlias = ALIAS()
SELECT ( NombArch )
SET ORDER TO ( NombEti )
SEEK ClaveLocal

DO WHILE NOT FOUND() AND LEN(ClaveLocal) > 0
   LargoCve = LEN ( ClaveLocal )
   ClaveLocal = LEFT ( ClaveLocal, LargoCve - 1 )
   SEEK ClaveLocal
   IF FOUND()
      EXIT
   ENDIF
ENDDO
ExprCve = IIF ( FOUND(), [KEY ClaveLocal, ClaveLocal], [] )
ON KEY LABEL ENTER KEYBOARD CHR(23)
ON KEY LABEL LEFTMOUSE KEYBOARD CHR(23)
BROWSE FIELDS &CampoEti ;
   TITLE ( Presione ENTER para seleccionar, ESC para cancelar ) ;
   &ExprCve
ON KEY LABEL ENTER
ON KEY LABEL LEFTMOUSE

IF LASTKEY() <> 27
   Key = &File..&CampoEti
ENDIF
```



Programe clases con código

También es posible programar clases e iniciarlas a través de código aislado. Yo prefiero utilizar el enfoque visual cuando es posible, pero por favor, note que puede hacer cualquier cosa en Visual FoxPro usando código aislado.

El código siguiente está aquí gracias a la generosidad de Steven Black, una superestrella canadiense dentro del mundo FoxPro, quien ha proporcionado muchos de los atajos que se usan en FoxPro hoy en día.

Note cómo se crea el formulario Miformaprueba como una ejemplificación de clases de Formulario FoxPro. Los botones objeto, de igual forma, están basados en Mibotoncomando, el cual está basado a su vez en CommandButton de FoxPro.

```
*============================================
*- Miforma
*============================================
DEFINE CLASS Miforma AS Form
   *- Propiedades particulares
   versión     = 1.02

   *- Modificación a propiedades nativas
   Comment     = "Mi forma genérica"
   ScaleMode   = 3
   Caption     = "miforma"
   ShowTips    = .T.
   Top         = 0
   Left        = 0
   Height      = 330
   Width       = 549
   BackColor   = RGB(192,192,192)
   BordeStyle  = 3
   Name        = "miforma"

ENDDEFINE

*============================================
*- Miformaprueba
*============================================
DEFINE CLASS miformaprueba AS miforma
   *- Propiedades particulares
   versión     = 1.02

   *- Modificaciones a propiedades nativas
   Top         = 0
   Left        = 0
   Height      = 330
   Width       = 573
```

```
   Caption      = "Escritorio de pruebas de Steve"
   Comment      = "Una forma de pruebas"

ADD OBJECT cmdQuit AS mibotonsalida ;
WITH ;
     Top        = 6, ;
     Left       = 1

ADD OBJECT cmdDebug AS mibotondepura ;
WITH ;
     Top        = 6, ;
     Left       = 88

ADD OBJECT cmdTrace AS mibotonexamina ;
WITH ;
     Top        = 6, ;
     Left       = 175

ADD OBJECT cmdView AS mibotonver ;
WITH ;
     Top        = 6, ;
     Left       = 262

ADD OBJECT cmdSuspend AS mibotonsuspender ;
WITH ;
     Top        = 6, ;
     Left       = 349

ADD OBJECT cmdVCRbuttons AS misbotonesvcr ;
WITH ;
     Top        = 6, ;
     Left       = 436

ENDDEFINE

*================================================
*- Mibotonsalida
*================================================
DEFINE CLASS mibotonsalida AS mibotoncomando
   *- Propiedades particulares
   versión        = 1.00
   *- Modificación a propiedades nativas
   Caption        = "Salida"
   StatusBarText =  ; ...
      "Presione para liberar este formulario"
   ToolTipText    = "Libere este formulario"

   PROCEDURE Click
   RELEASE THISFORM
   ENDPROC

ENDDEFINE
```

```
*=============================================
*- mibotondepuracion
*=============================================
DEFINE CLASS mibotondepuracion AS mibotoncomando
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   Caption       = "Depuración"
   StatusBarText =  ;
      "Presione para llamar a la ventana de depuración"
   ToolTipText   = "Invocar depuración"
   Name = "mibotondepuracion"

   PROCEDURE Click
   ACTIVATE WINDOW DEBUG
   ENDPROC

ENDDEFINE

*=============================================
*- mibotonexaminar
*=============================================
DEFINE CLASS mibotonexaminar AS mibotoncomando
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   Caption       = "seguimiento"
   StatusBarText =  ;
      "Presione para activar la ventana seguimiento"
   ToolTipText   = "Activar ventana seguimiento"
   Name = "mibotonexaminar"

   PROCEDURE Click
   ACTIVATE WINDOW TRACE
   ENDPROC

ENDDEFINE

*=============================================
*- mibotonver
*=============================================
DEFINE CLASS mibotonver AS mibotoncomando
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   Caption       = "Ver"
   StatusBarText =  ;
      "Presione para activar la ventana Ver"
   ToolTipText   = "Invoca ventana Ver"
```

```
   Name = "mibotonver"

   PROCEDURE Click
   ACTIVATE WINDOW VIEW
   ENDPROC

ENDDEFINE

*================================================
*- mibotonsuspender
*================================================
DEFINE CLASS mibotonsuspender AS mibotoncomando
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   Caption       = "Suspender"
   StatusBarText =  ;
      "Presione para suspender"
   ToolTipText   = "Suspender ejecución"
   Name = "mibotonsuspender"

   PROCEDURE Click
   SUSPEND
   ENDPROC

ENDDEFINE

 ================================================
 - mibotoncomando
*================================================
DEFINE CLASS mibotoncomando AS commandbutton
   *- Propiedades particulares
   versión       = 1.01

   *- Modificación a propiedades nativas
   Caption       = "Comando1"
   Height        = 32
   Width         = 85
   StatusBarText = "Presionar"
   ToolTipText   = "Presione"

ENDDEFINE

*================================================
*- misbotonesvcr
*================================================
DEFINE CLASS misbotonesvcr AS micontenedor
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   Width         = 135
   Heigth        = 32
   BackStyle     = 0
```

```
    ADD OBJECT comandoiniciar AS mibotoninicio ;
    WITH ;
        Width     = 32, ;
        Top       = 0,  ;
        Left      = 0   ;
        Caption   = "<<"

    ADD OBJECT comandoprevio AS mibotonprevio ;
    WITH ;
        Width     = 32, ;
        Top       = 0,  ;
        Left      = 34, ;
        Caption   = "<"

    ADD OBJECT comandosiguiente AS mibotonsiguiente ;
    WITH ;
        Width     = 32, ;
        Top       = 0,  ;
        Left      = 68, ;
        Caption   = ">"

    ADD OBJECT comandofinal AS mibotonfinal ;
    WITH ;
        Width     = 32,  ;
        Top       = 0,   ;
        Left      = 102, ;
        Caption   = ">>"

ENDDEFINE

*=============================================
*- micontenedor
*=============================================
DEFINE CLASS micontenedor AS container
   *- Propiedades particulares
   versión       = 1.00

   *- Modificación a propiedades nativas
   BackColor     = RGB( 192,192,192)
   BorderWidth   = 0
   versión       = 1.02

ENDDEFINE
*=============================================
*- mibotoninicio
*=============================================
DEFINE CLASS mibotoninicio AS mibotoncomando
   Caption       = "Inicio"
   ToolTipText   = "Inicio del archivo"
   StatusBarText = ;
      "Presione para ir al inicio del archivo"
   Name  = "mibotoninicio"

    PROCEDURE Click
```

```
   IF ! EMPTY( ALIAS() )
      LOCATE
      THISFORM.Refresh
      WAIT WINDOW "Principio de archivo" NOWAIT
    ELSE
      WAIT WIND "No hay tabla en el área de trabajo"
   ENDIF
   ENDPROC
ENDDEFINE

*=============================================
*- Mibotonprevio
*=============================================
DEFINE CLASS mibotonprevio AS mibotoncomando
   *- modificación apropiedades nativas
   Caption      = "Previo"
   ToolTipText  = "Subir un registro"
   StatusBarText = ;
      "Presione para subir un registro"
   Name = "mibotonprevio"

   PROCEDURE Click
   IF ! EMPTY( ALIAS() )
      IF ! BOF()
           SKIP -1
        ELSE
           LOCATE
           WAIT WINDOW "A principio de archivo" NOWAIT
      ENDIF
      THISFORM.Refresh
   ELSE
      WAIT WIND "No hay tabla en el área de trabajo"
   ENDIF
   ENDPROC

ENDDEFINE

*=============================================
*- mibotonsiguiente
*=============================================
DEFINE CLASS mibotonsiguiente AS mibotoncomando
   *- Modificación a propiedades nativas
   Caption       = "Siguiente"
   ToolTipText   = "Ir al siguiente registro"
   StausBar Text = ;
   "Presione para ir al registro siguiente"
   Name = "mibotonsiguiente"

   PROCEDURE Click
   IF ! EMPTY( ALIAS() )
      IF ! BOF()
           SKIP
        ELSE
```

```
            GO BOTTOM
            WAIT WINDOW "A final del archivo" NOWAIT
         ENDIF
         THISFORM.Refresh
      ELSE
         WAIT WIND "No hay tabla en el área de trabajo"
      ENDIF
      ENDPROC

ENDDEFINE

*============================================
*- mibotonfinal
*============================================
DEFINE CLASS mibotonfinal AS mibotoncomando
   *- Modificación a propiedades nativas
   Caption       = "Final"
   ToolTipText   = "Ir al final"
   Status BarText= ;
   "Presione para ir al final del archivo"
   Name = "mibotonfinal"

   PROCEDURE Click
   IF ! EMPTY( ALIAS() )
      GO BOTTOM
      WAIT WINDOW "A final de archivo" NOWAIT
      THISFORM.Refresh
   ELSE
      WAIT WIND "No hay tabla en el área de trabajo"
   ENDIF
   ENDPROC

ENDDEFINE
```

Note que, hablando en términos de programación, en cuanto a definir clases, usted las coloca debajo del programa PRINCIP, como lo hace con las funciones.

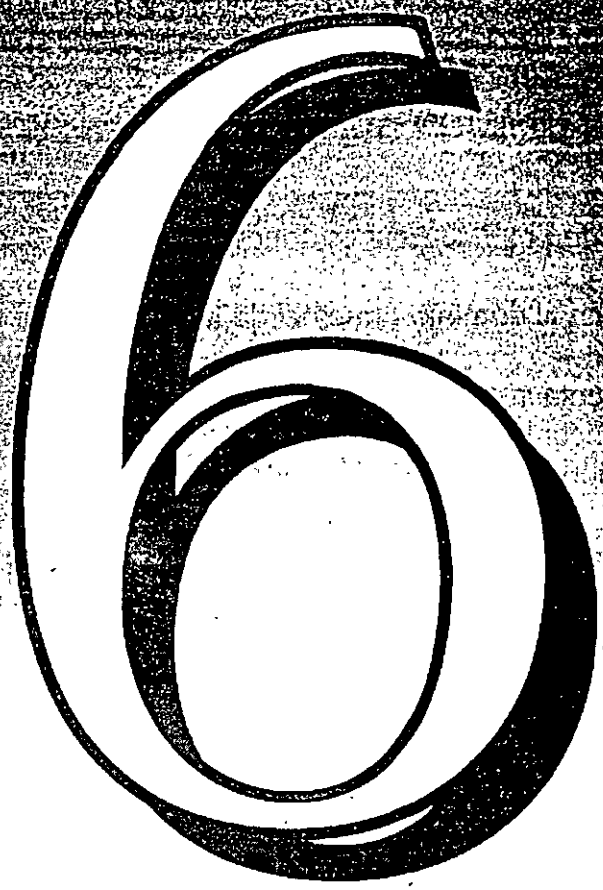
El Examinador de clases

Si usted tiene la versión profesional de Visual FoxPro, hay un nuevo elemento en el menú Herramientas, Examinador de clases, el cual tiene una característica interesante: Cuando usted genera una clase, se puede mostrar cómo se vería el código si se utilizara código en lugar del Generador de clases. Para usarlo, seleccione una clase, luego haga un clic en Ver código de clase, el tercer icono de izquierda a derecha, y estará viendo una versión de código de su clase. Desde luego, cualquier método que codifique se mostrará en los listados. Ésta es una gran forma para acostumbrarse a ver cómo puede usarse el código para construir y usar clases.

40

Conclusión

Éstas son unas cuantas maneras en las que usted puede escribir funciones y métodos dentro de Visual FoxPro. En los capítulos subsecuentes, se presentarán muchos más ejemplos de códigos. Para cuando termine de leer el libro, habrá visto maneras para realizar muchas de las cosas que las aplicaciones para bases de datos necesitan hacer.



El Generador de formularios

CAPITULO 6

EL Generador de formularios, el cual es virtualmente idéntico al Generador de clases, es donde usted pasará la mayor parte de su tiempo dentro de Visual FoxPro. Al igual que en FoxPro 2.6, todos los controles de pantalla están disponibles para usted en una tabla de herramientas, así como fragmentos de código y propiedades de control. Hay muchas, muchas más cosas funcionando dentro de los formularios de Visual FoxPro que las que había en versiones previas de FoxPro. Aquí es donde su aventura comienza.

Los formularios son el dominio principal de la programación orientada a objetos. A pesar de que existen interesantes objetos no visuales, mucho de su trabajo en programación orientada a objetos será hecho dentro de formularios. Y estoy convencido que encontrará que trabajar con los formularios Visual FoxPro es más fácil que con versiones anteriores de FoxPro.

Una vez que un formulario se ha generado, se guarda en una tabla con la extensión .SCX; los campos memo relacionados viven en un archivo con el mismo nombre y la extensión .SCT. Las propiedades y métodos del formulario se guardan, así como cada control, sus propiedades y métodos. Los archivos .SCX pueden ser usados, examinados y modificados igual que cualquier otra tabla.

Migración de FoxPro 2.6

Existen muchas, pero muchas diferencias entre Visual FoxPro y FoxPro 2.6, tales como usar propiedades y métodos en lugar de fragmentos de código dentro de ventanas. Aquí se verán sólo algunas de las diferencias importantes. A primera vista, usted podría intimidarse, pero las nuevas formas de hacer cosas son, a menudo, mucho más simples y se sentirá en casa una vez que se acostumbre a ellas.

Una de las primeras cosas que usted notará es que las pantallas ahora son llamadas *formularios*. Sin embargo, para compatibilidad con programas anteriores, el comando Modify Screen sigue funcionando y las extensiones para los archivos de formularios siguen siendo .SCX y .SCT.

Visualmente, hay algunas otras diferencias. La barra de herramientas Controles ahora se puede mover y está separada de la ventana de diseño. En FoxPro 2.6, cada ventana de diseño dentro de la pantalla tenía su propia barra de herramientas Controles en el costado izquierdo de la ventana. Ahora hay sólo una barra de herramientas, y puede tenerla flotando en el escritorio o acomodarla en cualquier extremo de la pantalla.

La barra de herramientas Controles también está basada en la biblioteca de clases que esté en uso (por omisión es la estándar), pero si usted hace clic en el segundo icono, la pequeña pila de libros, verá que puede cambiar bibliotecas de clases o agregar una biblioteca de clase nueva. La biblioteca de clase seleccionada se vuelve la fuente de la barra de herramientas, por lo que usted puede cambiar la barra de herramientas Controles. Cuando usted cambia bibliotecas de clase, la barra de herramientas cambia para reflejar que las nuevas herramientas están disponibles a través de la nueva biblioteca de clases.

La pantalla predeterminada pone líneas de cuadrícula dentro del formulario. Si no desea que aparezcan las líneas de cuadrícula, seleccione Ver/Líneas de cuadrícula, del menú y desactive las líneas de cuadrícula.

La pantalla Opciones/Formularios (ver figura 6-1) controla muchos aspectos del ambiente del Generador de formularios. Por ejemplo, para cambiar el valor por omisión del despliegue de líneas de cuadrícula a activado o desactivado, seleccione Herramientas, Opciones del menú, haga clic en la página Formularios y desactive las líneas de cuadrícula. Haga clic en el botón Establecer como predeterminado, para hacer que éste sea el valor predeterminado cada vez que utilice el Generador de formularios. Si no lo establece como un valor predeterminado, la configuración persistirá sólo para su sesión actual de Visual FoxPro.

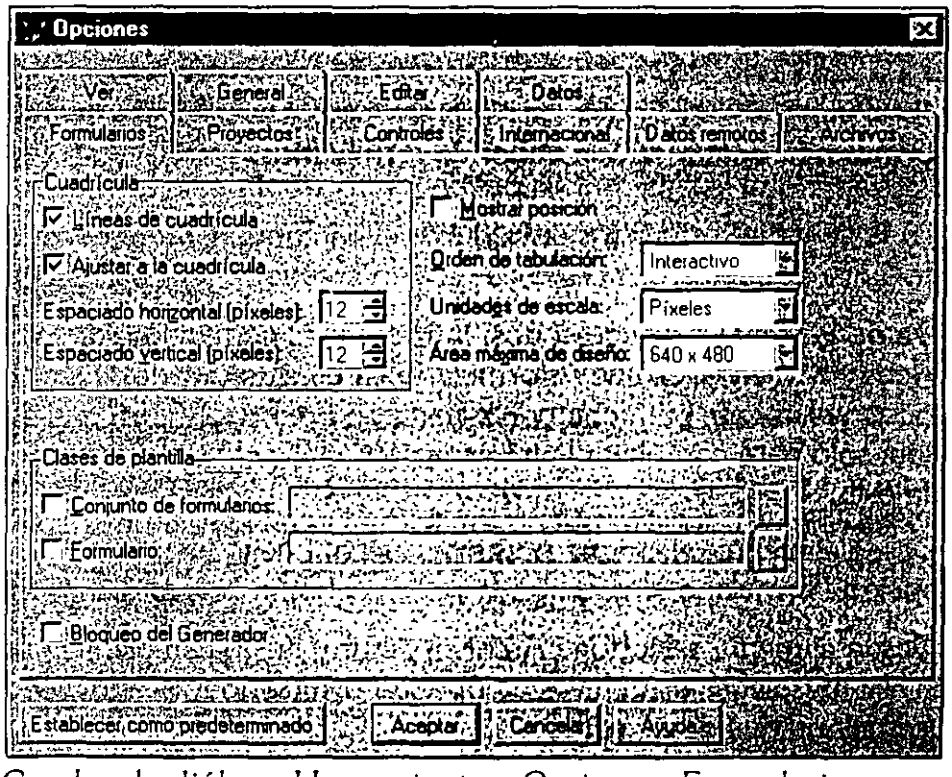


Figura 6-1

Cuadro de diálogo Herramientas, Opciones, Formularios.

4/4

Usted ya no usará formularios para generar código con GENSCR. (Sin embargo, seguirá usando GENMENU para generar programas de menú.) Ahora usted los ejecuta directamente del archivo .SCX, usando la sintaxis DO FORM *nombredeformulario*. Este hecho podría simplificar grandemente el proceso para aprender cómo generar formularios útiles en Visual FoxPro. Usted usaba archivos .SCX en FoxPro 2.6 junto con el programa GENSCR para generar código de programa en la forma de archivos .SPR. Como consecuencia, tenía que entender precisamente cómo trabajaba GENSCR y qué iba a hacer con su fragmento de código con el fin de controlar sus pantallas. Muchos programadores optaron por quitar todo el fragmento de código y ponerlo en otro archivo de programa, sólo para simplificar y acelerar el proceso de pruebas de pantallas.

Ahora, en lugar de saber cómo trabaja GENSCR, usted tiene que jerarquizar el modelo de evento, es decir, el orden en el que los eventos son ejecutados cuando se ejecuta DO FORM *nombredeformulario*. Afortunadamente es simple, siempre es el mismo y sus implicaciones son sencillas. Usted puede controlar muchos aspectos del ambiente del Generador de formularios mediante el uso de las configuraciones en el marco de página que se presenta cuando se selecciona Herramientas, Opciones, del menú de Visual FoxPro.

Cómo ejecutar el Generador de formularios

Hay seis formas de ejecutar el Generador de formularios:

- En la ventana Comandos, introduzca el comando Create Form o Modify Form (si es que el formulario ya ha sido creado).
- Desde la ventana de comandos, introduzca el comando Create Screen o Modify Screen (si es que el formulario ya ha sido creado). Este comando existe solamente para compatibilidad con FoxPro 2.6, así que es listado aquí sólo para completar la información. Utilice la sintaxis nueva en lugar de ésta.
- Desde el menú, seleccione Archivo, Nuevo. Luego haga clic en el botón de opción Formulario y haga clic ya sea en el botón Nuevo o en Asistente.
- Desde el menú, seleccione Archivo, Abrir y luego cambie el contenido de la lista desplegable Archivos de tipo, a Formulario (*.SCX). Seleccione el formulario que desee editar.

45

➢ Haga clic en el botón Nuevo de la barra de herramientas. Éste es el primer icono de la barra de herramientas predeterminada; se parece a una página blanca con la esquina superior izquierda doblada. Esto desplegará el cuadro de diálogo Nuevo, en el cual debe hacer clic sobre el botón de opción Formulario y luego en el botón Nuevo o Asistente.

➢ Desde el menú, seleccione Herramientas, Asistentes y luego seleccione Formularios. Esto lo guiará paso a paso a través del proceso de crear un formulario de una tabla existente.

Cambios al menú de Visual FoxPro en el Generador de formularios

Si trabaja dentro del Generador de formularios, el menú de Visual FoxPro cambiará interactivamente para ofrecerle algunas opciones de menú adicionales.

Cambios al menú Ver

* Diseño

Esto lo pone en el modo de diseño, el cual es también el modo por omisión cuando usted está editando un formulario. Este modo le permite poner controles en el formulario, mover los controles, hacer clic con el botón secundario del ratón para menús de propiedades y usar otras herramientas de diseño. La alternativa al modo de diseño es el modo de orden de tabulación, el cual le permite establecer rápidamente el orden de los tabuladores mediante un clic en el primer control en la pantalla, haciendo clic junto con la tecla MAYÚS en el resto de los controles para el orden que usted quiere, luego haciendo clic en Volver a ordenar o Cancelar en el cuadro de diálogo Orden de tabulación.

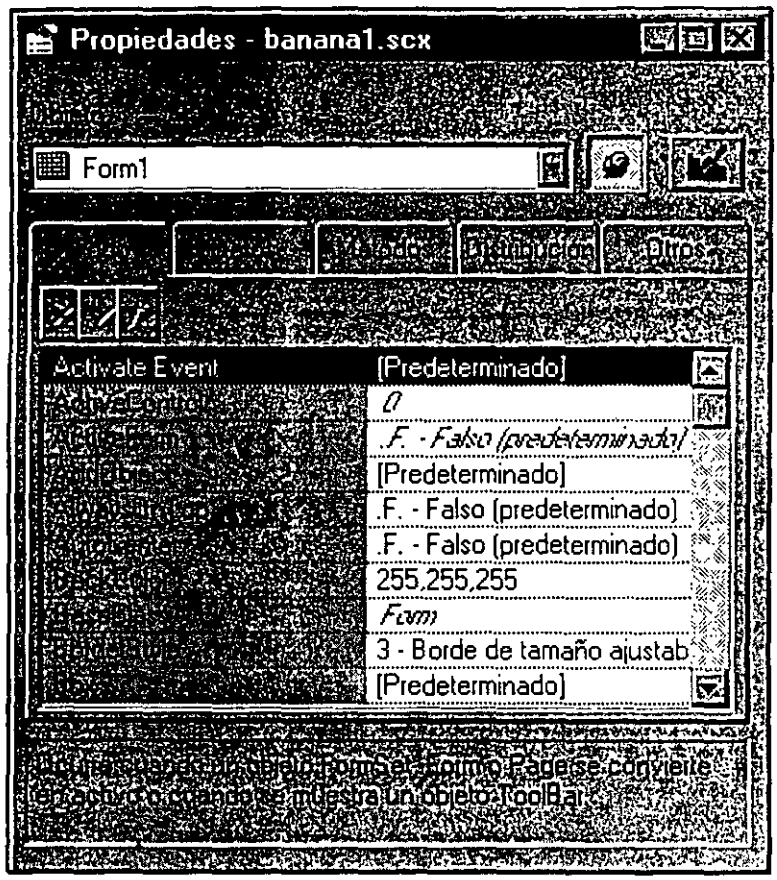
* Entorno de datos

Entorno de datos despliega una ventana que le permite definir las fuentes de datos disponibles, mientras está en el Generador de formularios. Cuando esta ventana se encuentra visible, el menú Entorno de datos es añadido al menú principal y le permite agregar o quitar tablas del entorno de datos.

* Propiedades

Esta opción despliega el cuadro de diálogo Propiedades, la cual se describe más adelante en el capítulo 7. Los formularios tienen propiedades, por ejemplo, color

Figura 6-2



Cuadro de diálogo Propiedades del formulario.

de fondo. También tienen código de eventos. El código Load event se ejecuta la primera vez que el formulario es cargado, y el código Activate se ejecuta siempre que usted regresa al formulario. Así que preferirá colocar código para abrir tablas en el código Load event, y seleccionar la tabla principal para el formulario en el código Activate. Además, puede asignar nuevas propiedades para el formulario. Las propiedades de los formularios son como variables locales adjuntas a cada elemento de su formulario. Es muy difícil cambiar los valores accidentalmente. A usted le va a gustar crear propiedades para sus formularios.

* Código

Despliega el cuadro de diálogo Código para el formulario, como se muestra en la figura 6-3, el cual le permite editar código que usted ha pegado a cualquiera de los eventos o métodos del formulario o a cualquiera de los controles en él. Las listas desplegables, en la parte superior de la ventana Código, le permiten seleccionar qué código desea editar. Si el pedazo de código que se muestra es menor que el largo de la ventana, presionar AvPag lo llevará automáticamente al siguiente fragmento de código.

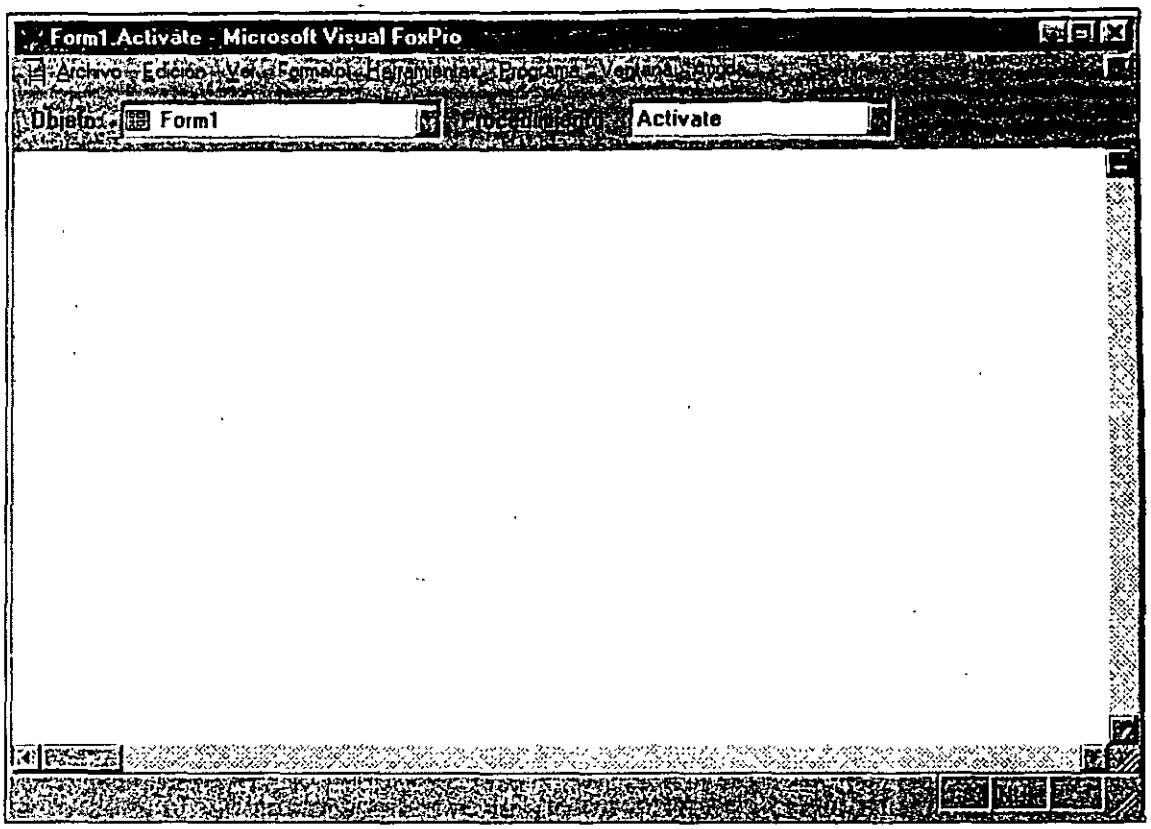


Figura 6-3

Ventana de código.

* Barras de herramientas Controles de formularios, Distribución y Paleta de colores

Estas selecciones cambian las barras de herramientas a activada o desactivada. Si alguna está activada, aparecerá en algún lugar de la pantalla. Si está flotando, podría estar cubierta por otras ventanas, por eso se recomienda que coloque su barra de herramientas en los extremos de la ventana Visual FoxPro. Para una descripción de cada barra de herramientas, vea la sección Barras de herramientas más adelante en este capítulo.

* Líneas de cuadrícula

Este parámetro controla el mostrar o no líneas de cuadrícula en la ventana, con el fin de proporcionar ayuda para la alineación. Usted podría pensar que sus controles tienen una molesta tendencia a ser ya sea demasiado largos, o demasiado pequeños; esto es debido a que el grano de la cuadrícula está establecido en 12 pixeles, el valor predeterminado de FoxPro. Puede cambiarlo a 1 o 2 para este formulario únicamente al seleccionar Formato, Configurar cuadrícula del menú mientras trabaja con su formulario, o puede cambiar el valor de granulación predeterminado en el marco de página Formulario, del menú

Herramientas, Opciones. Sin embargo, con las nuevas características de la barra de herramientas Distribución y las nuevas opciones del menú desplegable Formato, tal vez nunca tendrá que usar una cuadrícula de distribución.

* Mostrar posición

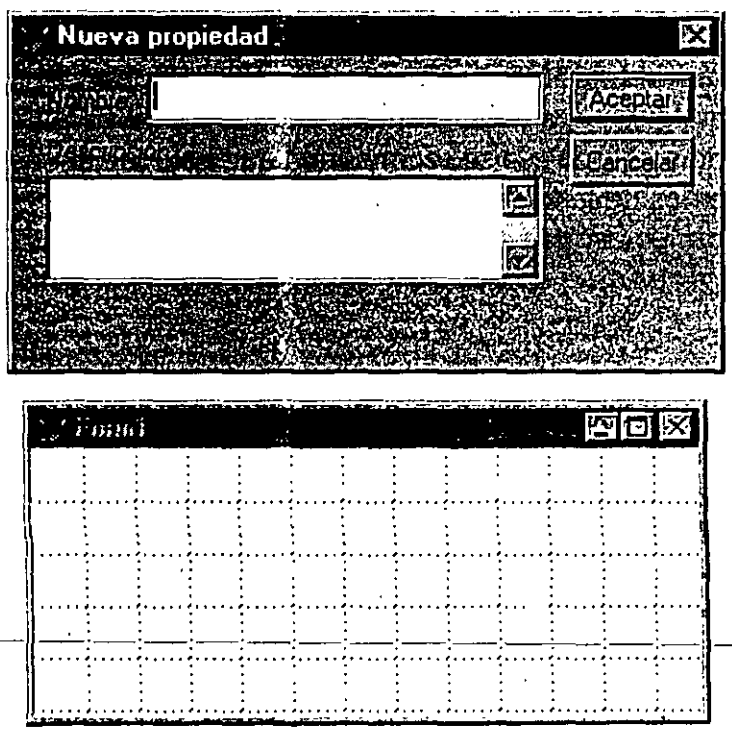
Activa o desactiva una pantalla en la barra de estado en el fondo de la pantalla, que despliega la posición y el tamaño exactos de los controles que ha seleccionado. Si hay muchos controles seleccionados, sólo es desplegada la posición; si no hay controles desplegados, pero usted hace clic y arrastra el ratón dentro del formulario para seleccionar un área, se desplegarán la posición y el tamaño del área seleccionada.

Cambios al menú Formulario

* Nueva propiedad/método

Despliega un cuadro de diálogo (ver figura 6-4) que le permite añadir una propiedad o método personalizados a su formulario. La propiedad o método se aplica localmente al formulario en forma automática, así que piense en ella como una variable local. Puede considerar a un método como un procedimiento local para el formulario.

Figura 6-4



Cuadro de diálogo Formulario/Nueva propiedad.

* Editar propiedad/método

Éste es esencialmente el mismo cuadro que el de Nueva propiedad/método, excepto porque aparece una lista desplegable de la cual puede seleccionar una propiedad o un método que desee editar. El cuadro le permite borrar la propiedad o el método.

* Crear conjunto de formularios

Le proporciona la posibilidad de añadir múltiples formularios. Estará completamente activo una vez que el formulario esté siendo ejecutado. Vea Agregar nuevo formulario y Quitar formulario.

* Quitar conjunto de formularios

Elimina la posibilidad de añadir formularios múltiples. Esta opción está disponible sólo si ha creado un juego de formularios y tiene solamente un formulario en el conjunto de formularios. En otras palabras, no puede Quitar un conjunto de formularios cuando existen formularios múltiples.

* Agregar nuevo formulario

Agrega un nuevo formulario vacío al conjunto de formularios actual. Esta opción no está disponible si no ha creado un conjunto de formularios.

* Quitar formulario

Quita el formulario actual del conjunto de formularios.

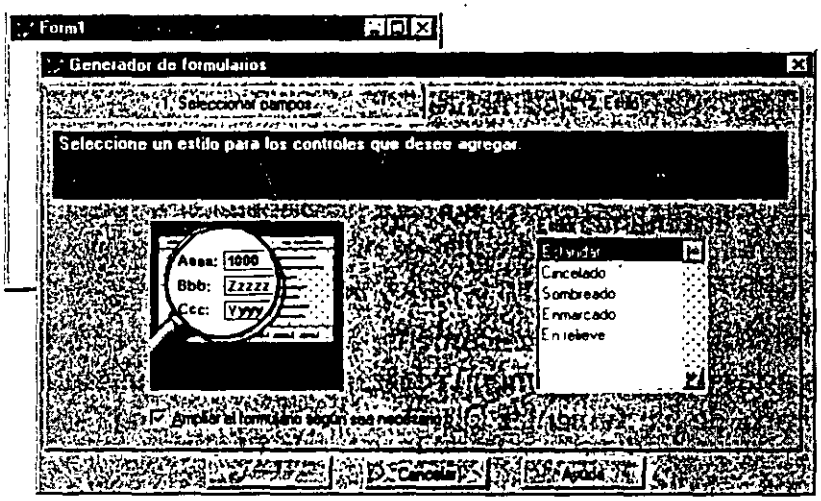


Figura 6-5

Fichas del Generador de formularios.

50

* Formulario rápido

Despliega el Generador de formularios, como se muestra en la figura 6-5, el cual permite seleccionar campos de una tabla y un estilo previamente definido. Luego genera un formulario por usted. Esto quiere decir que usted no tiene que llevar todos los campos hacia la pantalla y establecer todas las propiedades. Es un muy buen punto de inicio para pantallas de una sola tabla.

* Ejecutar formulario

Le preguntará si desea guardar cualquier cambio que haya hecho a su formulario y luego ejecutará en la pantalla el formulario, inmediatamente después de la pantalla del generador y todas sus barras de herramientas asociadas. Cuando termina de ejecutar el formulario, Visual FoxPro no regresa en forma automática al modo de diseño; usted tiene que editar el formulario explícitamente.

Cambios al menú Formato

* Alinear

Si al menos un elemento está seleccionado, Alinear despliega un submenú con las siguientes opciones:

- ➢ Alinear los bordes izquierdos
- ➢ Alinear los bordes derechos
- ➢ Alinear los bordes superiores
- ➢ Alinear los bordes inferiores
- ➢ Alinear los centros verticalmente
- ➢ Alinear los centros horizontalmente
- ➢ Centrar verticalmente
- ➢ Centrar horizontalmente

Si se selecciona un solo elemento, únicamente aparecerán las opciones de centrado. Todas estas opciones de menú realizan funciones equivalentes a las funciones de la barra de herramientas Distribución, descritas en la sección próxima.

* Tamaño

Si al menos un elemento está seleccionado, tamaño despliega un submenú con las siguientes opciones:

51

- Ajustar
- Ajustar a la cuadrícula
- Ajustar al más alto
- Ajustar al más corto
- Ajustar al más ancho
- Ajustar al más estrecho

Si está seleccionado un solo elemento, solamente se activarán Ajustar y Ajustar a la cuadrícula. Ajustar esencialmente encoge el objeto seleccionado al tamaño exacto en el que pueda contener la representación gráfica del mismo.

Por ejemplo, si usted crea un grupo de opciones, no sabe anticipadamente qué tan grande necesita ser para contener el texto o contener cuatro botones. Pero como puede darle dimensión para que ajuste con lo venidero, sólo cree cualquier tamaño del grupo de opciones (el valor predeterminado es de cerca de un centímetro cuadrado) y escriba los títulos. Haga clic con el botón secundario del ratón en el grupo de opciones y elija Editar, luego seleccione los múltiples botones de opciones y seleccione Formato, Tamaño, Ajustar, del menú. Todos los botones de opciones cambiarán su tamaño al largo del texto. Ellos podrían no estar totalmente visibles aún, pero esto es debido que usted también tiene que seleccionar el grupo de opciones y después seleccionar Formato, Tamaño, Ajustar, del menú otra vez. Esta vez, la caja que contiene al grupo se encogerá o expandirá para encerrar exactamente el número justo de botones de opciones que usted tenga y el ancho del título más largo.

Si selecciona Ajustar a la cuadrícula, el grupo toma el tamaño actual del objeto y lo redondea a las líneas de cuadrícula más cercanas. Note que esto podría quedar más pequeño que el tamaño del objeto, y podría quedar más grande. Si usted tiene activado Ajustar a la cuadrícula, los objetos se crean ajustándose a la cuadrícula, y al volver a darles tamaño se alínean automáticamente a las coordenadas de la cuadrícula, así que volver a ajustar a la cuadrícula no es necesario, a menos que usted haya ajustado alguno de los objetos al tamaño, lo cual ignora la cuadrícula.

Hay cuatro opciones de menú que permiten que el tamaño coincida con los extremos de los objetos seleccionados en ese momento. Ajustar al más alto, Ajustar al más corto, Ajustar al más ancho y Ajustar al más estrecho encogen o expanden todos los objetos al tamaño seleccionado.

* Espacio vertical y Espacio horizontal

Si tiene múltiples objetos seleccionados, Espacio vertical y Espacio horizontal le permiten establecer un tamaño estándar entre todos ellos o cambiar el espaciado entre objetos con espacio desproporcional. Las opciones del submenú son:

- ➢ Hacer igual
- ➢ Aumentar
- ➢ Disminuir

Hacer igual alinea cada objeto a su propia columna o fila (dependiendo de qué sea lo que seleccionó, Espacio vertical u horizontal) con espaciado igual entre todos los objetos. La cantidad de espaciado es la más pequeña que ya existe entre objetos que no están sobrepuestos o tocándose. Debido a esto, tenga cuidado de no tener objetos con espacio muy grande cuando seleccione esta opción, algunos objetos podrían ser colocados fuera de la parte visible de un formulario. Es aconsejable colocar objetos visualmente aproximados al lugar donde quiere que estén, y luego usar estas opciones para alinearlos precisamente. También recuerde que si hay espacios donde quiere deliberadamente dejar un objeto, esta opción no lo sabrá, así que mejor cree un objeto modelo, colóquelo en la ranura, haga el alineamiento, y luego borre el objeto modelo.

Una vez que ha igualado el espaciado, querrá afinar el ajuste de la cantidad de espaciado. Aumentar y Disminuir toman un grupo de objetos e incrementan o disminuyen el espacio entre cada uno de ellos, un pixel por vez.

* Traer al primer plano y Enviar al fondo

Estas opciones cambian el orden del objeto u objetos seleccionados. Todos los objetos en un formulario tienen un orden de objeto, el cual determina cómo son dispuestos gráficamente. El orden por tabulación está aparte, y puede ser diferente del orden de objetos. Tal vez quiera cambiar el orden de los objetos, por ejemplo, si usted tiene texto en la parte superior de un óvalo rojo. En ese caso, querrá que el texto esté en el frente del óvalo en el orden de objetos, sin importar en qué orden fueron creados (lo cual establece el orden de objetos predeterminado).

Si sólo un objeto se selecciona, se coloca al frente o fondo del orden de objetos. Si múltiples objetos están seleccionados, todos ellos se colocan en el frente o fondo. Si luego quiere determinar el orden entre el grupo seleccionado, puede seleccionar la opción de menú otra vez y ésta hará un giro entre ellos. Hacer una selección múltiple y cambiar el orden de los objetos por lo general se lleva a cabo, aun cuando el orden de los objetos sea irrelevante.

* Ajustar a la cuadrícula

Cambia el modo para el formulario en uso. Si usted quiere cambiar el valor por omisión para formularios nuevos, utilice el cuadro de diálogo Herramientas, Opciones, Formulario (ver figura 6-1).

* Configurar cuadrícula

Le permite cambiar la escala de la cuadrícula para el formulario en uso. Si quiere cambiar el valor predeterminado de la escala de la cuadrícula para formularios nuevos, utilice el cuadro de diálogo Herramientas, Opciones, Formulario (ver figura 6-1). Si cambia la escala de la cuadrícula, los objetos colocados en el formulario no se ajustarán a la nueva cuadrícula, aun si Ajustar a la cuadrícula está activado.

Barras de herramientas

Las barras de herramientas le permiten realizar tareas repetitivas con un solo clic del ratón. Puede diseñar sus propias barras de herramientas, pero por ahora se verán las que vienen con Visual FoxPro.

La barra de herramientas Generador de formularios

La barra de herramientas del Generador de formularios (ver figura 6-6) tiene toda la funcionalidad del menú Ver, como se describe antes en este capítulo. La única excepción es que en lugar de las opciones Líneas de cuadrícula y Mostrar posición, hay botones para el Generador de formularios (ver figura 6-7) y Autoformato. Este último se usa para desplegar el generador para el objeto que está seleccionado.

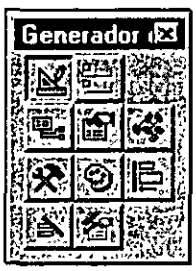


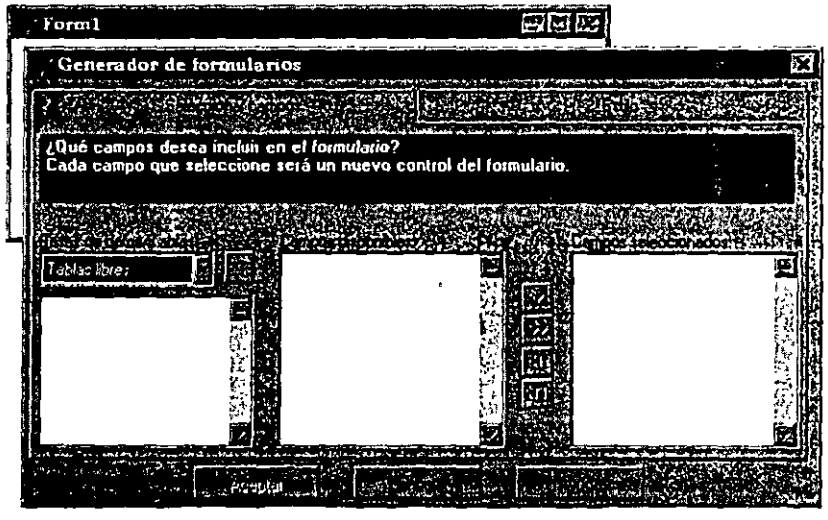
Figura 6-6

Barra de herramientas Generador de formularios.

La barra de herramientas Controles de formularios

Seleccionar objetos es el modo normal cuando usted no está colocando un objeto dentro de un formulario. Le permite hacer clic y arrastrar, cortar y pegar, seleccionar, hacer una selección múltiple y volver a dar tamaño a objetos. Si ha hecho clic en un botón de los Controles de formularios, pero no quiere crear un objeto, simplemente haga clic en el botón Seleccionar objetos, para cancelar.

Figura 6-7



Generador de formularios.

Para cambiar la barra de herramientas Controles de formularios (ver figura 6-8) y ver clases personalizadas u objetos OLE, haga clic en el botón Ver clases. Aparecerá un menú desplegable que le permite añadir controles .VCX o .OCX a la barra de herramientas, o se desplegará una lista predeterminada de controles OLE. Para definir la lista de controles OLE, utilice el cuadro de diálogo Herramientas, Opciones, Controles. Éste pedirá a su sistema una lista de objetos OLE en la base de datos de registro y permitira que usted seleccione de entre ellos para incluirlos en la barra Controles OLE, desde la cual se puede colocar cualquier control directamente en el formulario. ¿Podría ser más fácil?

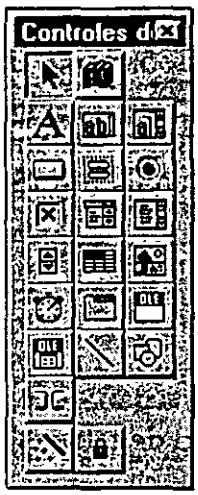


Figura 6-8

Barra de herramientas Controles de formularios.

La barra de herramientas Estándar

Hay 18 controles Visual FoxPro en la barra de herramientas Estándar. Aunque se hablará extensamente acerca de ellos, aquí hay un primer avance:

* Etiqueta

El control Etiqueta permite poner texto plano en su formulario. No permite introducir datos de texto, pero usted tiene control significativo en tiempo de ejecución sobre la apariencia y contenido.

* Cuadro de texto

Mostrado en la figura 6-9, éste es por mucho el control que se usa con más frecuencia. Se ve como un campo de entrada y eso es precisamente. Usted ahora usa cuadros de texto donde anteriormente usaba coordenadas @ GET nombredevariabledememoria, pero hay muchas, pero muchas diferencias.

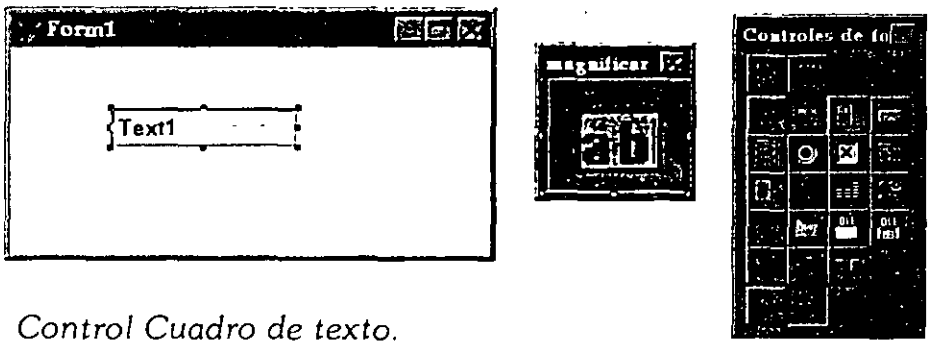


Figura 6-9

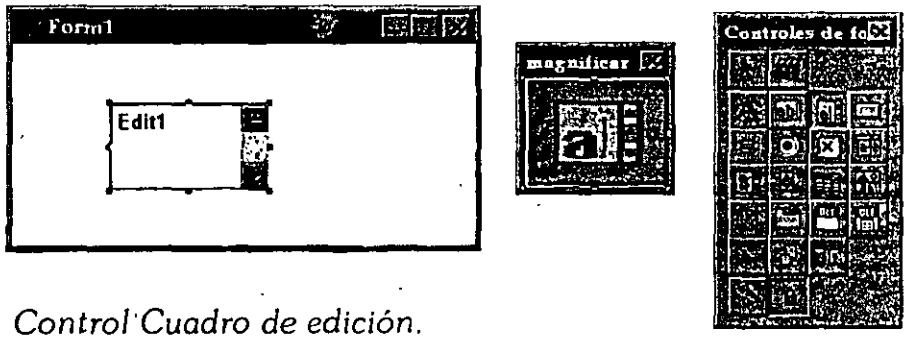
Control Cuadro de texto.

Los cuadros de texto se usan en muchos tipos de campos de entrada de datos: carácter, numérico, fecha y lógico. Así que usted tiene que proporcionar una fuente de datos, llamada *fuente control,* para ayudar a Visual FoxPro a imaginar qué tipo de datos debe esperar. Si utiliza, ya sea QuickScreen o el Asistente para formularios para generar pantallas, puede observar la propiedad ControlSource de los campos de datos para ver cómo se lleva a cabo esto.

* Cuadro de edición

Éste es básicamente un cuadro de texto con muchas líneas, como se muestra en la figura 6-10, sin tantas propiedades complejas. Utilícelo para introducir texto de formato libre, usualmente vinculado con campos memo dentro de tablas. Tenga cuidado con eso, mientras que los cuadros de edición son la mejor manera de trabajar con entradas de texto de formato libre, este control da por hecho que el usuario podría introducir más de un párrafo. De esta forma, la tecla ENTER se considera como parte de los datos introducidos, Visual FoxPro no toma a ENTER como una indicación de que el usuario desea salir del cuadro de edición del campo. En lugar de esto se debe presionar la tecla TAB o la combinación SHIFT +TAB. Parece ser una cosa pequeña, pero si los usuarios se irritan fácilmente, indicar esto podría ayudar mucho.

Figura 6-10



Control Cuadro de edición.

* Botón de comando

Éste es un botón sencillo (ver figura 6-11). A usted le gustará añadir código al evento Click de este control. Por ejemplo, un botón de navegación Siguiente, podría tener un código de evento Click parecido a esto:

```
SKIP
IF EOF()
   GO BOTTOM
ENDIF
THISFORM.Show
```

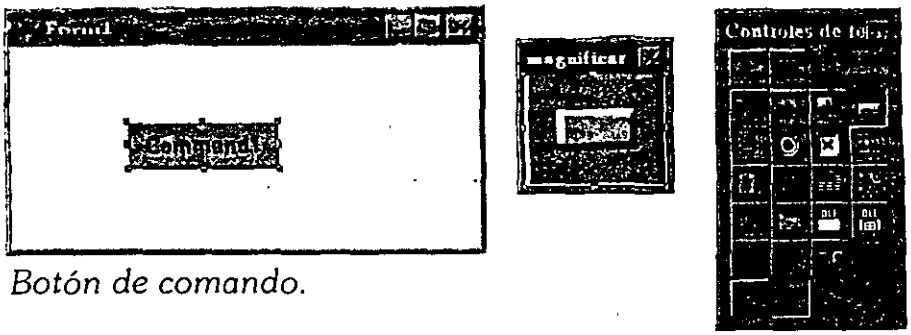


Figura 6-11

Botón de comando.

* Grupo de comandos

Éste es un simple contenedor de un grupo de controles de comandos, como se muestra en la figura 6-12. El contenedor tiene algunas propiedades y métodos, y puede responder a eventos separados de los botones que contiene. Utilice este control cuando tenga un conjunto de botones que están relacionados en funcionalidad. No tiene que usar una instrucción Case para decidir qué hacer, como era necesario en FoxPro 2.6, debido a que cada botón de comando en el contenedor tiene su propio evento Click.

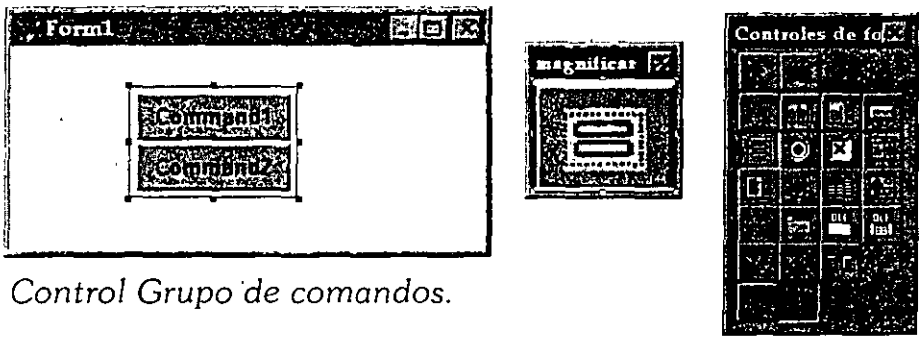


Figura 6-12

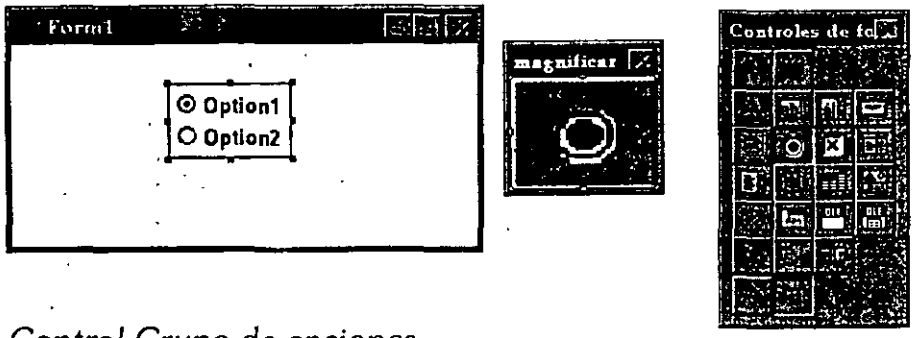
Control Grupo de comandos.

* Grupo de opciones

Este control (ver figura 6-13) es similar al grupo de comandos en el sentido de que es un contenedor para un conjunto de controles. Los botones de opciones contenidos (también conocidos como *botones de radio*) tienen sus propias propiedades y métodos separados, pero pueden ser manipulados como un grupo con la opción de objeto de grupo. También desempeña una función importante sobre los botones de opción que contiene, lo cual asegura que sólo uno es establecido por vez. Esto es lo que funcionalmente distingue a los botones de opción de las casillas de verificación. El grupo de opciones tiene un valor que corresponde al elemento que está marcado. Puede estar vinculado a un campo numérico en una tabla; así cualquier número de opciones mutuamente excluyentes puede ser almacenado como un número dentro de una tabla.

5p

Figura 6-13

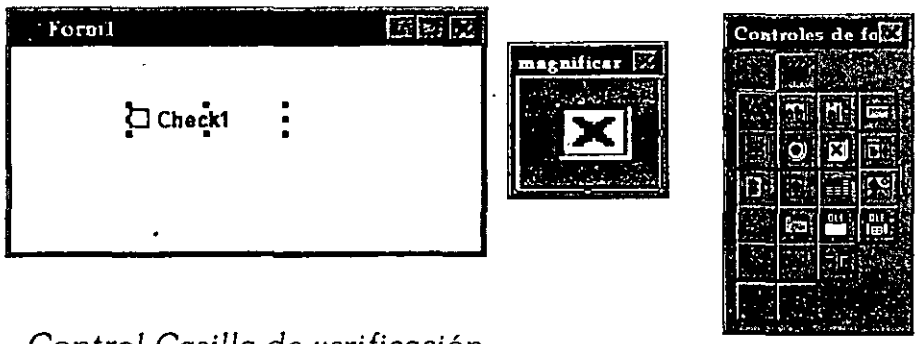


Control Grupo de opciones.

* Casilla de verificación

Éste es un control sencillo que tiene ya sea un estado de activado o desactivado (ver figura 6-14). Las casillas de verificación difieren de los botones de grupo de opciones debido a que ellas no dependen del valor de otros controles. Sólo un botón del grupo de opciones puede ser seleccionado a la vez, mientras que varias casillas de verificación pueden estar marcadas a la vez. Aunque era de esperar que los controles de casillas de verificación fueran de tipo lógico, cuando se vinculan a campos de tablas tienen un valor de 0 o 1.

Figura 6-14



Control Casilla de verificación.

Si usted quiere que un grupo de casillas de verificación actúe como grupo de opciones, puede programar las casillas de verificación para reaccionar una con otra; por ejemplo, cuando una se activa, el resto se desactiva.

* Cuadro combinado

Un cuadro combinado (ver figura 6-15) también es conocido como lista desplegable. Se llama cuadro combinado debido a que usted puede, escribir un valor en la región de edición o hacer clic en el botón para desplegar una lista. Estos campos pueden estar ligados a campos carácter dentro de tablas y son una forma simple de validar datos de entrada con una serie restringida de valores.

59

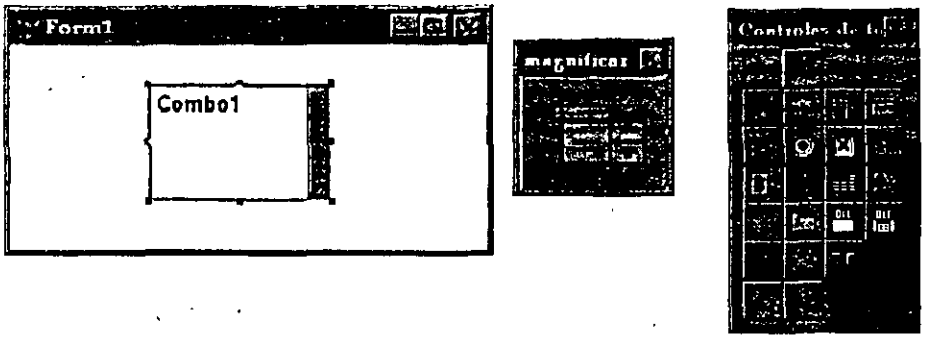


Figura 6-15

Control Cuadro combinado.

* Cuadro de lista

Es una lista de tamaño fijo (ver figura 6-16), a diferencia de los cuadros combinados. Usted puede volver a darles tamaño, pero el usuario no. El valor puede estar ligado a un campo dentro de una tabla y el contenido puede ser ligado a arreglos o tablas y cambiarse dinámicamente. Los cuadros de lista pueden tener varias columnas, las cuales pueden volver a ser dimensionadas dinámicamente. El valor del cuadro de lista puede ser ligado a cualquiera de las columnas. Por ejemplo, la primera columna puede ser el nombre del cliente, la segunda la edad y la tercera el número telefónico y el valor del cuadro de lista puede estar ligado a la columna edad.

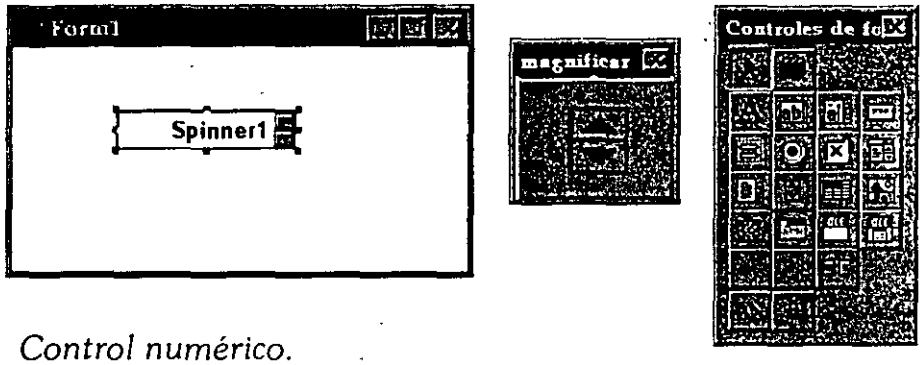
Figura 6-16

Control Cuadro de lista.

* Control numérico

El Control numérico, mostrado en la figura 6-17, es un control numérico que tiene pequeñas flechas al lado derecho del campo de entrada de datos, que incrementa el valor en el campo con una cantidad específica. El control puede tener un máximo y mínimo predeterminados, que puede ser diferente para introducción por teclado o por control numérico, y puede tener una máscara de entrada. Como pasa con cualquier control de entrada de datos, puede estar ligado a un campo numérico dentro de una tabla.

Figura 6-17

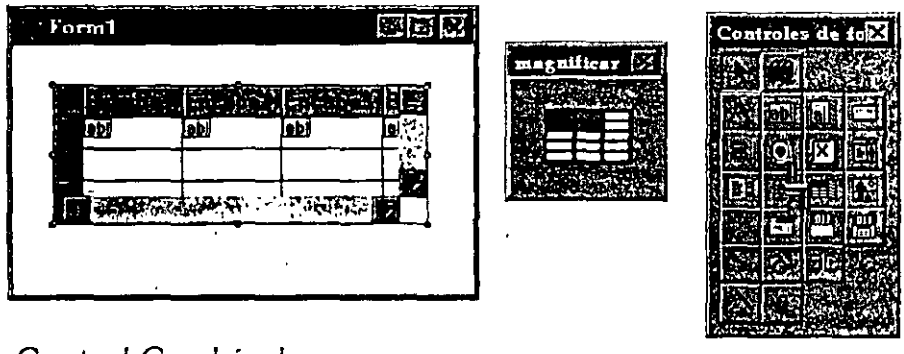


Control numérico.

* Cuadrícula

El control Cuadrícula (ver figura 6-18) es uno de los controles estándar más poderosos de Visual FoxPro. Para aquellos que están familiarizados con FoxPro 2.6, es esencialmente lo mismo que una ventana Examinar, excepto que ahora puede sobreponerse en un formulario y puede desplegar muchos datos.

Figura 6-18



Control Cuadrícula.

Las cuadrículas también pueden contener otros objetos, incluidos casillas de verificación, cuadros combinados e incluso otras cuadrículas. Una cuadrícula es un contenedor para objetos de columna, los cuales pueden ser controlados en forma separada. Cada uno puede tener propiedades y métodos propios, y responder a eventos de manera independiente.

61

* Imagen

El control Imagen, mostrado en la figura 6-19, se denominó control Picture en FoxPro 2.6. Puede desplegar una imagen gráfica dentro de un formulario la cual puede ser controlada dinámicamente. Por ejemplo, usted puede desplegar las fotografías de empleados, imágenes de productos o registros de inventario mientras circule por una base de datos de empleados. La imagen puede ser transparente u opaca, permitiendo a los objetos de fondo ser visibles por debajo, incluidos texto y otros controles.

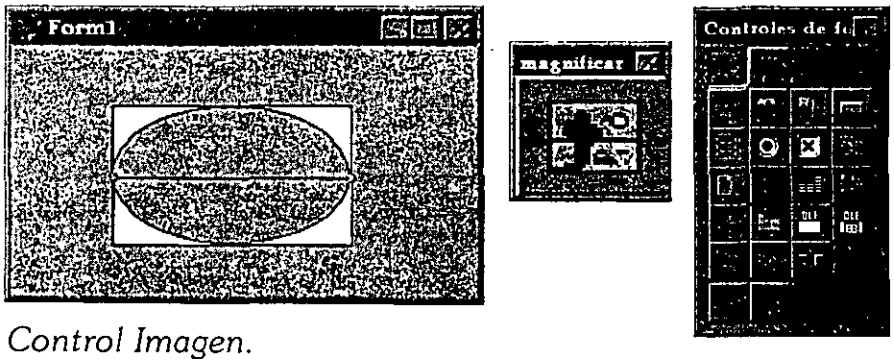


Figura 6-19

Control Imagen.

* Cronómetro

El Cronómetro (ver figura 6-20) le proporciona control de programación sobre eventos de cronómetro. Con el Cronómetro se puede tener una ejecución de procedimiento en intervalos establecidos, por ejemplo, revisar la inactividad y salir de un formulario si los usuarios han abandonado los escritorios, con el fin de desbloquear el registro que han bloqueado dentro de la red.

Hay un límite para la resolución de los eventos de cronómetro mismo que es impuesto por el cronómetro del hardware. A pesar de que el control Cronómetro le da control en milisegundos sobre los eventos de cronómetro, sólo hay 19.2 tics de reloj por segundo. Por ejemplo, un cronómetro que mide centésimas de segundo avanza en incrementos de un poco más de .05 segundos y no en incrementos de .01 segundos.

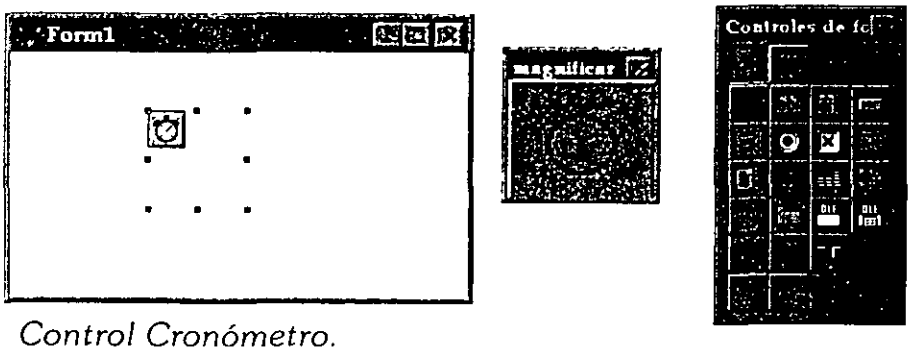


Figura 6-20

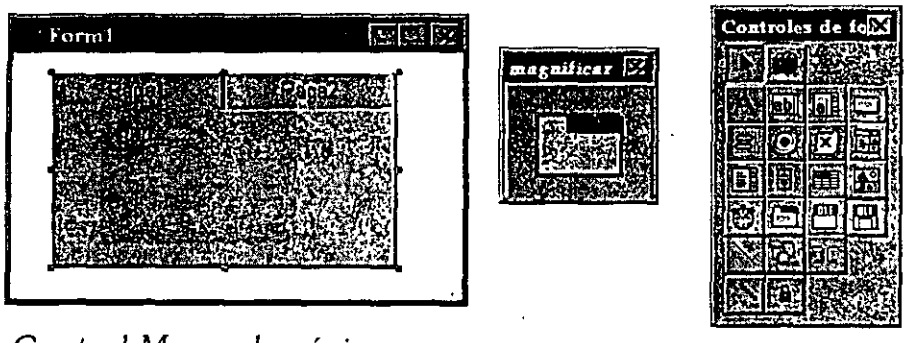
Control Cronómetro.

Cuando usted coloca un control Cronómetro en el formulario, aparece como un reloj pequeño, pero no está visible en el formulario mientras se ejecute.

* Marco de página

El control Marco de página (ver figura 6-21) es otro nuevo y poderoso control dentro de FoxPro. El control Marco de página le permite crear cuadros de diálogo con cualquier número de fichas. Con este control usted puede crear incluso pantallas de tipo Asistentes o formularios cuyos contenidos cambien interactivamente, con base en cualquier lógica que desee.

Figura 6-21



Control Marco de página.

Los Marcos de página son contenedores, al igual que el grupo de comandos, el grupo de opciones y los controles de cuadrícula. Los objetos que contienen están ordenados en forma jerárquica. Los objetos pueden ser colocados en el frente de los marcos de página. Sin embargo, parecen estar contenidos en cada marco de página, pero no tienen que ser duplicados dentro de cada página.

* Control Contenedor OLE

El control Contenedor OLE creará un contenedor en el formulario para cualquier objeto contenedor OLE. Esto incluye las hojas de cálculo de Microsoft Excel, los documentos de Microsoft Word, las imágenes de Paintbrush y los objetos de Microsoft Graph, dibujos de Visio, películas y sonidos. El número de las aplicaciones de contenedores OLE 2.0 está creciendo exponencialmente y con el lanzamiento de Windows 95, la compatibilidad con OLE 2.0 se convertirá en el estándar. Recuerde que entre los sistemas de alto nivel, la funcionalidad OLE 2.0 tiende a ser un cuello de botella en cuanto a desempeño, así que use este control con poca frecuencia. Los controles .OCX le darán a las hojas de cálculo una funcionalidad más económica (en cuanto a desempeño) que un vínculo a Excel, así que considere todas las opciones antes de crear un contenedor OLE en la aplicación.

* Control OLE dependiente

Este control OLE está limitado a un campo dentro de una tabla.

* Línea

Este control, mostrado en la figura 6-22, le permite dibujar líneas en cualquier parte dentro del formulario, de cualquier color y largo. Éstas tienen desencadenantes de eventos, de forma que usted puede dibujar una línea abajo de la mitad de la pantalla y tener un desencadenante de evento siempre que el ratón se mueva de una mitad de la pantalla hacia la otra.

* Forma

Utilice este control, mostrado en la figura 6-23, para generar cualquier forma, abarcando desde un rectángulo hasta un círculo o una elipse. También puede crear un rectángulo con los extremos redondeados con la propiedad de curvatura. Una forma con curvatura 0 es un rectángulo, y una forma con curvatura 99 es un círculo, o una elipse si la proporción dimensional no es 1:1. Las formas pueden tener cualquier color de frente o fondo, cualquier ancho de línea y tienen desencadenantes de eventos.

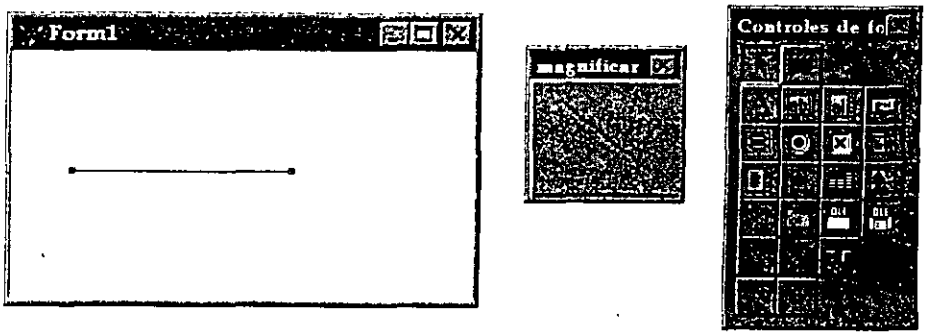


Figura 6-22

Líneas.

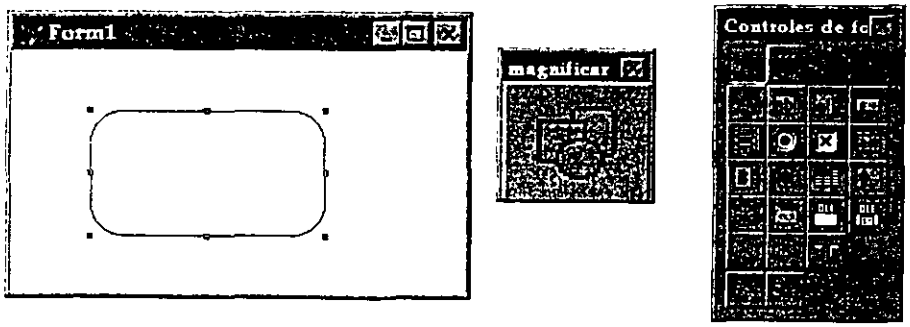


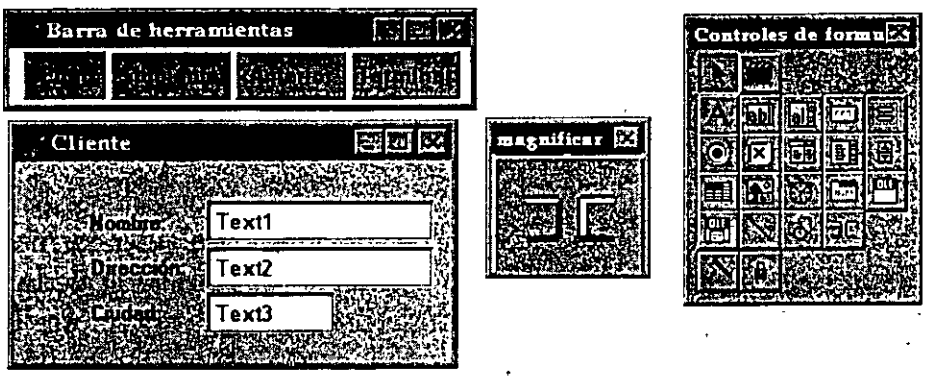
Figura 6-23

Formas.

* Separador

El control Separador (ver figura 6-24) es para usarse con la clase Barra de herramientas personalizada. Por ejemplo, cuando usted emplea la clase Barra de herramientas para crear clases propias, puede usar el separador para poner espacio entre dos botones.

Figura 6-24



Separador.

* Bloqueo del Generador y Bloqueo del botón

El bloqueo del generador no es un control; es un apagador que determina si el generador aparece automáticamente cuando usted crea una nueva instancia de un objeto. El bloqueo del botón tampoco es un control; es un apagador que determina si un botón se mantiene oprimido después de que lo utilizan dentro de un formulario. La conducta predeterminada es regresarle a usted al modo de seleccionar objeto, pero si se tiene activado el bloqueo de botón, podrá poner 20 casillas de verificación dentro de un formulario, sin tener que hacer 20 veces clic en el botón de casilla de verificación.

La barra de herramientas Distribución

Cualquiera que haya desarrollado en un entorno visual conoce la agonía de estar parado frente a la pantalla, a dos pulgadas de distancia e intentar mover el ratón lo suficientemente lento de tal manera que pueda alinear objetos al pixel vertical exacto dentro de la pantalla. Luego, tan pronto como se levanta el dedo del ratón para dejar el objeto exactamente en el pixel correcto, el ratón se mueve y debe empezarse todo de nuevo.

Desde que usé un generador de aplicaciones visuales en una máquina Silicon Graphics en la escuela, me sentí decepcionado de que FoxPro no tuviera herramientas de alineación. Bueno ahora las tiene. Una vez que se haya acostumbrado a diseñar con las herramientas mostradas en la figura 6-25, su pantalla no sólo se verá mucho más clara y profesional, sino que arreglará los defectos rápidamente.

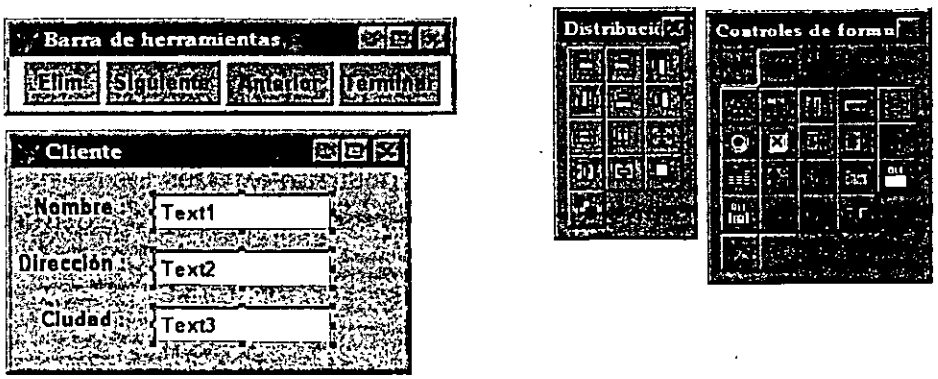


Figura 6-25

Barra de herramientas Distribución.

Para ser activadas casi todas las herramientas requieren que más de un objeto esté seleccionado. ¿Cómo se alinea un objeto consigo mismo? Las excepciones son Centrar verticalmente, Centrar horizontalmente, Traer al primer plano y Enviar al fondo, aunque cada una de ellas también trabajará cuando se seleccionen múltiples objetos.

Alinear los bordes izquierdos Mueve todos los objetos seleccionados, de forma que los bordes izquierdos se alinien con el objeto seleccionado lo más lejos posible hacia la izquierda.

Alinear los bordes derechos Mueve todos los objetos seleccionados, de forma que los bordes derechos se alinien con el objeto seleccionado lo más lejos posible hacia la derecha.

Alinear los bordes superiores Mueve todos los objetos seleccionados, de forma que los extremos superiores se alinien con el borde más alto del objeto seleccionado.

Alinear los bordes inferiores Mueve todos los objetos seleccionados, de forma que los extremos inferiores se alinien con el extremo más bajo del objeto seleccionado.

Alinear centros verticalmente Mueve todos los objetos seleccionados, de forma que se alinien verticalmente, centrados por un eje central común. El eje central es determinado por la siguiente regla: si los límites verticales (los extremos izquierdo y derecho) de un objeto seleccionado encierran a todos los demás objetos seleccionados, el eje central vertical de dicho objeto se usará. Si no existe ese objeto, entonces se hace un promedio de los ejes centrales de todos los objetos seleccionados, y éste es usado.

Alinear centros horizontalmente Mueve todos los objetos seleccionados, de forma que se alinien horizontalmente, centrados por un eje central común. El eje central se determina por la siguiente regla: si los límites horizontales (los extremos superior e inferior) de un objeto seleccionado encierran a todos los demás objetos seleccionados, se usa el eje central horizontal de ese objeto. Si no existe ese objeto, entonces se hace un promedio de los ejes centrales de todos los objetos seleccionados, y se usa éste.

Mismo ancho Alarga todos los objetos seleccionados, de forma que horizontalmente tengan el mismo tamaño que el objeto seleccionado más ancho. Todos los objetos se expanden hacia abajo y hacia la derecha.

Mismo alto Alarga todos los objetos seleccionados, de forma que verticalmente tengan el mismo tamaño que el objeto seleccionado más alto. Todos los objetos se expanden hacia abajo y hacia la derecha.

Mismo tamaño Alarga todos los objetos seleccionados hasta que tengan horizontalmente el mismo tamaño que el objeto más ancho y verticalmente el mismo tamaño que el objeto seleccionado más alto. Note que el objeto más ancho puede crecer en altura y el objeto más alto puede crecer en ancho. Todos los objetos se expanden hacia abajo y hacia la derecha.

Centrar horizontalmente Mueve todos los objetos seleccionados, de forma que los ejes verticales sean el eje del formulario.

Centrar verticalmente Mueve todos los objetos seleccionados, de forma que los ejes horizontales sean el eje del formulario.

Traer al primer plano Trae el objeto seleccionado al primer plano en el orden de los objetos. Si están seleccionados varios objetos, al oprimir repetidamente el botón, pasarán uno tras otro al primer plano.

Enviar al fondo Envía el objeto seleccionado al final del orden de los objetos. Si están seleccionados varios objetos, al presionar repetidamente el botón pasarán uno tras otro al fondo.

La barra de herramientas Paleta de colores

Utilice la barra de herramientas Paleta de colores, mostrada en la figura 6-26, para establecer rápidamente el color de los objetos. Para cambiar el color de un objeto, primero selecciónelo (o seleccione muchos para cambiar el color de varios objetos de una sola vez). Cuando tenga una selección, escoja el botón Color de primer plano o Color de fondo en la barra de herramientas Paleta de colores, y luego haga clic en el color que desea establecer. Son botones de activación, así que se pueden cambiar inadvertidamente ambos colores si no se es cuidadoso. Desde luego, se pueden cambiar los colores del frente y fondo de un objeto al mismo color, en caso de querer ocultarlo. Por ejemplo, usted podría hacer esto en un campo de contraseña si desea ocultar el texto que está siendo introducido en él.

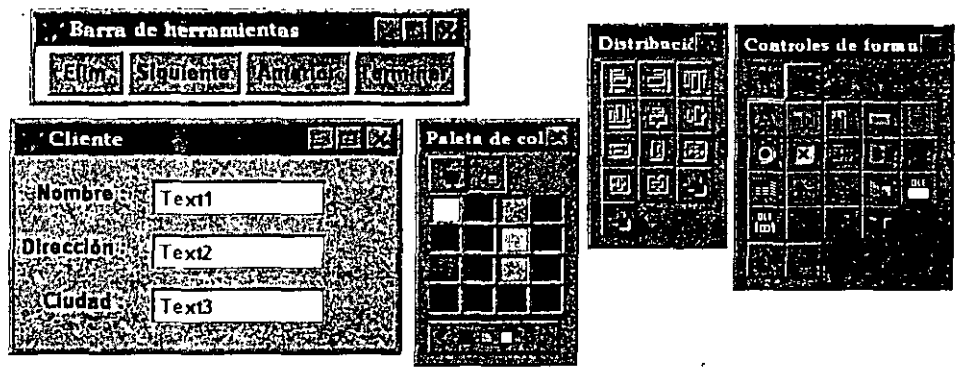


Figura 6-26

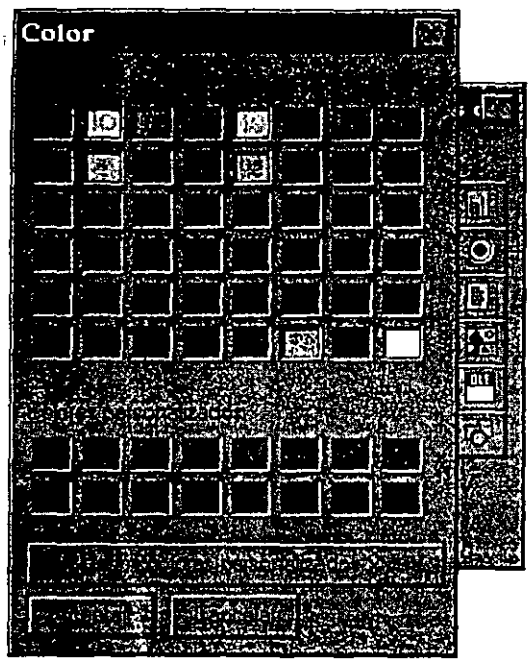
Barra de herramientas Paleta de colores.

Si el color que busca no es uno de los 20 colores previamente definidos por Windows, seleccione Otros colores y obtendrá el cuadro de diálogo Color (ver figura 6-27). Esto despliega los 48 colores básicos de Windows y otros 16 adicionales de los que puede seleccionar alguno. Si aun así no es suficiente, hay un cuadro de diálogo de colores personalizados (ver figura 6-28). Con esta extensión al cuadro de diálogo Color básico, usted puede agregar colores a la lista de Colores personalizados y acceder a cualquiera de un total de 16 millones de colores (si el monitor puede manejarlos). Los colores que no estén disponibles para usted se verán como patrones temblorosos en un intento de aproximarse al color en cuestión.

Usted puede definir un color al arrastrar el ratón sobre el campo de color, hasta que el color que usted busca aparezca en la pantalla Color/Sólido, o puede

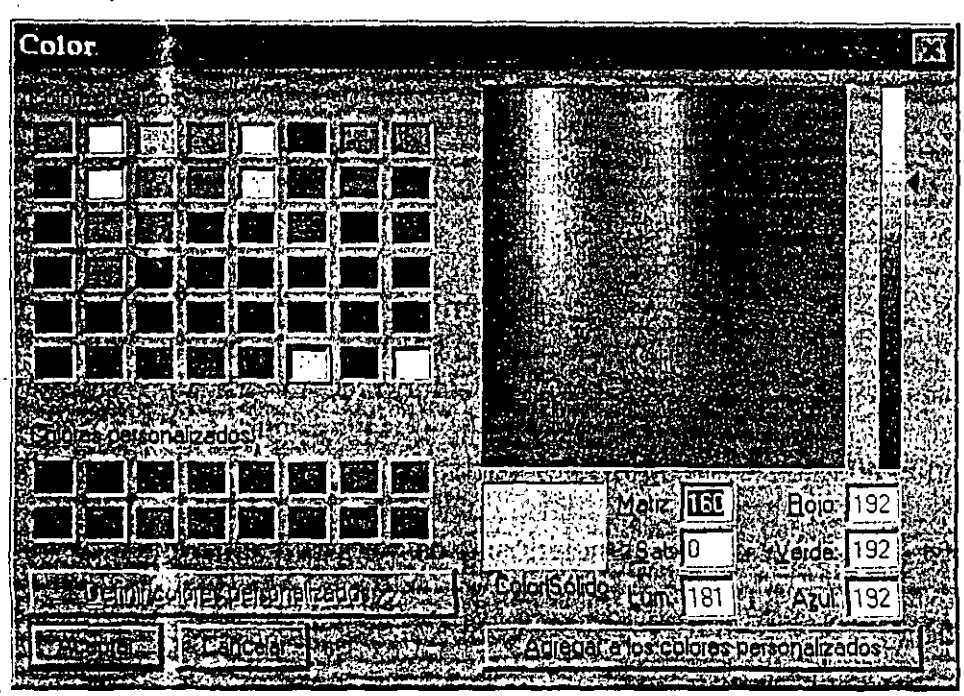
introducir valores específicos para los componentes rojo, verde y azul o para el matiz, la saturación y la luminosidad. Éstas son dos formas estándares de representar colores numéricamente.

Figura 6-27



Cuadro de diálogo Color.

Figura 6-28



Extensión del cuadro de diálogo Color para colores personalizados.

Trabaje con el Generador de formularios

Usted ha visto una muestra de las herramientas, controles y opciones que puede usar cuando está diseñando pantallas, pero utilizarlas es otra historia. Aquí es' donde los ejemplos proporcionados por este libro y el modelo de código de Visual FoxPro son la única solución. Sin embargo se proporcionarán algunos consejos para ayudarlo a evitar trampas, así como algunas lecciones para ahorrar tiempo.

Los formularios también tienen propiedades

El formulario en sí es un objeto mucho más robusto que cualquiera de los otros objetos de controles individuales. Tiene más propiedades, métodos y desencadenamientos de eventos que cualquiera de los controles. Es fácil ver qué tanto puede hacerse con las propiedades y los métodos, pero si no está consciente de la capacidad de programación de los formularios, pronto se verá atrapado en un callejón sin salida.

Otro capítulo en el libro habla sobre estas características en mayor detalle, pero esté consciente de ellas cuando use el Generador de formularios y tendrá a su disposición un juego de herramientas mucho más poderoso.

Las propiedades son dinámicas

Todas las propiedades pueden ser modificadas "sobre la marcha", es decir, mientras un programa está ejecutándose. Resulta trivial cambiar el color de cualquier objeto, incluido el formulario, con base en cualquier desencadenante, como un evento de cronómetro, la presión de un botón o un cierto valor introducido dentro de un cuadro de texto.

No se tire por la borda por esto, mejor modifique las propiedades dinámicamente, con el fin de crear aplicaciones más "inteligentes". Cuando ciertas acciones no deban ser realizadas, desactive el botón o elimínelo de la pantalla. Si sólo están disponibles algunas opciones, haga que éstas sean las únicas visibles. Si quiere proporcionar al usuario una indicación visual, como cambiar el color del texto a rojo en señal de advertencia, sepa que es muy fácil hacerlo.

Usted incluso puede mover dinámicamente los objetos alrededor de la pantalla, dependiendo de la funcionalidad requerida.

Plantillas de clases

Una de las primeras cosas que usted deseará hacer después de elaborar un formulario será cambiar algunas de las propiedades predeterminadas. Por ejemplo, se usó gris claro como fondo estándar debido a que la textura 3-D de los controles es mucho más efectiva así. Usted puede tener una fuente estándar que quiera usar, o incluso código estándar en el evento Activate. Quizá siempre use controles de cronómetro y tal vez el 90% del tiempo tenga un grupo de comandos Aceptar/Cancelar en alguna parte de sus formularios. Aun si alguno de éstos equivale a tres o cuatro clics de ratón (que sí sucede), si usted efectúa cada uno de ellos cada vez que crea un formulario, serán 30 segundos que estará perdiendo.

Las plantillas de clases le permiten crear un formulario estándar (o un conjunto de formularios) y luego usarlo como la base para cada formulario nuevo que cree. Para utilizar esta característica, simplemente cree la plantilla de formulario, guárdela, y luego seleccione el cuadro de diálogo Herramientas, Opciones, Formulario (ver figura 6-1) para establecer ese formulario como plantilla de clase. Si cambia los valores predeterminados en cualquier formulario individual, estos valores no son guardados en la plantilla. Si hace cambios a la plantilla, los cambios afectarán a todos los formularios que cree en el futuro, pero no serán incorporados automáticamente a todos los formularios que haya creado usando esa plantilla.

Agregue controles

Para agregar controles a un formulario, simplemente haga clic en la barra de herramientas Controles sobre el control que quiera agregar. El botón en la barra de herramientas quedará oprimido, y el cursor cambiará a una cruz. Seleccione el área dentro del formulario que desea que el control ocupe. El significado de esta área será diferente, dependiendo del tipo de control. Por ejemplo, un botón de comando o forma será literalmente del tamaño del área que usted seleccionó. Sin embargo, las casillas de verificación y los cuadros combinados están limitados en altura, así que el área de despliegue será de una altura constante, sin importar qué tan grande sea el área de selección. Algunos controles, como el cronómetro, no se despliegan, así que el tamaño del área seleccionada no es importante.

71

Para colocar múltiples controles del mismo tipo repetitivamente dentro de un formulario, seleccione el icono de bloqueo de botón (el que se parece a un candado). Esto forzará a que el botón del control se quede presionado hasta que usted físicamente selecclone otro control o el botón para seleccionar objetos. La conducta predeterminada (sin usar el bloqueo de botón) lo regresa al modo de seleccionar objetos, tan pronto como el objeto sea colocado.

Agregue campos de datos

Para agregar campos de datos, simplemente coloque un cuadro de texto dentro del formulario, de la misma manera que con cualquier otro control, y establezca la propiedad Fuente Control para el campo de datos que quiera enlazar a él.

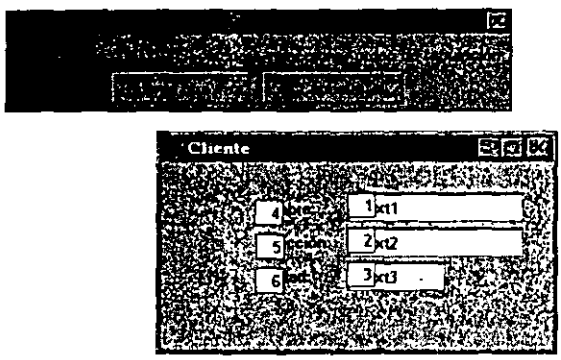
Una forma más sencilla consiste en sacar provecho de las propiedades del entorno de datos del formulario, para crear un entorno de datos que describa las tablas y relaciones usadas en el formulario. Cuando la ventana Entorno de datos está activa, usted puede simplemente hacer clic y arrastrar los campos dentro del formulario, lo cual creará automáticamente un cuadro de texto del tamaño correcto, vinculado a la tabla. Este enfoque utiliza la clase base Textbox de FoxPro, sin embargo, es un recurso limitado si usted usa sus propias clases.

Vuelva a ordenar paradas de tabulación

El orden en el que usted agrega campos a un formulario determina el orden de examinación, es decir, a dónde se dirige el cursor cuando se presiona ENTER para abandonar un campo. Aunque usted introduzca campos en un orden perfectamente razonable y luego los vuelva a ordenar, FoxPro no sabe que usted ya no quiere saltar al campo Código Postal después de abandonar el campo Estado. Cuando usted introduzca datos, el cursor brincará de campo en campo en la misma forma aleatoria. Esto puede ser desconcertante para el usuario.

La forma más sencilla de cambiar el Orden de tabulación de los controles dentro del formulario consiste en selecionar Ver/Orden de tabulación del menú. Aparecerá una pequeña ventana con los botones Ordenar otra vez y Cancelar, así como pequeñas cajas con un paro de tabulación próximas a cada control, dentro del formulario (ver figura 6-29). Cuando haga clic en alguna de las pequeñas cajas, ésta se convertirá en la primera Parada del tabulador. Y si luego deja presionada alguna de las teclas MAYÚS y continúa haciendo clic en la cajas, el orden en el que usted haga clic sobre ellas será el Orden de tabulación de los controles dentro del

Figura 6-29



Modo de parada del Orden de tabulación.

formulario, y el orden numérico se desplegará en las cajas. Cuando termine, haga clic en el botón Ordenar otra vez o Cancelar, para mantener el orden anterior.

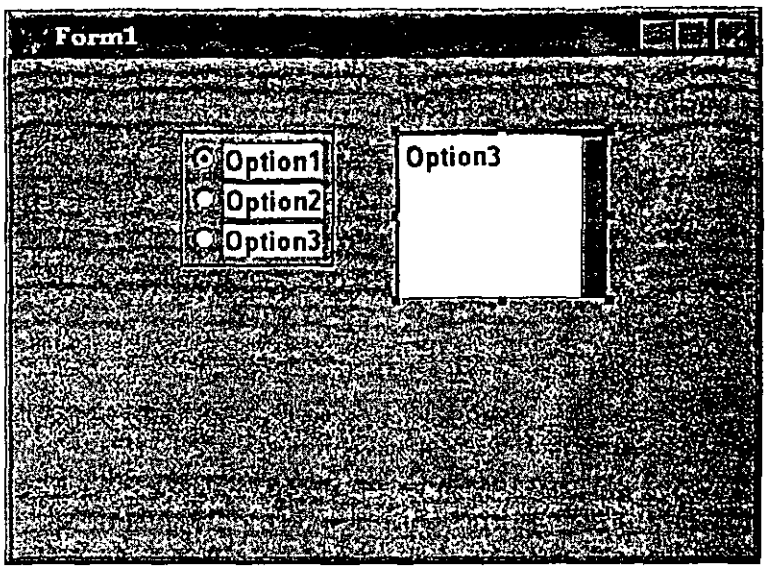
Introducción de datos vs. controles

Para un buen diseño de interfaz es importante recordar que los cuadros de texto no son la única forma dè introducir datos. Si usted arrastra desde la ventana Entorno de datos, siempre se crearán cuadros de texto, sin importar que el campo sea de carácter, numérico o lógico. Considere la utilidad y saque provecho de los controles de la interfaz de usuario que se han convertido en los estándares GUI actuales.

Por ejemplo, podría usar casillas de verificación para campos lógicos mientras que los numéricos podrían usar el control numérico. Esto proporciona al usuario un control de ratón para introducir datos, el cual puede ser mucho más eficiente. Los archivos .OCX estarán disponibles para muchos otros formularios de entrada de datos que pueden ligarse a campos de tablas. Por ejemplo, puede tener sentido ligar Controles deslizables y cuadrantes (tipo calculadora), a campos numéricos. Algunas veces los campos numéricos son utilizados para representar un número pequeño de opciones, por ejemplo:

1 = Extragrande
2 = Grande
3 = Mediano
4 = Pequeño

Figura 6-30



Ejemplo de grupo de opciones y cuadro combinado.

En este caso, un grupo de opciones o un cuadro combinado podrían ser mejores controles que un cuadro de texto, con código complejo que valide y traduzca los valores de texto a equivalentes numéricos (ver figura 6-30).

Es muy importante no exagerar. Los cuadros de lista o combinados que despliegan cientos de entradas son de utilidad cuestionable. Además, como en muchos casos la funcionalidad de los cuadros combinados y los grupos de opciones es intercambiable, el hecho de que los cuadros combinados utilicen mucho menos área de la pantalla debe tomarse en cuenta. El atractivo de los grupos de opciones en 3-D es mucho mayor, pero más de dos o tres grupos de opciones en un formulario crea confusión, mientras que cualquier cantidad de cuadros combinados dentro de un formulario son intuitivos y fáciles de usar.

Valide las entradas del usuario

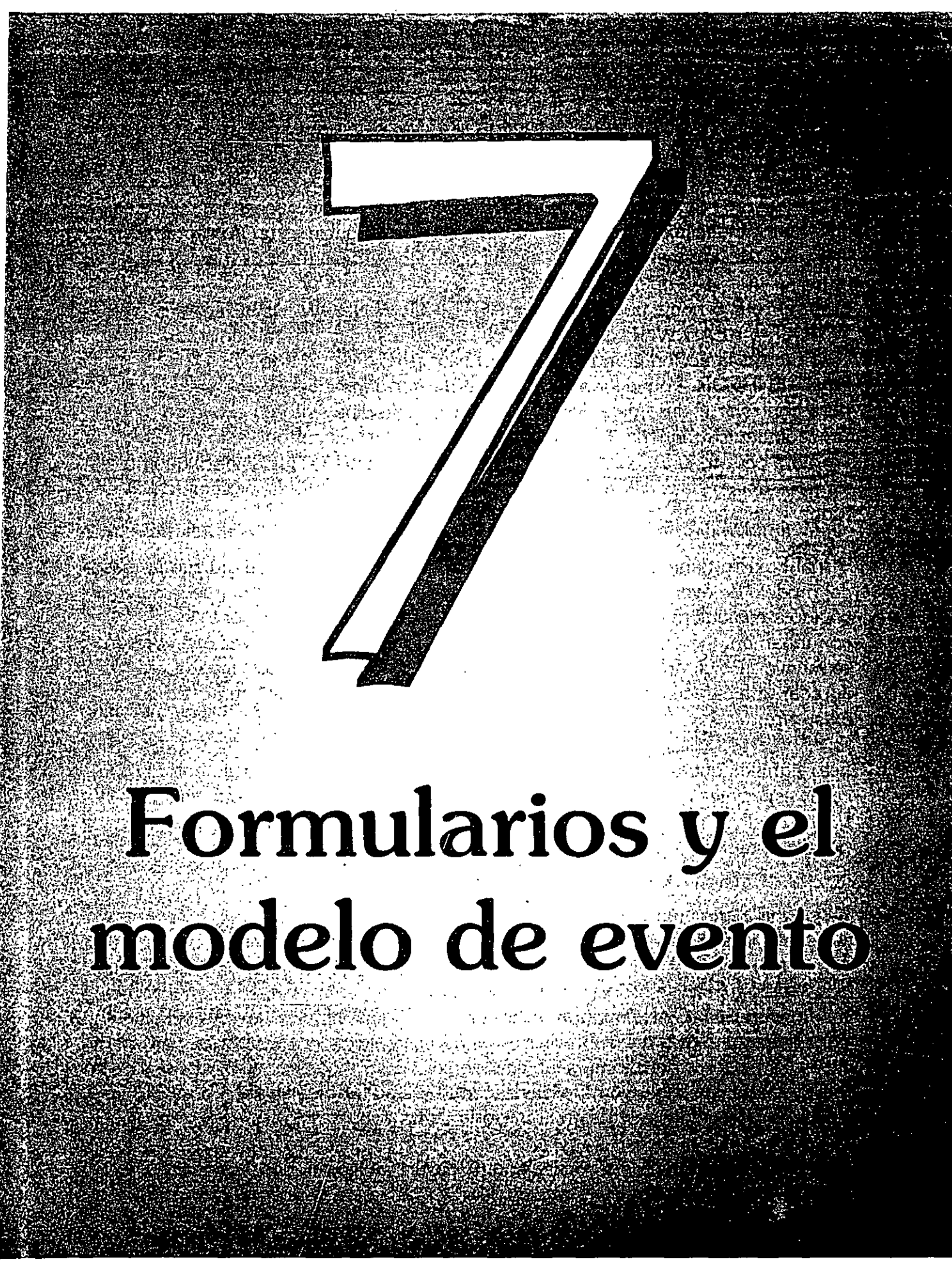
Para validar las entradas del usuario en cuadros de texto, o en cualquiera de los otros controles (aunque la mayoría de los otros controles tiende a validarse por sí misma debido a su naturaleza), utilice el evento Valid para desencadenar el código. Éste es equivalente a la cláusula Valid de FoxPro 2.6. Si la entrada del usuario es válida, regresa un valor falso (.F.), y el usuario recibirá un mensaje. Además, el punto de interés no abandonará al objeto.

Por ejemplo, si usted quiere asegurar que los usuarios introduzcan solamente Sí o No dentro del campo ListadeCorreo.Cliente puede usar algo tan simple como lo siguiente dentro de la ventana de código Valid:

```
IF NOT THIS.Value $ [SN]
   WAIT WINDOW [ Por favor introduzca Sí o No ] TIME 1
   RETURN .F.
ENDIF
```

Conclusión

En este capítulo usted dio un vistazo rápido a las herramientas utilizadas en el Generador de formularios. En el próximo capítulo verá con más detalle cómo utilizar estas herramientas para construir una interfaz realmente amigable para el usuario.



7

Formularios y el modelo de evento

CAPITULO 7

DE todas las mejoras dentro de Visual FoxPro, por mucho, la más importante y que llega más lejos es la implementación del modelo de evento de Windows. Los eventos ocasionan la ejecución del código asociado con dichos eventos de Windows. Un clic del ratón es un evento. Cerrar una ventana es un evento. Seleccionar un renglón de una lista desplegable es un evento. Hacer una selección del menú es un evento y hay muchos, pero muchos más.

En FoxPro 2.6 había pocos eventos: ACTIVATE WINDOW, DEACTIVATE WINDOW, READ WHEN, READ VALID y algunos más. Los mecanismos que controlaban la acción eran DO CASE...ENDCASE, DO WHILE...ENDDO y FOR...ENDFOR. Foundation Read, la *madre de todas las lecturas,* se reunió para controlar el desorden. Acercarse a un modelo de evento en el código de procedimientos requería de instrucciones CASE masivas en todo sitio en que fueran a ocurrir, junto con el código fallido (la condición OTHERWISE) que no siempre manejaba todas la excepciones posibles a pesar del mejor esfuerzo de todos.

Visual FoxPro tiene docenas de eventos. La Foundation Read, que formó las bases del modelo de eventos en FoxPro 2.6, se ha ido. En su lugar usted puede agregar código a cualquiera de los muchos eventos, mismos que son podados continuamente. Se puede ejecutar un fragmento de código cada vez que el ratón se mueve. ¡Por todos los cielos! De esta forma, usted queda posibilitado para agregar una característica nueva a un programa -*sin tocar siquiera el código existente.*

Esto significa que usted puede volver a usar código mucho más fácilmente. Los casos especiales están aislados, y no se incorporan al código. Así que aunque Visual FoxPro tiene una curva de aprendizaje significativa, ciertamente vale la pena el esfuerzo.

El enfoque en este libro

Si usted va a Francia, no tiene que aprender todas las palabras del idioma con el fin de sobrevivir. Como mínimo, necesita saber algunas frases clave. Desde luego, para hacer algo más que sobrevivir usted necesita aprender más frases, pero no todas las palabras del idioma. Aun cuando conociera todas las palabras del idioma y la sintaxis para saber cómo ponerlas juntas, usted terminaría utilizando sólo un pequeño porcentaje de ellas en la mayoría de las conversaciones.

Visual FoxPro contiene un vocabulario de cientos de comandos, funciones, objetos, eventos, métodos y propiedades. Aun así, para el 95% de las necesidades

72

de programación, sólo necesitará conocer algunos de ellos. La documentación que viene con el producto es exhaustiva, pero es material de referencia. Este capítulo es como un libro de frases de Visual FoxPro.

Eventos, métodos y propiedades

El *modelo de evento* significa usar una lista predeterminada de eventos o interrupciones, a los cuales usted puede pegar su código. Cuando una interrupción ocurre, el código que usted le agregó se ejecuta.

El *modelo de objeto* está basado en pegar propiedades y código a objetos. Un campo de entrada es un ejemplo de un objeto, y puede contener muchos pequeños compartimientos, con parámetros de color, conductas y código fuente almacenado dentro de ellos. Copie el objeto y copiará el contenido de todos los compartimientos. Cambie la propiedad Color y el objeto cambiará el color. Invoque el método Refresh de campo y éste se volverá a desplegar.

Los métodos son como funciones y las propiedades son como variables. Cada tipo de objeto dentro de Visual FoxPro tiene su propia lista de propiedades, eventos y métodos. Los cuadros de texto no tienen las mismas propiedades, eventos y métodos que las cuadrículas, pero cambiar la propiedad BackColor de cualquier objeto tiene exactamente el mismo efecto. Una vez que usted sabe que una propiedad trabaja para un objeto, sabe cómo usarla en cualquier otro.

Los métodos son funciones integradas. Cada tipo de objeto tiene su propio juego de métodos. Para ejecutar uno, teclee su nombre completo, por ejemplo, Form1.Grid1.Refresh (Refresh vuelve a desplegar la cuadrícula). Puede parecer poca cosa, pero es mejor que escribirlo usted mismo.

Ya que los métodos están pegados a objetos específicos en formas específicas, usted no tiene que decirle a su programa todo lo que usted quiere que haga debido a que Visual FoxPro sabe, por ejemplo, cómo cambiar el color de fondo de un botón de opción. Usted indica que se cambie la propiedad BackColor, y Visual FoxPro sabe qué hacer. Cambiar la propiedad del color de fondo ejecuta un programa interno para cambiar el color. Para volver a desplegar todos los campos en un formulario una vez que los datos subrayados han cambiado, utilice Form1.Refresh. Visual FoxPro sabe qué hacer para volver a desplegar un formulario.

Así, algunas veces usted cambia una propiedad y otras invoca un método. En cualquier caso, es una línea de código en lugar de 30. Usted hace mucho menos trabajo.

Los eventos son diferentes en ambos aspectos. El juego de eventos reconocidos por Visual FoxPro es fijo; usted no puede agregar los suyos. Imagínese el modelo de evento como la interfaz de Visual FoxPro para el sistema operativo. Usted no puede cambiar el sistema operativo, y no puede cambiar el modelo de evento.

Además, usted puede crear tanto propiedades como métodos personalizados. Éstos se agregan a formularios, clases y objetos, exactamente de la misma manera que las propiedades y métodos de Visual FoxPro. Sus propiedades no cambiarán la apariencia de un objeto o función en el momento que usted cambie el objeto, como las propiedades de objetos de FoxPro; en ese sentido, sus propiedades son más parecidas a variables locales. Pero sus métodos trabajan justo como los métodos de Visual FoxPro. Usted los nombra y éstos se ejecutan.

Modifique eventos, métodos y propiedades

Usted puede alterar las propiedades, métodos y eventos de todos los objetos en diferentes formas. La que usted decida utilizar depende tanto de la situación como del gusto personal. También puede agregar nuevas propiedades y métodos a formularios o clases.

Agregue nuevos métodos

Para agregar un método nuevo, usted debe estar en el Generador de formularios o de clases. Del menú Formulario o Clase, seleccione Nuevo método, luego escriba el nombre del método. En este punto puede agregar código al método; tiene que abrir la ventana Editar código, en alguna de las siguientes tres formas:

- Haga clic en Ver, Código del menú.
- Haga doble clic en cualquier parte dentro del fondo del formulario.
- Utilice el botón secundario del ratón para desplegar el menú contextual, luego haga clic en Código. Usted verá la pantalla mostrada en la figura 7-1.

Haga clic en el cuadro combinado de la izquierda para seleccionar el nombre del formulario o de la clase, el cual será el primer objeto en la lista; luego utilice el cuadro combinado de la derecha para seleccionar el método que desee editar. Verá tres tipos de entradas:

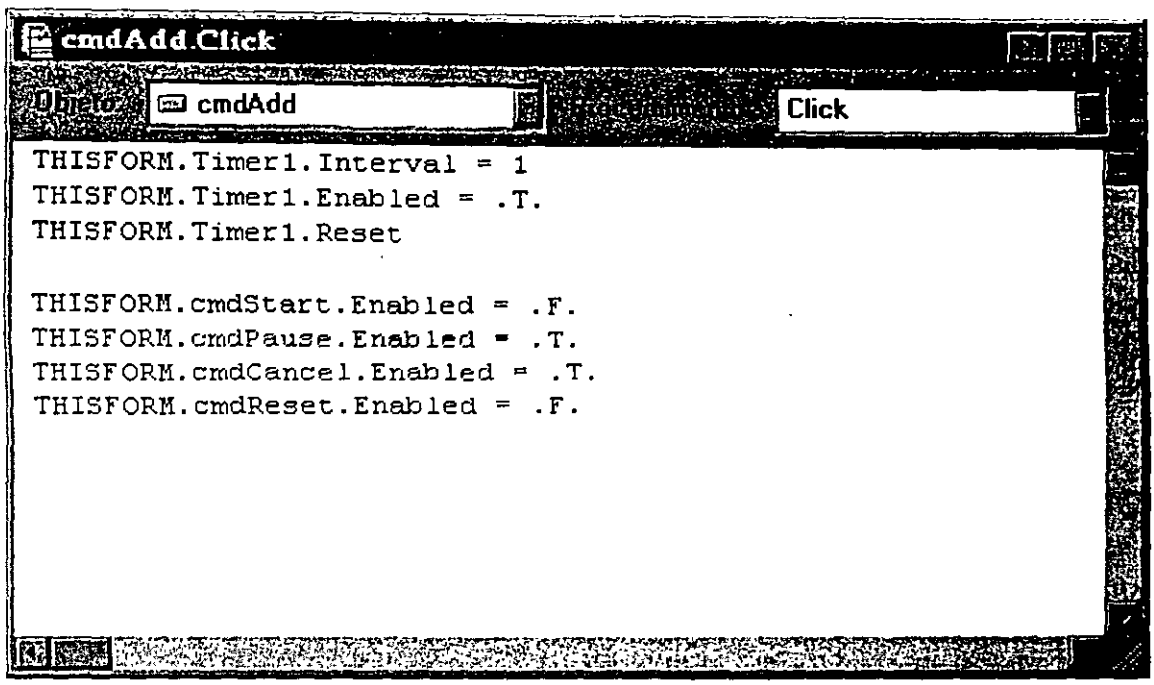


Figura 7-1

Ventana de edición de código.

- ➢ Métodos no modificados de Visual FoxPro en tipo de letra normal.
- ➢ Métodos modificados de Visual FoxPro en tipo de letra negrita.
- ➢ Los nombres de sus métodos.

Así que sus métodos se listan junto con los de Visual FoxPro. Estos también funcionan de la misma forma. Para llamar un método de Visual FoxPro, por ejemplo Show, utilice FORM1.Show; para llamar la función Reload para actualizar las pantallas con cuadros de lista basados en algunos parámetros de ambiente, utilice FORM1.Reload.

Agregue propiedades nuevas

Para agregar nuevas propiedades a un formulario, usted tiene que hacer casi lo mismo. Despliegue el menú Formulario o Clase, haga clic en Nueva propiedad y escriba su nombre. Luego haga clic en el icono Propiedades en la barra de herramientas Generador de formularios (o seleccione Ver, Propiedades del menú) y haga clic en Otros. Ahí está su nueva propiedad. Usted puede incluso asignarle un valor inicial.

Haga doble clic para editar código

Cuando usted hace doble clic en un objeto (en modo de diseño), una ventana de código aparece con el primer procedimiento (alfabéticamente) del objeto que tiene código en él. Si ninguno de los procedimientos ha sido alterado todavía, entonces se lista el primer procedimiento. Por coincidencia, sucede que esto es el evento Click, el cual es el evento más común que contiene código. Para el formulario, el primer procedimiento es Activate, el cual generalmente también contiene algún código.

Ver, Propiedades

Hay tres formas para hacer que aparezca la ventana Propiedades:

- Hacer clic en Ver, Propiedades, del menú de Visual FoxPro.
- Hacer clic con el botón secundario del ratón, luego seleccionar Propiedades del menú contextual.
- Hacer clic en el icono Propiedades en la barra de herramientas Generador de formularios.

La ventana Propiedades aparece, apuntando al objeto que está seleccionado en ese momento. Desde aquí, usted puede seleccionar las fichas para editar Datos, Distribución y Otras propiedades, o modificar código en los procedimientos de evento o método. Para cambiar a otro objeto o al formulario mismo, haga clic en el cuadro combinado de la ventana Propiedades, que contiene los nombres de todas las propiedades del formulario en uso.

Para cambiar una propiedad, primero debe activar la hoja Propiedades y seleccionar la propiedad. Puede efectuar esto desplazándose por la lista hasta que la encuentre, pero CTRL-ALT más la primera letra de la propiedad, por lo general es más rápido.

El menú contextual del botón secundario del ratón

Si usted hace clic con el botón secundario del ratón, aparecerá un menú contextual, que le permite desarrollar algunas acciones seleccionadas en el objeto. Las opciones más comunes, tanto para formularios como para controles, son:

- Propiedades
- Generador
- Código
- Ayuda

La opción Propiedades despliega la ventana Propiedades con las propiedades y procedimientos del objeto. La opción Generador despliega el generador para el objeto seleccionado, si es que existe. La opción Código despliega la ventana Código, con el primer procedimiento (en orden alfabético) alterado desplegado. Si el procedimiento no ha sido alterado, entonces se depliega el primer procedimiento para el objeto.

La opción Ayuda despliega el archivo de ayuda para el objeto seleccionado. Algunas selecciones son diferentes según el tipo de objeto que usted haya seleccionado:

Formularios Ejecutar, que equivale a seleccionar Formulario, Ejecutar del menú (capítulo 6); Pegar, que pegará lo que haya en el portapapeles dentro del formulario, incluyendo un objeto cortado o copiado, y Entorno de datos, el cual es equivalente a seleccionar Ver, Entorno de datos, del menú (consulte el capítulo 6).

Controles Cortar, que mueve el objeto al portapapeles y lo borra, y Copiar, que copia un objeto dentro del portapapeles, pero no lo borra.

Controles OLE Cualesquiera verbos registrados que sean para el objeto, los cuales generalmente incluyen Modificar y Abrir, y en ocasiones Imprimir.

Eventos que usted necesita conocer

Los primeros tres eventos de formularios son Load, Init y Activate. Load es primero y crea cualquier cosa utilizada por otros objetos de formularios, por ejemplo, para código SQL en el cuadro combinado RowSources en el formulario.

El evento Init ocurre cuando se crea el formulario. Este evento ocurre después de que todos los objetos dentro del formulario ya han sido creados (y su código Init ejecutado), por lo tanto el código Init del formulario se puede referir a objetos en el formulario. El evento Init puede regresar falso (.F.), cancelando la creación del formulario. Si esto sucede, entonces el evento Delete no se desencadena.

82

Por ejemplo, el siguiente código en el evento Init de un formulario ejecutará el formulario sólo si VentCliPrin está activo en ese momento. VentCliPrin es el nombre (propiedad) del formulario que usted quiere ver en ejecución antes de que el nuevo formulario se ejecute:

```
IF .NOT. WEXIST("VentCliPrin")
   RETURN .F.
ENDIF
```

El evento Init también es el lugar para iniciar las variables y arreglos (matrices) que tendrán el mismo tiempo de vida que el formulario. El evento Activate, por otro lado, no sabe, ni le importa, cuánto tiempo está el formulario en existencia. Es llamado si un formulario pierde el enfoque y luego lo vuelve a adquirir. También se desencadena cada vez que se llama el método Refresh del formulario. El evento Init es llamado una vez, y sólo una vez, en el tiempo de vida de un formulario -al principio. Todos los eventos Activate, incluyendo el inicial, ocurren después del evento Init.

El evento Activate es llamado cada vez que el formulario se convierte en la ventana activa. Por ejemplo, si usted tiene múltiples formularios en la pantalla, el código Activate de cada formulario debe incluir una línea para seleccionar la tabla primaria del formulario, de esta forma SKIP o BROWSE no harán cosas extrañas.

El evento Destroy es análogo a Init. Es llamado como el evento final del formulario. Sin embargo, a diferencia del evento Init, usted no puede prevenir la destrucción del formulario regresando falso (.F.) desde el evento Destroy. Sólo es una oportunidad para liberar a las variables y hacer limpieza después de que un formulario está en su camino de salida.

El evento Deactivate ocurre si el formulario pierde el enfoque o foco. Es llamado después del evento LostFocus del control que tiene el foco en el formulario. Esto sucede, por ejemplo, cuando el usuario hace clic en otro formulario o ventana.

Indicaciones de cuándo está activo un formulario

Aquí hay algunas formas para darle al usuario pistas visuales de cuándo está activo o inactivo un formulario. El siguiente código debe ir en la ventana de código del evento Deactivate del formulario:

```
THISFORM.Caption = "Ventana de clientes inactiva"
```

Este código va en la ventana de código del evento Activate del formulario:

```
THISFORM.Caption = "Clientes"
```

Las dos líneas de código cambian la barra de título del formulario a Ventana de clientes inactiva, cuando los usuarios hacen clic en cualquier otra ventana, y restablecen Clientes como el título cuando los usuarios reactivan el formulario al hacer clic dentro de él. El siguiente código va en la ventana de código del evento Deactivate del formulario:

```
THISFORM.BackColor = RGB(255,255,255)  && Blanco
THISFORM.Refresh
```

El siguiente código va en la ventana de código del evento Activate del formulario:

```
THISFORM.BackColor = RGB(192,192,192)  && Gris claro
THISFORM.Refresh
```

Esto cambia el color de fondo del formulario a blanco cuando está inactivo, y regresa a gris claro (el color de fondo asumido como normal) cuando se reactiva. El código siguiente va en la ventana de código del evento Deactivate del formulario:

```
THISFORM.WindowState = 1  && Minimizado
```

Esto ocasiona que el formulario sea minimizado cuando se hace clic en cualquier otra ventana. Se desplegará hacia la parte inferior de la pantalla como un icono y será restablecido otra vez a su tamaño normal cuando se haga doble clic sobre él. Si esto se hace sistemáticamente en cada formulario dentro de la aplicación, ocasionará una acumulación constante de ventanas disponibles que serán representadas por iconos en la parte inferior de la pantalla, y automáticamente seguirá la regla de que sólo una pantalla puede ser maximizada a la vez.

Métodos que usted necesita conocer

Load, Init, Activate y Deactivate son eventos; ejecutan su código cuando ciertas condiciones los desencadenan. Por otro lado, los métodos son funciones que usted llama en su código. Usted obtendrá el máximo uso de los siguientes cuatro métodos:

- Hide
- Refresh
- SetAll
- Release

El método Hide establece la propiedad Visible del formulario a falso (.F.). Note que el conjunto de formularios y todos los controles dentro del formulario retienen sus propiedades Visible. El método Show establece la propiedad Visible de un formulario a verdadero (.T.) y refresca el formulario. Usted también puede utilizarlo para establecer la modalidad del formulario. Si el parámetro de la modalidad es omitido, se utiliza el valor de la propiedad WindowType.

Mientras un formulario está invisible (u oculto), todavía está activo y aún recibe eventos, y puede aceptar métodos. Si las propiedades del o los controles en él formulario se cambian, cuando el formulario se vuelva visible otra vez se desplegarán sus nuevos valores.

Un uso muy común del método Hide (y la propiedad Visible) es cuando cerrar un formulario se convierte en una operación recurrente y un grupo de formularios necesita ser cerrado. En lugar de permitir al usuario ver el retraso al cerrar ventanas debido al procedimiento, todas las ventanas pueden ocultarse de una sola vez, entonces el proceso y el cierre del formulario pueden ocurrir mientras están ocultas. El retraso continúa, pero no es tan obvio como cuando diez ventanas desaparecen de la pantalla una por vez, cada medio segundo. Es mucho más atractivo ocultar todos los formularios y colocar una ventana Wait, un control de llenado o un cuadro de diálogo explicando el breve retardo, el cual desaparece cuando el retraso ha terminado. El siguiente código demuestra esta técnica:

```
THISFORMSET.SetAll('Visible', .F.)
WAIT WINDOW "Cerrando tablas. Por favor espere..." NOWAIT
* Código de limpieza que consume mucho tiempo
WAIT CLEAR
```

Note que el método SetAll (descrito posteriormente) es equivalente a programar un Hide en cada formulario dentro del conjunto de formularios. Si usted sólo quiere incluir unos cuantos formularios, ejecute el método Hide o establezca la propiedad Visible de cada uno de los formularios individualmente.

Usted puede utilizar el método Refresh para actualizar el contenido (el control fuente) de todos los controles en un formulario. Note que los objetos individuales también tienen métodos Refresh. Casi siempre que se cambia la apariencia de un

85

formulario, usted debe incluir un Refresh con el fin de que el cambio tenga efecto precisamente en la pantalla. Cuando un formulario se refresca, cada objeto en el formulario también se refresca, así que este proceso no tiene que ser explícitamente desarrollado en cada control del formulario.

Utilice el método Release para liberar el formulario de la memoria. Esto es útil si no fueron usadas variables de memoria para referirse al objeto formulario. Así que usted puede incluir un ThisForm.Release o un _Screen.ActiveForm.Release sin una variable de objeto que haga referencia al formulario, con el fin de usar el comando Release MiForma.

Utilice el método SetAll para establecer una propiedad de todos los objetos o de algunos objetos seleccionados dentro de un contenedor a un valor particular. Por ejemplo, usted puede establecer el color de fondo de todos los formularios con BackColor en un conjunto de formularios o todos los botones de opción en un grupo de opciones una sola vez. El ejemplo previo de los eventos Activate y Deactivate se modificó para mostrar cómo es útil esto. El siguiente código va en la ventana de código del evento Deactivate de un formulario:

```
THISFORM.BackColor = RGB(255,255,255)  && Blanco
THISFORM.SetAll('BackColor', RGB(255,255,255))
THISFORM.Refresh
```

y este código va en la ventana de código del evento Activate de un formulario:

```
THISFORM.BackColor = RGB(192,192,192)  && Gris claro
THISFORM.SetAll('BackColor', RGB(192,192,192))
THISFORM.Refresh
```

Esto cambia el color de fondo del formulario y de todos los objetos que están dentro de él a blanco, cuando está inactivo, y regresa a gris claro (el color de fondo asumido como normal) cuando se reactiva. Si usted ejecutó el ejemplo cuando estaba sin el método SetAll, vio el color de fondo diferente, pero se mantuvo el mismo color de fondo de todos los objetos dentro del formulario.

Usted puede hacer un refinamiento que vaya más allá con el método SetAll, que está basado en el hecho de que los cuadros de texto por lo general tienen un fondo blanco, sin importar el color de fondo del formulario. Por lo tanto, si usted cambia el fondo de todos los objetos dentro del formulario a gris claro, habrá cambiado el fondo de los cuadros de texto a gris claro, lo cual es incorrecto.

Deje el código Deactivate como aparece y modifique el código Activate como sigue:

```
THISFORM.BackColor = RGB(192,192,192)  && Gris claro
THISFORM.SetAll('BackColor', RGB(192,192,192))
THISFORM.SetAll('BackColor', RGB(192,192,192), 'TextBox')
THISFORM.Refresh
```

Esto limpia el aspecto y hace que los cuadros de texto mantengan el color de fondo en blanco, mientras que el color de fondo de todos los otros objetos es gris claro. Si va a cambiar constantemente el color de fondo de su formulario, es mejor establecer BackStyle para que sea transparente para todos los controles del formulario. Esto permite que el fondo del formulario sea el color de fondo de todos los otros objetos dentro del formulario. El ejemplo anterior es simplemente para mostrar el método SetAll.

El método SetAll es una forma muy sencilla de desactivar un grupo de controles dentro de un formulario:

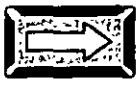
```
THISFORM.SetAll('Enabled',.F.)
```

de establecer todo un grupo de casillas de verificación a verdadero (.T.) de una sola vez:

```
THISFORM.SetAll('Value',.T., 'CheckBox')
```

o incluso de minimizar todos los formularios dentro de un conjunto de formularios:

```
THISFORMSET.SetAll('WindowState', 1)
```



Propiedades que usted necesita conocer

* Caption

Esta propiedad establece el texto del título del formulario. Si el formulario está minimizado, el título es el texto desplegado debajo del icono. Usted puede establecer esto en el momento del diseño y dejarlo sin cambios, o hacer que se modifique dinámicamente, dependiendo del estado del formulario. Por ejemplo, usted podría querer desplegar si el formulario está en el modo Modificar o en modo Ver y que sea parte del título. Algunas aplicaciones que se han visto utilizan la barra de título para dar instrucciones al usuario más notorias que si estuvieran en la barra de estado. Por ejemplo, usted podría querer que aparezca un formulario que simplemente tenga un control de cuadrícula, como una pantalla de

examinación personalizada. O podría querer dar a los usuarios instrucciones para que opriman Esc cuando hayan terminado de ver los datos con el fin de regresar a la pantalla principal. La barra de título puede ser un buen lugar para poner esto. Usted puede incluso utilizar el cronómetro y deslizar el texto, como un desfile.

* BackColor

Esta propiedad establece el color de fondo del formulario. Si la propiedad BackStyle es 0 (transparente), el estilo del fondo es ignorado. Si es 1 (opaco), entonces el color de fondo se establece al valor de BackColor tan pronto como éste sea establecido. Pero como es opaco, sobrescribe todos los controles del formulario. Para arreglar esto, ejecute el método Refresh del formulario inmediatamente después de cambiar el valor de BackColor. Esto se repetirá a través de los objetos del formulario, refrescándolos, y sobrescribiendo el color opaco de fondo. La propiedad Picture permite establecer un archivo .BMP para colocarlo como el color de fondo del formulario. Si lo establece, hace a un lado la propiedad BackColor, aun si BackStyle está establecido en opaco. Note también que todas las etiquetas y muchos controles deben establecerse a transparente para que el archivo de mapa de bits se muestre como fondo, a diferencia de establecer simplemente el color de fondo de todos los objetos en el mismo color. Opaco es el valor predeterminado para la mayoría de los objetos. La figura 7-2 es un ejemplo de un formulario con un archivo de mapa de bits utilizado como fondo.

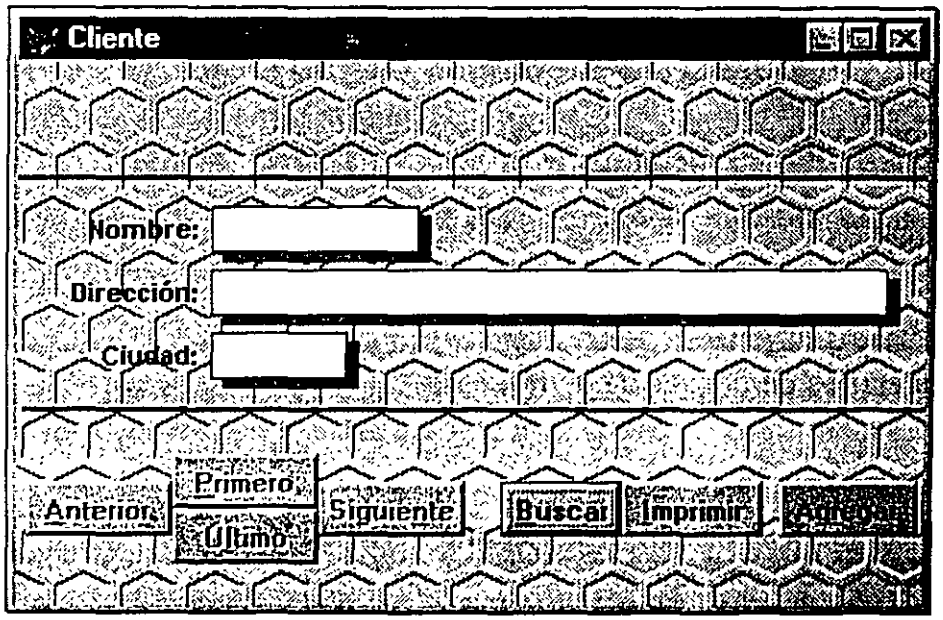


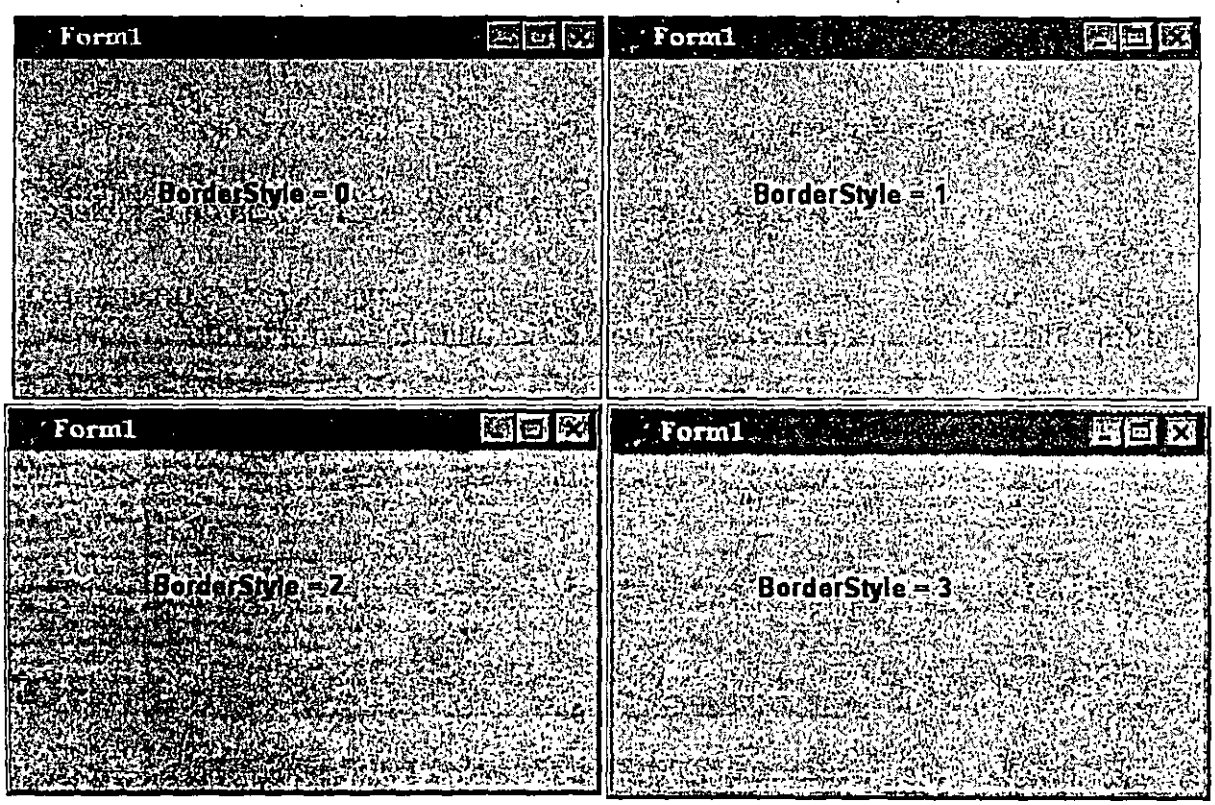
Figura 7-2

Un formulario con la propiedad Picture establecida con MARBLE.BMP.

* BorderStyle

Esta propiedad determina el estilo de los bordes alrededor del formulario, como se muestra en la figura 7-3. El valor predeterminado, 3, hace que la ventana se comporte como cualquier otra ventana de Windows, con un color y ancho de borde determinados por el Panel de control de Windows y actuando como un redimensionador para el formulario. Note que si la propiedad Resizable es falsa (.F.), entonces será el estilo del borde, pero la función de redimensionamiento estará inactiva. Establecer BorderStyle en 0 provoca que la ventana no tenga bordes.

Figura 7-3



Diferentes estilos de bordes de ventanas.

Esto significa que si se despliega la ventana sobre un fondo que es del mismo color que la ventana, no habrá líneas de división. La barra de título aún existirá, pero será la única pista visual para el usuario, así como el tamaño de la ventana. BorderStyle 1 y 2 son, respectivamente, ancho sencillo y doble de las líneas de bordes. Ninguno de estos dos permite que la ventana se pueda volver a dimensionar.

89

* Enabled

Esta propiedad determina si el formulario puede responder a eventos definidos por el usuario, tales como presionar teclas o hacer clics con el ratón. Si el formulario está desactivado, todos los controles dentro de él también están desactivados.

* Icon

Esta propiedad determina qué archivo de icono representa al formulario cuando éste está minimizado. Por omisión se usa el icono Visual FoxPro (ver figura 7-4), pero usted puede usar cualquier icono, incluyendo alguno que usted haya diseñado. El título (texto) desplegado debajo del icono se establece al utilizar la propiedad Caption del formulario. Usted puede implementar iconos animados utilizando el control de cronómetro y la propiedad Icon.

Figura 7-4

La ventana principal de Visual FoxPro con forma de icono.

* Show Tips

Esta propiedad determina si las herramientas de ayuda que se muestran son para objetos en el formulario cuando el ratón se detiene en ellas, como se muestra en la figura 7-5. Para formularios, están predeterminadas en falso (.F.). Si las herramientas de ayuda están encima y el ratón continúa un control por un periodo corto sin acción, aparecerá un pequeño cuadro amarillo con una descripción del control. Esta descripción está controlada por la propiedad ToolTipText del control.

Figura 7-5

Un control con una herramienta de ayuda desplegada.

* Sizable

Esta propiedad determina si el usuario puede redimensionar el formulario del todo. Si no, entonces las esquinas de redimensionamiento están inactivas y la opción Tamaño del cuadro del menú Control está desactivada. Si Sizable se establece en verdadero (.T.), y BorderStyle no es igual a 3, el formulario no podrá redimensionarse.

* MaxHeight, MaxWidth, MinHeight y MinWidth

Utilice estas propiedades para establecer límites en el tamaño de su formulario si es redimensionable, pero hay límites superiores e inferiores. Usted puede calcular estos tamaños basándose en el contenido de su formulario, ya sea en el tiempo de diseño o sobre la marcha.

Una vez que el formulario ha sido redimensionado, es muy factible que usted necesite redimensionar y mover algunos de los controles individuales dentro del formulario para quitar el lugar o reemplazar el espacio vacío creado por la modificación en las dimensiones del formulario. Usted puede simplificar esto ideando cuidadosamente el diseño del formulario, pero el mejor enfoque es no permitir que sus formularios sean redimensionables a menos que exista alguna buena razón.

* Visible

Esta propiedad determina si un formulario es o no visible. La propiedad FormSet.Visible no la toma en cuenta sólo si es falsa. Usted puede utilizar los métodos Hide y Show para cambiar esta propiedad, o puede cambiarla directamente. Si un formulario no es visible está aún activo, todavía recibe eventos del sistema, responde a llamadas de métodos y puede tener cambios en propiedades. Si su función se extiende más allá de su visibilidad (por ejemplo, hacer beep cada diez segundos), desplegará esta función.

* WindowState

Esta propiedad determina el estado de la ventana Visual FoxPro. Los estados válidos son 0 para normal, 1 para minimizada y 2 para maximizada. Esto puede cambiarse en el momento de la ejecución, como usted vio anteriormente en el ejemplo para el evento Deactivate en este capítulo. Usted puede programar el formulario a cualquier tamaño, pero recuerde volver a dar tamaño a los controles dentro del formulario si cambia entre los estados 0 (normal) y 2 (maximizado).

* WindowType

Esta propiedad determina el tipo de ventana que el formulario utilizará. Las ventanas pueden ser Modeless (sin modo), Modal, Read (lectura) o Read Modal. Usted debe usar ya sea ventanas sin modo o modales. Las dos últimas se incluyen para compatibilidad con productos anteriores. Una ventana modal simplemente quiere decir que el control no puede dejar la ventana sin que la ventana se haya ido. Por ejemplo, el cuadro de diálogo Archivo abrir/Guardar de cualquier aplicación es modal. Usted debe hacer ya sea una selección válida o cancelar antes de dejar la ventana. Incluso el menú no trabajará si usted está en una ventana

KeyPress Este evento ocurre cuando el formulario tiene el foco y una tecla se presiona y se suelta. Como se dijo previamente para el evento GetFocus, es raro que el formulario tenga el foco. Si usted quiere obstruir golpes de teclas en un nivel de formulario, utilice la propiedad KeyPreview del formulario. Si ésta es establecida, el formulario obtiene golpes de tecla antes de que repercutan en el control con el foco.

Load Ocurre justo después de que un formulario y cualquier otro objeto en el formulario han sido iniciados. El evento Init ocurre después de que los objetos del formulario se han iniciado. Para prevenir que el formulario sea creado, regrese falso (.F.) del evento Init. Regresar falso del evento Load no tiene efecto. Utilice Load para crear variables empleadas en código SQL en los cuadros de lista, y para administrar tablas usadas en el formulario si usted no quiere que el entorno de datos del formulario lo haga por usted.

LostFocus Este evento ocurre cuando el formulario pierde el foco. El formulario rara vez tendrá el foco, sin embargo, un formulario debe estar desprovisto de controles o no tener controles activos en ese momento con el fin de obtener el foco en primer lugar.

MouseDown Sucede cuando se hace clic con el botón del ratón y no se suelta en un área en blanco del formulario. El formulario no debe tener el foco para que este evento ocurra, pero si se hace clic con el botón del ratón sobre un objeto de un formulario, el objeto obtendrá el evento en lugar del formulario. No hay evento MousePreview (como lo hay para el teclado).

MouseMove Esto ocurre cada vez que la posición del ratón cambia en relación con la ventana. Note que esto puede pasar si la ventana se mueve por programación, aunque el ratón no se mueva. ¡Tenga cuidado al usar este evento! Si usted observa qué tan a menudo pasa este evento (se despliega un mensaje en la ventana cada vez que sucede), verá que este evento se desencadena continuamente. Sea muy cuidadoso con el código que coloque ahí, debido a que se ejecutará hasta 20 veces por segundo.

MouseUp Ocurre cuando el botón del ratón se suelta de una posición oprimida y el puntero del ratón está sobre un área en blanco del formulario. Note que este evento normalmente seguirá al evento MouseUp por una cantidad de tiempo insignificante.

Paint Este evento ocurre después de que un formulario se ha movido o ha cambiado su tamaño, y aparece una porción que anteriormente estaba cubierta.

Reposition Event Este evento ocurre siempre que se mueve un formulario, ya sea interactivamente o mediante programación.

Resize Event Ocurre siempre que se cambia de tamaño un formulario, ya sea interactivamente o por programación.

Unload Esto ocurre cuando el objeto es liberado de la memoria. Note que esto sucede después del evento Destroy. Utilice el evento Unload para regresar un valor de un formulario modelo, con la siguiente sintaxis:

```
DO FORM (nombre) TO RetVal
```

Otros métodos de formularios

AddObject Este método agrega un objeto al formulario en el tiempo de ejecución, desencadenando el evento Init del objeto agregado. Utilice el método RemoveObject para quitar objetos del formulario en el tiempo de ejecución.

Box Este método dibuja un rectángulo dentro del formulario, con un estilo determinado por las propiedades DrawWith, DrawMode y DrawStyle del formulario. El relleno puede ser controlado con las propiedades FillColor y FillStyle del formulario.

Circle Éste dibuja un círculo o arco dentro del formulario con un estilo determinado por las propiedades DrawWith y DrawStyle del formulario. Si la figura dibujada está coloreada, el color de relleno puede ser controlado con las propiedades FillColor y FillStyle del formulario.

Cls Este método borra todos los gráficos en tiempo de ejecución (como aquellos creadas por los métodos Box, Circle, Line y PSet) y restablece las propiedades CurrentX y CurrentY a 0. Los controles no son afectados por el método Cls.

Line El método Line dibuja una línea en el formulario. El estilo de la línea puede ser controlado con las propiedades DrawWith, DrawMode y DrawStyle. La inclinación de la línea puede controlarse con la propiedad LineSlant.

95

Move Este método mueve el formulario hacia una nueva posición en la pantalla. Las coordenadas dadas son respecto al origen (0,0).

Point Este método da el color RGB del punto específico en el formulario. Si el punto no está dentro del formulario, se obtiene el valor -1.

Print Éste imprime una cadena de caracteres en el formulario en las coordenadas CurrentX y CurrentY. Para saltar una línea, imprima el carácter CHR(13). La fuente del texto impreso puede ser controlada con las propiedades FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontOutline, FontShadow y FontSize.

PSet Establece un color específico en un punto dentro del formulario. El estilo del punto puede ser controlado por las propiedades DrawWith, DrawMode y DrawStyle.

ReadExpression Este método da la expresión almacenada a una propiedad en el formulario. Está disponible sólo en el tiempo de creación.

ReadMethod Este método da el texto de un método específico. Está disponible sólo en el tiempo de creación.

RemoveObject Utilice este método para quitar un objeto de un formulario en el tiempo de ejecución.

AddObject Emplee este método para agregar un nuevo objeto.

SaveAs Utilice este método para crear en el programa un formulario y guardarlo en un archivo .SCX.

SaveAsClass Utilice este método para crear en el programa un formulario y salvarlo en un archivo .VCX.

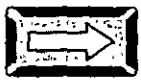
TextHeight Este método da el largo de una cadena de texto si ésta se imprimió utilizando las propiedades de fuentes en ese momento. Si los retornos de carro están dentro de la cadena, se regresa la altura acumulada de todas la líneas de la salida.

TextWidth Este método da el ancho de una cadena texto si ésta se imprimió utilizando las propiedades de fuentes en ese momento. Si los retornos de carro están dentro de la cadena, se regresa la longitud de la línea más larga de la salida.

WriteExpression Este método escribe una expresión en una propiedad al momento de creación.

WriteMethod Este método escribe texto en un método al momento de creación.

ZOrder Este método establece el orden gráfico de un objeto durante el tiempo de ejecución. No cambia el orden de los separadores de controles, ni puede colocar objetos gráficos (tales como cuadros, líneas y círculos) en el frente de los controles. Hay dos capas gráficas en un formulario: la capa para dibujar y frente a ella, la capa de objetos. Los objetos pueden ser ordenados sólo dentro de su capa específica.



Otras propiedades de formularios

AlwaysOnTop Esta propiedad determina si el formulario sigue colocado al principio de los otros formularios, aun cuando los otros formularios estén activos. La ventana de propiedades se comporta de esta manera si el botón de tachuela está oprimido. La única forma en la que un formulario puede ponerse encima de un formulario con esta propiedad es sólo si también se establece la propiedad AlwaysOnTop. Desde luego, cualquier otra aplicación Windows puede ponerse encima de cualquier aplicación Visual FoxPro que esté siendo ejecutada.

AutoCenter Esta propiedad determina si el formulario se centra a sí mismo automáticamente en la ventana Visual FoxPro la primera vez que se despliega. La propiedad puede ser establecida en el tiempo de creación en el programa con los procedimientos Load o Init.

BackStyle Determina si un color de fondo se utiliza para el formulario. Si BorderStyle = 0 el fondo del formulario es transparente; si BorderStyle = 1 el fondo del formulario es opaco.

Class Da la clase en la cual está basado el formulario. Es de sólo lectura en los tiempos de creación y de ejecución. Esta propiedad se utiliza junto con la siguiente.

ClassLibrary Esta propiedad da el nombre de archivo de la biblioteca que contiene la clase de formulario. Es para uso de usted cuando trabaja con un formulario, en el caso de que olvide cómo funcionan los objetos.

97

Closable Esta propiedad determina si el cuadro de menú Control en la esquina superior izquierda de la ventana le permite cerrar un formulario, ya sea al hacer doble clic en él o al seleccionar Cerrar en su menú.

Comment Simplemente almacena información acerca de un objeto. No utilice esta propiedad para almacenar datos usados en tiempo de ejecución. Para eso son las propiedades personalizadas.

ControlBox Determina si aparece el cuadro de menú Control en la esquina superior izquierda de la ventana.

CurrentX Esta propiedad es la posición horizontal en la cual el próximo objeto gráfico o texto empezarán a dibujarse. Cada método que dibuja un objeto gráfico actualiza los valores tanto de las propiedades CurrentX como de CurrentY.

CurrentY Esta propiedad es la posición vertical en la cual el próximo objeto gráfico o texto empezarán a dibujarse. Cada método que dibuja un objeto gráfico actualiza los valores tanto de las propiedades CurrentX como de CurrentY.

DeskTop Especifica si el formulario puede moverse hacia cualquier parte dentro del escritorio de Windows, o si está confinado a la ventana principal de Visual FoxPro.

DrawMode Determina cómo los objetos gráficos tales como cuadros, círculos, líneas y puntos son dibujados dentro del formulario. Hay 16 diferentes modos que determinan cómo toman lugar los dibujos. No se listarán aquí, pero basta decir que si usted va a dibujar mucho, necesitará estar extremadamente familiarizado con los 16.

DrawStyle Determina cómo las líneas de objetos gráficos tales como cuadros, círculos, líneas y puntos son dibujados dentro del formulario. Las líneas pueden ser sólidas, con guiones, punteadas, con puntos y guiones, sólida por dentro o transparente. Le dejo a su imaginación cómo es la última de éstas.

DrawWidth Determina el espesor de los puntos y el ancho de la línea en pixeles o de objetos gráficos tales como cuadros, círculos y líneas.

FillColor Determina el color con el que se pueden llenar formas cerradas que han sido dibujadas en la capa gráfica de los formularios, tales como cuadros y círculos.

98

FillStyle Determina el estilo del llenado para un objeto gráfico. Los estilos de llenado incluyen sólido, transparente y líneas que pueden ser horizontales, verticales, diagonales o cruzadas.

FontBold Determina si el texto impreso en el formulario es desplegado en negritas.

FontItalic Determina si el texto impreso en el formulario se despliega en itálicas.

FontName Determina la fuente utilizada para desplegar texto dibujado dentro del formulario.

FontSize Esta propiedad determina el tamaño de la fuente que se utiliza para desplegar texto que está dibujado dentro del formulario. Cuántos caracteres encajarán dentro de un campo de entrada es una función de las propiedades FontName y FontSize. Usted debe decidir acerca de éstas antes de crear su formulario.

FontStrikeThru Esta propiedad determina si el texto que se imprime en el formulario es desplegado con una línea a través de él.

FontUnderline Esta propiedad determina si el texto que se imprime en el formulario va subrayado.

ForeColor Determina el color del primer plano utilizado para desplegar objetos en el formulario.

HalfHeightCaption Determina si una barra de título de la mitad del tamaño normal, con el tamaño de la fuente ajustado, es utilizada como la barra de título del formulario.

Height Determina la altura de la ventana del formulario.

HelpContextID Determina qué contexto ID de ayuda utiliza FoxPro para proporcionar al usuario ayuda sensible al contexto. El archivo de ayuda está determinado por los parámetros del comando Set Help To.

KeyPreview Esta propiedad permite al formulario interceptar golpes de tecla antes de que vayan a los controles dentro del formulario. Esto permite programar eventos de golpe de tecla a nivel de formulario sin tener que agregar código a cada control.

Left Esta propiedad determina la columna más alejada hacia la izquierda del formulario en relación con la pantalla de Visual FoxPro, o las coordenadas absolutas si está en el escritorio de Windows.

MaxButton Determina si el formulario tiene un botón para maximizar en el extremo derecho de la barra de título, y si la opción está disponible en el cuadro de menú Control. Esto no es aconsejable si usted no ha codificado su pantalla para poder cambiarla de tamaño; habrá simplemente mucho espacio vacío en el formulario si el usuario lo maximiza y éste ha sido diseñado para un tamaño específico.

MaxLeft Esto establece una distancia máxima a la que el formulario puede moverse desde el lado izquierdo de la ventana principal de Visual FoxPro.

MaxTop Establece una distancia máxima a la que el formulario puede moverse desde el extremo superior de la ventana principal de Visual FoxPro.

MDIForm Determina si el formulario actúa como un formulario estándar de Windows con interfaz para múltiples documentos, o como una ventana estándar de Visual FoxPro. En tiempo de ejecución es de sólo lectura.

MinButton Determina si el formulario tiene un botón para minimizar en la esquina superior derecha de la barra de título, y si la opción está disponible en el cuadro de menú Control. Si usted establece esto en verdadero (.T.) es aconsejable establecer un icono para el formulario o éste utilizará por omisión el icono de Visual FoxPro.

MousePointer Determina la forma del icono mientras está sobre el formulario. Los valores posibles incluyen la forma predeterminada (una flecha), una cruz, una letra I, flechas para cambiar el tamaño apuntando hacia todas las direcciones posibles, o un reloj de arena (para esperar algo) y un "no colocar." El puntero no colocar es un círculo con un corte a través de él. Yo creo que es un muy buen detalle cambiar el puntero del ratón a un reloj de arena si alguna acción puede tomar algo de tiempo.

Movable Esta propiedad determina si el usuario puede mover un formulario en el tiempo de ejecución al hacer clic en la barra de título y arrastrándolo.

Name Esta propiedad especifica el nombre del objeto a nivel de código. Si usted se refiere a un objeto con el fin de cambiar su nombre y su nivel más alto de objeto, entonces debe usar una variable de objeto para referirse a él.

ParentClass Esta propiedad da el nombre de la clase relacionada en la cual está basado el formulario. Es de sólo lectura tanto en el tiempo de creación como en el de ejecución.

ScaleMode Determina las unidades de medición para los métodos para dibujar gráficos de los formularios. Para foxeles, establezca ScaleMode en 0. Para pixeles, establézcala en 3. A menos que usted tenga una muy buena razón para no hacerlo, utilice pixeles.

TabIndex Determina el orden que ocupa el formulario en el conjunto de formularios. Si se cambia este parámetro en el tiempo de ejecución, usted debe tener cuidado de establecer los formularios en el conjunto de formularios.

TabStop Determina si la tecla TAB lleva al usuario de campo en campo dentro del formulario.

Top Esta propiedad determina la fila más alta del formulario en relación con la pantalla de Visual FoxPro, o las coordenadas absolutas si está dentro del escritorio de Windows.

Width Esta propiedad determina la columna más alejada hacia la izquierda del formulario en relación con la pantalla de Visual FoxPro, o las coordenadas absolutas si está dentro del escritorio de Windows.

Window Esta propiedad determina si el formulario está contenido dentro de otro formulario. Si es así, el nombre del formulario contenedor será el valor de la propiedad Window.

Controle eventos con Read Events

Recuerde el programa principal en el capítulo 3, el cual fue "activado" por el comando Read Events. Tan pronto como usted programe un Read Events, FoxPro toma control, pasando los eventos de los formularios y controles correctos y ejecutando el código editado en los eventos. Finalmente, Clear Events detiene todos los procesamientos de eventos que se iniciaron con Read Events, pero mantiene abierto el formulario.

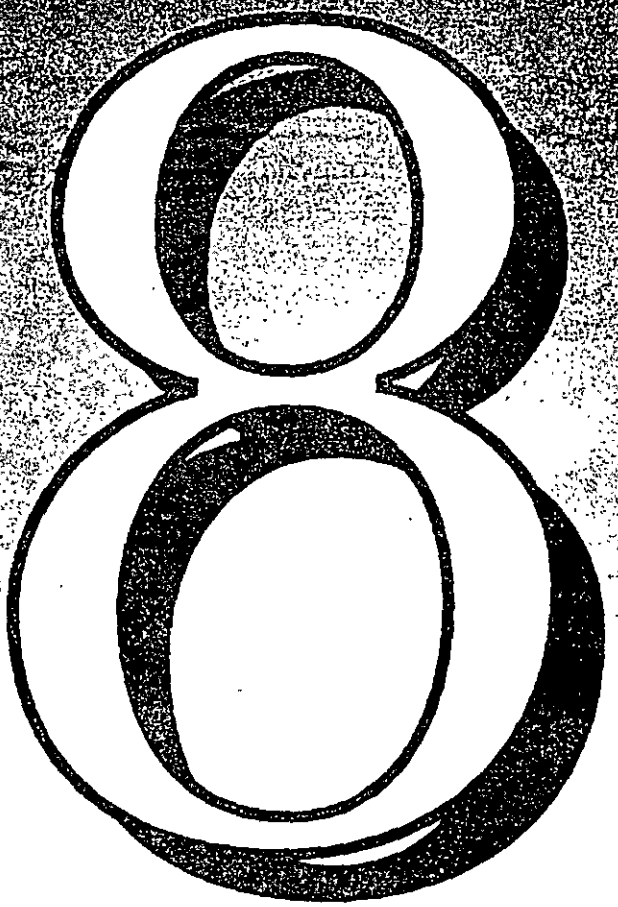
Los formularios inician y terminan automáticamente el procesamiento de eventos. La manera predilecta de terminar un procesamiento y cerrar un formulario es liberando el formulario, ya sea con el comando Release (por ejemplo, Release

ThisForm) o mediante el método Release del objeto formulario. Esto libera el formulario, pero mantiene el ciclo del evento activo para todos los formularios que estén activos en ese momento.

Usted podría incluir una barra de menús con el título Salir, con un comando CLEAR EVENTS. Cada formulario tiene un botón Salir cuyo evento Click consiste en el sencillo comando THISFORM.Release.

Resumen

La enorme cantidad de nuevas propiedades y métodos dentro de Visual FoxPro es intimidante, pero no deje que nadie le diga que necesita conocerlos todos. Usted puede hacer software muy impresionante utilizando sólo un subconjunto de las nuevas características de Visual FoxPro.



El Generador de menús

CAPITULO 8

TODOS hemos admirado el sistema de menús de Windows. Una vez que los usuarios se han familiarizado con él, les parece una manera sencilla e intuitiva de desplazarse por su aplicación. Funciona como una especie de sistema de ayuda para usuarios nuevos, quienes instintivamente presionan F10 y despliegan el menú para ver qué contiene. Los usuarios avanzados ni siquiera ven los menús -sus dedos recuerdan que ALT-C, B es el Balance de clientes. Y las teclas de acceso rápido que pueden adjuntarse a las opciones de menú o submenús reducen al mínimo el número de golpes requerido para ejecutar funciones que se usan frecuentemente.

Panorama general

A diferencia de cualquier otra pieza de Visual FoxPro, el Generador de menús genera un código que se utiliza en el formulario generado como parte de su aplicación. El menú que usted crea se almacena en un par de archivos con las extensiones .MNX y .MNT, donde el archivo .MNT almacena texto de los campos memo en el archivo .MNX. Me avergüenza admitir que cuando vi por primera vez la versión 1 de FoxPro, intenté escribir mi propio código de menú utilizando los muchos comandos de menú descritos en el manual de referencia del lenguaje. Me llevó un buen tiempo comprender que uno no tiene que escribir el código de menú, éste se genera por GenMenu, y luego se compila justo en su aplicación. Muchos de los comandos relacionados con clases en Visual FoxPro son similares; aunque usted puede crear clases con código, es mucho más fácil utilizar el Generador de clases.

Los menús consisten en menús principales y menús desplegables. Los menús desplegables están formados por opciones de menú. Mientras un menú desplegable está activo, usted puede presionar la primera letra de cualquier opción de menú para seleccionarlo, a menos que una tecla de acceso (una letra subrayada) haya sido definida utilizando la cadena \< inmediatamente antes de la tecla de acceso en la opción de menú. Además, si ha sido definido un método abreviado, éste aparecerá a la derecha de la opción de menú, y puede presionarse aun cuando el menú no esté desplegado o activo.

La manera usual de activar un menú en su código de aplicación es incluir una línea como ésta:

```
DO MENU.MPR
```

e· ' programa Principal. FoxPro reemplazará el menú del sistema de Visual
I ro con su código. Cuando desee mostrar su menú, incluya cualquiera de los dos comandos siguientes:

```
SET SYSMENU TO DEFAULT
POP MENU _MSYSMENU
```

El último solamente es aplicable si, antes de instalar su menú, usted incluyó el comando:

```
PUSH MENU _MSYSMENU
```

El comando Push Menu va en el código Setup del menú, como comprobará en un momento. El código Setup se ejecuta justo antes de que corran los comandos de su menú. Push y Pop son como operadores de pila; Push hace una copia del menú y Pop regresa el último elemento copiado, en la modalidad LIFO (*Last In First Out*: último dentro, primero fuera).

¿Cuál debe usar? Si sólo tiene un menú, simplemente utilice Set SysMenu To Default como parte de su rutina de Salir, y no se moleste en especificar Push Menu _MSysMenu antes de instalar su menú.

Una de las muchas características agradables de los menús de Visual FoxPro es el hecho de que usted puede habilitar y deshabilitar opciones individuales o menús principales con facilidad. El efecto Set Skip del menú principal es constantemente evaluado, por lo que si usted adjunta, por ejemplo, la condición:

```
SKIP FOR Editando
```

a un menú principal, en el instante en que la variable de memoria Editando acepta un valor de .T., el menú principal correspondiente se minimiza y es inhabilitado. Usted puede sacar provecho de esta característica para crear en sus aplicaciones una extrema facilidad de uso para el usuario. También puede utilizar Skip For para crear una versión demo de su software que disminuya unas cuantas opciones importantes del menú hasta, y a menos, que sus presuntos clientes registren su software, en cuyo caso usted les envía una clave de activación que abre las opciones inhabilitadas del menú.

Los menús no son sólo para los usuarios. Cada vez que trabajo en una aplicación, yo creo un pequeño menú principal que se instala a la derecha de mi menú de Visual FoxPro, proporcionándome métodos abreviados para la media docena de funciones más comunes que tengo que realizar durante una jornada típica de trabajo.

Finalmente, los menús son el mecanismo que dirige las aplicaciones. Cuando los usuarios quieren hacer algo, lo encuentran en el menú, y entonces, o bien hacen clic con el ratón o presionan ENTER. En versiones anteriores de FoxPro, el poner varias ventanas en la pantalla era una molestia considerable. Con Visual FoxPro, se vuelve trivial. Literalmente no se requiere de una programación adicional para escribir aplicaciones de pantalla múltiple. Si una opción está en el menú, los usuarios pueden seleccionarla y abrir su ventana sin cerrar otras ventanas en la pantalla. Probablemente algunas aplicaciones no deberían operar de esta forma, pero puede resultar muy conveniente.

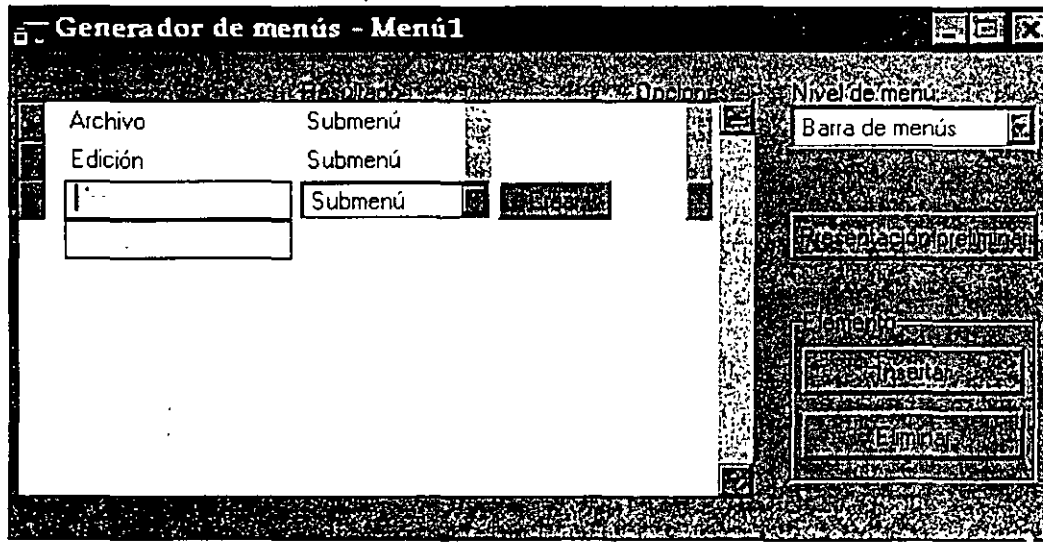
Ahora se verá el Generador de menús y lo que puede hacer por usted.

Active el Generador de menús

Como es usual, hay varias maneras de empezar a generar un menú. Usted puede:

- Presionar CTRL-N (para Nuevo) y luego elegir Menú.
- Hacer clic en Archivo, Nuevo y elegir Menú.
- Escribir MODIFY MENU nombre en la ventana Comandos.
- En el Administrador de proyectos, resaltar Menú (bajo la ficha Otros) y luego hacer clic en Nuevo.

Figura 8-1



Pantalla del Generador de menús.

➢ En el Administrador de proyectos, resaltar Menú (bajo Otros), presionar F10 y seleccionar Proyecto, Nuevo archivo del menú desplegable Proyectos. Verá la pantalla del Generador de menús (figura 8-1).

Una vez que el Generador de menús aparece en la pantalla, se realizan dos cambios en el menú de Visual FoxPro: el menú Ver contiene dos opciones nuevas: Opciones generales y Opciones de menú, y aparece un nuevo menú principal llamado Menú.

Cambios al menú Ver

Las dos opciones nuevas del menú Ver le permiten crear fragmentos de código y fijar parámetros para generar el menú. La figura 8-2 muestra el cuadro de diálogo Opciones generales del menú. Éste tiene tres áreas de interés:

* Ventana de procedimientos

Cualquier cosa que se introduce aquí (o en la versión expandida de la misma ventana de código que aparece si hace clic en Editar) se genera después de una instrucción que se lee:

```
! SELECTION MENU _MSYSMENU aquí su código
```

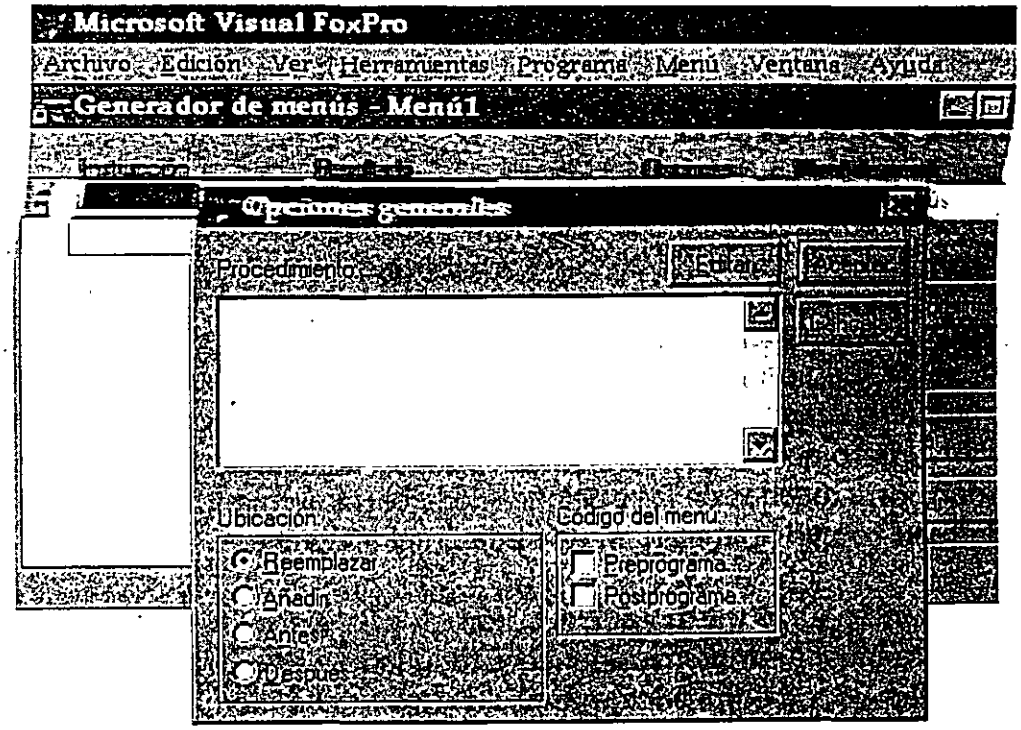


Figura 8-2

Cuadro de diálogo Opciones generales.

* Dónde agregar el menú

Puede utilizar un menú para obtener el nombre del menú principal de FoxPro como un punto de referencia; las selecciones Antes o Después se refieren al menú principal que usted eligió. Por ejemplo, para agregar un menú Utilerías a la derecha del menú de Visual FoxPro utilice Después, Ayuda. Reemplazar y Añadir se refieren a todo el menú de Visual FoxPro; por ejemplo, si se verifica cualquiera de éstos, su menú reemplaza al menú existente o se añade al mismo.

* Preprograma, Postprograma

Éstos abren ventanas de edición de texto para que usted introduzca el código que se ejecutará, respectivamente, antes o después de la definición del menú. Si usted introduce algo en cada una de las dos ventanas, luego genera el código del menú y lo mira, verá hacia dónde se dirige. El código Preprograma puede utilizarse para definir variables de memoria utilizadas en cláusulas Skip For, por ejemplo, Editando. Asegúrese de hacer pública cualquiera de dichas variables de memoria. El código Postprograma se usa comúnmente para inhabilitar inicialmente opciones de menú seleccionadas. El código de la opción de menú se vuelve parte de una línea de código de menús que comienza así:

```
ON SELECTION POPUP ALL código de opciones de menú
```

y se ejecuta cada vez que se elige una opción de menú del menú generado. No he tenido oportunidad de usarlo, pero me parece que puede llegar a ser útil.

El Menú principal

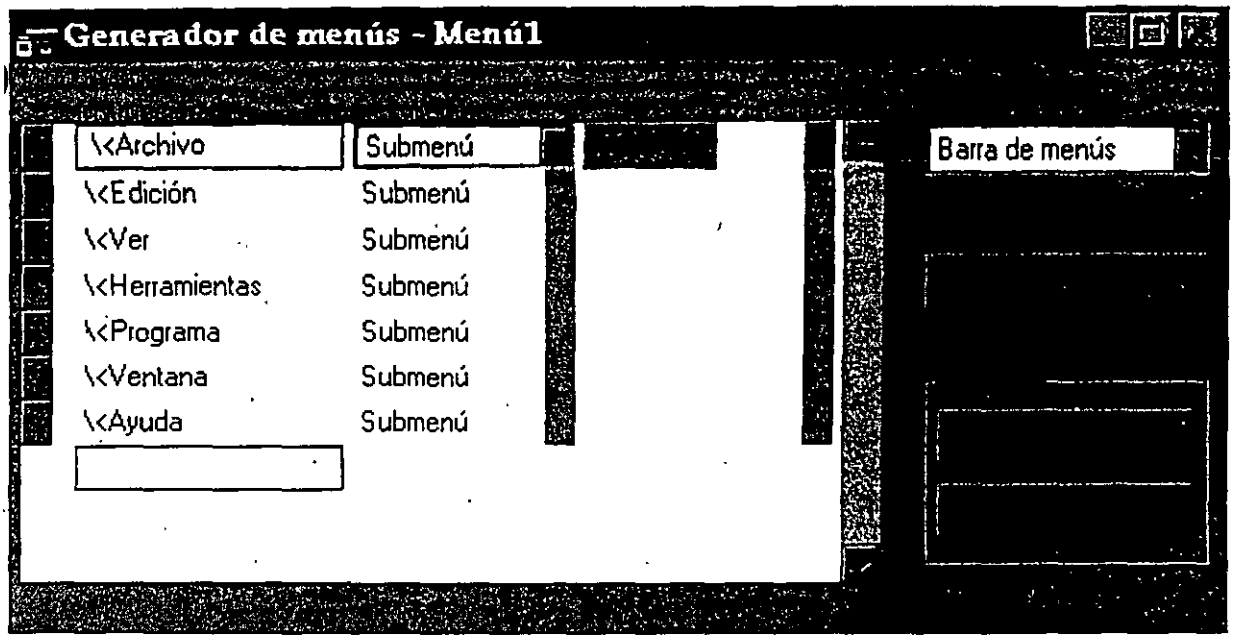
El Menú principal, que sólo es accesible mientras el Generador de menús está en la pantalla, consiste en cinco opciones de menú:

Menú rápido Éste crea un menú estándar como el que se muestra en la figura 8-3. Es útil sobre todo para mostrar todas las cosas que usted puede hacer en un menú. El menú generado tiene opciones como Depuración y Seguimiento, que yo siempre utilizo, pero que obviamente nunca pondría en la aplicación de un cliente. También menciona muchas de las opciones de la barra de menús con sus nombres internos de FoxPro que aparecen en una lista en el apéndice A.

Insertar elemento Es lo mismo que el botón Insertar elemento de la pantalla del Generador de menús, y simplemente mueve todo lo que está debajo de la línea actualmente seleccionada una línea más abajo e inserta un elemento en blanco.

108

Figura 8-3



Ejemplo del Menú rápido.

Eliminar elemento Éste es el mismo botón que Eliminar elemento de la pantalla del Generador de menús; borra la línea en curso y mueve todo lo que está bajo la línea seleccionada actualmente una línea más arriba.

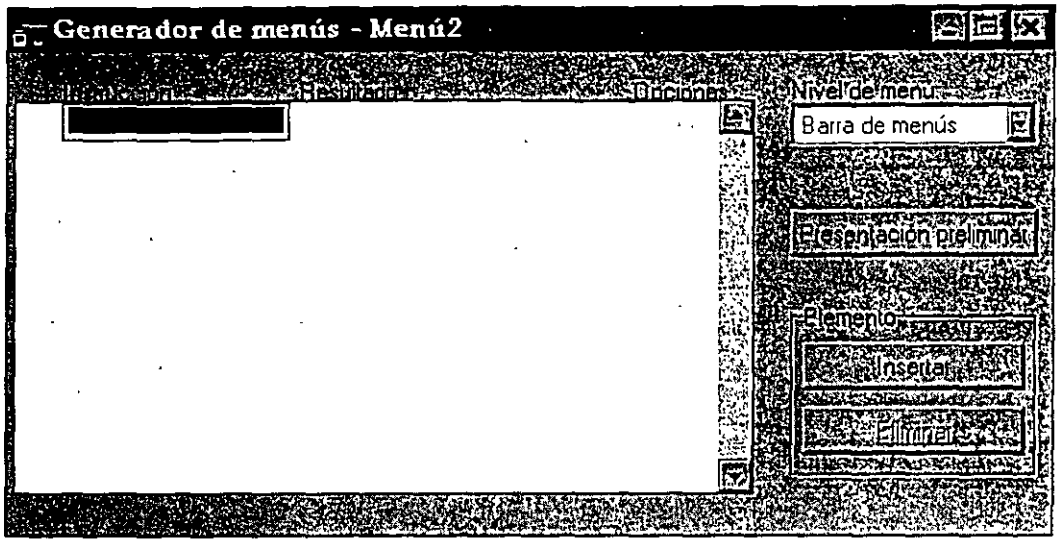
Generar Escribe un archivo, nombredemenu.MPR, incluyendo su código de menú generado. Tómese el tiempo para escribir algunos códigos de preprograma y postprograma y genere un menú; luego mire el código generado para ver dónde los pone GenMenu.

Presentación preliminar Esto le permite "probar antes de comprar". Si hace clic en Presentación preliminar, Visual, entonces el menú de FoxPro es temporalmente reemplazado por el suyo, aunque no ejecuta ninguno de los comandos o procedimientos que usted haya creado hasta ahora en el menú. Se le proporciona un cuadro de diálogo para finalizar la presentación preliminar para permitirle regresar a su sesión con el Generador de menús con el menú de Visual FoxPro restablecido.

Cree un menú

Se hará un menú de prueba. Utilice CTRL-N para abrir el cuadro de diálogo Nuevo; luego haga clic abajo, en Menú. O escriba MODIFY MENU MIMENU y presione

Figura 8-4



Pantalla del Generador de menús.

ENTER. Obtendrá la pantalla en blanco del Generador de menús que aparece en la figura 8-4.

Para la primera línea, introduzca el indicador Cliente y luego salte al campo Resultado. Probablemente Resultado tiene sentido para los tipos que generaron esta pantalla, pero Tipo es un mejor nombre. Existen cuatro tipos:

Comando Se usa para comandos de una sola línea como Do Form Cliente. Si necesita un código de más de una línea, utilice Procedimiento.

Número de barra Se usa para introducir nombres del menú de sistema de Visual FoxPro, como se describe en el apéndice A.

Submenú Se utiliza para crear menús desplegables.

Procedimiento Se utiliza para comandos de líneas múltiples; por lo demás, es igual a Comando.

Puesto que Do Form Cliente se asocia con el formulario Cliente, la mejor opción es Comando, así que escriba DO FORM CLIENTE en el siguiente campo.

Ejecute sus programas desde el menú

He escrito en tres líneas: la primera selección, Cliente, se asocia con el comando Do Form Cliente. La segunda hace lo mismo para el formulario ordenes. Así que

Si los usuarios hacen clic en Cliente, se ejecutará el formulario Cliente; si hacen clic en ordenes, se ejecutará el formulario ordenes justo encima del formulario Cliente. Como sucede con todas las aplicaciones de Windows, los usuarios pueden elegir cualquier formulario haciendo clic en cualquier parte de él para ponerlo "encima".

Si usted ha trabajado en FoxPro 2.6, probablemente recuerda los aros a través de los que tenía que saltar para poner varias ventanas en la pantalla y permitir a los usuarios hacer clic en una ventana subyacente para activar su código. Todo eso es historia: Visual FoxPro maneja automáticamente el cambio de pantallas. Hablé fuerte y con frecuencia contra la programación de ventanas múltiples en FoxPro 2.6; con Visual FoxPro, no existe ninguna razón en lo absoluto para no utilizar este método. ¡De hecho, programar cualquier otro tipo de interfaz requiere de un esfuerzo extra!

Añada elementos desde el menú de Visual FoxPro

La tercera selección se llama Utilerías, y su resultado es un submenú. Haga clic en Crear para añadir un submenú, como se muestra en la figura 8-5.

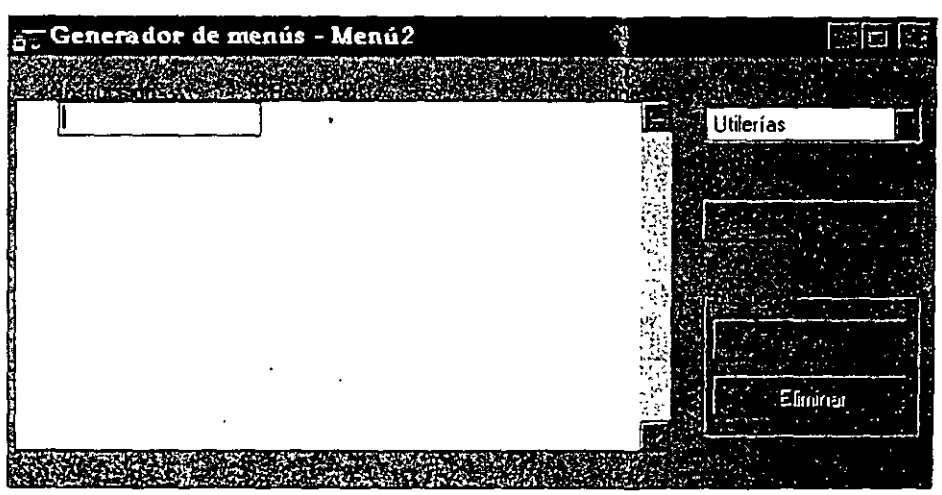


Figura 8-5

Crear un submenú de utilerías.

Para utilizar dos de las características atractivas integradas a FoxPro, introduzca Calendario/Agenda para el primer indicador, presione el tabulador hasta Resultado, haga clic en el botón y elija Número de barra. Luego escriba lo siguiente:

```
_MST_DIARY
```

Introduzca una segunda línea con el indicador Calculadora, y después de elegir la opción Número de barra, introduzca:

```
_MST_CALCU
```

¿Y cómo regresa ahora al principio del menú? Haga clic en el cuadro combinado Nivel de menú, y cuando se despliegue la ruta de acceso a su sitio actual en el menú, haga clic en la entrada de arriba, Barra de menús. Esto no resulta del todo intuitivo para mí, pero así es como se hace. El tercer menú, Salir, necesita un pequeño código extra:

```
#NAME salirmenu
CLEAR WINDOWS
RELEASE WINDOW CALENDAR
RELEASE WINDOW CALCULATOR
SET DEBUG OFF
SET DEBUG ON
SET SYSMENU TO DEFAULT
CLEAR EVENTS
```

La peculiaridad de GenMenu que resulta sumamente extraña son los nombres que Visual FoxPro asigna a los procedimientos generados. Si usted escribe SYS(3) en la ventana Comandos o Depuración, verá de dónde provienen. Aunque probablemente usted ni siquiera verá el código generado, si hay un error, el mensaje que lo identifica puede ser ilegible.

Si desea forzar a Visual FoxPro a utilizar nombres particulares para estos procedimientos, puede incluir el siguiente comando al principio de cualquier fragmento de procedimiento:

```
#NAME nombredelprocedimiento
```

Por ejemplo: deseo llamar al procedimiento de código de salida Codigosalida, por lo que le precedo con la directiva:

```
#NAME Codigosalida
```

El nombre del procedimiento en el código generado será Codigosalida en lugar de _QRB14EXI. Si tiene errores en su código de menú, esto podría hacerlos más fáciles de localizar.

Puesto que los usuarios pueden abrir varias ventanas, usted necesita cerrarlas al entrar a Windows. Esto cuida las ventanas del usuario, pero no el Calendario/ Agenda ni la Calculadora, que tienen que cerrarse de manera explícita.

112

Finalmente, puesto que pudo haber activado el depurador durante el programa, especifique Set Debug Off, cerrando así las ventanas Depuración y Seguimiento; o vuélvalas a accionar. Set Debug On no es lo mismo que Set Step On; simplemente vuelve a habilitar Set Debug On. Luego restablezca el menú de sistema de Visual FoxPro. Para terminar, finalice el comando Read Events que estaba activado en PRINCIP.PRG. Hasta que se ejecuta Clear Events, el programa está "pegado" a Read Valid .T., sólo que mejor.

Advierta que este código está incluido en el procedimiento Salir para mayor claridad. En la práctica, la mayoría de las personas simplemente utilizaría CLEAR EVENTS en el código Salir del menú y llevaría el resto del código al programa Principal, justo después de la línea READ EVENTS. El resultado neto es exactamente el mismo.

Genere su programa de menús

Ahora genere este menú y pruébelo. Haga clic en Menú, Generar, Aceptar. El programa GenMenu que envía con Visual FoxPro lee su archivo de menús y utiliza su contenido para escribir un programa llamado nombremenu.MPR (para el programa de menús).

Haga clic en Archivo, Cerrar para regresar a la ventana Comandos. Si no le otorgó un nombre de archivo al iniciar, utilice MIMENU. Ahora su programa de menús está listo para correr. Si desea echarle un vistazo al código generado, escriba MODIFY COMMAND MIMENU.MPR NOMODIFY. Tome nota especialmente de los lugares donde se incluya cualquier código de preprograma o postprograma.

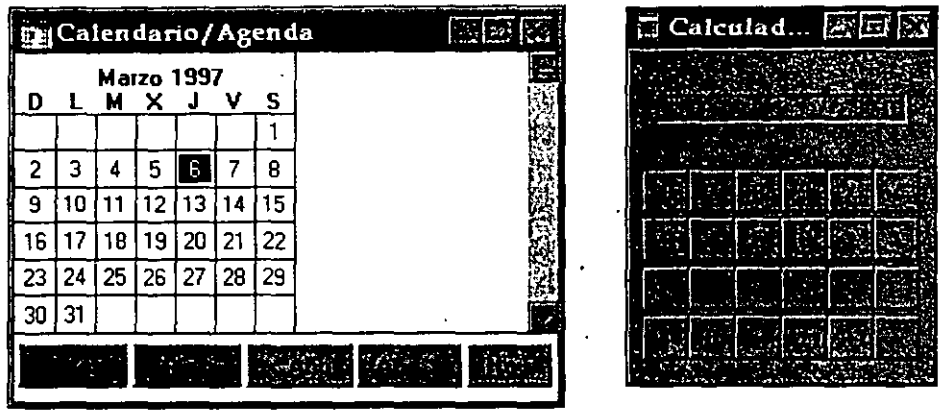
Ejecute su programa de menús

Cuando haya regresado a la ventana Comandos, escriba:

```
DO MIMENU.MPR
```

Haga clic en Utilerías y aparecerá el submenú. Haga clic en Calendario/Agenda y verá la pantalla de la figura 8-6. Haga clic en Salir y verá que no reaparece su menú original de Visual FoxPro. ¡No hay menú Archivo, no hay menú Ventana y, peor aún, no hay Herramientas, desde donde usted podría llamar a las ventanas Seguimiento y Depuración si su programa tuviera problemas! Está claro que su menú no se limpió a sí mismo.

Figura 8-6



Calendario/Agenda y Calculadora de Visual FoxPro.

Si desea ver su menú de Visual FoxPro después de que corra su menú, tiene que poner algo en su menú que lo haga reaparecer. La manera sencilla de hacerlo es incluir la siguiente línea en el código del menú Salir:

```
SET SYSMENU TO DEFAULT
```

De cualquier forma, si su menú necesita regresar a otro de los menús definidos por el usuario, en vez de al menú de Visual FoxPro, escriba:

```
PUSH MENU _MSYSMENU
```

en su código de preprograma, y:

```
POP MENU _MSYSMENU
```

en su código Salir. ¿Por qué no puede insertar POP MENU _MSYSMENU en el código de postprograma del menú? Vea el código generado. El código de postprograma se ejecuta inmediatamente después de que se ha instalado el menú, una fracción de segundo después de que ha instalado su menú. Si pone POP MENU _MSYSMENU en su código de postprograma, verá a su menú literalmente aparecer y desaparecer rápidamente ante sus ojos, seguido por el regreso inmediato del menú de Visual FoxPro.

Evite problemas

Esto lleva a la pregunta de qué más podría incluir en el código correspondiente a la opción Salir en el menú. ¿Qué pasa si sus usuarios han dejado abiertas una o dos ventanas en la pantalla? Tendrían tareas inconclusas que atender, como salvar

114

ediciones o terminar de añadir un registro. ¿Cómo se asegura de que la rutina de da cuide de esos cabos sueltos? La respuesta puede sorprenderle.

En mi hoja informativa tengo una columna llamada "Doctor, me duele cuando hago esto". Es la línea inicial de una vieja broma, cuya réplica es "...y el doctor dijo: Bueno, no lo haga." Si quiere asegurarse de que los usuarios no se salgan de la aplicación durante una edición, No ponga el código de postprograma en la rutina de salida; simplemente inhabilite la rutina Salir del menú mientras lo está editando. Ésta es la forma de hacerlo: en su Princip.PRG, añada la línea:

```
Editando = .F.
```

En el código del evento Click para sus botones Modificar y Agregar, incluya la siguiente línea:

```
Editando = .T.
```

En los botones Guardar y Cancelar, ponga:

```
Editando = .F.
```

Finalmente, haga clic en el botón Opciones a la derecha del indicador Salir, después haga clic en Skip For, y escriba Editando en el cuadro de texto de Skip For. Es tan sencillo como eso. Ahora, cada vez que edite o añada algo, los usuarios no podrán seleccionar Salir porque el menú Salir del menú estará inhabilitado.

Ésta es la clave del software amigable para el usuario. Si sus usuarios no pueden hacer las elecciones equivocadas, harán las correctas y se sentirán muy listos. A todo el mundo le gusta sentirse listo.

Construya versiones demo de su aplicación

Puede utilizar esta característica para crear versiones de demostración de su aplicación. En su menú, elija unas cuantas características clave que los usuarios necesiten para hacer buen uso de su aplicación, y en la cláusula Skip For de las opciones de su menú, introduzca:

```
FILE ( "llave.DBF" )
```

o algún otro nombre de archivo aparentemente inocuo. Ahora, cuando corra su programa, mientras el archivo llave.DBF se encuentra en su directorio de aplicaciones, las opciones de menú correspondientes estarán inhabilitadas.

Cuando llamen para registrarse, borre el archivo llave.DBF. ¡Una vez que el archivo ha sido borrado, la opción funciona!

Existen maneras más sofisticadas de crear condiciones Skip For en su aplicación; éste es un método muy simple.

Cambie la tecla de acceso

Por omisión, la primera letra de cada opción de menú es la tecla de acceso para esa opción. El comportamiento de las teclas de acceso al menú es controlado por la configuración actual de Set Confirm o su contraparte en los campos Herramientas, Opciones, Pantalla general, Tabulador, o ENTER para Salir de los campos. Si Confirm está activado, si se presiona una tecla de acceso para una de las opciones en el menú desplegable, el resaltado brincará a esa opción pero no la seleccionará. Si se presiona la misma tecla de acceso y otra opción del menú inicia con la misma letra, se resaltará el siguiente suceso de la misma tecla de acceso. Por ejemplo, si usted tiene dos opciones de menú, Ordenar y Organizar, y está activado Set Confirm, oprimir la tecla O repetidas veces hará que el resaltado fluctúe entre dos opciones.

Si, por otro lado, usted pone Set Confirm Off, presionar O en el ejemplo anterior hará que su programa elija el siguiente suceso de dicha tecla de acceso; en otras palabras, si se resalta Ordenar, presionar O hará que se seleccione la opción Organizar. Eso es todavía peor que el comportamiento fluctuante que obtiene si pone Set Confirm On.

La solución, por supuesto, consta de dos partes: especifique Set Confirm On y no utilice la misma tecla de acceso dos veces en el mismo menú desplegable. Para cambiar las teclas de acceso, utilice los caracteres \< a la izquierda de la tecla de acceso deseada en cada indicador de opción de menú.

Añada métodos abreviados de teclado

Las teclas de acceso rápido se confunden con frecuencia con los métdos abreviados. Se parecen en que se oprime una tecla de acceso para ejecutar una opción del menú, pero los métodos abreviados de teclado son diferentes en varios aspectos. No tienen que ser una de las letras en el indicador del menú; de hecho, las teclas de F2 a F10, de ALT-F2 a ALT-F10, y de CTRL-F2 a CTRL-F10 se utilizan con frecuencia como métodos abreviados. Además, los métodos abreviados están activos aun cuando el menú no está desplegado.

Para añadir un método abreviado, haga clic en el botón Opciones a la derecha de la opción de menú a controlar; luego, cuando aparezca el cuadro de diálogo Opciones de instrucción, haga clic en Método abreviado. Podrá ver el cuadro de diálogo de la figura 8-7.

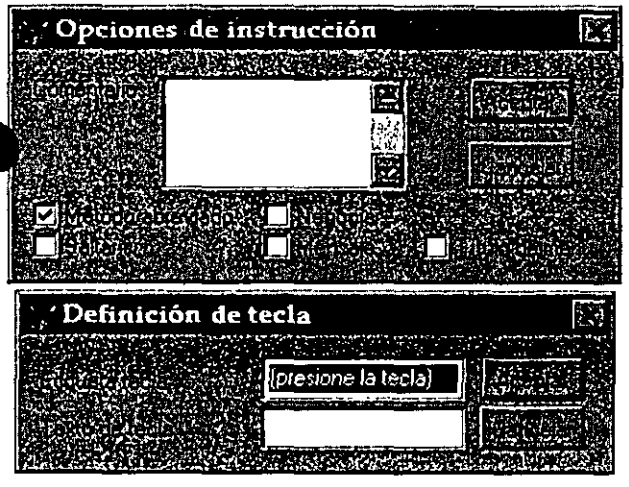


Figura 8-7

Cuadro de diálogo de definición de métdos abreviados.

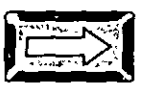
El cuadro de diálogo Definición de tecla del método abreviado es distinto a casi cualquier otra cosa en Visual FoxPro, con excepción de la macro de definición. Usted no escribe en el campo Etiqueta tecla, sino que presiona cualquier combinación de teclas y Visual FoxPro despliega su elección en la ventana Etiqueta tecla. FoxPro lleva entonces cualquier cosa que usted llame en la combinación de teclas en el cuadro Texto de tecla y la pone al lado derecho del menú desplegable. Así que puede introducir Oprima ALT más la tecla D, y eso es lo que verá su usuario a la derecha cuando sea desplegado el menú. El indicador puede tener hasta 19 caracteres, aunque probablemente sea mejor si es más
to.

Colores del menú

En el código de menú generado, los menús utilizan una combinación de color 3 y los menús desplegables utilizan la combinación de color 4. Sin embargo, si usted escribe las siguientes dos líneas desde la ventana Comandos, nada le sucede a los colores del menú o a los del menú desplegable:

```
SET COLOR OF SCHEME 3 TO W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R
SET COLOR OF SCHEME 4 TO W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R,W/R
```

Visual FoxPro utiliza las configuraciones de color del Panel de control de Windows para sus menús. La cláusula Combinación de color se utiliza en DOS, pero no en Windows.



Barras de herramientas

Las barras de herramientas no son, propiamente hablando, parte del sistema de menús de FoxPro, pero se utilizan de manera similar. Las barras de herramie automatizan las funciones que los usuarios podrían seleccionar de un menú, por ejemplo, imprimir una copia del formulario que está en la pantalla. Son como controles móviles; se asocian con el escritorio, y pueden moverse fuera de los límites de la ventana desde la que son lanzadas.

Las barras de herramientas tienen dos requerimientos: deben añadirse a un conjunto de formularios y deben definirse primero como una clase. Lo primero no representa ningún problema, aun si todos sus formularios son ventanas individuales; un conjunto de formularios puede tener un solo formulario, por lo que usted simplemente abre su formulario, se despliega el menú Formulario y hac clic en Crear conjunto de formularios. Si no lo hace, Visual FoxPro le pedirá que lo haga cuando intente agregar la barra de herramientas, por lo que convertir un formulario en un conjunto de formularios de antemano, en lugar de realmente añadir la barra de herramientas, es opcional. El segundo requerimiento tampoco es difícil; simplemente es la primera vez que usted diseña una clase. Y el Generador de clases se parece tanto al Generador de formularios que usted ya debe sentirse como en casa a estas alturas. Así que comience.

Se creará una barra de herramientas para imprimir un informe del formulario en pantalla. Para crear una clase de barra de herramientas, utilice CTRL-N; luec seleccione la clase y haga clic en Archivo, Nuevo (puesto que no existe un

118

Asistente en este momento). Verá la pantalla que muestra la figura 8-8. Introduzca H mientas para el nombre de clase. Luego, despliegue el cuadro combinado Basada en y seleccione la barra de herramientas del final de la lista. Finalmente, tiene que almacenar el nuevo archivo .VCX en algún lugar, así que póngale un nombre. Advierta que el nombre de clase Herramientas está reservado, por lo que Visual FoxPro no le permitirá usarlo.

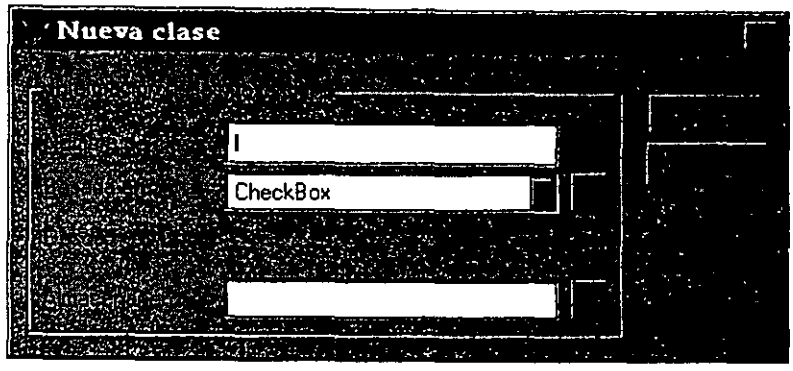


Figura 8-8

Cuadro de diálogo Nueva clase.

Ahora tiene que añadir un control. Su barra de herramientas consistirá en un solo botón de comando con una impresora y un cuadro de consejo que dice "Imprima este formulario", por si nadie sabe que el icono es una impresora. Si la barra de h mientas Controles de formularios no aparece, utilice el menú Ver, l amientas para hacerla visible. Entonces haga clic en el botón de comando, mueva el puntero del ratón a la ventana Clase barra de herramientas y haga clic una vez. Acaba de crear una barra de herramientas.

Y ahora agregue algunos detalles. Haga clic con el botón secundario del ratón en cualquier lugar de la pantalla del Generador de clases y seleccione Propiedades en el menú contextual. Haga clic en la ficha Distribución.

La imagen es la primera propiedad que usted desea establecer. Encontrará una buena colección de imágenes .BMP en el directorio VFP\MISC, aunque ninguna de ellas es una impresora. Puede utilizar el programa ImagEdit que viene con Visual FoxPro y diseñar el suyo propio, o comprar uno de esos CD-ROM con 500 imágenes. Yo finalmente encontré uno bueno en el directorio \VFP\WIZARDS\WIZBMPS.

StatusBarText pone un mensaje en la barra de estado en la base de la pantalla cuando el mensaje de ayuda está activado. La barra de estado es lo suficientemente amplia para permitir una descripción bastante completa de todo lo que hace la herramienta; por desgracia, requiere que la barra de estado sea p te del diseño de su pantalla.

ToolTipText es donde usted añade un pequeño mensaje que se despliega en texto negro sobre un pequeño rectángulo amarillo justo debajo y a la derecha de su icono.

Finalmente, haga clic en Métodos, seleccione Click e introduzca el código apropiado. Para que una herramienta imprima un informe correspondiente al formulario actual en la pantalla, una sola línea de código bastará:

```
REPORT FORM nombinforme NEXT 1 TO PRINT
```

Tendrá que diseñar un informe y darle un nombre, pero aún no se ha llegado al diseño de informes.

Guarde la clase, y luego entre al formulario que desea. Tendrá que añadir la clase de barra de herramientas que ha creado, así que haga visible la barra de herramientas Controles de formularios y luego haga clic en la pequeña pila de libros. Haga clic en Añadir y seleccione el nombre de la clase de barra de herramientas que creó de los archivos .VCX disponibles. La barra de herramientas Controles de formularios cambiará a una más pequeña con solamente cinco elementos, como se muestra en la figura 8-9. Su barra de herramientas es la pa. y el pico. Haga clic en ella y luego haga clic en su pantalla. Haga clic en Formulario, Ejecutar formulario, para ver cómo funciona. Está listo.

Figura 8-9

Barra de herramientas Controles de formularios con la barra de herramientas .VCX cargada.

Verifique que el mensaje de ayuda y el mensaje de la barra de estado estén coordinados. Así de sencillo es añadir barras de herramientas a sus pantallas. Considérelas una especie de "menús fuera del menú" para conocedores.

Hay un viejo proverbio que dice: "Los hijos del zapatero van descalzos." Significa que no debe olvidar que lo que usted hace por su cliente se aplica también a usted mismo. Los programadores olvidan con frecuencia que la computadora es una herramienta, y una herramienta de la que usted como programador puede hacer buen uso.

Una de las maneras más fáciles de programar herramientas para escribir es una lición al menú de Visual FoxPro que permite realizar tareas comunes con teclas de acceso rápido. Escriba MODI MENU Herramie. Éste es un submenú, puesto que se desplegará a la derecha de la Ayuda en el menú de Visual FoxPro.

Escriba Herramientas en el indicador. Ahora presione la tecla TAB sobre el botón de comando a la derecha y haga clic en él. Verá el cuadro de diálogo Opciones de instrucción. La opción importante aquí es la instrucción Método abreviado, que define la tecla de acceso rápido para este menú principal. Aunque los menús de Visual FoxPro responden a ALT-A para Archivo, ALT-E para Edición, y así consecutivamente, no son automáticos. El usuario presiona ALT para activar el menú, libera la tecla ALT y entonces presiona una de las teclas de acceso al menú. Para unir una tecla de acceso rápido ALT-TECLA a un menú principal, tiene que hacerlo de manera explícita.

Haga clic en Método abreviado, luego sostenga la tecla ALT y oprima H. Las opciones Etiqueta tecla y Texto de tecla cambiarán a ALT+H. Haga clic en Aceptar dos veces para regresar al modo de diseño de menú. ¿Qué tal si le asigna un nombre a la hoja? Probablemente no necesitará uno aquí, puesto que éste sólo se ejecuta desde la ventana Comandos, y si algo sale mal usted sabe dónde está el problema.

Ahora haga clic en Menú, Generar, responda Sí al cuadro de diálogo Guardar cambios, y luego haga clic de nuevo en Generar. Regrese a la ventana Comandos y escriba:

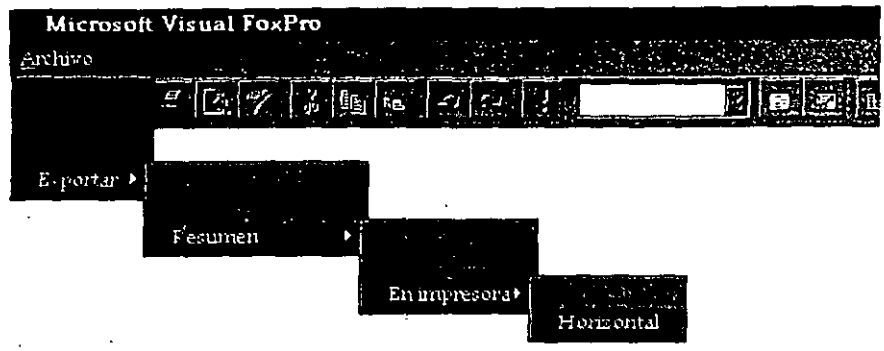
```
DO herramie.MPR
```

El menú tendrá una nueva opción, Heramientas. Utilice ALT-H para desplegarlo. La aparición instantánea del menú sólo tiene dos opciones, y no son útiles todavía porque la aplicación que las utiliza y genera el error en el archivo del registro está en el próximo capítulo, pero usted puede darse una idea.

Genere menús

Generar menús es más un arte que una ciencia. Si usted fuera a utilizar una aproximación puramente lógica, todos sus menús se verían como árboles de decisión, con ramificaciones en cada opción posible. El resultado sería muchos, muchos niveles. Considere la estructura en la figura 8-10. Este menú es el resultado de una estructura lógica que proporciona una ramificación en cada pción. Este tipo de estructura de menú es lógica, pero no es muy útil.

Figura 8-10



Un menú lógico pero excesivamente complejo.

Yo prefiero construir menús que tengan, a lo mucho, tres niveles. Así que una vez que se activa un menú, los usuarios nunca tienen que presionar más de tres teclas para alcanzar el resultado deseado.

Marque opciones de menú

Si su aplicación tiene configuraciones que pueden tener dos estados, el menú proporciona un excelente lugar para desplegar y controlar tales configuraciones. Por ejemplo, los usuarios novatos en una aplicación usualmente prefieren moverse con mayor lentitud a través de los campos de entrada de datos y los menús. Si usted especifica Set Confirm On, tienen que presionar la tecla ENTER para salir a un campo de ingreso o elegir una selección de menú. Por esto añada un submenú llamado Opciones (utilice la selección Título de menú, y haga la primera elección Modo experto. Luego haga clic en Ver, Opciones generales, y escriba la línea:

```
PUBLIC Experto
```

para crear una variable de memoria lógica al mismo tiempo que se ejecute el programa del menú. Finalmente, añada un indicador de menú al submenú desplegable Opciones con el texto indicador Modo experto, haga clic en Resultados, elija Procedimiento e ingrese el siguiente código en la ventana de edición de código:

```
Experto = NOT Experto
IF Experto
     SET CONFIRM OFF
   ELSE
     SET CONFIRM ON
ENDIF
SET MARK OF BAR 1 OF OPTIONS TO Experto
```


122

Genere su programa de menú y ejecútelo. Luego abra el menú desplegable O ...ies y haga clic en Modo experto. Cuando la variable de memoria Experto se fij... .T., tiene una marca de verificación; cuando Experto es .F., Modo experto no se marca.

Éste es un ejemplo de cómo los menús pueden controlar fácilmente configuraciones globales en su aplicación. Sin duda usted se encontrará con muchos otros. Como es recomendable con todo lo relacionado con la programación, no lo haga demasiado.

Indicadores de variables de menú

Otro truco de herramientas se refiere a las variables de memoria en su indicador de menú. Puesto que Visual FoxPro evalúa las opciones de menú muy a menudo, usted puede crear opciones de menú sensibles al contexto.

Considere una aplicación donde los usuarios tienen dos pantallas: Cliente y Ordenes. Su botón Imprimir puede reflejar cuál de los formularios es la ventana que está más al frente en la pantalla si usted realiza lo siguiente: en su menú, ...ya un botón Imprimir en el menú Opciones que acaba de definir. Para su ...ición Skip For, utilice esto:

```
(  NOT [CLIENTE] $ WONTOP( ) ) AND ( NOT [ORDENES] $ WONTOP ( ) )
```

La opción Imprimir se difuminará si ninguna de las pantallas Cliente u Ordenes está al principio. Aquí está el código generado:

```
PUBLIC Experto
SET SYSMENU TO
SET SYSMENU AUTOMATIC

DEFINE PAD OPCIONES OF _MSYSMENU PROMPT "Opciones" COLOR SCHEME 3
ON PAD OPCIONES OF _MSYSMENU ACTIVATE POPUP opciones

DEFINE POPUP opciones MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR 1 OF opciones PROMPT "Usuario experto"
DEFINE BAR 2 OF opciones PROMPT "Imprimir" ;
       SKIP FOR (  NOT [CLIENTE] $ WONTOP ( ) ) ;
            AND ( NOT [ORDENES] $ WONTOP ( )  )
ON SELECTION BAR 1 OF opciones ;
       DO _qxm0rx59j ;
       IN LOCFILE ("CAP08\MARCADOR" , "MPX;MPRFXP;PRG" , "¿Dónde está ;
        el marcador?")
ON SELECTION BAR 2 OF opciones ;
```


123

```
        DO _qxm0rx59t ;
        IN LOCFILE ("CAP08\MARCADOR" , "MPX;MPR|FXP;PRG" , "¿Dónde está el ;
        marcador")

PROCEDURE _qxm0rx59j
Experto = NOT Experto
IF Experto
     SET CONFIRM OFF
   ELSE
     SET CONFIRM ON
ENDIF
SET MARK OF BAR 2 OF OPCIONES TO Experto
PROCEDURE _qxm0rx59t
DO CASE
       CASE [CLIENTE] $ UPPER ( WONTOP( ) )
              REPORT FORM CLIENTE TO PRINT NEXT 1
       CASE [ORDENES] $ UPPER ( WONTOP( )  )
               REPORT FORM ORDEN TO PRINT NEXT 1
ENDCASE
```

Advierta que la referencia LOCFILE en el código del menú generado la pone GenMenu. Si está creando un menú que se sostenga solo (no como parte de un proyecto), los nombres del procedimiento que trata de ejecutar sólo pueden encontrarse si Visual FoxPro puede leer el archivo en el que se encuentran. El código es, por lo tanto, probablemente innecesario en su caso, pero es bueno saber por qué las cosas son como son.

Aquí, la opción Imprimir está inhabilitada si ordenes o Cliente no están al frente de la ventana. Sin embargo, sería bueno que el indicador pudiera reflejar cuál está al frente, en caso de que el usuario no lo sepa. Como es usual, existe una manera de hacerlo. En el método Activate para cada uno de sus dos formularios, añada la siguiente línea:

```
SET BAR 2 OF OPCIONES PROMPT "Imprimir  " + WONTOP( )
```

y en el método Deactivate, añada esto:

```
SET BAR 2 OF OPCIONES PROMPT "Imprimir"
```

Ahora, cada vez que se carga cualquiera de los dos formularios, se cambia el indicador en el menú. El código Skip For en el programa del menú generado se encarga de inhabilitar la opción cuando ninguna está activada, y el código Unload en cada formulario asegura que el indicador no tenga un nombre de formulario junto al indicador Imprimir después de que el formulario ha sido descargado. Pequeños detalles como éste casi no requieren código, y pueden hacer que el usuario se sienta mucho más seguro.

124

Resumen

Este capítulo muestra cómo los menús no solamente permiten navegar a través de una aplicación, sino que también proporcionan atajos y flexibilidad. En el próximo capítulo, usted tomará la sencilla aplicación creada en el capítulo 3 y aplicará algunas de las herramientas desarrolladas en éste y en los capítulos anteriores.

9

Mejore una aplicación de muest

CAPITULO 9

USTED ha visto muchas herramientas desde el capítulo 3, cuando desarrolló una aplicación simple. Es tiempo de regresar y aplicar algunas de esas nuevas ideas para extender la original.

En este capítulo, usted creará de la nada una aplicación de tres pantallas. Esta aplicación es un sistema de ingreso de órdenes para una librería. La persona que toma la orden introduce la información del cliente, busca los libros requeridos y los introduce en la orden.

Los libros se buscan bien por su ISBN (Número Internacional de Libro Estándar, un código de diez dígitos), o por título, autor o palabra clave. Los títulos se ordenan alfabéticamente omitiendo cualquier inicial, *un, una, el, los,* etc. Cada vez que un empleado introduce una tecla, todos los concordantes parciales se despliegan en una lista de elección.

Cree el archivo del proyecto

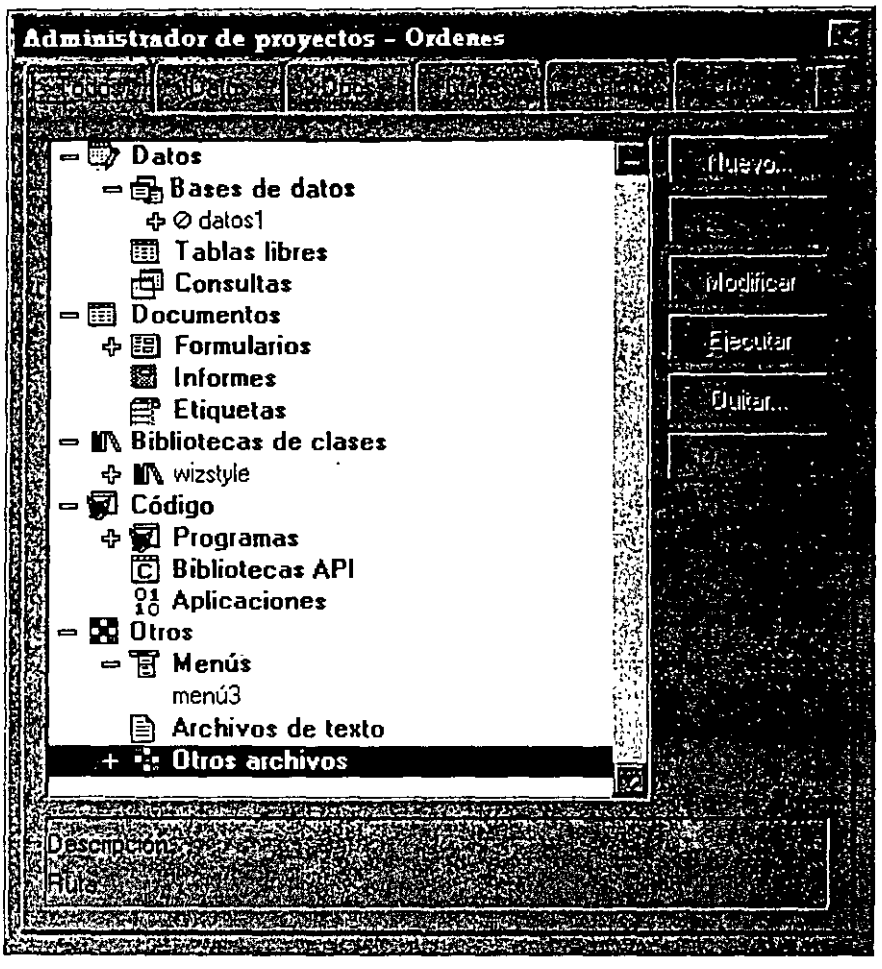
Como es usual, usted empezará con un archivo de proyecto, llamado Ordenes, puede utilizar MD\PINTER\CAP09 para crear un directorio de trabajo. Luego especifique SET DEFAULT TO\PINTER\CAP09 para llegar al directorio una vez que haya sido creado. También puede utilizar el comando CD, que es soportado por Visual FoxPro.

Haga clic en Archivo, Nuevo, seleccione Proyecto y haga clic en Nuevo archivo. Verá el conocido cuadro de diálogo del Administrador de proyectos que se muestra en la figura 9-1. Escriba ORDENES sobre el nombre implícito PROYCT1.PJX y presione ENTER.

El entorno de datos

Su aplicación tiene clientes, libros y órdenes, y requiere cinco tablas: Las tablas Cliente y Libros son "archivos simples", pero las facturas requieren una tabla de órdenes con un registro por factura, más otra tabla de detalles con un registro por elemento de línea por factura. Además, necesitará una tabla Control para almacenar el último número de factura utilizado, el nombre y domicilio de la tienda para imprimir en las facturas. La tabla Cliente es similar a los ejemplos anteriores:

Figura 9-1



Archivo del proyecto Ordenes.

Campo	Nombre de campo	Tipo	Ancho	Dec	Expr Índice
1	CODIGOCLIE	Carácter	1		CODIGOCLIE
2	NOMBRE	Carácter	20		
3	APELLIDO	Carácter	20		
4	DIRECCION1	Carácter	40		
5	DIRECCION2	Carácter	40		
6	CIUDAD	Carácter	20		
7	ESTADO	Carácter	3		
8	CODPOST	Carácter	10		
9	PAIS	Carácter	10		
10	TELEFONO1	Carácter	14		
11	TELEFONO2	Carácter	14		
12	LIMCREDITO	Numérico	6		
13	BALANCE	Numérico	7	2	
14	CREDITOHABIL	Lógico	1		
15	NUMTARJETA	Carácter	16		
16	FECHAEXPED	Carácter	5		
17	COMPRAS	Numérico	7	2	

La tabla Libros contiene la información necesaria sobre el precio y el título para llenar una factura, así como campos para un control rudimentario de inventario y pedidos:

Campo	Nombre de campo	Tipo	Ancho	Dec	Expr Índice
1	ISBN	Carácter	10		ISBN
2	TITULO	Carácter	60		UPPER(TITULO)
3	PRECIOLISTA	Numérico	7	2	
4	ADICIONAL	Memo	10		
5	AUTOR	Carácter	20		UPPER(Autor)
6	PRECIOVENTA	Numérico	7	2	
7	EXISTENTES	Numérico	4		
8	PEDIDOS	Numérico	4		
9	ORDENESPECIAL	Numérico	4		
10	LLEGARAN	Fecha	8		
11	CONDICIONES	Carácter	3		
12	CODIGO	Carácter	10		
13	CATEGORIA	Carácter	10		
14	TIPO	Carácter	10		
15	UBICACION	Carácter	10		

La expresión de índice de Título es de hecho un poco más complicada de lo que parece. Usted desea una lista alfabética de títulos como si ningún título empezara con *un, una, el, los,* etcétera, por lo que en realidad la expresión es:

```
INDEX ON UPPER(    ;
  IIF ( LEFT (UPPER(Titulo),2)=[Un ], SUBSTR(Titulo,3),   ;
  IIF ( LEFT (UPPER(Titulo),3)=[Una ], SUBSTR(Titulo,4),   ;
  IIF ( LEFT (UPPER(Titulo),4)=[El ], SUBSTR(Titulo,5),Titulo))))   ;
  TAG TÍTULO
```

Como se muestra en la orden de ejemplo en la figura 9-2, una orden tiene un área de encabezados, que consiste en un número de factura, fecha e información sobre el cliente; un área de detalles con una línea para cada libro comprado, y un área al pie para el subtotal, impuestos y la suma total. Para almacenar una orden, usted necesita dos archivos.

El archivo de facturas contiene una copia con la información del cliente en el momento de la venta:

Campo	Nombre de campo	Tipo	Ancho	Dec	Expr Índice
1	NUMORDEN	Carácter	10		NUMORDEN
2	FECHA	Fecha	8		
3	CODIGOCLIE	Carácter	10		CODIGOCLIE
4	NOMBRE	Carácter	40		
5	DIRECCION1	Carácter	40		
6	DIRECCION2	Carácter	40		
7	CIUDAD	Carácter	20		

8	ESTADO	Carácter	3	
9	CODPOST	Carácter	10	
10	PAIS	Carácter	10	
11	SUBTOTAL	Numérico	7	2
12	IMPUESTO	Numérico	6	2
13	TOTAL	Numérico	7	2
14	ENVIADO	Lógico	1	

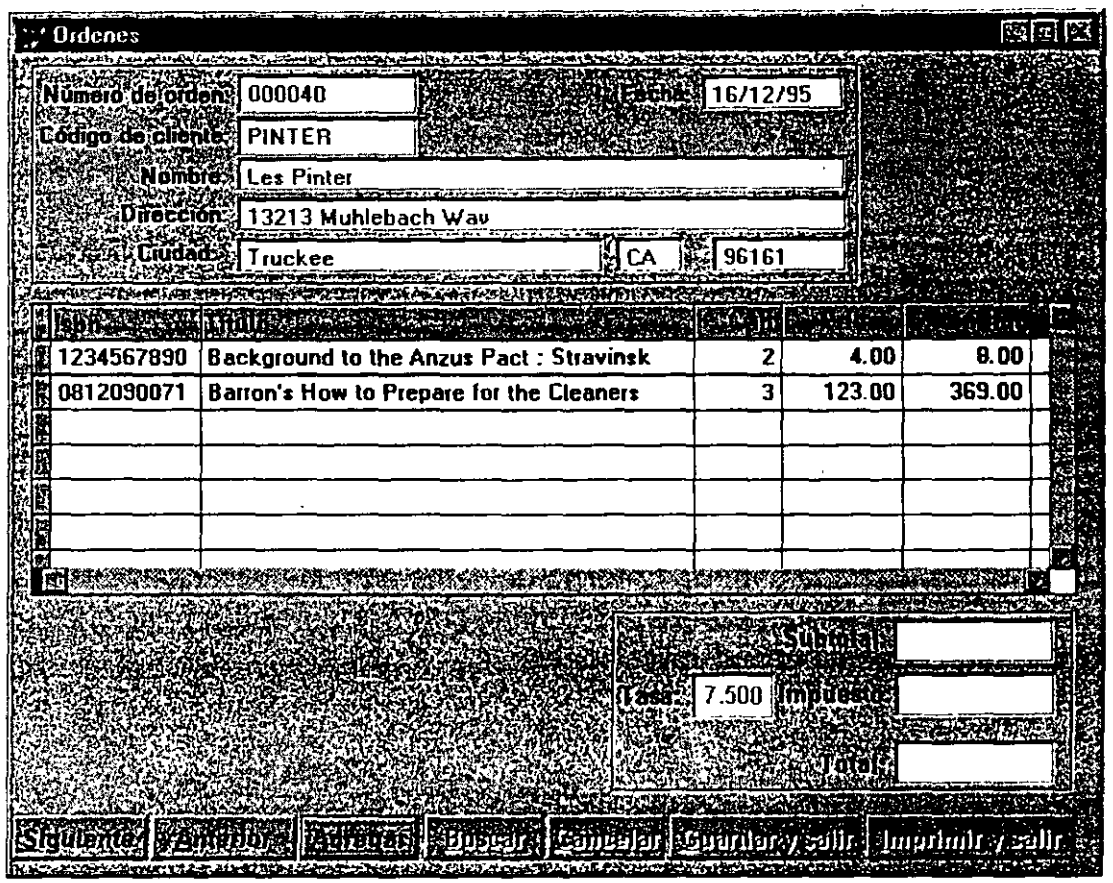


Figura 9-2

Una factura con áreas de encabezados, detalles y pie.

El archivo DETALLES.DBF es donde están almacenadas las líneas individuales de la factura:

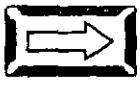
Campo	Nombre de campo	Tipo	Ancho	Dec	Expr Índice
1	NUMORDEN	Carácter	10		NUMORDEN
2	ISBN	Carácter	10		
3	CANTIDAD	Numérico	1		
4	TITULO	Carácter	40		
5	PRECIO	Numérico	7	2	
6	ADICIONAL	Numérico	7	2	
7	ESPECIAL	Lógico	1		

Finalmente, el siguiente archivo de control contiene el nombre de la tienda (para la impresión de las facturas), así como un campo que contiene el último número de factura utilizado. Usted va a estar tratando cuestiones de usuarios múltiples, y ésta es la mejor forma de manejar una numeración secuencial.

Campo	Nombre de campo	Tipo	Ancho	Dec
1	ULTIMAORDEN	Numérico	6	
2	NOMBRETIENDA	Carácter	40	
3	DIREC1TIENDA	Carácter	40	
4	DIREC2TIENDA	Carácter	40	
5	DIREC3TIENDA	Carácter	40	
6	MENSAJE1	Carácter	40	
7	MENSAJE2	Carácter	40	

Advierta que todos los campos numéricos son lo suficientemente amplios para contener un total de siete dígitos, incluyendo dos lugares decimales para los centavos y el mismo punto decimal. El campo para divisas apoyado por Visual FoxPro, de hecho, proporcionará un número mucho mayor, pero como en este momento aún es un poco escaso, no se está usando para las aplicaciones de muestra. Sustituya campos de divisas por estos campos de pesos, y estará en forma.

La figura 9-3 muestra la pantalla Entorno de datos para la aplicación. Las teclas que unen a las tablas están esbozadas, aunque con frecuencia las relaciones se controlan directamente en las aplicaciones.



Los formularios

En el capítulo 3, usted utilizó el Asistente para formularios para crear un formulario que tuviera un sombreado atractivo, un juego completo de navegación y otros botones de control al pie de la pantalla, y una cierta cantidad de lógica interna. Ése es el poder de las clases. La desventaja de utilizar clases es que usted hace las cosas a la manera de éstas (aunque, finalmente, usted diseñará sus propias clases).

Por ahora, no se usará ninguno de los formularios creados por el Asistente para formularios. Una razón es porque a mí me gusta que las pantallas tengan un cierto aspecto: yo cambio el color de fondo de la celda de entrada actual a texto blanco sobre un fondo rojo, para que los usuarios sepan dónde están. Cuando abandona

131

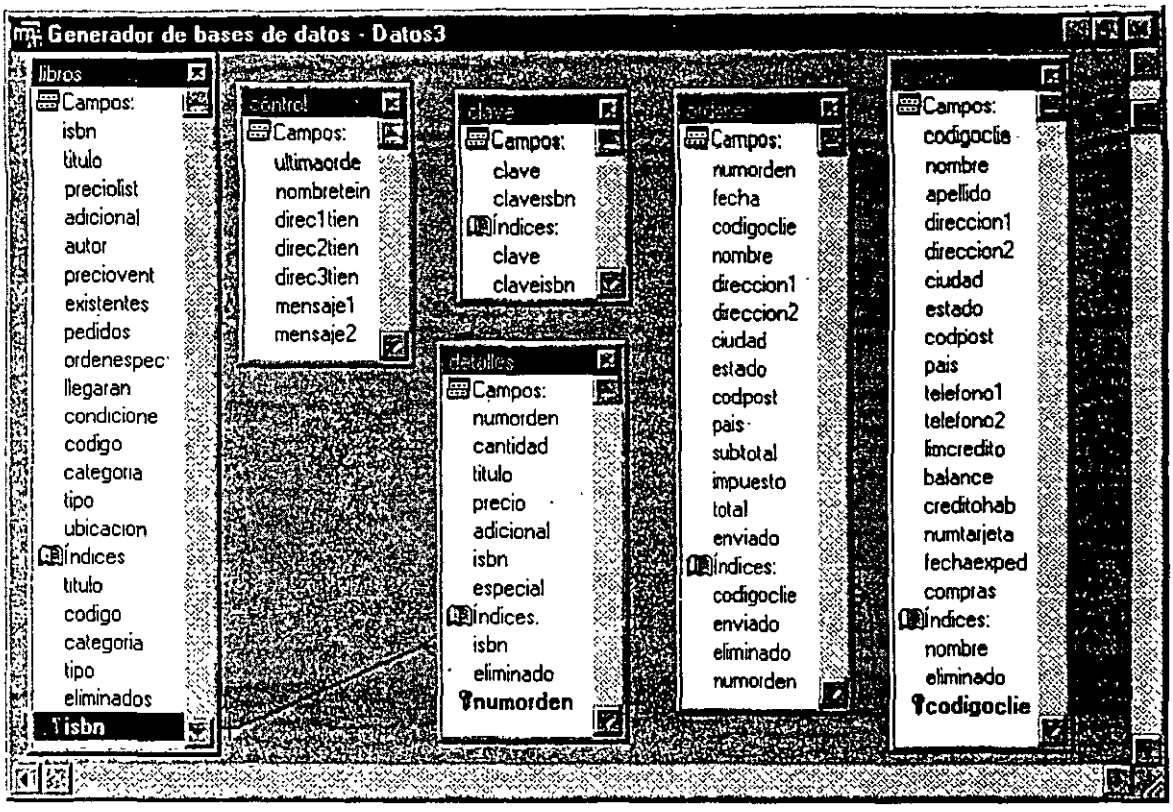


Figura 9-3

Entorno de datos para la aplicación de muestra.

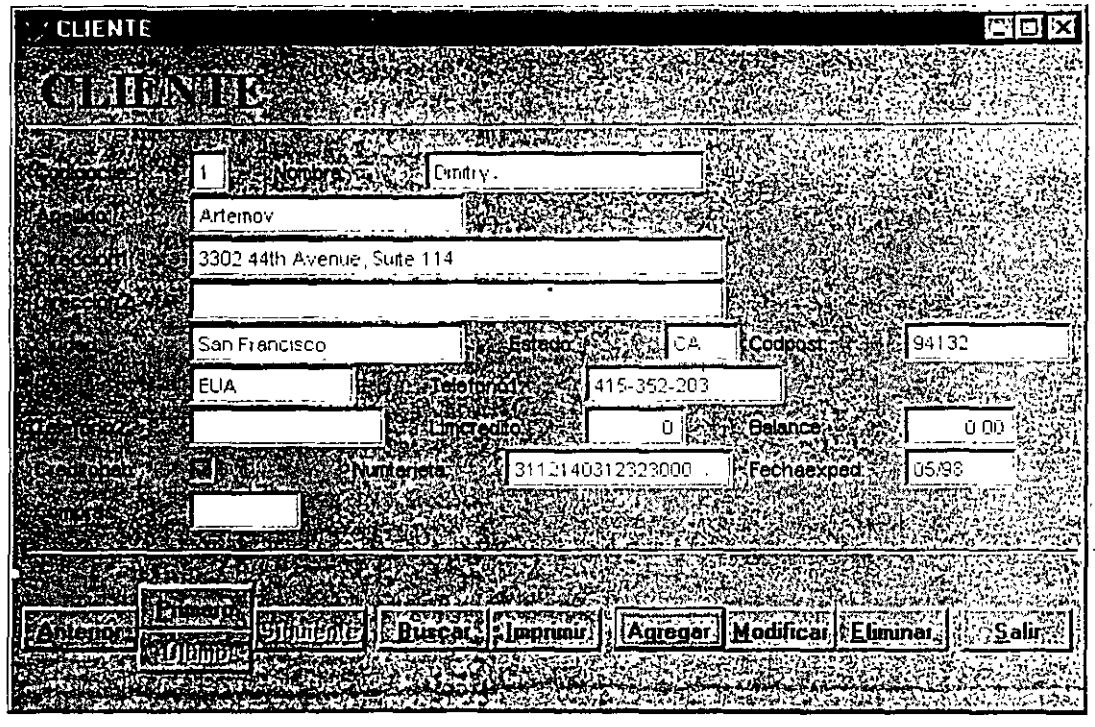
el campo, el fondo se revierte a texto estándar. Si usted regresa al ejemplo en el capítulo 3 y hace clic en cualquier objeto de la pantalla, advertirá que, de hecho, hay tres partes en cada campo:

- La etiqueta con el nombre de campo para un encabezado
- El campo de entrada en sí
- Una forma que sirve como la sombra del campo de entrada

Para dirigirse a un objeto de entrada de datos, tiene que referirse a él por su nombre. En el caso del código del cliente en la pantalla realizada por el asistente, el nombre del campo de ingreso es Form1.CodigoCli1.Text1. Cuando se escribe código, no puede referirse simplemente a la última parte del nombre; es necesario el nombre completo. Aunque los nombres de los objetos no son muy, muy importantes, a veces se nombra a los objetos de la pantalla para que el código sea más fácil de leer cuando haya que referirse a los objetos por su nombre.

Así que haga clic en Formulario, Nuevo, y luego haga clic en Nuevo formulario (no en el Asistente para formularios). El formulario vacío Cliente, con el nombre Form1, aparecerá como se muestra en la figura 9-4.

Figura 9-4



El formulario Cliente.

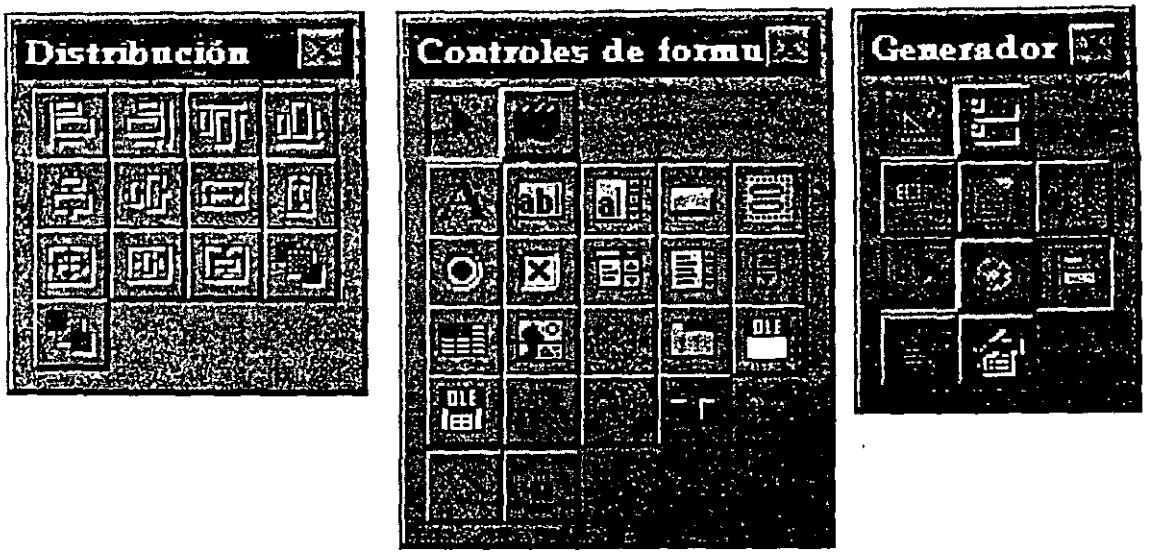
Establezca la mesa de trabajo

Para trabajar en el formulario, necesita tres barras de herramientas que estén disponibles con facilidad: Controles de formularios, Generador de formularios y Distribución.

Haga clic en Ver, Barras de herramientas desde el menú de Visual FoxPro, y marque la casilla de las tres barras de herramientas que desea utilizar. Luego llévelas al lado derecho de la pantalla. La barra de herramientas Generador de formularios es la de la derecha, ajustada al tamaño de cuatro iconos de altura y tres de ancho. Esto pone al icono Propiedades en la segunda fila a la izquierda, que lo hace fácil de encontrar. Lo usará mucho. Ahora, haga la barra de herramientas Distribución de cuatro por cuatro y sitúela a la derecha de la barra de herramientas Generador de formularios. Finalmente, dé a la barra de herramientas Controles de formularios un tamaño de cinco iconos de altura y póngala debajo de las otras dos. La disposición resultante, que aparece en la figura 9-5, es la manera en que yo establezco mi mesa de trabajo cada vez que laboro en un formulario, por lo que no tengo que preguntarme dónde están las cosas.

133

Figura 9-5



Barras de herramientas arregladas para trabajar con un formulario.

Cree el formulario Cliente

El formulario Cliente aún se llama Form1. Usted desea cambiar el nombre, el encabezado y el color de fondo antes de hacer cualquier otra cosa. Haga clic en el icono Propiedades en la barra de herramientas del Generador de formularios, y luego haga clic en la ficha Otros. Name es el séptimo elemento hacia abajo. Resáltelo y luego escriba Cliente. Lo resaltado está en la propiedad Name, pero el cursor está en la ventana de ingreso como a una tercera parte del camino en la ventana Propiedades.

Ahora, haga clic en la ficha Distribución, resalte la entrada Caption (la quinta hacia abajo) y vuelva a escribir Cliente. Advierta que el nombre del formulario está integrado a su código, mientras que el encabezado es lo que aparece en la parte superior del formulario.

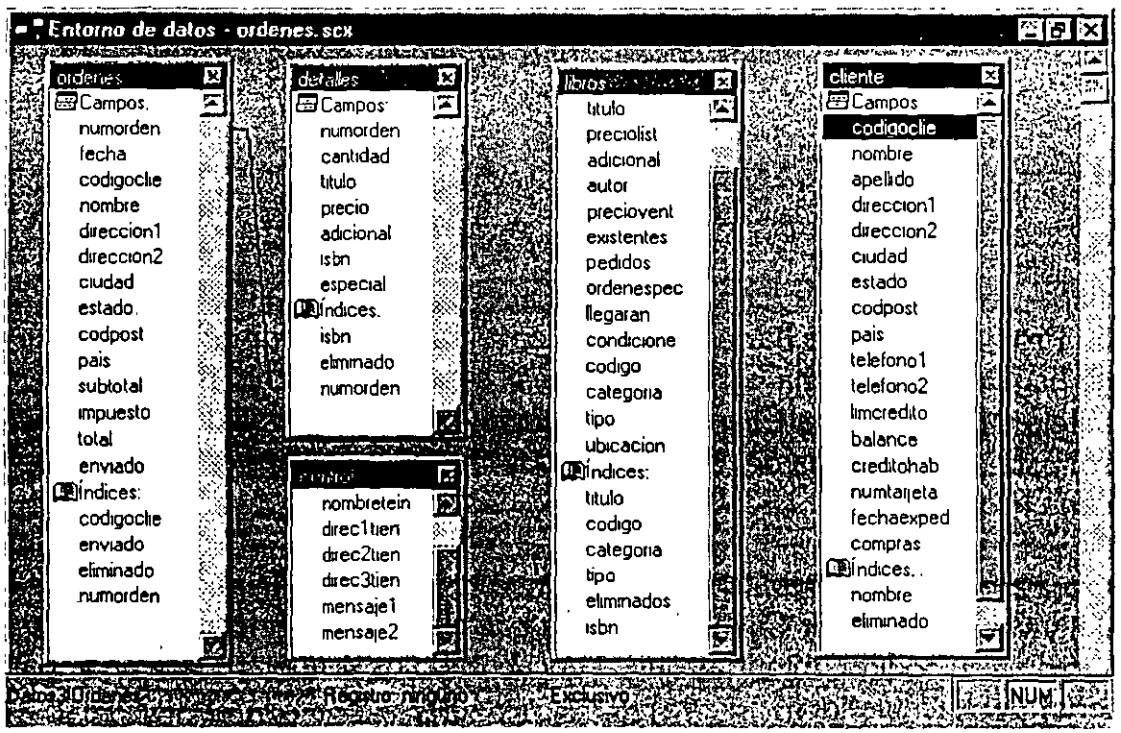
Finalmente, haga doble clic en la tercera propiedad, BackColor. Cuando aparezca el Generador de colores, haga clic en el rectángulo gris en la fila de abajo de selecciones de color y luego haga clic en Aceptar. Para centrar el formulario, haga clic dos veces en AutoCenter (la segunda propiedad hacia abajo en la barra Distribución).

Debe repetir estos pasos para cada formulario, pero si genera un formulario maestro, llamado Plantilla, puede especificar todas estas configuraciones una sola vez. Luego serán usadas cada vez que añada un formulario nuevo utilizando su plantilla como diseño.

Utilice el entorno de datos del formulario

Puesto que está creando un formulario para utilizarlo con la tabla Cliente, lo primero que debe hacer es unir la tabla al Entorno de datos del formulario. Entonces, cada vez que haga clic en una propiedad ControlSource, el Generador de formularios le proporcionará una lista de nombres de campo de la tabla Cliente. Haga clic en Ver, Entorno de datos desde el menú para obtener la ventana Entorno de datos; haga clic con el botón secundario del ratón para obtener el menú contextual, y luego haga clic en Añadir. Elija Cliente de la lista de tablas. La figura 9-6 muestra lo que verá. Haga clic en el botón Minimizar en la esquina superior derecha de la ventana Entorno de datos para minimizarlo.

Figura 9-6



Entorno de datos con la tabla Cliente añadida.

Dimensión de la pantalla

La ventana Propiedades quedará ajustada debajo de las barras de herramientas si a su pantalla le da tamaño de manera tal que usted tenga lugar para todo. Yo diseño la mayoría de las pantallas para 640 x 480, la resolución más baja de VGA, puesto que nunca sé en qué tipo de laptop o monitor correrá la aplicación.

135

Sin embargo, trabajar en una pantalla de 640 x 480 no provee de mucho espacio para colocar sus herramientas, especialmente con el arreglo de herramientas de Visual FoxPro. Por esto, generalmente cambio la resolución de la pantalla ya se a 800 x 600 o a 1,024 x 768, y luego fijo el área máxima de diseño a 640 x 480 en la ficha Formularios de la página Opciones, Herramientas. Eso me da algo de espacio para barras de herramientas y la ventana Propiedades a la derecha de mi formulario.

Añada campos de entrada y etiquetas de campo

Cada campo de datos en el formulario necesitará una etiqueta para indicar qué información hay en el campo. Haga clic en la A mayúscula en la barra de herramientas Controles; cree un pequeño rectángulo en la esquina superior izquierda del formulario vacío. Luego, haga clic en el icono del cuadro de texto

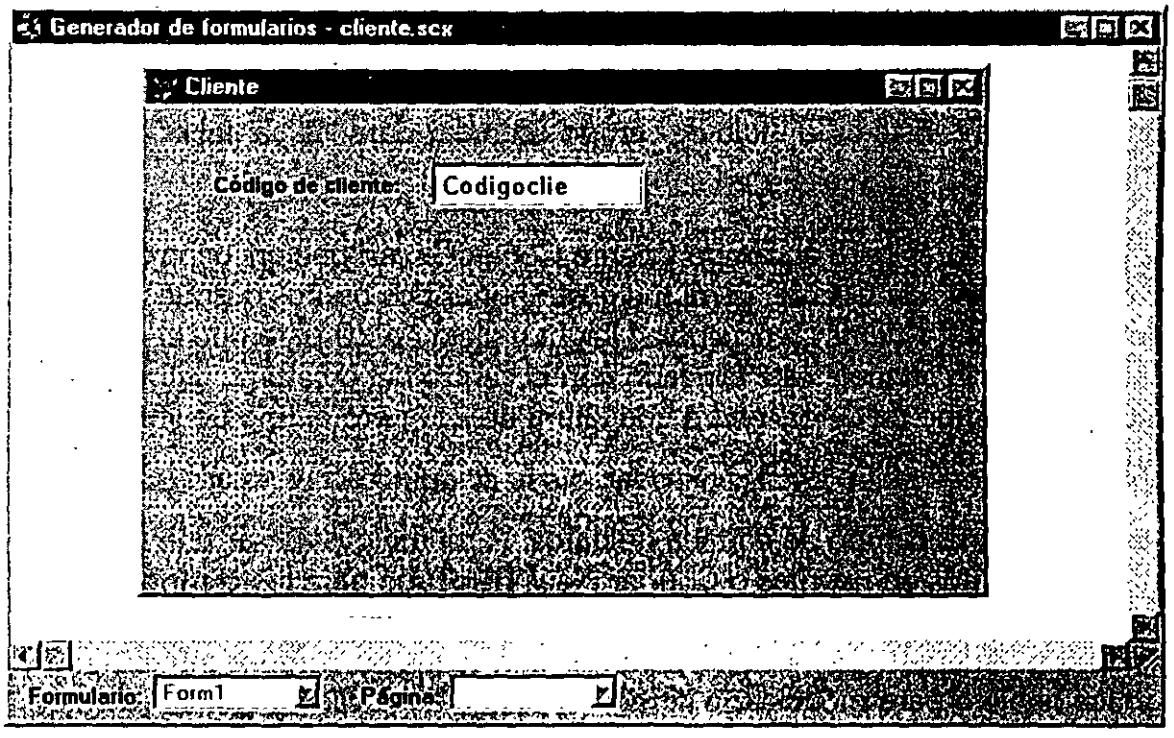


Figura 9-7

Agregue el campo CodigoClie al formulario Cliente.

justo a la derecha de la A, y ponga en la pantalla un rectángulo ligeramente más amplio a la derecha del primero. El resultado se muestra en la figura 9-7. Advierta que también puede abrir el entorno de datos y arrastrar y soltar nombres de campo en el formulario.

Desafortunadamente, la técnica no funcionará con clases, por lo que no tiene sentido acostumbrarse a ella.

Hasta ahora, usted tiene un rectángulo con el encabezado Label1 y otro con el encabezado Text1. Haga clic en Label1 para que los cuadrados negros lo enmarquen por todos sus lados y esquinas. Luego haga clic en el icono más alejado hacia la izquierda en la segunda fila de la barra de herramientas Generador de formularios, el icono Propiedades. Obtendrá el cuadro de diálogo Propiedades justo debajo de sus barras de herramientas.

Su próximo trabajo será crear un par de elementos, una etiqueta y un campo de entrada, que tengan las propiedades que desea. Luego puede copiar este par de objetos para cada campo en la pantalla. Haga clic en el objeto Label1 y luego en el icono Propiedades. Haga clic en Distribución, luego encuentre Width al pie de la lista de propiedades y escriba 100 (por 100 pixeles). Después, encuentre la propiedad BackStyle en la misma lista y haga clic dos veces para fijar el valor en 0, Transparente. Finalmente, localice Alignment y haga clic dos veces para fijar el valor en 1, Right. Éste es el título de campo estándar para las pantallas de ejemplo.

Este primer título es para el campo Cliente.CodigoCli, así que encuentre el campo Encabezado y escriba Cliente: (a mí me gusta poner dos puntos después de los títulos de campo). Además, encuentre Height y fíjelo en 20 para que el encabezado de la etiqueta esté alineado con el centro del objeto de datos a la derecha. (El campo de datos tendrá una altura de 25, pero el tipo de letra será distinto, por lo que harán juego.)

Lo que sigue es el campo de entrada de datos. A mí me gusta usar una fuente Courier 11 puntos en negritas o Courier New TrueType para los campos de entrada de datos; utilice las propiedades FontBold y FontName en la página Distribución. Además, yo inicio las pantallas con todos los campos inhabilitados, y luego los habilito para modificar o agregar. Así que haga clic en la ficha Otros y luego haga clic dos veces en Enabled, para fijar la propiedad en .F.

Mientras añade sus cuadros de texto, Visual FoxPro los nombra consecutivamente de acuerdo con su tipo; por ejemplo, Text1, Text2, y así sucesivamente. Si usted

cambia el nombre de los cuadros de entrada de texto al mismo del campo de fuente de datos en su tabla, al menos puede mirar la pantalla y ver qué es que.

Aquí hay un ejemplo: si deseo hacerle algo a todos los campos de ingreso en la pantalla, por ejemplo, cambiar su color de fondo a azul, tengo que escribir algo como esto:

```
FOR I = 1 TO THISFORM.ControlCount
        THISFORM.Controls(I).BackColor = RGB ( 0, 0, 175 )
ENDFOR
```

Pero esto cambia el color de todo lo que hay en el formulario: controles, líneas, formas, cualquier cosa. Por otro lado, si usted añade un prefijo de dos caracteres a cada objeto del formulario, puede escribir métodos genéricos. Si desea cambiar una propiedad para todos los miembros de una clase en un formulario utilice el método SetAll. Si, por el contrario, tiene campos de la misma clase que necesitan ser tratados de manera distinta, existe una forma de hacerlo con prefijos para el nombre de objeto. Yo uso una convención para nombrar en la cual pongo prefijos con I_ a los campos de entrada y con B_ a los botones de comando. Por ejemplo, el objeto Apellido se llama I_Apellido, mientras que el botón Siguiente se llama B_Siguiente. Aun puede leerse en la pantalla, y le permite referirse a sus objetos de formularios con un código de método de una manera más descriptiva que, por ejemplo, ThisForm.Text3.

Finalmente, haga clic en la ficha Datos. Visual FoxPro une en sus tablas objetos de entrada a los campos correspondientes, por lo que buffering trabaja de manera transparente. Éste es uno de los grandes beneficios de Visual FoxPro, y no cuesta más ir en primera clase. Así que resalte ControlSource y luego haga doble clic. El primer campo en el archivo Cliente, Cliente.CodigoCli, aparecerá en el campo de valor de la propiedad ControlSource.

Si usted hace clic en la flecha hacia abajo del cuadro combinado sobre el área de la lista de propiedades, verá todos los campos en CLIENTE.DBF. Hacer clic sucesivamente en la línea de la propiedad ControlSource realizará un ciclo a través de todos los campos disponibles. Si su entorno de datos está formado por varias tablas, realizar el ciclo puede ser la manera lenta de llegar al campo que necesita; en este caso, haga clic en el cuadro combinado y elija el campo deseado de la lista desplegable.

Finalmente, puede establecer la propiedad InputMask para este objeto de datos. Será diferente para cada campo, por supuesto, y es fácil olvidarse de cambiarlo cuando hace una copia del primer par de objetos. Establezca la propiedad InputMask en !!!!!!!!!!.

Muestre el campo de ingreso actual

A mí me gusta facilitar a los usuarios que determinen en qué campo están. En Visual FoxPro, usted dice que el objeto actual tiene el foco, y que pierde el foco cuando algún otro objeto se convierte en el objeto actual. La mejor manera de hacer eso es cambiar el color del objeto actual cuando llega al foco, y luego regresarlo al normal cuando el objeto pierde el foco. A mí me gusta el texto en color blanco sobre un fondo rojo para el objeto actual, y texto negro sobre un fondo blanco para todo lo demás.

Esto solía ser una lata en FoxPro 2.6, pero en Visual FoxPro es una nadería. Simplemente añada el código siguiente al código del evento GotFocus del objeto de entrada:

```
THIS.ForeColor = RGB ( 255, 255, 255 )
THIS.BackColor = RGB ( 255,   0,   0 )
```

Luego introduzca lo siguiente en el código LostFocus del objeto:

```
THIS.ForeColor = RGB (   0,   0,   0 )
THIS.BackColor = RGB ( 255, 255, 255 )
```

Duplique los objetos del modelo

El par de objetos en el código anterior constituye su modelo para todos los campos de entrada de datos en el formulario. Todo lo que tiene que hacer es mantener presionada la tecla MAYÚS (SHIFT) y hacer clic en los dos objetos para seleccionarlos a ambos, y luego oprimir CTRL-C para copiarlos a la memoria del portapapeles. Una vez que están copiados, la opción Pegar en el menú Editar se habilita, indicando que puede volverse a pegar en el formulario.

Pegue el par de objetos al formulario para usarlos como el siguiente campo de entrada. El siguiente campo de entrada será Nombre del cliente, que consta de 4 caracteres. Para que el formulario se vea balanceado, voy a poner el nombre y la dirección directamente debajo del campo Código del cliente.

El método abreviado para Pegar es CTRL-V. En FoxPro 2.6 para DOS, usted po seleccionar un lugar en la pantalla antes de pegar; no sucede así con Visual FoxPro para Windows. Los contenidos del búffer Pegar están unidos en la mism

posición relativa que la del objeto que está siendo copiado y marcado como seleccionado. Haga clic en cualquier lugar del par de objetos que acaba de añadir y arrástrelos al área justo debajo del campo Código del cliente. Por el momento no se preocupe por alinearlos perfectamente; eso es fácil en Visual FoxPro. Se muestra el resultado en la figura 9-8.

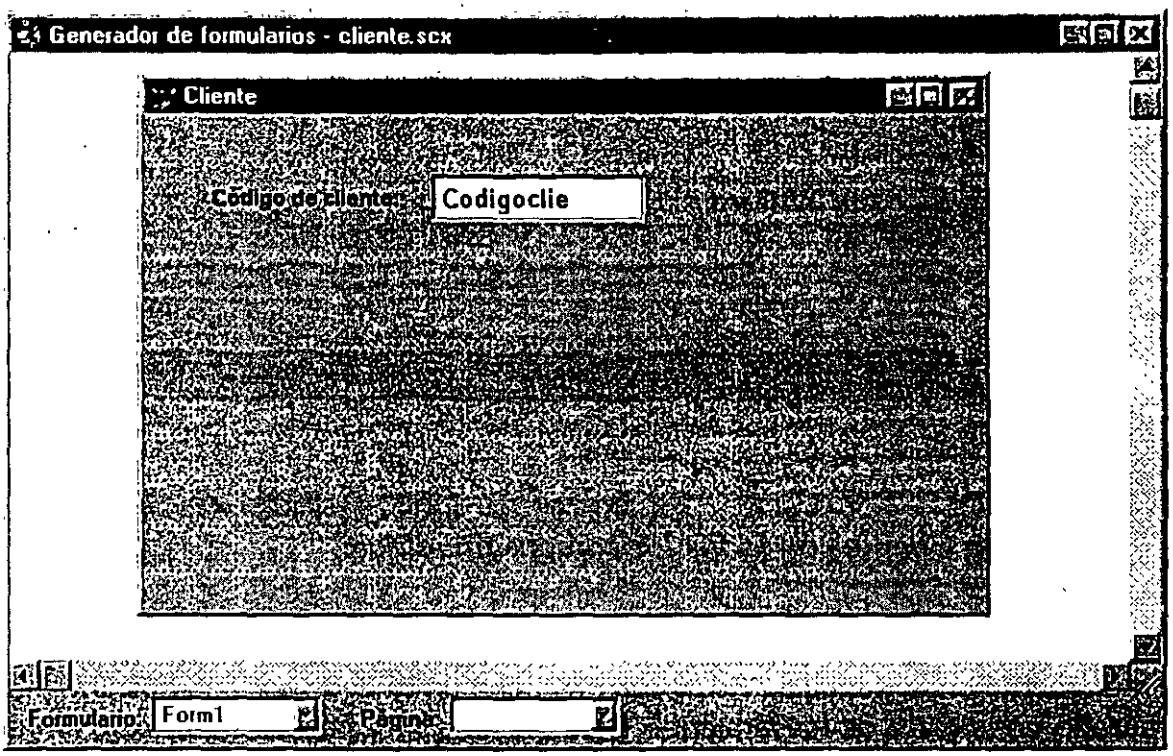


Figura 9-8

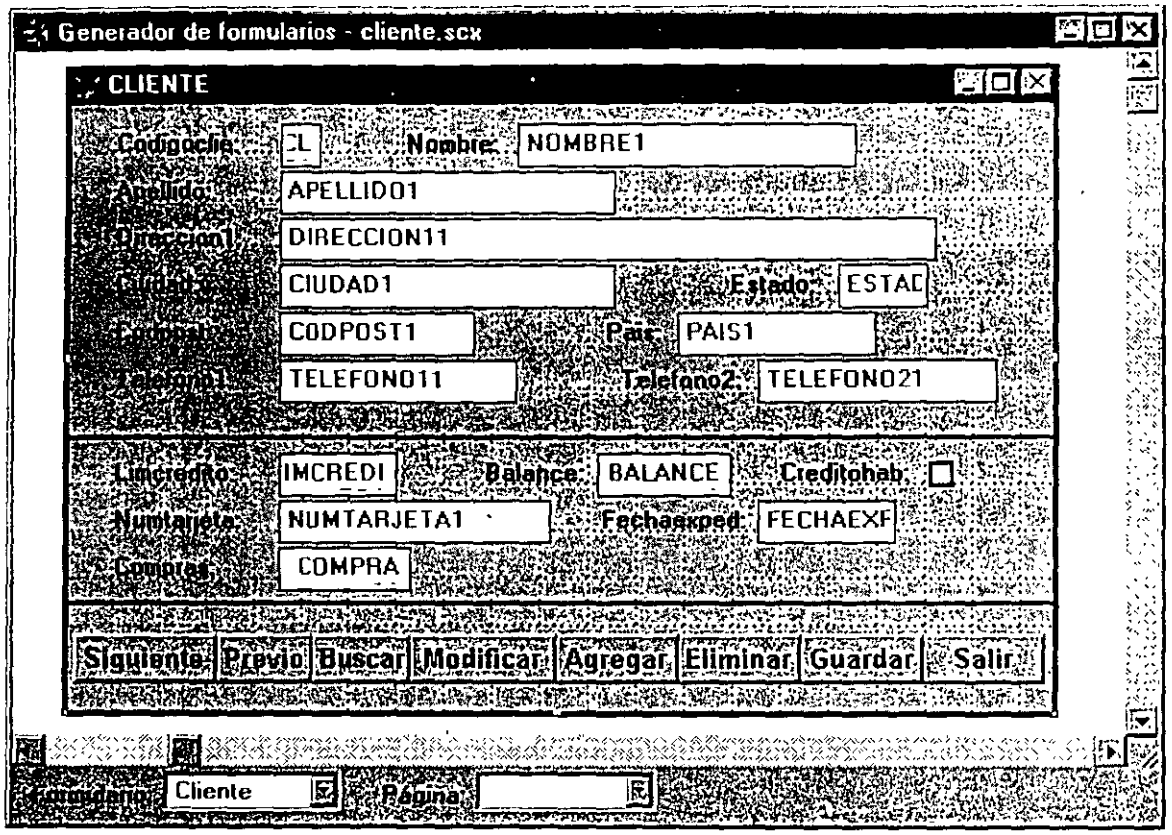
Agregue un segundo título y objeto de entrada al formulario.

De ahora en adelante, cada campo requiere sólo tres cambios:

- El encabezado de la etiqueta
- Las propiedades DataSource del campo de entrada
- Las propiedades Width, MaxLength e Input Mask del campo de entrada

Copie el par modelo y haga los cambios necesarios hasta que haya creado la pantalla que aparece en la figura 9-9.

Figura 9-9



El formulario Cliente con todos los campos añadidos.

El paso final es asegurarse de que todos los elementos están alineados correctamente en la pantalla. Visual FoxPro tiene un nuevo juego de herramientas, localizado en la barra de herramientas Distribución, para hacer este trabajo en un chasquido. Ésta también es la razón por la que hice todas las etiquetas de la misma longitud y justificadas a la derecha. Para alinearlas todas de manera que los dos puntos al final de los indicadores estén perfectamente alineados, sostenga la tecla MAYÚS (SHIFT) y haga clic en todas las etiquetas a lo largo del lado izquierdo del formulario hasta que estén seleccionadas; luego encuentre el icono para alinear los extremos derechos y haga clic en él. Ha terminado.

Propiedades y métodos de los formularios

Al igual que los cuadros de texto y etiquetas, los formularios tienen un conjunto de eventos y propiedades. El primero en ocurrir es el evento Load. Usted desea asegurarse de que el formulario tiene abierta y seleccionada la tabla Cliente, y el evento Load del formulario es el lugar para hacerlo.

Primero, asegúrese de que el formulario está seleccionado. Haga clic en cualquier parte del formulario que no sea un control. El nombre Cliente debe aparecer en el cuadro de lista Objetos en la parte superior de la ventana Propiedades. Si no sucede así, haga clic en el cuadro de lista y elija el elemento de la parte superior de la misma. Si olvidó ponerle un nombre al formulario Cliente, hágalo ahora.

Ahora va a necesitar unas cuantas variables locales: una para recordar en qué número de registro estaba antes de intentar añadir un nuevo registro y otra para decir a las rutinas Guardar y Cancelar que está en el proceso de añadir un registro. En FoxPro 2.6 usted utilizó variables locales. En Visual FoxPro usted crea dos propiedades de formulario, Cliente.ÚltimoRegistro y Cliente.Adicionando. Puede utilizar éstas de la misma manera que utilizó las variables de memoria.

¿Por qué no puede asignar simplemente UltimoRegistro y añadir valores en Init, como podía hacerlo en el código de inicio en FoxPro 2.6? El motivo es que Load, Init y Activate se llaman como procedimientos del formulario, por lo que cualquier variable de memoria creada en ellos no es visible para el resto del formulario.

Haga clic en el menú Formulario, luego seleccione Agregar propiedad y escriba UltimoRegistro en la ventana Nueva propiedad, como se muestra en la figura 9-10. Repita los pasos para añadir otra propiedad de formulario llamada Añadir. Puede utilizar el cuadro Descripción para documentar el propósito de la propiedad.

También puede añadir métodos a un formulario. Éste es un buen momento para considerar la diferencia entre eventos y métodos. El conjunto de eventos de Visual FoxPro está fijo; no se le puede agregar nada. Cada evento tiene una ventana de código correspondiente, donde usted puede escribir el código que desea ejecutar

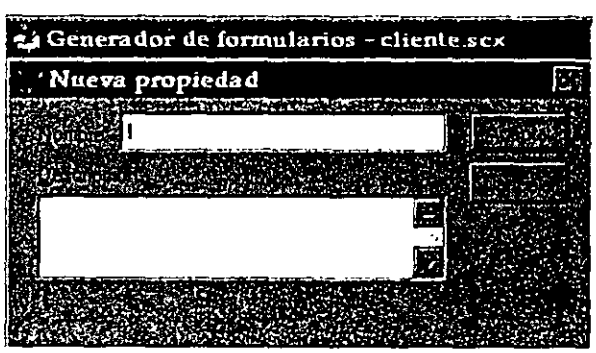


Figura 9-10

Agregue un nombre de propiedad a un formulario.

cada vez que el evento (por ejemplo, Load o Unload Form) tiene lugar. Por otro lado, puede ejecutar métodos cada vez que lo desee usando simplemente su nombre como un comando, por ejemplo, ThisForm.Refresh. Esto se aplica no sólo al código que corresponde a los eventos, sino también a los métodos que no tienen un evento correspondiente (como Show y Hide) y métodos que usted ha escrito. Son como funciones y procedimientos locales, excepto que la sintaxis es un poco diferente.

Por el momento, añada el código en el evento Load necesario para asegurar que la tabla está abierta y seleccionada. Haga clic en la página Métodos de la ventana Propiedades, y luego haga doble clic en Load. Cuando la ventana del código se abra, escriba lo siguiente:

```
IF NOT USED ( "CLIENTE" )
   USE CLIENTE IN 0
ENDIF
SELECT CLIENTE
SET ORDER TO CODIGOCLIE
=CURSORSETPROP("Buffering", 3)
```

La última línea es esencial. Modo 3, buffering en filas, se llama "bloqueo de registro optimista", y significa que Visual FoxPro hará por usted lo siguiente:

- Cuando usted añade un registro en blanco o cambia los contenidos de campos en un registro existente, los cambios son guardados en la memoria sólo hasta que usted, o bien salta a otro registro, o ejecuta el comando TableUpdate( ). Si usted, por el contrario, ejecuta el comando TableRevert( ), sus cambios son desechados.
- Si usted guarda sus datos, Visual FoxPro cierra el registro automáticamente, aplica los cambios y abre el registro.
- En un entorno de usuarios múltiples donde se está compartiendo un archivo, usted debe cerrar y abrir el registro mientras escribe en él. El buffering por filas lo hace de manera transparente y automática.

Existen otras formas de tratar con el bloqueo de registros y archivos, pero para el 99 por ciento de sus necesidades ésta es suficiente. Ahora que ha asegurado que el archivo Cliente está abierto y seleccionado, puede añadir los controles.

Agregue controles

Necesita un juego de controles para manejar los formularios de "archivo simple". Yo utilizo éstos:

Siguiente Un registro adelante.

Anterior Un registro atrás.

Buscar Encontrar un registro.

Agregar Añadir un nuevo registro.

Modificar Cambiar los datos del formulario.

Eliminar Quitar datos actuales de la tabla.

Salir Cerrar el formulario y abandonar.

Haga clic en el icono Botón de comando de la barra de herramientas Controles, luego añada un botón de comando. Normalmente, tendría que seleccionar el tipo de control cada vez que deseara poner otro en el formulario. Pero si hace clic en el último icono de la barra de herramientas Controles de formularios, el botón se bloquea, y entonces el icono selector Botón de comando permanecerá seleccionado. La próxima vez que usted haga clic en el formulario, será añadido otro botón de comando. Siguiente y Anterior son fáciles de codificar:

```
* Siguiente
IF NOT EOF()
   SKIP
   IF EOF()
      GO BOTTOM
   ENDIF
ENDIF
THISFORM.Show
THISFORM.Botones

* Anterior
IF NOT BOF()
   SKIP -1
   IF BOF()
      GO TOP
   ENDIF
ENDIF
THISFORM.Show
THISFORM.Botones
```

Cada uno de éstos va en la ventana del código Click para el botón de comando correspondiente. El método Show es necesario porque Visual FoxPro no vuelve a exhibir los objetos en una forma simplemente porque usted ha cambiado los datos a los que apuntan los objetos.

El método Botones al que se refiere el listado anterior es un método definido por el usuario, que se muestra en el siguiente listado. Controla la habilitación de los botones de los comandos Siguiente y Anterior, basado en si existe o no un registro anterior. Primero intenta saltar hacia adelante; si no hay un lugar a donde ir, inhabilita la opción Siguiente. Luego regresa a su punto de partida y trata de saltar hacia atrás, inhabilitando B_Anterior si el registro actual está en la parte superior del archivo.

```
SalvaReg = recno ()
IF EOF()
   THISFORM.B_Siguiente.Enabled = .F.
  ELSE
   THISFORM.B_Siguiente.Enabled = .T.
   SKIP
   IF EOF()
      THISFORM.B_Sigiente.Enabled = .F.
      GO BOTTOM
   ENDIF
ENDIF
IF BETWEEN ( SalvaReg, 1, RECCOUNT())
   GO ( SalvaReg )
ENDIF
IF BOF()
   THISFORM.B_Anterior.Enabled = .F.
  ELSE
   THISFORM.B_Anterior.Enabled = .T.
   SKIP -1
   IF BOF()
      THISFORM.B_Anterior.Enabled = .F.
      GO TOP
   ENDIF
ENDIF
IF BETWEEN ( SalvaReg, 1, RECCOUNT() )
   GO ( SalvaReg )
ENDIF
```

El comando Buscar puede ser tan simple o tan complejo como usted lo desee. Por el momento, puede utilizar Browse con un pequeño truco para elegir el registro que está resaltado cuando el usuario oprime ENTER o hace clic en una línea con el botón primario del ratón:

145

```
THISFORM.UltimoRegistro.Value = RECNO()
ON KEY LABEL ENTER       KEYBOARD CHR(23)
ON KEY LABEL RIGHTMOUSE KEYBOARD CHR(23)
BROWSE NOMODIFY NOAPPEND
ON KEY LABEL ENTER
ON KEY LABEL ENTER RIGHTMOUSE
IF LASTKEY() = 27
   GO  ( THISFORM.UltimoRegistro )
ENDIF
THISFORM.Show
THISFORM.Botones
```

Si el usuario oprima Esc, usted necesita regresarlo al registro en el que estaba antes de que empezara a vagar alrededor de la tabla. Puesto que Browse mueve el puntero de registros, usted debe recordar su lugar anterior. En este contexto, la propiedad UltimoRegistro del formulario se comporta exactamente como una variable de memoria local. Advierta que tiene que referirse específicamente al valor de la propiedad.

Puede hacer que esto se vea mejor añadiendo el cambio de una línea Browse a lo siguiente:

```
DEFINE WINDOW BROWSER FROM 3,10 TO 40, 90
BROWSE                                          ;
     WINDOW BROWSER                             ;
     NOMODIFY NOAPPEND                          ;
     FIELDS                                     ;
       Nombre=TRIM(Nombre)+[ ] + Apellido:30,   ;
       Direccion = Direccion1:30,               ;
        Telefono casa = Telefono1
RELEASE WINDOW BROWSER
```

On Key Label se utiliza aquí para alterar temporalmente el uso de la tecla ENTER y el botón secundario del ratón. CHR(23) es lo mismo que presionar CTRL-W, en xBASE para "He terminado". En el instante en que el usuario oprime ENTER o hace clic con el botón primario del ratón, Browse está concluido.

Modificar es un poco distinto. Todos los campos en el formulario están inhabilitados, por lo que necesita habilitarlos. En la documentación de Visual FoxPro se indica que usted puede inhabilitar todos los controles en un formulario con el siguiente código:

```
FOR I = 1 TO THISFORM.ControlCount
   THISFORM.Control(I).Enabled = .F.
ENDFOR
```

Pero esto es cierto sólo si todos los objetos en el formulario tienen una propiedad Enabled. Si usted pone líneas, formas u otros objetos que no pueden ser habilitados o inhabilitados en el formulario, este código estallará porque ThisForm.Shape1 no tiene una propiedad Enabled. Sin embargo, si usted ha utilizado un prefijo como I_ para todos los campos de entrada, puede usar lo siguiente:

```
FOR I = 1 TO THISFORM.ControlCount
   IF THISFORM.Controls(I).Name = [I_]
      THISFORM.Controls(I).Enabled = .T.
   ENDIF
ENDFOR
```

O para habilitar solamente botones de comando (por ejemplo), utilice esto:

```
FOR I = 1 TO THISFORM.ControlCount
   IF THISFORM.Controls(I).Name = [B_]
      THISFORM.Controls(I).Enabled = .T.
   ENDIF
ENDFOR
```

Por supuesto, usted puede utilizar el método SetAll para habilitar o inhabilitar cualquier cosa dentro del formulario:

```
FORM1.SetAll ( "Enabled", .T. )
```

o todos los controles de un cierto tipo:

```
FORM1.SetAll ( "Enabled", .F., "TextBox" )
```

pero puede ser que aún necesite caracterizar sus campos de una manera distinta a como lo hace Visual FoxPro; si es así, lo que usted está buscando son convenciones para nombrar. Por cierto, Microsoft recomienda CMD para botones de comando, TXT para campos de texto y así consecutivamente. Eso también es correcto.

Ahora que sabe cómo habilitar todos los campos de entrada e inhabilitar todos los botones de comandos, añada dos métodos al formulario. Haga clic en cualquier parte del formulario y vea el nombre en el cuadro combinado ObjetoActual para verificar que el formulario mismo sea el objeto seleccionado. Luego haga clic en Formulario, Nuevo método en el menú y escriba HabilitarTodo. Luego escriba el siguiente código:

```
FOR I = 1 TO THISFORM.ControlCount
   IF THISFORM.Controls(I).Name = [I_]
      THISFORM.Controls(I).Enabled = .T.
   ENDIF
ENDFOR
```

Añada otro método, InhabilitarTodo, y cópielo en el mismo código, cambiando la .T. en la tercera línea a .F. Mientras introduce datos, sólo debe tener dos botones de comandos accesibles: Guardar, para guardar cambios y agregados, y Cancelar, para deshacer cambios. Así que necesita dos métodos de formulario más. Utilice Formulario, Agregar método para agregar el nombre BotonesEn, y escriba esto:

```
FOR I = 1 TO THISFORM.ControlCount
   IF THISFORM.Controls(I).Name = [B_]
        THISFORM.Controls(I).Enabled = .T.
      ENDIF
ENDFOR
```

Copie el código en otro método llamado BotonesApagado y cambie la .T. en la tercera línea a .F. El comando Modificar ahora está codificado así:

```
Editando = .T.
THISFORM.HabilitarTodo
THISFORM.BotonesApagado
THISFORM.B_Guardar.Enabled = .T.
THISFORM.B_Cancelar.Enabled = .T.
THISFORM.I_CodigoCli.Enabled = .F.
THISFORM.I_Apellido.GotFocus
```

Advierta que inhabilité el código del cliente, que no quiero que cambien los usuarios, y moví el enfoque al primer objeto modificable, I_Apellido. Agregar es parecido, excepto por que CodigoClie también es modificable:

```
THISFORM.SalvaReg = RECNO()
THISFORM.Adicionando = .T.
APPEND BLANK
THISFORM.Show

Editando = .T.
THISFORM.HabilitarTodo
THISFORM.BotonesApagado
THISFORM.B_Salvar.Enabled = .T.
THISFORM.B_Cancelar.Enabled = .T.
THISFORM.I_CodigoClie.GotFocus
```

Valide el código del cliente

Cuando se introduce un código de cliente, yo deseo que suceda una de tres cosas:

- Si el nombre y la dirección ya han sido llenados, no hacer nada.
- Si el nombre y la dirección están vacíos y los usuarios escriben un código de cliente correctamente, llene los campos del nombre y la dirección en la pantalla.
- Si el nombre y la dirección están vacíos y los usuarios escriben un código de cliente que no hace juego exacto con nada en la tabla del cliente, exhiba la tabla del cliente como una lista para escoger, empezando cerca de lo que escribieron. Si resaltan un nombre y oprimen ENTER o hacen clic en un nombre con el botón primario del ratón, utilícelo para llenar los campos de nombre y dirección.

El siguiente código en la ventana de código del evento Valid del objeto CodigoClie hace precisamente eso:

```
SELECT CLIENTE
SET ORDER TO CodigoClie
SEEK THISFORM.I_CodigoClie.Value
DO CASE
   CASE FOUND()
   THISFORM.I_Nombre.Value    TRIM(CLIENTE.Nombre);
                      +[ ] +  TRIM(CLIENTE.Apellido)
   THISFORM.I_Direccion.Value    = CLIENTE.Direccion1
   THISFORM.I_Ciudad.Value       = CLIENTE.Ciudad
   THISFORM.I_Estado.Value       = CLIENTE.Estado
   THISFORM.I_CodPost.Value          = CLIENTE.CP
   THISFORM.I_Telefono1.Value    = CLIENTE.Telefono1
   THISFORM.I_Pais.Value         = CLIENTE.Pais
   CASE NOT FOUND()
      DEFINE WINDOW BROWSER FROM 1,1 TO 30, 100
      BROWSE WINDOW BROWSER NOMODIFY NOAPPEND
      RELEASE WINDOW BROWSER
         IF LASTKEY() <> 27
         THISFORM.I_CodigoClie.Value    = CLIENTE.CodigoClie
         THISFORM.I_Nombre.Value     = TRIM(CLIENTE.Nombre);
                              +[ ] + TRIM(CLIENTE.Apellido)
         THISFORM.I_Direccion.Value = CLIENTE.Direccion1
         THISFORM.I_Ciudad.Value    = CLIENTE.Ciudad
         THISFORM.I_Estado.Value    = CLIENTE.Estado
         THISFORM.I_CodPost.Value   = CLIENTE.CodPost
         THISFORM.I_Telefono1.Value = CLIENTE.Telefono1
         THISFORM.I_Pais.Value      = CLIENTE.Pais
```

```
   ENDIF
ENDCASE
SELECT ORDENES
RETURN .T.
```

El comando Guardar puede utilizar entonces la función TablaActualizar( ) para hacer cualquier cambio en la pantalla permanente:

```
=TablaActualizar()
Editando = .F.
THISFORM.Adicionando = .F.
THISFORM.InhabilitarTodo
THISFORM.BotonesEncendido
THISFORM.Botones
THISFORM.B_Guardar.Enabled = .F.
THISFORM.B_Cancelar.Enabled = .F.
THISFORM.B_Siguiente.GotFocus
```

El código del botón Cancelar utiliza TablaRevertir( ), y se encarga de la complicación adicional de regresar el apuntador del registro a donde estaba antes de que Agregar anexara un espacio en blanco.

```
=TablaRevertir()
IF THISFORM.Adicionando
   GO ( THISFORM.SalvaReg )
ENDIF
Editando = .F.
THISFORM.Adicionando = .F.
THISFORM.Show
THISFORM.InhabilitarTodo
THISFORM.BotonesEn
THISFORM.Botones
THISFORM.B_Guardar.Enabled = .F.
THISFORM.B_Cancelar.Enabled = .F.
THISFORM.B_Siguiente.GotFocus
```

El botón Borrar es bastante directo, y le permite echar un vistazo a la función MessageBox, que le da a su aplicación un aspecto Windows bonito y agradable:

```
IF MESSAGEBOX ( "¿Borrar este registro?", 4 + 32 + 256 ) = 6
   DELETE NEXT 1
   BLANK
   SET DELETED ON
   GO TOP
   THISFORM.Show
ENDIF
```

Los parámetros MessageBox aparecen en una lista en el apéndice al final de este libro. 4 significa que Sí y No son las dos opciones, 32 es el icono a exhibir en el recuadro (un signo de interrogación), y 256 indica que el valor predeterminado que será escogido si los usuarios presionan ENTER sin leer el mensaje en la pantalla, cosa que seguramente harán, es el segundo botón, No.

El comando Blank, que es nuevo en Visual FoxPro, es el equivalente de las cuatro líneas de código siguientes:

```
SCATTER MEMVAR BLANK
=RLOCK()
GATHER MEMVAR
UNLOCK
```

Set Deleted On asegura que el registro borrado no aparezca en la pantalla, y Go Top asegura que, cuando llame al método Show en la última línea, tendrá algo que mostrar.

Ahora ejecute el formulario. Intente modificar, agregar y eliminar. Creo que estará de acuerdo en que esta interfaz cubrirá las necesidades de la mayoría de los usuarios. Aun si es sólo un prototipo, es un formulario bastante utilizable para un trabajo de menos de una hora.

Construya el formulario Libros

La tabla Libros tiene requerimientos ligeramente distintos. Para empezar, tiene una región de edición que contiene una descripción extensa del libro. Puesto que uno no puede juzgar un libro por su portada, estas descripciones ayudan a los clientes a entender el auditorio preciso del volumen particular.

Además, aunque es razonable buscar a un cliente con el Código del Cliente, que será probablemente una combinación del apellido y el primer nombre, los libros se identifican únicamente por su Número Internacional de Libro Estándar, o ISBN, que muy poca gente conocerá al buscar un libro. Los títulos ni siquiera son particularmente útiles, porque los lectores con frecuencia recuerdan un título incorrectamente, o recuerdan sólo unas cuantas palabras claves de él.

En xBASE hay un comando substring, invocado con el carácter del signo de pesos, $. Usted puede crear una lista de todos los títulos que contengan la palabra 'fue' utilizando esta sintaxis:

151

```
LIST TITLE FOR 'fue' $ Title
```

o, mejor aún, escriba:

```
LIST TITLE FOR 'FUE' $ UPPER(title)
```

Sin embargo, aun en una computadora extremadamente rápida, este tipo de búsqueda puede llevarse medio minuto. Y si usted añade dos palabras clave, por ejemplo:

```
LIST TITLE FOR 'FUE' $ UPPER(title) AND 'VIENTO' $ UPPER(title)
```

es muy probable que haya podido leer el libro que es objeto de la búsqueda para cuando éste aparezca en su pantalla.

Por esta última razón, se añadió una búsqueda de palabra clave al archivo de títulos. Las palabras clave se forman extrayendo todas las palabras significativas del título, poniéndolas en mayúsculas y almacenándolas en una tabla separada Keywords junto con el ISBN del título. De esa forma usted puede permitir que los usuarios busquen una o dos palabras clave, visualicen el subjuego usual de títulos que tienen cada una de las palabras clave que fueron introducidas, y elijan de entre lo que ven.

La búsqueda de palabras clave será llamada desde una cláusula válida que parece un signo de interrogación en los campos cuadrícula ISBN o Título del formulario Ordenes, utilizando el siguiente código:

```
IF THIS.Name <> [?]
   RETURN
ENDIF
DO FORM Enconlib
```

Verá el formulario Enconlib de búsqueda por palabras clave después. Por el momento, se diseñó la pantalla Libros. Usted podría simplemente ir al Administrador de proyectos, hacer clic en Formulario, Nuevo y empezar a escribir, pero entonces tendría que volver a escribir o al menos copiar todos los métodos que escribió para el formulario Cliente. Limpie este formulario y haga de él una plantilla.

Ponga el puntero del ratón en la parte inferior derecha del último campo del formulario antes de los botones de comando, presiónelo y expanda el área rectangular hasta que contenga cada campo de entrada en la pantalla. Luego suelte el botón del ratón y verá que todos los campos de entrada han sido

etiquetados. Ahora sostenga la tecla MAYÚS (SHIFT) y haga clic en la etiqueta Código del cliente y en los campos de ingreso para quitarles la etiqueta. Presione la tecla SUPR, haga clic en la ficha Distribución, luego en Encabezado, y escriba Nuevo.

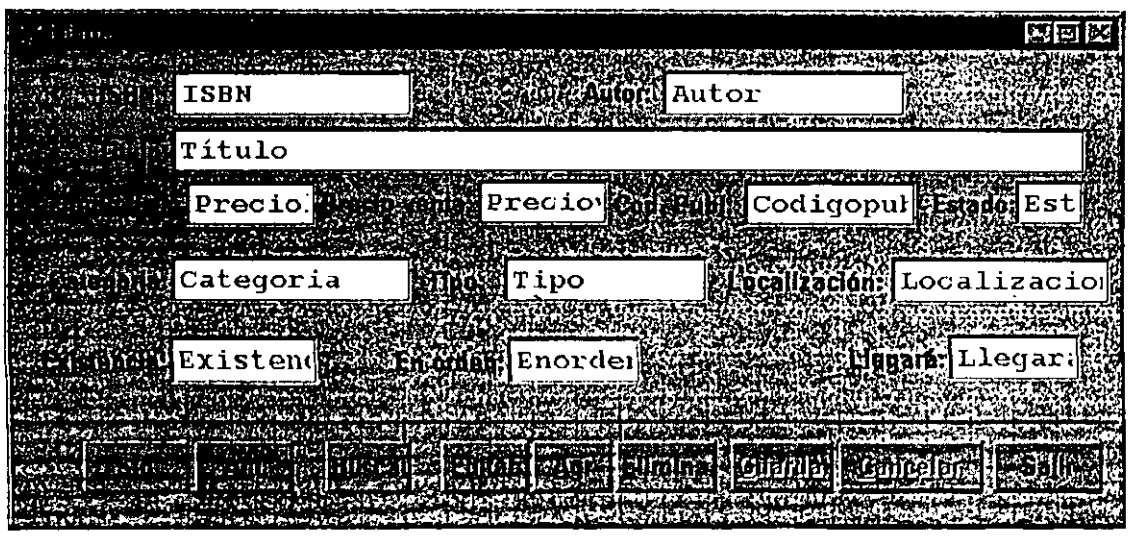
Haga clic en Guardar como clase y nombre al archivo Master. Luego haga clic en Herramientas, Opciones y seleccione la ficha Formularios. En la parte inferior izquierda, bajo Plantillas de clases, haga clic en Formulario, y luego elija Master de la lista de clases visuales (archivos .VCX) en el menú desplegable. Haga clic en establecer como predeterminado y luego en Aceptar.

Ahora cuando haga clic en Formulario, Nuevo, obtendrá la plantilla en lugar de tener que empezar con una forma en blanco. Guárdela como Libros, y ya podrá empezar.

Cambie la primera etiqueta a ISBN:, el primer nombre para campo de ingreso a I_ISBN y la fuente de datos a CLIENTE.ISBN. Ahora repita el proceso de añadir todos los campos de la tabla hasta que tenga el formulario que se muestra en la figura 9-11.

Para el formulario Libros, haga clic en cualquier parte de él con el botón secundario de ratón y abra el entorno de datos. Luego haga clic de nuevo con el botón secundario del ratón y añada la tabla Libros. Haga clic una vez más con el botón secundario del ratón, seleccione Propiedades y establezca BufferMode Override en 3 y Orden en ISBN. Ahora ¡ya no necesita introducir ningún código al evento Load! Luego haga clic en Formulario, Ejecutar, y pruébelo. Puesto que fue capaz de utilizar todo el código desde el formulario Cliente, esto le tomó aún menos tiempo.

Figura 9-11



El formulario Libros.

153

Agregue el programa Principal

El programa Principal es casi idéntico al que usted desarrolló en el capítulo 3. Se han añadido unos cuantos detalles, de los que se hablará más adelante. Aquí está el programa Principal:

```
#INCLUDE C:\VFP\FOXPRO.H
SET TALK OFF
SET SAFETY OFF
SET CONFIRM ON
SET STATUS BAR OFF
CLEAR
=OpenTabs()
PUSH MENU _MSYSMENU
DO MENU.MPR
DO FORM BACK
ON ERROR DO ERRTRAP WITH PROGRAM(), MESSAGE(), MESSAGE(I)
READ EVENTS
ON ERROR
CLEAR WINDOW
CLOSE DATABASE
CLOSE ALL
CLEAR ALL
POP MENU _MSYSMENU
```

Utilice un archivo Incluir

Visual FoxPro incluye un archivo que contiene los códigos utilizados por muchas funciones internas. Por ejemplo, la función MessageBox( ) regresa un valor de 6 si el usuario presiona el botón Sí. En el archivo FOXPRO.H encontrará la afirmación:

```
#DEFINE IDYES 6
```

Esto es más fácil de entender:

```
IF MESSAGEBOX("¿Continuar?",;
MBYesNo + MBIconQuestion + MB_DEFButton3 ) = IDYes
```

que leer esto:

```
IF MESSAGEBOX("¿Continuar?",4+32+256) = 6
```

Usted puede utilizar archivos Incluir con formularios individuales. Si abre un formulario y hace clic en el menú Formulario, verá la opción de menú Archivo Incluir. Yo puse el archivo Incluir en el programa Principal en lugar de los formularios porque retrasa notablemente el almacenamiento en los formularios. Esto puede resultar diferente en la versión de Visual FoxPro con la que usted está trabajando.

Atrape errores

Atrapar y manejar errores es importante en programación. Cuando usted está escribiendo código, a veces las cosas salen mal. Si tiene varios errores en una fila, puede ser difícil recordar qué sucedió. Además, cada vez que el programa estalla, hay una serie de comandos que con frecuencia son necesarios para restaurar su entorno de programación. No es nada divertido escribirlos de nuevo una y otra vez. Es mejor escribir un manipulador de errores que por un lado almacene mensajes de error en un archivo y por el otro le ofrezca restaurar su entorno. El manipulador de errores que yo utilizo ahora es como se muestra a continuación:

```
FUNCTION ATRAPARERROR
PARAMETERS PROG, ERRMSG, CODE
GuardarAlias = ALIAS()
msglen     = 60
WAIT WINDOW
          [Error: ] + LEFT(ERRMSG+SPACE(msglen),msglen) + CHR(13)  ;
        + [ Code: ] + LEFT(CODE  +SPACE(msglen),msglen) + CHR(13)  ;
        + [ (Presione C para cancelar, ]                           ;
        + [ D para depurar, ]                                      ;
        + [ o Enter para continuar )] TO keyin
IF keyin $ [CdDd]
   IF TXNLEVEL() > 0
      WAIT WINDOW [ Transacción cancelada (Presione ENTER) ]
      ROLLBACK
   ENDIF
ENDIF
IF NOT FILE ("ERRORLOG.DBF")
   CREATE TABLE ERRORLOG    ;
   ( DATE D(8),             ;
     TIME C(5),             ;
     PROGRAM C(100),        ;
     MESSAGE C(100),        ;
     BADCODE C(100)         )
ENDIF
INSERT INTO ERRORLOG VALUES (DATE(), TIME(), PROG, ERRMSG, CODE)
IF NOT EMPTY (GuardarAlias )
   SELECT    (GuardarAlias )
ENDIF
DO CASE
```

```
   CASE keyin $ [Cc]
      ON ERROR
        POP MENU _MSYSMENU
        CANCEL
   CASE keyin $ [Dd]
        ON ERROR
        POP MENU _MSYSMENU
        SET STEP ON
ENDCASE
```

Para llamarlo, incluya la línea:

```
ON ERROR DO ATRAPARERROR WITH PROGRAM(), MESSAGE(), MESSAGE(1)
```

en su programa Principal. Unas cuantas explicaciones estarán bien. GuardarAlias guarda el nombre del archivo que fue seleccionado cuando tuvo lugar el error. Puesto que el programa ATRAPARERROR pudo haber tenido que crear una nueva tabla ErrorLog, y crear una tabla va a la primera área de trabajo que aún no está en uso, tendrá que proveer una manera de regresar en caso de que desee continuar en donde abandonó el programa. Algunos errores no son fatales.

Msglen = 60 está ahí porque existe un límite al tamaño máximo de un mensaje en una ventana de espera.

Si TxnLevel( ) (nivel de transacción) no es cero, hay una instrucción Begin Transaction aún viva, y tendrá que cancelarla con un Rollback si no va a continuar con el programa.

Si elige Cancelar o Depurar, necesitará de nuevo el menú de Visual FoxPro, por lo que en ambos casos yo incluí un comando Pop Menu_MSysMenu. Si usted tiene una extensión del menú del programador en MSysMenu, esto lo regresará; si no es así, o existe la posibilidad de que tenga una profundidad de más de un menú, utilice Set SysMenu To Default. Después de todo, el propósito de esta línea de código es asegurar que el menú Herramientas con las opciones de ventanas de Seguimiento y Depuración sean visibles. De otra forma, depurar es prácticamente imposible.

ATRAPARERROR exhibe el mensaje de error al igual que la línea de código que produjo el error en una ventana en la pantalla, y luego le da varias opciones. Podría construir algo muy similar a esto utilizando MessageBox:

```
Meg =  [Error: ] + LEFT(ERRMSG+SPACE(msglen),msglen) + CHR(13);
     + [Codigo:] + LEFT(CODE  +SPACE(msglen),msglen)
RetCode = MESSAGEBOX ( Msg, 2 + 48 + 512, "ERROR" )
```

```
DO CASE
   CASE RetCode = 3    && Abortar
   CASE RetCode = 4    && Reintentar
   CASE RetCode = 5    && Ignorar
ENDCASE
```

Los mensajes de error son almacenados en un archivo llamado ERRORLOG. Si éste ha sido borrado, simplemente crea uno nuevo. ¿Recuerda la rutina en el capítulo 8 para añadir una hoja de programador al sistema de menús, con una opción para Ver Archivo Bitácora Error? Esto es lo mismo.

Agregue el menú

Ahora, añada un archivo de menús justo como aquél del capítulo 3, excepto que esta vez tendrá unos cuantos menús más:

- Libros
- Introducir Orden
- Reconstruir palabras clave

Ya ha hecho las pantallas Libros y Cliente. El programa Palabcve se ejecuta como una opción de menú separada, porque este capítulo no incluye la lógica para mantener automáticamente el archivo de las palabras clave cuando se agregan, modifican o borran libros. No es difícil, pero no quiero que usted sufra de un caso de "demasiado; muy pronto". Aquí está lo que hace la opción Crear palabras clave:

```
PROCEDURE adcvepal
* Crea las palabras clave de los títulos
IF FILE ( "KEYWORDS.DBF" )
   IF USED ( "KEYWORDS" )
        USE IN KEYWORDS
   ENDIF
   DELETE FILE KEYWORDS.DBF
ENDIF
IF FILE ( "KEYWORDS.CDX" )
   DELETE FILE KEYWORDS.CDX
ENDIF
CREATE TABLE KEYWORDS ( KEYWORD C(20), KEYISBN C(10) )
PRIVATE i
DIMENSION word(30)
STORE SPACE(10) TO word
```


157

```
nwords = 1
forbidden = ;
        "Y / O / EL / A / DE / PARA / EN / POR / CON / LO / LA / QUE "
SELECT LIBROS
=InitTermo()
SCAN
   =MosTermo()
   DO parser
   I = 1
   DO WHILE i <= nwords
       IF NOT ( word(I) $ forbidden ) AND NOT EMPTY ( word(I) )
          INSERT INTO KEYWORDS VALUES ( word(i), LIBROS.isbn )
       ENDIF
       I = i + 1
   ENDDO
   SELECT LIBROS
ENDSCAN
=QuiTermo()
DEFINE WINDOW PROGRESS FROM 5,5 TO 25, 70 DOUBLE SHADOW ;
   TITLE [ Creando índices de palabras clave ]
ACTIVATE WINDOW PROGRESS
SET TALK ON WINDOW PROGRESS
SELECT KEYWORDS
INDEX ON KEYWORD TAG KEYWORDS
INDEX ON KEYISBN TAG KEYISBNS
SET TALK OFF
RELEASE WINDOW PROGRESS

PROCEDURE parser
STORE " " TO word
tstring = UPPER ( TRIM ( LIBROS.TITULO ) )
tlen = LEN(TRIM(tstring))
i = 1
string = " "
DO WHILE i <= tlen
   string = string + ;
     IIF  ( SUBSTR(tstring,i,1) $ [#&(:),.] , [ ], SUBSTR(tstring,i,1) )
   i = i + 1
ENDDO
nwords = 0
DO WHILE .T.
   endword = AT ( " " , string ) - 1
   IF endword = -1
      endword = LEN ( string )
   ENDIF
   nwords = nwords + 1
   word ( nwords ) = SUBSTR ( string , 1 , endword )
   IF ( (endword+2) > LEN(string) ) OR ( LEN(string) = 0 )
       EXIT
   ENDIF
   string = SUBSTR ( string , endword + 2 )
   string = LTRIM(string)
ENDDO
```

```
* ProgramaID....: InitTermo.PRG
* Propósito...: Inicialización para llamar a la rutina termómetro
PARAMETERS vTitulo
SET COLOR OF SCHEME 17 TO ;
 W+/BR, W+/BR, W+BR, W+BR, W+BR, W+BR, W+BR, N/N
SET TALK OFF
IF PARAMETERS() = 0
   TitlVent = [ Progreso ]
 ELSE
   TitlVent = vTitulo
ENDIF
IF TYPE ( [MaxTermo] ) = [U]
   IF NOT EMPTY ( DBF () )
        PUBLIC MaxTermo
        MaxTermo = RECCOUNT()
     ELSE
        WAIT WINDOW ;
        [ No se ofreció un valor a la rutina termómetro para contar hasta ] ;
        + [ (Presione Enter) ]
        RETURN
    ENDIF
ENDIF
IF TYPE ( [ContTermo] ) = [U]
   PUBLIC  ContTermo
   ContTermo = 0
ENDIF
DEFINE WINDOW TERMO           ;
   FROM 18, 8 TO 24, 72       ;
        DOUBLE                ;
        SHADOW                ;
        COLOR SCHEME 17       ;
        TITLE ( TitlVent )
ACTIVATE WINDOW TERMO
@ 0, 1 SAY '....10....20....30....40....50....60....70....80....90....100'
RETURN

* ProgramaID.....: MosTermo.PRG
* Propósito.....: Despliega un termómetro para representar el progreso
* Requiere..: Llamar primero INITTERMO, cerrar con QUITERMO
PARAMETERS contador
IF TYPE ( 'ContTermo' ) = [U]
   PUBLIC ContTermo
   ContTermo = 0
ENDIF
IF PARAMETERS() <1
   ContTermo = ContTermo + 1
  ELSE
   ContTermo = contador
ENDIF
ACTIVATE WINDOW TERMO
@ 1,1 SAY REPLICATE ( 'X', ( ContTerm / MaxTerm ) * 60 ) ;
     COLOR W+/BR
```

```
* ProgramaID.....: QuiTermo
* Propósito.....: Limpia la ventana del termómetro, y
*                  deshabilita la rutina de escape

IF WEXIST('TERMO')
   RELEASE WINDOW TERMO
ENDIF
IF TYPE ( 'ContTermo' ) <> [U]
   RELEASE ContTermo
ENDIF
IF TYPE ( 'MaxTermo'   ) <> [U]
   RELEASE MaxTermo
ENDIF
ON ESCAPE
```

Usted creará un termómetro más elegante en el capítulo referente a las clases.

Agregue el formulario Ordenes

La única pieza que falta de esta aplicación es la pantalla en la que se introducen las órdenes del cliente. Éste es un formulario clásico padre-hijo. No utilicé el uno ra-muchos que genera el Asistente para formularios porque, en primer lugar, éste no le permitirá añadir al formulario líneas de detalles.

De nuevo, esto puede haber cambiado para el momento en que usted lea estas páginas. Aun así, un poco de código no le hará ningún daño.

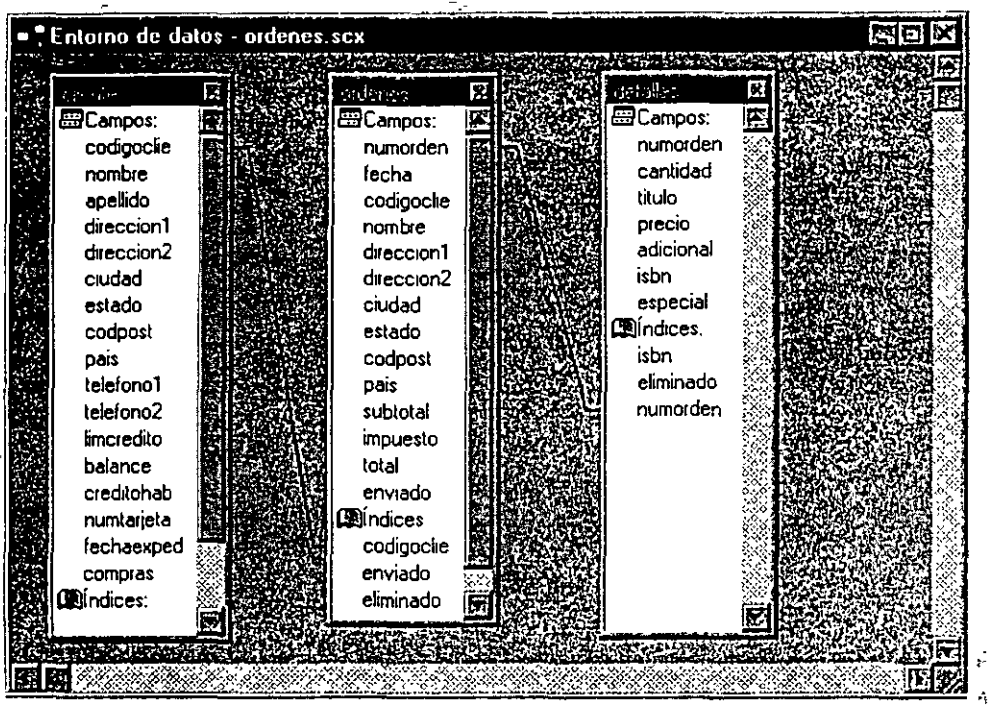
Genere el formulario básico

Haga clic en Archivo Nuevo Formulario, y obtendrá la plantilla. Cambie el título del formulario a Ordenes, y ponga las tablas Cliente, Ordenes y Detalles en el entorno de datos, como se muestra en la figura 9-12.

Ahora, cambie el encabezado de la etiqueta a Factura #: y la fuente de datos para el primer cuadro de texto del campo de entrada a Órdenes.OrdernNum. Duplique el resto de los campos en ORDENES.DBF hasta que el formulario se vea como la figura 9-13.

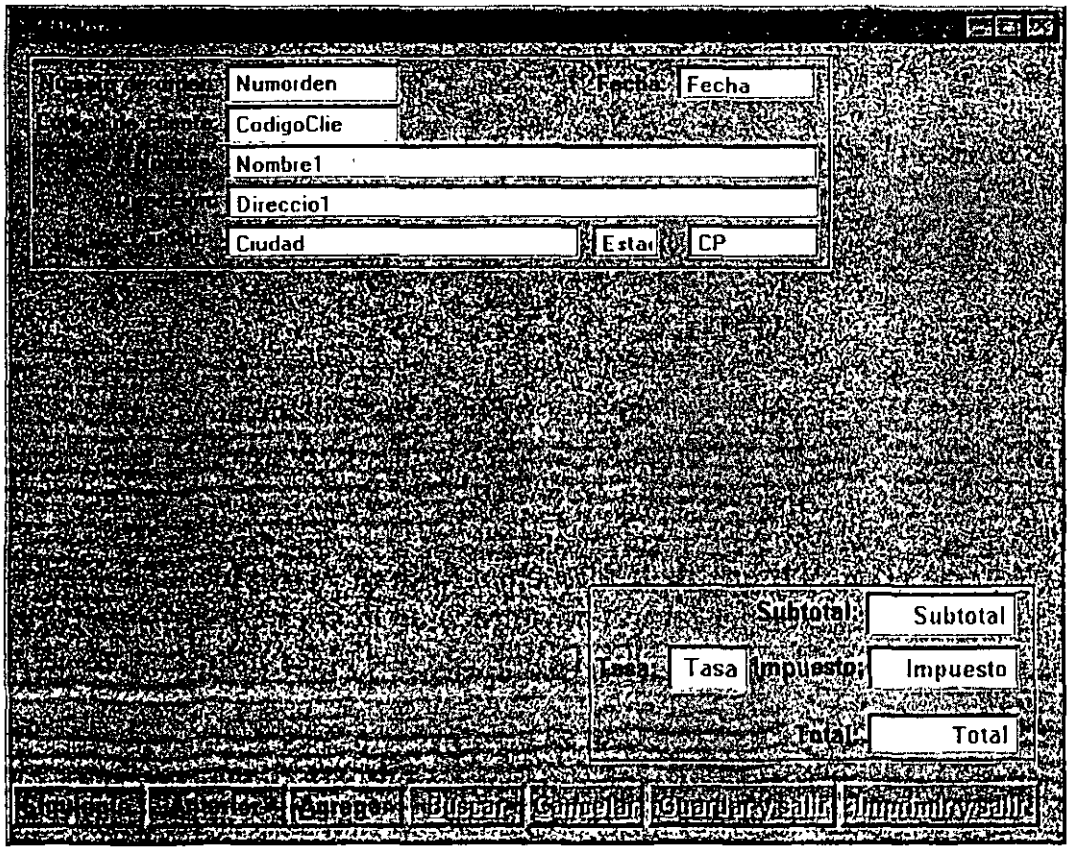
Para que el objeto cuadrícula funcione, tiene que proveer las tablas y las relaciones. Haga clic con el botón secundario del ratón, seleccione Código del menú contextual y abra la ventana del código Load Form. O haga clic en Ver,

Figura 9-12



Entorno de datos para el formulario Ordenes.

Figura 9-13



El formulario Ordenes sin el objeto cuadrícula.

Código desde el menú. (Si no está seguro sobre cómo hacer esto, practique un poco. Una sugerencia: para seleccionar la ventana del código Load Form el formulario tiene que ser el objeto seleccionado en el cuadro combinado Objeto al frente de la pantalla.) Luego escriba esto:

```
SELECT ORDENES
SET ORDER TO OrdenNum
SELECT DETALLES
SET ORDER TO OrdenNum
SET FILTER TO DETALLES.OrdenNum = ORDENES.OrdenNum
SELECT ORDENES
SET RELATION TO OrdenNum INTO DETALLES
SET SKIP TO DETALLES
```

Para coordinar estos dos archivos dentro de un objeto cuadriculado padre-hijo, el archivo hijo debe estar en el índice OrdenNum. Aquí Detalles es el archivo hijo. Ordenes también está en la orden OrdenNum, pero sólo por coincidencia. Las siguientes tres líneas:

```
USE ORDENES
SET RELATION TO OrdenNum INTO Detalles
SET SKIP TO Detalles
```

aseguran que cada vez que usted se cambie a otro registro en el archivo ordenes el objeto cuadrícula mostrará los registros de Detalles que hagan juego. Y lo siguiente:

```
SELECT DETALLES
SET FILTER TO DETALLES.OrdenNum INTO ORDENES.OrdenNum
```

asegura que sólo aparezcan registros hijos que hagan juego con el registro padre. Advierta que usted puede llevar a cabo todos estos pasos en el entorno de datos del formulario excepto para establecer un filtro en el archivo DETALLES.

Agregue el objeto cuadrícula

Haga clic en el objeto cuadrícula en la barra de herramientas Controles, luego haga clic en la esquina superior izquierda del lugar donde debería empezar el objeto cuadrícula y arrastre la esquina inferior derecha al lado derecho del formulario, llenando el espacio que dejó a un lado para la cuadrícula. Ahora haga clic en el icono Propiedades de la barra de herramientas Generador de formularios. Haga clic en Datos y escriba los elementos que se ven en la figura 9-14. Las propiedades importantes son:

- ChildOrder: OrdenNum
- LinkMaster: Ordenes
- RecordSource: Detalles
- RelationalExper: OrdenNum

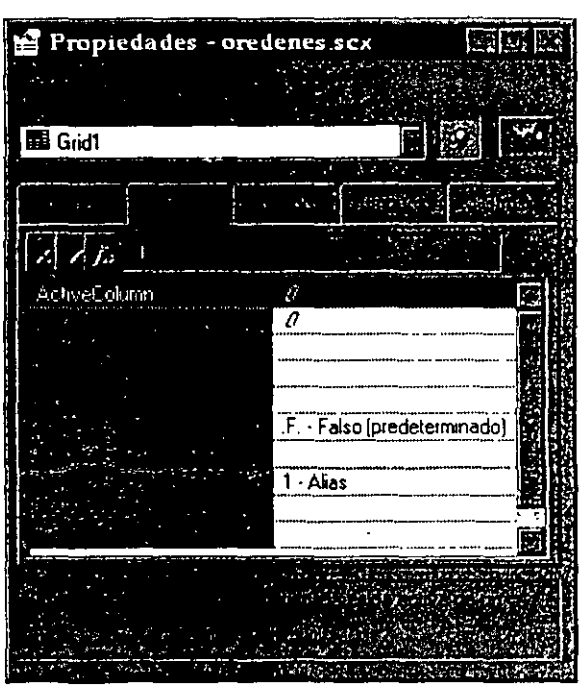
A diferencia de Browse, no tiene que hacer ningún truco para poner a funcionar una cuadrícula con su formulario. Para Visual FoxPro, es sólo otro objeto de entrada del formulario.

Cambios al código de los botones de navegación

Los tres botones que pueden llevarlo a otra orden son Siguiente, Anterior y Buscar. Debido al comando Set Skip si está apuntando a una orden con tres libros, el comando, tendría que saltar tres veces para llegar a la orden siguiente. Eso es lo que hace Set Skip; cuenta el número de registros saltados en el archivo hijo en vez del archivo padre.

Pero existe un arreglo fácil: antes de saltar, especifique Set Skip To (para desactivarlo), luego vuélvalo a activar cuando haya terminado de moverse. Aquí está el código del método correcto para estos tres controles:

Figura 9-14



Propiedades de datos para el objeto cuadrícula de Ordenes.

```
* B_Sig Código del método Click:
SELECT ORDENES
SET SKIP TO
IF NOT EOF()
   SKIP
   IF EOF()
      GO BOTTOM
   ENDIF
ENDIF
SET SKIP TO DETALLES
THISFORM.Show
THISFORM.Buttons

* B_Prev Código del método Click:
SELECT ORDENES
SET SKIP TO
IF NOT BOF()
   SKIP -1
   IF BOF()
      GO TOP
   ENDIF
ENDIF
SET SKIP TO DETALLES
THISFORM.Show
THISFORM.Buttons

* B_Busc Código del método Click:
SELECT ORDENES
SET SKIP TO
ON KEY LABEL ENTER KEYBOARD CHR(23)
BROWSE
ON KEY LABEL ENTER
SET SKIP TO DETALLES
THISFORM.Show
THISFORM.Buttons
```

Búsqueda de un libro

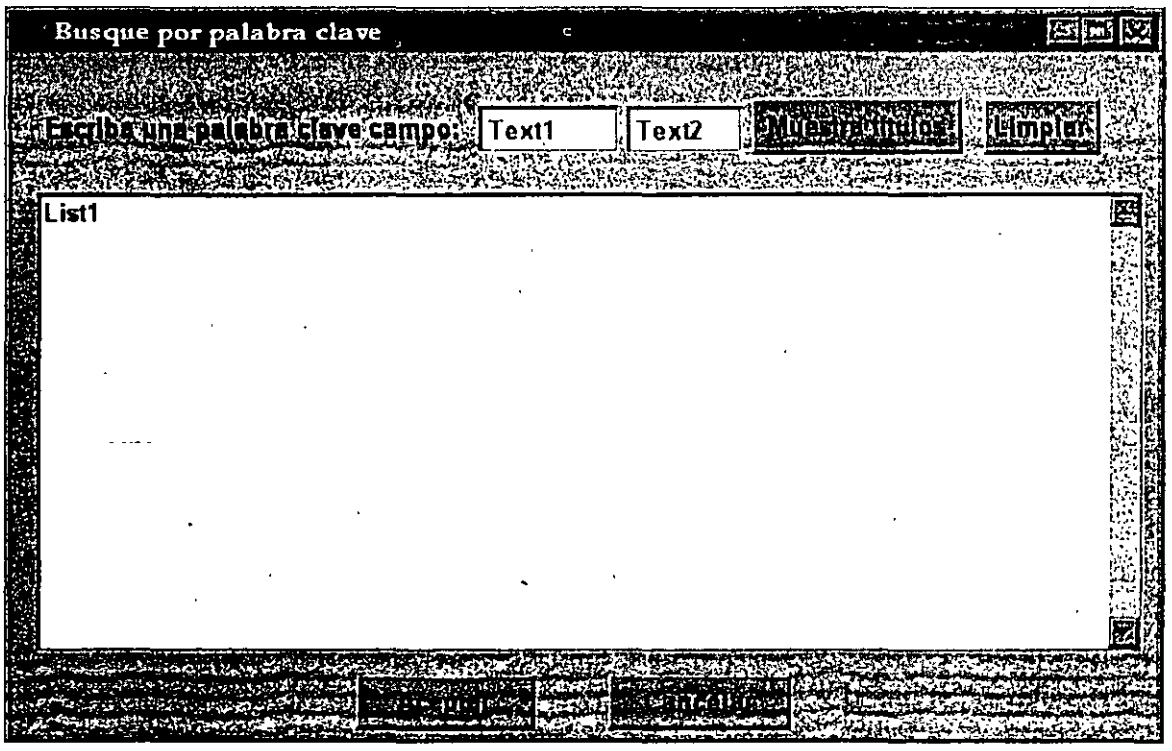
Para buscar un libro utilice un signo de interrogación ya sea en el campo del ISBN o en el de Título. ¿Por qué no dejar que el usuario simplemente escriba un ISBN? Porque los ISBN fueron inventados para las máquinas, no para las personas. ¿Por qué no hacer que los usuarios escriban las palabras clave en el campo del título y permitan que la computadora averigüe si lo que escribieron es un título o algunas palabras clave? Si usted ejecuta remotamente esta aplicación, necesitará saber cuántos títulos coinciden antes de descargar en un módem todos ellos. Por esto necesita una herramienta especializada para contar títulos que coincidan, reducir la búsqueda y minimizar el tiempo de espera.

El formulario Busque por palabra clave aparece en la figura 9-15. Éste permite a los usuarios introducir una o dos palabras clave y luego cuenta cuántos títulos contiene cada una de ellas. El botón Mostrar títulos cuenta cuántos títulos tienen las dos palabras clave, si es que se introdujeron dos, y muestra la lista de ISBN y títulos que coinciden. Limpiar limpia la lista en preparación para crear una lista nueva.

El código Load para el formulario asegura que todos los archivos necesarios estén abiertos:

```
IF NOT USED ( "LIBROS" )
   USE LIBROS IN 0
ENDIF
SELECT LIBROS
SET ORDER TO ISBN

IF NOT USED ( "KEYWORDS" )
   USE KEYWORDS IN 0
ENDIF
SELECT KEYWORDS
SET ORDER TO KEYWORDS
```

Figura 9-15



El formulario Busque por palabra clave.

Los campos de entrada Texto1 y Texto2 utilizan el evento KeyPress para buscar inmediatamente palabras clave que coinciden. Cada vez que usted da un golpe de tecla en cualquiera de estos campos, el código del método KeyPress añade la nueva tecla al contenido del campo (puesto que KeyPress sucede aún antes de que la tecla que usted presionó sea aceptada dentro del campo), busca y cuenta todos los juegos en el archivo de palabras clave (KeyWords), y exhibe la cuenta bajo el campo de ingreso correspondiente:

```
Parameters nKeyCode, nMayusAltCtrl
IF BETWEEN ( nKeyCode, ASC("A"), ASC("Z") ) ;
OR BETWEEN ( nKeyCode, ASC("a"), ASC("z") )
   LastChar = UPPER(CHR(nKeyCode))
  ELSE
   LastChar = []
ENDIF
Key = TRIM(THISFORM.Texto2.Value) + LastChar
SEEK Key
COUNT TO THISFORM.c2 WHILE Keyword = Key
THISFORM.Contar2.Caption = ALLTRIM(STR(THISFORM.c2)) + [ encontrados]
```

El código del evento KeyPress es idéntico a éste exceptuando que se refiere a Texto2, C2 y Contar2. C1 y C2 son propiedades del formulario inicializadas en cero. Contar1 y Contar2 son campos de texto transparentes localizados respectivamente bajo sus campos de palabras clave.

El código Click para el botón Mostrar coincidencias aparece en el siguiente listado. Básicamente, usted toma la palabra clave que tenía menos coincidencias y cuenta cuántos de los títulos contienen también la otra palabra clave. Empieza con el que tenía menos juegos porque, por supuesto, sólo los títulos que contienen ambas palabras clave pueden coincidir, así que sólo tiene que considerar la lista más corta:

```
Key1 = TRIM(THISFORM.Texto1.Value)
Key2 = TRIM(THISFORM.Texto2.Value)
IF NOT EMPTY ( Key2 )
   DO CASE
         CASE THISFORM.c1 < THISFORM.c2
            Key1 = TRIM(THISFORM.Texto1.Value)
            Key2 = TRIM(THISFORM.Texto2.Value)
      OTHERWISE
            Key1 = TRIM(THISFORM.Texto2.Value)
            Key2 = TRIM(THISFORM.Texto1.Value)
   ENDCASE
ENDIF
```

```
SELECT KEYWORDS
SEEK Key1
JointHits = 0
SCAN WHILE KEYWORDS.KEYWORD = Key1
   SELECT LIBROS
   SEEK KEYWORDS.KEYISBN
   IF       EMPTY(Key2)          ;
   OR ( NOT EMPTY(Key2)          ;
        AND Key2 $ UPPER(Title) )
      JointHits = JointHits + 1
      THISFORM.List1.AddItem(LIBROS.ISBN+[-]+LIBROS.Titulo)
   ENDIF
   SELECT KEYWORDS
ENDSCAN
THISFORM.Contar3.Caption = ALLTRIM(STR(JointHits)) + [ joint hits]
THISFORM.List1.Setfocus
```

El código del botón Limpiar es:

```
THISFORM.List1.Clear
THISFORM.List1.Refresh
THISFORM.Texto1.SetFocus
```

Si el usuario hace clic en uno de los títulos seleccionados, esto es lo que sucede:

```
IF THIS.ListItemID > 0
   SELECT LIBROS
   Key = THIS.List(THIS.ListItemID)
   Key = LEFT(Key,10)
   SEEK Key
   THISFORM.B_Aceptar.Click
ELSE
   THISFORM.B_Cancelar.Click
ENDIF
```

Los botones Aceptar y Cancelar contienen esto:

```
* El código B_Aceptar.Click
ORDENES.CancelarBusqueda = .T.
RELEASE THISFORM

* El código B_Cancelar.Click
RELEASE THISFORM
```

La propiedad CancelarBusqueda( ) del formulario Ordenes puede establecerse en .T., de manera que el programa Ordenes sepa que no ha sido seleccionado ningún libro. Aquí está el código válido para las columnas de las cuadrículas ISBN y Título:

```
IF THIS.Value = [?]
   THISFORM.CancelarBusqueda = .F.
   DO FORM EncontrarLibro
   IF NOT ORDENES.CancelarBusqueda
        THISFORM.Cuadr1.Column1.Texto1.Value = Libros.ISBN
        THISFORM.Cuadr1.Column2.Texto1.Value = Libros.Titulo
        THISFORM.Cuadr1.Column3.Texto1.Value = 1
        THISFORM.Cuadr1.Column4.Texto1.Value = Libros.PrecioVenta
        THISFORM.Cuadr1.Column5.Texto1.Value = Libros.PrecioVenta
   ENDIF
ENDIF
```

Esto es llamado cada vez que el enfoque sale fuera de las columnas ISBN o Título del objeto cuadrícula, lo que puede suceder mientras se agrega o modifica.

Agregue una orden

Debido a la existencia del objeto cuadrícula, agregar ahora una nueva orden implica un poco más de esfuerzo. Para empezar, el objeto cuadrícula, como su primo Browse, no puede señalar al aire. Necesita tener registros a los cuales apuntar. Por esto, añada algunos registros en blanco a la tabla, y luego deseche los que no necesite.

```
* Código del método B_Adicionar:
BEGIN TRANSACTION
Editando = .T.
THISFORM.Adicionando = .T.
SELECT CONTROL
REPLACE NEXT 1 UltimaOrden WITH UltimaOrden + 1
SELECT ORDENES
SET RELATION OFF INTO DETALLES
SET SKIP TO
THISFORM.GuardarRegistro = RECNO()
APPEND BLANK
REPLACE NEXT 1 OrdenNum WITH ;
    TRANSFORM ( CONTROL.UltimaOrden, "@L ######" )
SELECT DETALLES
SET DELETED OFF
SET ORDEN TO DELETED
FOR I = 1 TO THISFORM.MaxChildRecs
   LOCATE FOR DELETED()
   IF NOT FOUND()
        APPEND BLANK
   ENDIF
   REPLACE NEXT 1 OrdenNum WITH ORDENES.OrdenNum
ENDFOR
SET ORDER TO OrdenNum
SELECT ORDENES
```

```
SET RELATION TO OrdenNum INTO DETALLES
SET SKIP TO DETALLES
THISFORM.Show
THISFORM.EnableAll
THISFORM.ButtonsOff
THISFORM.B_Guardar.Enabled = .T.
THISFORM.B_Cancelar.Enabled = .T.
THISFORM.I_CodigoClie.SetFocus
```

Tiene que desactivar Set Relation y Set Skip mientras añade registros de detalles en blanco, introduce el número de orden apropiado, un número de orden del archivo Ordenes de seis caracteres llenado con ceros, y luego vuelva a activarlos a ambos.

¿Qué le sucede a la secuencia de órdenes en el archivo si el usuario cancela la orden? Puede guardar una transacción abortada, lo que es probablemente una buena idea, puesto que un tipo de fraude de empleados implica cancelar transacciones en la computadora y luego completar la venta y guardarse el dinero. Sin embargo, probablemente sea mejor no permitir hacerlo a los usuarios. Como la columna "Doctor, me duele cuando hago esto" en mi hoja informativa, la respuesta siempre es: "Entonces ¡no lo haga!"

Validación del código del cliente

El código del cliente es algo que muy probablemente no recuerden los empleados, por lo que debe proporcionarles una búsqueda sencilla. Puede ser mejorada en forma considerable, pero ¿por qué tendría que reservarse toda la diversión?

```
SELECT CLIENTES
SET ORDER TO CodigoCli
Key = TRIM(THISFORM.CodigoCli.Value)
SEEK Key
DO CASE
   CASE FOUND() AND NOT EMPTY      ( THISFORM.Nombre.Value )
      RETURN
   CASE FOUND() AND     EMPTY      ( THISFORM.Nombre.Value )
        THISFORM.Nombre.Value      = TRIM(CLIENTE.Nombre) + [ ];
                                   + TRIM(CLIENTE.Apellido)
        THISFORM.Direccion1.Value  = CLIENTE.Direccion1
        THISFORM.Ciudad.Value      = CLIENTE.Ciudad
        THISFORM.Estado.Value      = CLIENTE.Estado
        THISFORM.CodPost.Value     = CLIENTE.CodPost
        THISFORM.Telefono1.Value   = CLIENTE.Telefono1
        THISFORM.Pais.Value        = CLIENTE.Pais
```

```
   CASE NOT FOUND()
       DO WHILE LEN(Key) > 0 AND NOT FOUND()
          SEEK Key
         IF NOT FOUND()
             Key = LEFT ( Key, LEN(Key)-1 )
          ENDIF
          SEEK Key
       ENDDO
       IF NOT FOUND()    && Ni siquiera la primera letra existe...
          GO TOP
        ENDIF
        DEFINE WINDOW BROWSER FROM 1,1 TO 30, 100
        BROWSE WINDOW BROWSER NOMODIFY NOAPPEND
        RELEASE WINDOW BROWSER
        IF LASTKEY() <> 27
           THISFORM.CodigoClie.Value =  CLIENTE.CODIGOCLIE
           THISFORM.Nombre.Value = TRIM(CLIENTE.Nombre)+[ ];
                                 + TRIM(CLIENTE.Apellido)
           THISFORM.Direccion1.Value = CLIENTE.Direccion1
           THISFORM.Ciudad.Value     = CLIENTE.Ciudad
           THISFORM.Estado.Value     = CLIENTE.Estado
           THISFORM.CodPost.Value    = CLIENTE.CodPost
           THISFORM.Telefono1.Value  = CLIENTE.Telefono1
           THISFORM.Pais.Value       = CLIENTE.Pais
        ENDIF
ENDCASE
SELECT ORDENES
RETURN .T.
```

Modifique una orden

El siguiente listado es el código Modificar para la pantalla Ordenes. Es un poquito más directo:

```
Editando = .T.
THISFORM.HabilitarTodo
THISFORM.I_OrdenNum.Enabled = .F.
THISFORM.I_Fecha.Enabled = .F.
THISFORM.ButtonsOff
THISFORM.B_Guardar.Enabled = .T.
THISFORM.B_Cancelar.Enabled = .T.
THISFORM.I_CodigoClie.SetFocus
BEGIN TRANSACTION
```

Faltan dos características sin las que un software robusto y profesional no puede existir: no se añadieron más líneas en blanco y no existe una rutina Borrar línea. No son difíciles, pero, una vez más, se dejará algo para después.

Guarde los cambios

Cuando usted ha terminado de añadir o modificar, tiene que guardar sus cambios o cancelar lo añadido. Los cambios ya están en las tablas, ¿lo están? Si usted utiliza un buffering de fila, tan pronto como pasa de una línea detallada a la siguiente, los datos son escritos en el archivo. Es por eso que usted puso Begin Transaction en las rutinas Agregar y Modificar. Aquí está el código Guardar:

```
END TRANSACTION
Editando = .F.
SET DELETED OFF
IF THISFORM.Adicionando
   SELECT DETALLES
   SCAN FOR OrdenNum = ORDENES.OrdenNum
      IF EMPTY ( DETALLES.ISBN )
            DELETE NEXT 1
            BLANK NEXT 1
      ENDIF
ENDSCAN
ENDIF
SET DELETED ON
SELECT ORDENES
THISFORM.InhabilitarTodo
THISFORM.ButtonsOn
THISFORM.Buttons
THISFORM.B_Guardar.Enabled = .F.
THISFORM.B_Cancelar.Enabled = .F.
THISFORM.B_Siguiente.GotFocus
```

End Transaction escribe todo lo que el usuario introdujo en el disco. Pero Visual FoxPro no sabe o no le importa que algunas de las líneas de detalle estén en blanco; para el diminuto cerebro de la computadora, datos son datos. Así que tiene que borrar las líneas que se han quedado en blanco en la orden. (Recuerde, otros usuarios podrían estar consignados, por lo que se requiere el FOR OrdenNum = Ordenes.OrdenNum.) Finalmente, vuelva a inhabilitar todos los campos de entrada y habilite todos los botones, excepto Guardar y Cancelar.

Cancele los cambios

¿Y para cancelar cambios? Es aún más sencillo. Begin Transaction puede finalizar de dos maneras: End Transaction o Rollback. Aquí está el código B_Cancelar:

```
ROLLBACK
SELECT ORDENES
IF THISFORM.Adicionando
```

```
   GO ( THISFORM.GuardarRegistro )
ENDIF
Editando = .F.
THISFORM.Adicionando = .F.
THISFORM.Show
THISFORM.InhabilitarTodo
THISFORM.ButtonsOn
THISFORM.Buttons
THISFORM.B_Guardar.Enabled = .F.
THISFORM.B_Cancelar.Enabled = .F.
THISFORM.B_Siguiente.GotFocus
```

Si usted ha puesto pantallas padre-hijo bajo un código FoxPro 2.6, o cualquier lenguaje, para el caso, apreciará el poco trabajo que se requiere aquí.

Elimine una orden

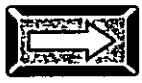
Para eliminar una orden, tiene que quitar el encabezado de la orden de la tabla Ordenes, y cualquier registro hijo que haga juego con la tabla Detalles. Éste es un ejemplo de integridad referencial. Puede ser útil en situaciones más complejas, pero creo que utilizarlo para esta situación trivial es una medida excesiva. Aquí está el código:

```
IF MESSAGEBOX ( "¿Borrar este registro?" , 4 + 32 + 256 )= IDYes
* Nota: IDYes está definida como 6 foxpro.h
   SET DELETED OFF
   SELECT DETALLES
   SCAN FOR OrdenNum = ORDENES.OrdenNum
        BLANK NEXT 1
        DELETE NEXT 1
   ENDSCAN
   SELECT ORDENES
   BLANK
   DELETE NEXT 1
   SET DELETED ON
   GO TOP
   THISFORM.Show
ENDIF
```

Set Deleted On es necesario porque usted no desea ver los registros después de que han sido borrados. Advierta que cuando usted añade registros de detalle durante una adición, recicla estos registros borrados así como cualquiera no utilizado durante una adición previa. De esa formà, generalmente hay pocos registros borrados en el archivo en cualquier momento, y usted nunca tiene que comprimir el archivo.

Salga del formulario

Cuando usted abandona un formulario padre-hijo, necesita deshacer cualquier comando Set Relation o Set Skip, porque puede desear ver Ordenes o Detalles de alguna otra manera. Por ejemplo, existe un indicador orden-especial con el que no ha hecho nada hasta ahora. Las órdenes especiales son órdenes que apresuran libros que no están en existencia, por lo que si un cliente tiene sólo un libro de orden especial en una orden, usted necesita trabajar sobre él inmediatamente. Tiene que ser capaz de leer el archivo Detalles sin que esté unido al archivo Ordenes.



Conclusión

Esta aplicación, a la que le faltan algunos elementos esenciales (como imprimir una factura y marcar una orden como enviada), es sorprendentemente completa. Representa quizás un día de codificar, lo que es notablemente poco para software utilizable. Antes, prototipos con mucha menos funcionalidad se llevaban días, incluso semanas. Espero que usted esté empezando a apreciar el verdadero poder de este entorno.

En los capítulos que siguen, usted se abastecerá de algunos otros fundamentos. El Generador de informes y el Generador de consultas son dos elementos especiales del producto Visual FoxPro, y saber lo que hay ahí puede darle la confianza de continuar con un proyecto con sólo las especificaciones mínimas.

Genere un informe nuevo

Hay cuatro maneras de abrir el Generador de informes:

- Haga clic en Archivo, Nuevo, luego seleccione Informe y después elija Archivo nuevo.
- Haga clic en Archivo, Nuevo, luego seleccione Informe y después elija Asistente.
- Desde la ventana Comandos, teclee MODIFY REPORT (Modificar informe), dé un nombre nuevo y oprima ENTER.
- Teclee MODIFY REPORT y oprima ENTER. Después introduzca un nombre de archivo y haga clic en Abrir.

Una vez que abra el Generador de informes, obtendrá la ventana de la figura 10-1. Lo que se hace en seguida, depende de lo que usted quiere que su informe realice. Para una introducción a los diferentes tipos de informes, se emplearán los asistentes para informes de Visual FoxPro. Después, si desea realizar variaciones sobre uno de estos temas, contará con buenos fundamentos.

Genere un informe nuevo con un Asistente para informes

Existen tres clases de asistentes para informes: de tabla única, de grupos/totales y de uno a varios. Cada uno produce diferentes tipos de informes y demuestra algunas de las muchas características del Generador de informes.

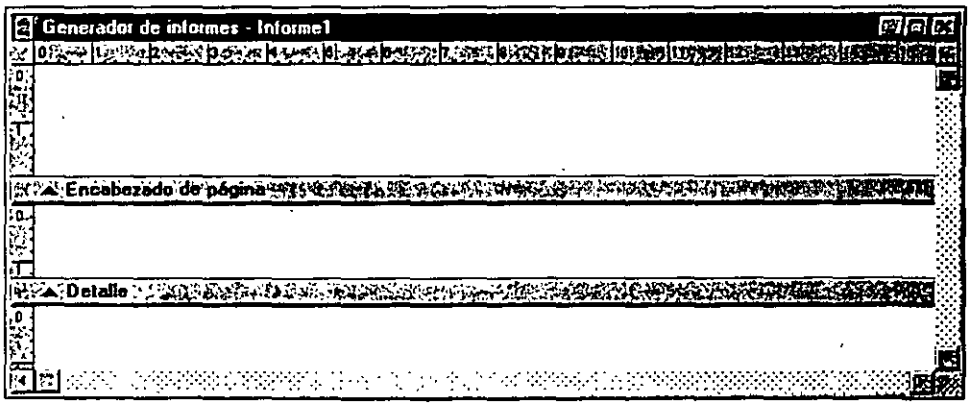
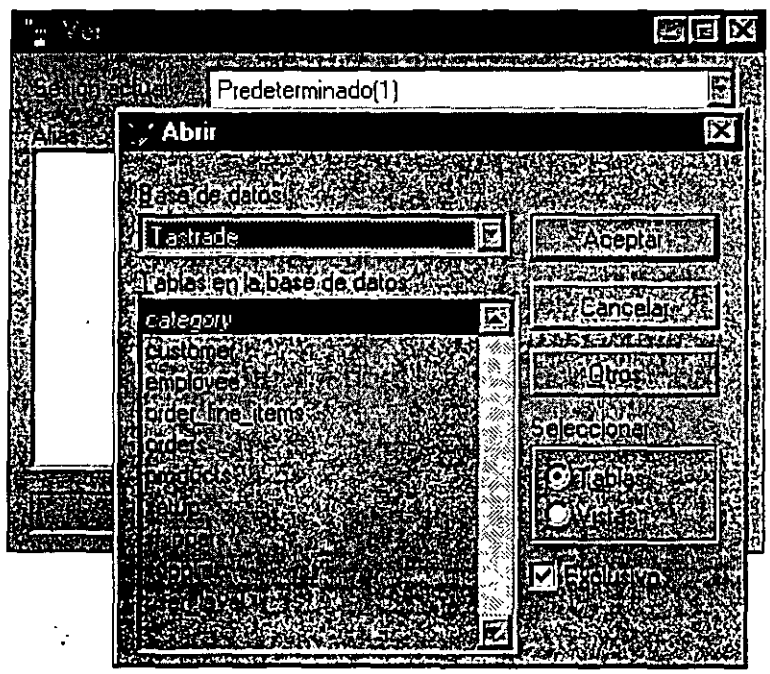


Figura 10-1

Pantalla del Generador de informes.

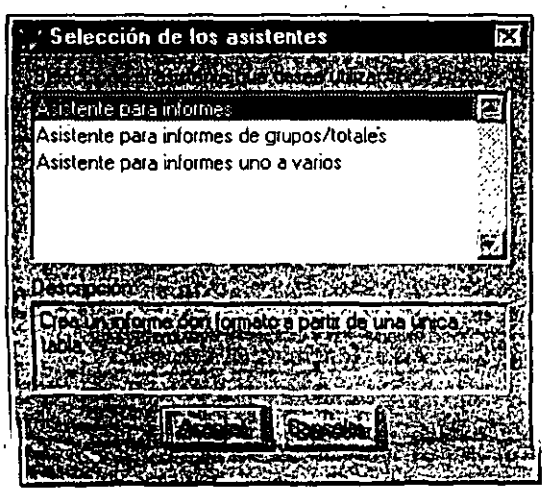
Se comenzará con el archivo Cliente. Si nunca ha trabajado con él, haga clic en Ventana, Ventana Ver y luego en Abrir. Verá el cuadro de diálogo Abrir que aparece en la figura 10-2. Haga clic en Cliente, Aceptar y cierre la ventana. Luego haga clic en Archivo, Nuevo, Informe y seleccione Asistente. Verá el cuadro de diálogo Selección de los asistentes (figura 10-3). Se empezará con el tipo de informe más sencillo. Escoja la primera opción, Asistente para informes que, de acuerdo con la descripción, crea un informe formateado a partir de una sola tabla. Resalte Cliente y los campos en el archivo Cliente aparecerán en la segunda lista.

Figura 10-2



Cuadro de diálogo Abrir sobre la ventana Ver.

Figura 10-3



Cuadro de diálogo para seleccionar los asistentes.

177

Ahora debe decidir qué hará. Si hace clic en el icono con doble flecha, todos los campos en la lista Campos disponibles se moverán a la lista Campos seleccionados, en orden alfabético. Si hace eso, tendrá que volver a ordenar todos los campos seleccionados en la lista del recuadro de la derecha. Es posible hacerlo, pero no es divertido. Yo prefiero escoger los campos uno a la vez y luego pasarlos a la lista Campos seleccionados en el orden en que los escogí. Cuando termine, haga clic en Siguiente. Obtendrá la ventana de la figura 10-4.

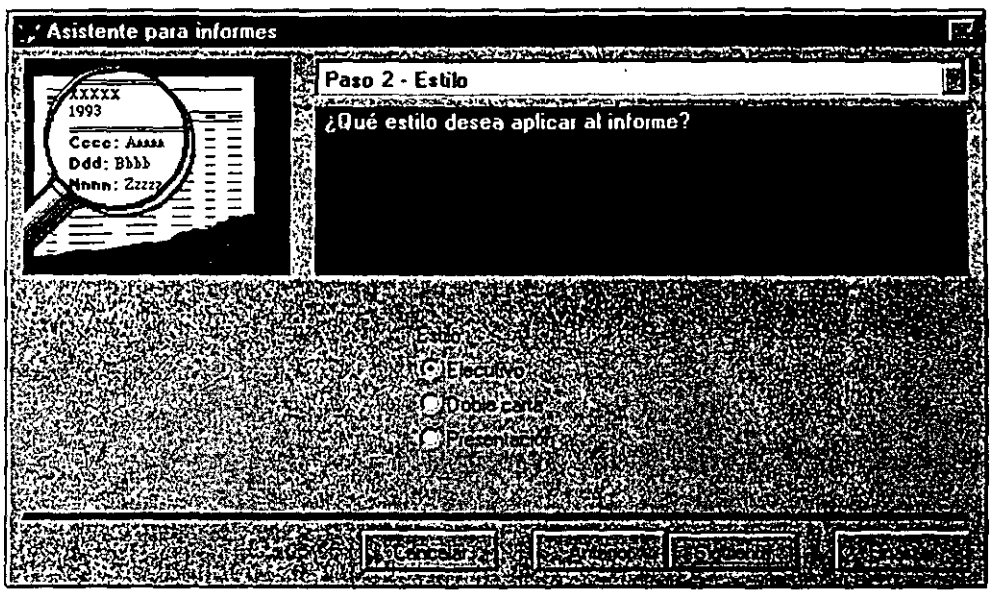


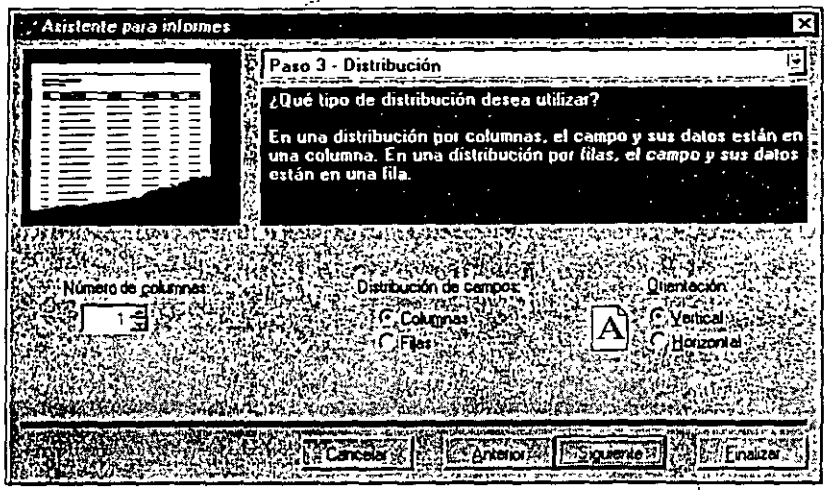
Figura 10-4

Pantalla de selección de estilo del Asistente para informes.

Los tres formatos son buenos, pero el estilo Doble carta tiene un poco más de qué hablar, de modo que haga clic en Doble carta y después en Siguiente. Obtendrá la tercera ventana, Distribución, que se muestra en la figura 10-5. Esta ventana le permite escoger entre dos disposiciones de campos: columnas o filas. Columnas produce un informe tabular como una hoja de cálculo, en tanto que Filas pone un campo en cada línea del informe, algo que podría utilizar para un formato de informe con un registro por página. Por el momento, seleccione Columnas.

Orientación se refiere a si el informe se imprime de costado. Si tiene mucha información en cada registro, la impresión de costado, llamada *horizontal*, le permite incluir más registros en cada página. Sin embargo, para poder utilizar esta opción, debe tener una bandeja de papel con orientación horizontal en su impresora. Así que seleccione Vertical.

Figura 10-5

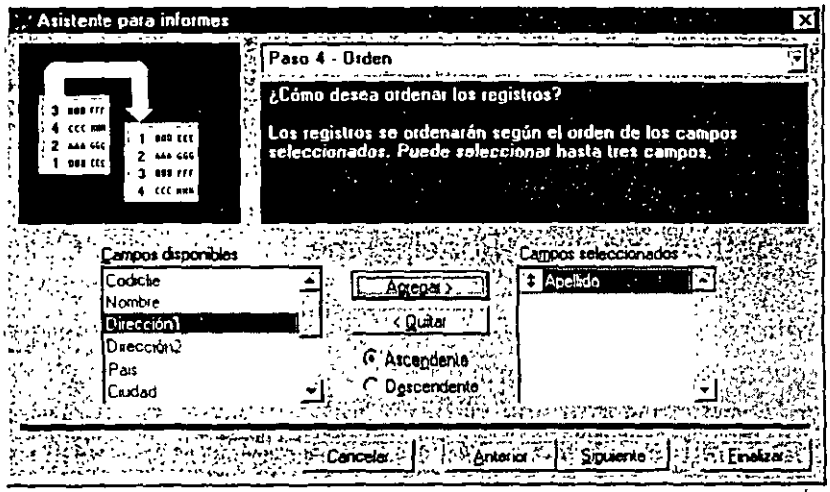


Pantalla de selección de distribución del Asistente para informes.

Puede indicar al Asistente para informes que ponga dos o más registros por grupo de detalle utilizando el selector de columnas en la esquina inferior izquierda de la ventana del asistente, pero este primer ejemplo no se beneficiará con esa opción, de modo que déjela en 1. Por último, deje el tamaño del papel en Carta, que es el parámetro establecido por omisión. Haga clic en Siguiente para continuar.

El paso 4, Orden, se ilustra en la figura 10-6. Las listas de clientes se realizan con mayor frecuencia ordenadas por apellido para facilitar la localización de los clientes, de manera que haga clic en Apellido para moverlo al lado derecho de la pantalla. Note que puede seleccionar varias claves. Si selecciona, por ejemplo,

Figura 10-6



Selección del orden para el informe.

Apellido y luego Nombre, obtendrá Aguirre, Efrén antes de Aguirre, Sonia. El Asistente soporta hasta tres claves. También puede ordenarlos ya sea en forma ascendente o descendente, si esto tiene sentido. En este caso no lo tiene, de modo que deje seleccionado el botón de opción Ascendente y haga clic en Finalizar.

El paso 5 del Asistente para informes, Finalizar, le permite crear el informe. Luego puede ya sea guardar el informe para usarlo después o modificarlo antes de guardarlo. También puede hacer clic en el botón Presentación preliminar, el cual crea el informe y le muestra cómo se verá una vez impreso. La presentación preliminar para este informe se muestra en la figura 10-7. La barra de herramientas de Presentación preliminar que se produce en esta etapa del proceso le permite hacer un acercamiento en partes del informe para examinar detalles. Cuando se hace clic en el icono de libro abierto se cierra la Presentación preliminar, como lo indicará el nombre del icono. Cierre la Presentación preliminar, luego haga clic en el segundo botón de opción para ver algunos de los detalles del

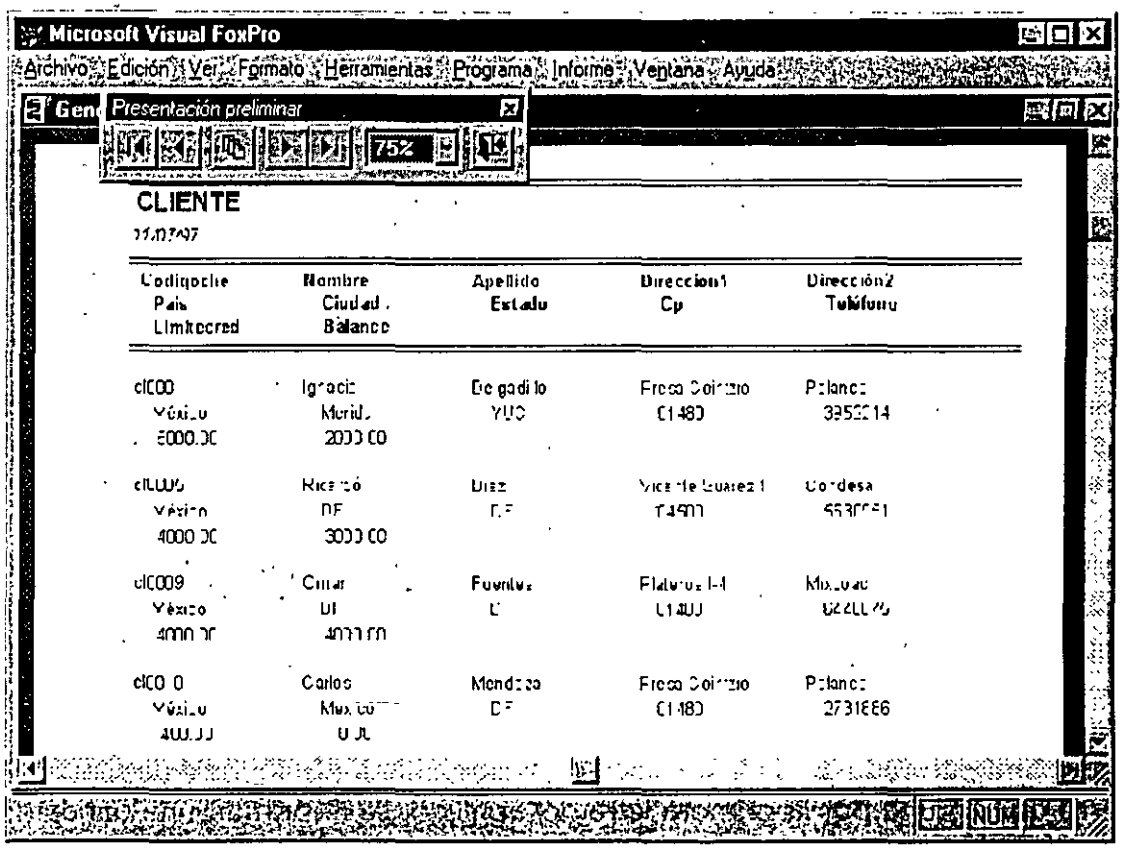


Figura 10-7

Presentación preliminar del informe generado por el Asistente.

informe generado. Visual FoxPro le pedirá un nombre para el informe; llámelo Cliente. La extensión .FRX se agrega en forma automática.

Un informe de Visual FoxPro se almacena en un par de archivos con las extensiones .FRX y .FRT. Los archivos contienen un registro por cada elemento en su informe, además de algunos otros registros para definir los recursos usados por el informe, como las tablas de entrada. También puede visualizar los archivos del informe, aunque necesita ser cuidadoso para no cambiar nada si no está seguro de las consecuencias del cambio. Una vez que sabe qué contienen los diversos campos, puede escribir programas para crear archivos de informes. Eso es lo que hace Foxfire!.

En Visual FoxPro, los informes consisten en bandas. Hay varias bandas estándar, además de algunas bandas opcionales. Las bandas se procesan en un orden particular. Dentro de cada banda, los objetos también se procesan en un orden determinado. Usted pone los objetos (campos, texto, imágenes o recuadros) en una banda y FoxPro sabe cómo imprimirlos.

El menú principal Informe

Hay un nuevo menú en la barra de menús de Visual FoxPro. Despliéguelo y verá el menú que aparece en la figura 10-8.

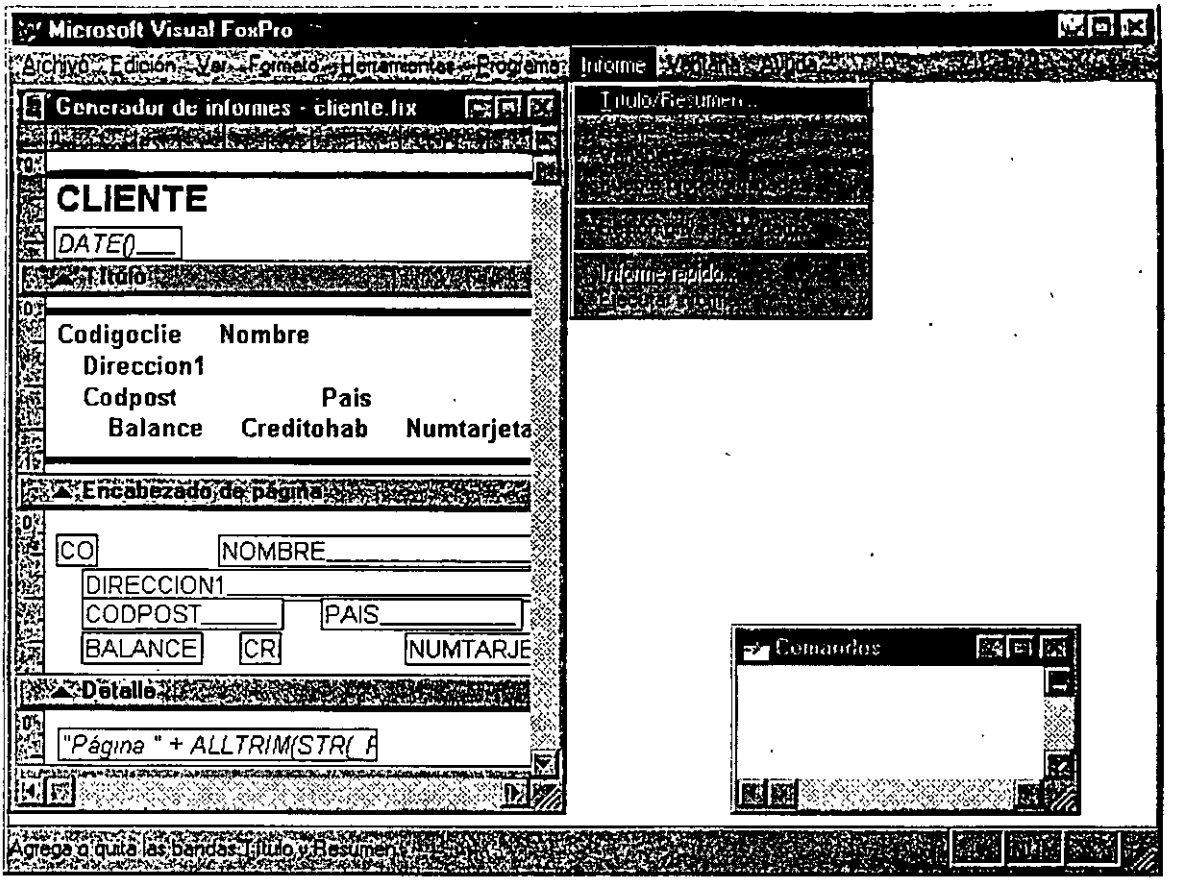
Título/Resumen controla lo primero y lo último que se imprime en su informe. Puede imprimir una página de título y resumen o imprimir la información del título y resumen en la primera y la última página del informe. El informe generado por el Asistente para informes pone el título en la parte superior de la primera página, en lugar de hacerlo en una sola página.

Agrupar datos le permite separar visualmente grupos de registros en casos en que los datos estén indexados u ordenados. Por ejemplo, puede imprimir el listado de clientes con un encabezado para los apellidos que comienzan con A, luego B y así en forma sucesiva, como un directorio.

El agrupamiento requiere la indexación correspondiente. Si le indica a Visual FoxPro que agrupe por la primera letra del apellido, se imprimirá una banda de grupo cada vez que cambie la primera letra del apellido. Si los datos están ordenados al azar, tendrá un aspecto raro. Basta decir que se selecciona el agrupamiento con base en la etiqueta de índice que se planee usar.

181

Figura 10-8



Menú Informe.

Puede usar variables para incluir conceptos en el informe que no son campos en ninguna tabla abierta. Por ejemplo, si quiere calcular e imprimir un subtotal de operación y un porcentaje del total general además de la cantidad en dólares, necesitará una variable del subtotal de operación y un total general de depósito para usarlos en el cálculo del porcentaje del total.

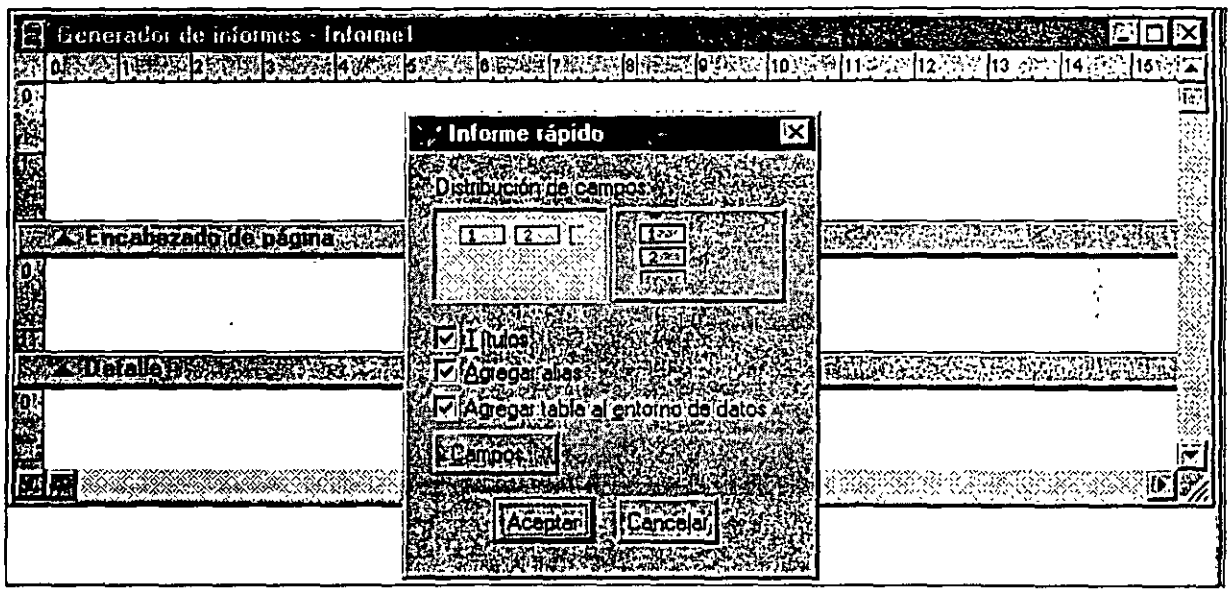
La fuente predeterminada se aplica en cada objeto nuevo de su informe. Si usted pone una serie de encabezados o campos en un informe, este parámetro asegurará que se emplee la fuente correcta.

Sesión privada de datos significa que no quiere que el movimiento del puntero de registro, conforme se imprime el informe, afecte la localización del puntero de registro en el mismo archivo en cualquier otra parte de su programa. Por ejemplo, si desea imprimir un informe de clientes mientras busca el registro de un cliente en la pantalla, seleccione la sesión privada de datos para el informe con el fin de proteger la localización del puntero de registro en la vista de la tabla en pantalla.

Esto es como el comando USE AGAIN (Usar otra vez) en versiones previas de FoxPro.

La opción Informe rápido se habilita sólo si la ventana del Generador de informes está limpia, es decir, si aún no ha puesto nada en la ventana. El Informe rápido es como un pequeño asistente, como se ilustra en la figura 10-9. Realiza menos funciones y en realidad puede es preferible si desea hacer listados muy simples y rápidos.

Figura 10-9



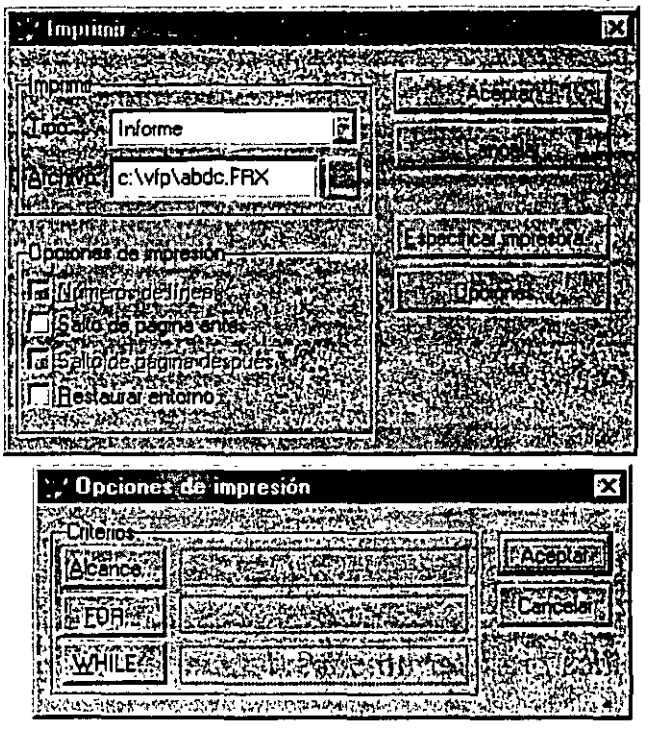
Informe rápido.

Por último, use Ejecutar informe para probar su informe sin salir del Generador de informes. Es diferente de la opción Presentación preliminar del menú Archivo porque envía la salida a la impresora y sólo puede usarla con parte de los datos en su archivo, con SCOPE (por ejemplo, siguientes 50, registro # o "resto del archivo"), así como las opciones PRINT FOR (Imprimir para) y PRINT WHILE (Imprimir mientras).

La figura 10-10 muestra el cuadro de diálogo Imprimir y el cuadro de diálogo asociado Opciones de impresión. Nótese que el nombre del archivo en el cuadro está en el directorio \VFP. Visual FoxPro realiza gran parte de su trabajo en archivos temporales. Si establece en C:\TEMP o C:\WINDOWS\TEMP la localización de los archivos temporales en la ficha Archivos del cuadro de diálogo Opciones, que se obtiene haciendo clic en Herramientas, Opciones, ahí es a donde se envían dichos archivos de trabajo. Es evidente que se deben limpiar los directorios de trabajo de vez en cuando.

183

Figura 10-10



Ejecute el informe con el cuadro de diálogo Opciones de impresión abierto.

Ésa fue una plática muy considerable acerca de los informes antes de pasar a lo que en realidad conforma el informe que respondió algunas de sus preguntas. Pero ahora se avanzará y verá la disposición real de un informe.

Bandas en el Generador de informes

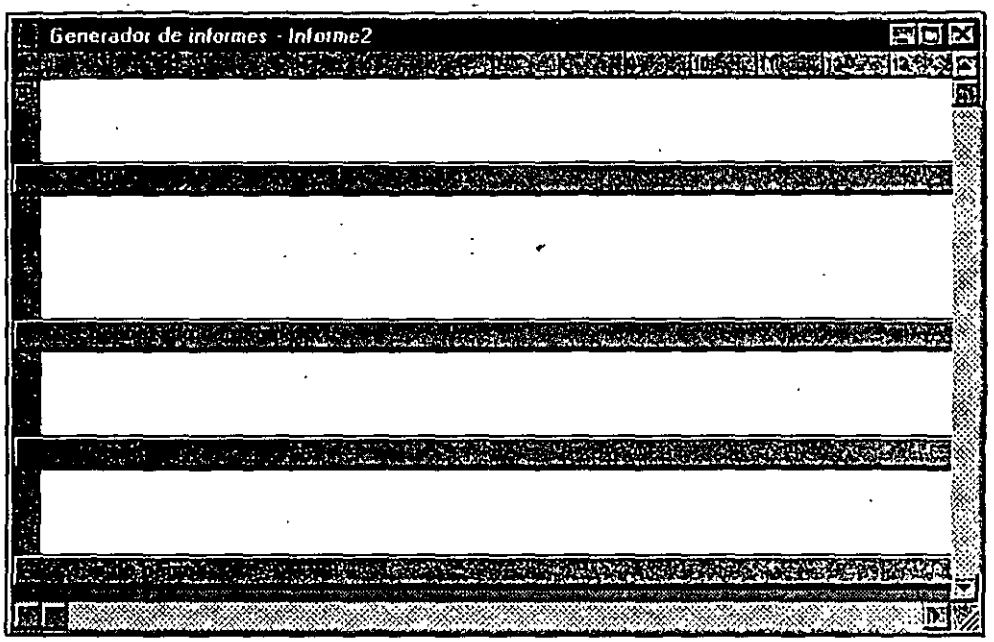
La ventana Generador de informes se muestra en la figura 10-11. Inicialmente consiste en cuatro bandas:

- Título
- Encabezado de página
- Detalle
- Pie de página

La banda Título se imprime sólo una vez, antes que el resto del informe. Si lo desea, puede imprimirla en una página separada, pero el Asistente para informes la genera en la parte superior de la página. Para forzar una página nueva después

184

Figura 10-11



Las cuatro bandas predeterminadas del Generador de informes.

de la página de título, haga clic en Informe, Título/Resumen y marque la casilla de verificación Nueva página.

La banda Encabezado de página se imprime en la parte superior de cada página. Aquí es donde se ponen los nombres de los campos que aparecerán como columnas en la banda Detalle, de modo que los usuarios pueden decir qué contiene cada columna.

Nótese que, dado que las bandas Detalle pueden contener más de una fila de campos, la banda Encabezado de página podría tener más de una fila de títulos. Generar un informe de detalles con líneas múltiples de modo que se puedan leer es más arte que ciencia. Así que debe trabajar con la parte izquierda del cerebro (¿o es con la derecha?).

La banda Detalle es donde se da la acción. Puede extender la banda Detalle para que sea tan grande como desee. Puede tener varias filas de detalle y agregar cuadros para enfatizar, o para que los usuarios escriban comentarios. Es posible suprimir la impresión de líneas que están vacías porque los registros actuales no contienen datos para los campos en la fila del informe. Puede envolver el texto alrededor de la línea siguiente si el campo contiene demasiados caracteres no blancos para ajustar el ancho de la columna asignada. Y mucho más.

La banda Pie de página, si se usa, se imprime en la parte inferior de cada página. Para algunos tipos de informes de contabilidad es deseable imprimir un total en la parte inferior de cada página. No obstante, si tuviera un total general en mente, éste no es el sitio en donde se ubica. Para los totales que deben imprimirse al final del informe entero, agregue una página de resumen con la opción Título/ Resumen del menú Informe.

Estas cuatro bandas son el lugar en que se realiza casi todo lo que tiene que ver con la generación de un informe. Un poco más adelante se estudiará más a fondo la banda Detalle. Por el momento, hay algunas otras herramientas sobre las cuales debe aprender.

El menú del botón secundario del ratón en el Generador de informes

Haga clic en cualquier área que no contenga ningún objeto con el botón secundario del ratón (por lo general el derecho) y obtendrá un menú contextual con las siguientes opciones:

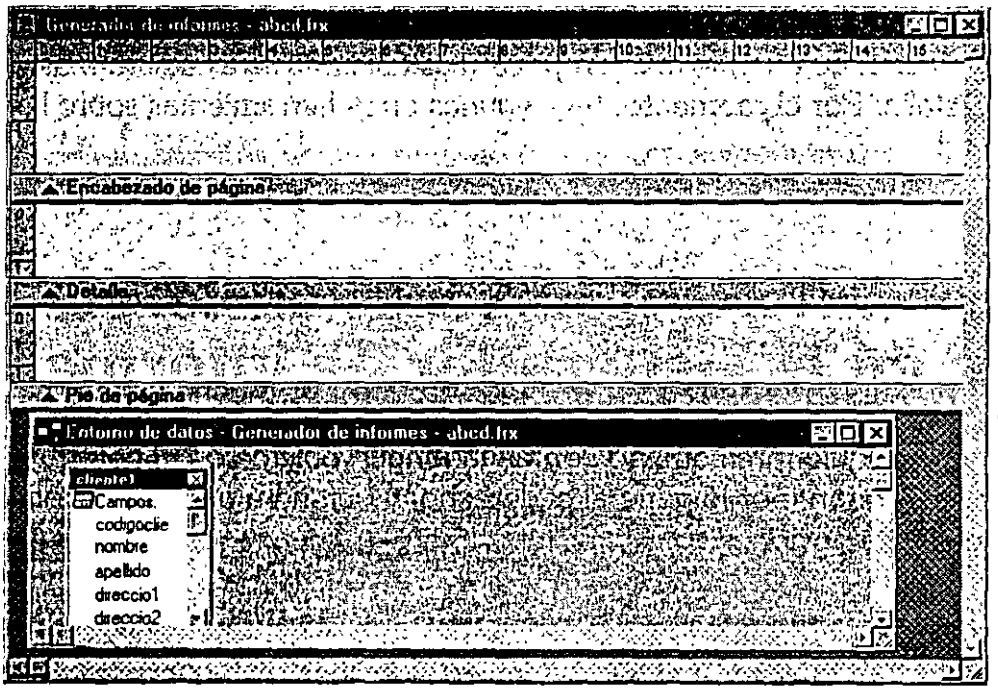
- Pegar
- Presentación preliminar
- Imprimir
- Entorno de datos
- Agrupar
- Ayuda

Si hace clic en un objeto de texto o datos y después hace clic con el botón secundario del ratón, verá un menú más pequeño:

- Cortar
- Copiar
- Pegar
- Propiedades
- Ayuda

Estas opciones también pueden elegirse en otras partes, pero el menú contextual sabe en dónde se encuentra usted y, por tanto, lo que es probable que desee hacer. Por ejemplo, si necesita agregar otra tabla de modo que sus campos también se incluyan en el informe, escoja Entorno de datos y agregue el nombre de la tabla. El Entorno de datos, que aparece en la figura 10-12, muestra qué tablas están disponibles para generar el informe.

Figura 10-12



Entorno de datos en el Generador de informes.

Barras de herramientas

Es posible utilizar tres barras de herramientas cuando se está en el Generador de informes:

Controles de informes Se utiliza para poner campos, texto, recuadros e imágenes en el informe.

Distribución Se emplea para alinear objetos.

Paleta de colores Usted tiene esta barra de herramientas cuando es lo suficientemente afortunado (o desafortunado) como para tener una impresora a color.

Cuando hace clic en un par de objetos, se habilitan los iconos en la barra de herramientas Disposición. La alineación de objetos en la pantalla, que con FoxPro

187

2.6 era una tarea tediosa, es una tarea sencilla con Visual FoxPro. Sólo haga clic en todo lo que deba alinear mientras mantiene oprimida la tecla MAYÚS (SHIFT), luego haga clic en la herramienta de disposición apropiada (por ejemplo, Alinear los bordes superiores), y eso es todo. La barra de herramientas Controles de informes sólo contiene seis objetos:

- Etiqueta
- Campo
- Imagen/Control OLE dependiente
- Línea
- Rectángulo
- Rectángulo redondeado

Además, hay un bloqueo de botón. Cuando está desactivado, cada vez que hace clic en el informe, Visual FoxPro agrega el mismo tipo de objeto (por ejemplo, etiqueta o campo) como el que se agregó previamente. Si el bloqueo de botón no está activado, el botón Seleccionar objetos es el parámetro prederminado. Si el bloqueo de botón no está activado, Seleccionar objetos lo desactiva. Es evidente que el bloqueo de botón por sí mismo alterna la activación y desactivación. Esto puede ahorrar tiempo cuando se agrega un grupo de campos o etiquetas.

Manipule los objetos del informe

Para generar informes útiles, necesita familiarizarse con las herramientas del Generador de informes.

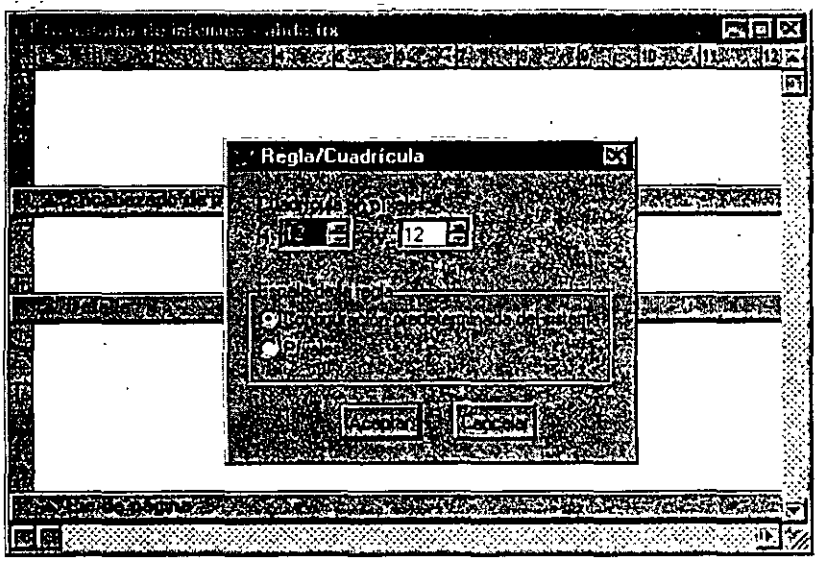
Cuadrícula y granularidad

Cuando el Generador de informes está abierto, el menú Formato contiene dos opciones: Ajustar a la cuadrícula y Configurar cuadrícula. El primer lugar en que se debe detener es en la segunda opción, que produce el cuadro de diálogo que aparece en la figura 10-13.

La cuadrícula no aparece en el Generador de informes, pero su efecto sí; cuando coloca un objeto en la pantalla, "cambia" a las coordenadas de la cuadrícula más

122

Figura 10-13



Cuadro de diálogo Regla/Cuadrícula.

cercana, siempre que Ajustar a la cuadrícula esté activada. Esta operación solía ser muy frustrante. En el pasado, sólo se podía agarrar el objeto con el ratón y moverlo alrededor mientras se mantenía oprimido el botón del ratón, o se hacía clic en el objeto para seleccionarlo y luego se usaban las teclas de flecha para empujarlo suavemente por la pantalla. La barra de herramientas Disposición facilita esta tarea en Visual FoxPro. Además, tal vez prefiera establecer un tamaño más pequeño para la cuadrícula, por ejemplo, 4 x 4, de modo que el efecto visual no sea tan desconcertante.

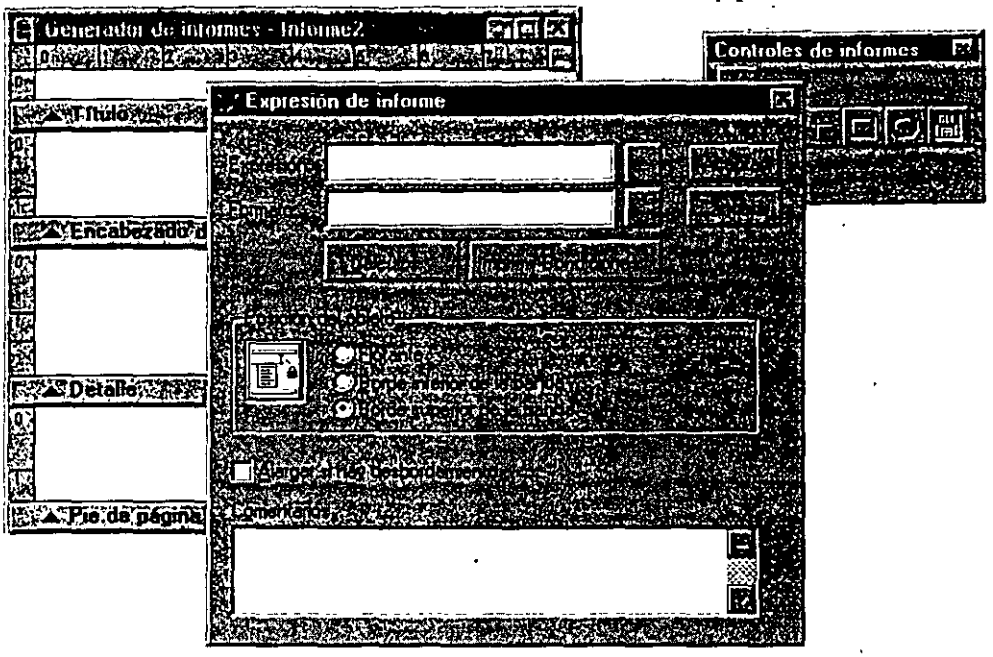
Agregue texto a una banda

Las etiquetas identifican los campos en sus informes y pueden darles cierta personalidad. Puede cambiar la fuente, el tamaño de la fuente y el estilo (negritas, itálicas, subrayado) de cualquier etiqueta. Para agregar una etiqueta, haga clic en el icono de A mayúscula, luego coloque el apuntador del ratón donde desee.

Agregue campos a una banda

Para agregar campos a un informe, haga clic en el icono de campo y luego haga clic en cualquier otra parte en el formulario; se creará un objeto rectangular. Puede cambiar su tamaño antes de soltar el botón primario (por lo general el izquierdo) del ratón. Aparecerá el cuadro de diálogo Expresión de informe que se muestra en la figura 10-14.

Figura 10-14



Cuadro de diálogo Expresión de informe.

Expresión es donde usted introduce el nombre del campo que se va a imprimir. Por lo regular, teclear el nombre del campo es casi lo peor que puede hacer. FoxPro no sabe cuán grande es su campo hasta que lo seleccione, y a menos que lo haga, de la lista de campos en las tablas registradas con el Entorno de datos. Y, aun si la tabla está en el Entorno de datos, si introduce el nombre del campo, Visual FoxPro dará por predeterminación una longitud de campo no asociada con el tamaño real del campo. De modo que será necesario hacer clic en el botón al final del campo, donde se puede introducir el nombre del campo y seleccionarlo de la lista de selección.

Formato es el lugar en que se indica a Visual FoxPro cómo dar formato a la salida, en especial la de los campos numéricos. Si quiere desplegar precios con una cláusula de imágenes como #,###.##, éste es el sitio en que debe hacerlo.

Las opciones de edición cambian con el tipo de campo. Por supuesto, si selecciona un nombre de campo de una tabla registrada en el Entorno de datos, se verificará el tipo de datos correcto. Si introdujo el nombre del campo, usted tiene el control.

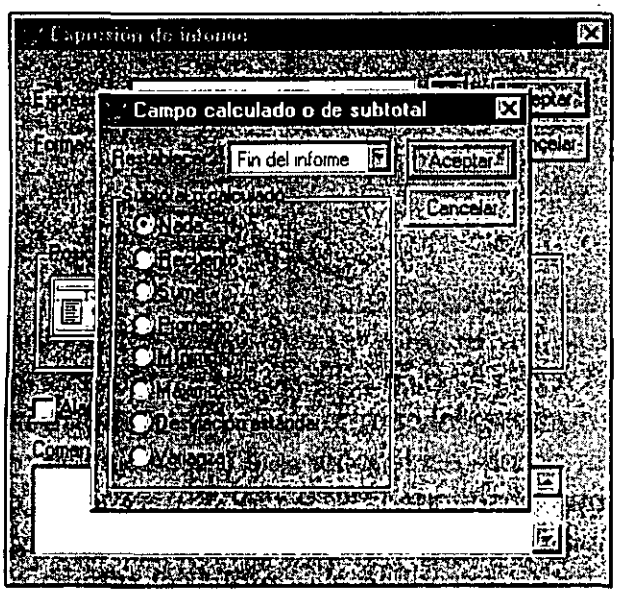
Cálculos

Los campos son los elementos más importantes en un informe. Despliegan los datos de sus tablas. Pero, ¿qué sucede si los datos que necesita ver no están

exactamente esperándolo en una tabla? Los subtotales son un ejemplo de información derivada de sus tablas; las cuentas son otro ejemplo.

Para obtener este tipo de concepto, sólo haga clic en el botón de cálculos, luego seleccione la opción que desee de la lista que aparece en la figura 10-15. Aunque Suma y Recuento son dos de las opciones que se emplean con mayor frecuencia, hay algunas un cuanto más complejas. Utilice Nada para cancelar cualquier opción establecida anteriormente.

Figura 10-15



Opciones del cálculo.

Imprimir-Condiciones

Puede establecer la opción Imprimir valores repetidos, del botón Imprimir-Condiciones, en No para destacar valores únicos de una columna.

Imprimir-Condiciones también se usa para asegurarse de que un encabezado o elemento de detalle importante esté presente en la parte superior de la página siguiente, aun si ya estaba impreso en la parte superior del grupo de detalle. Si el texto envuelto provoca que la línea de detalle se despliegue en otra página, esto asegura que se repitan los detalles importantes. Además, puede forzar los detalles para que se repitan cuando ocurra una división particular a nivel de grupo.

Quitar líneas en blanco se utiliza para suprimir por completo la salida de líneas en que todos los controles en la fila se refieren a campos que están vacíos.

Imprimir sólo cuando Expresión sea verdadera es un nivel de control adicional de los elementos de despliegue. Se pueden incluir o excluir condicionalmente campos

191

particulares, por ejemplo, información monetaria delicada que se muestra sólo a quienes tienen la contraseña adecuada.

Posición del objeto

Posición del objeto selecciona un ancla relativa para el objeto. Las opciones son:

- Flotante
- Borde inferior de la banda
- Borde superior de la banda

Esta opción es particularmente útil, por ejemplo, si tiene un recuadro de tamaño fijo para escribir comentarios en el informe. Agregar el recuadro a la parte inferior de la banda se ve mucho mejor que tener otros campos de texto que han envuelto y fluido en filas adicionales.

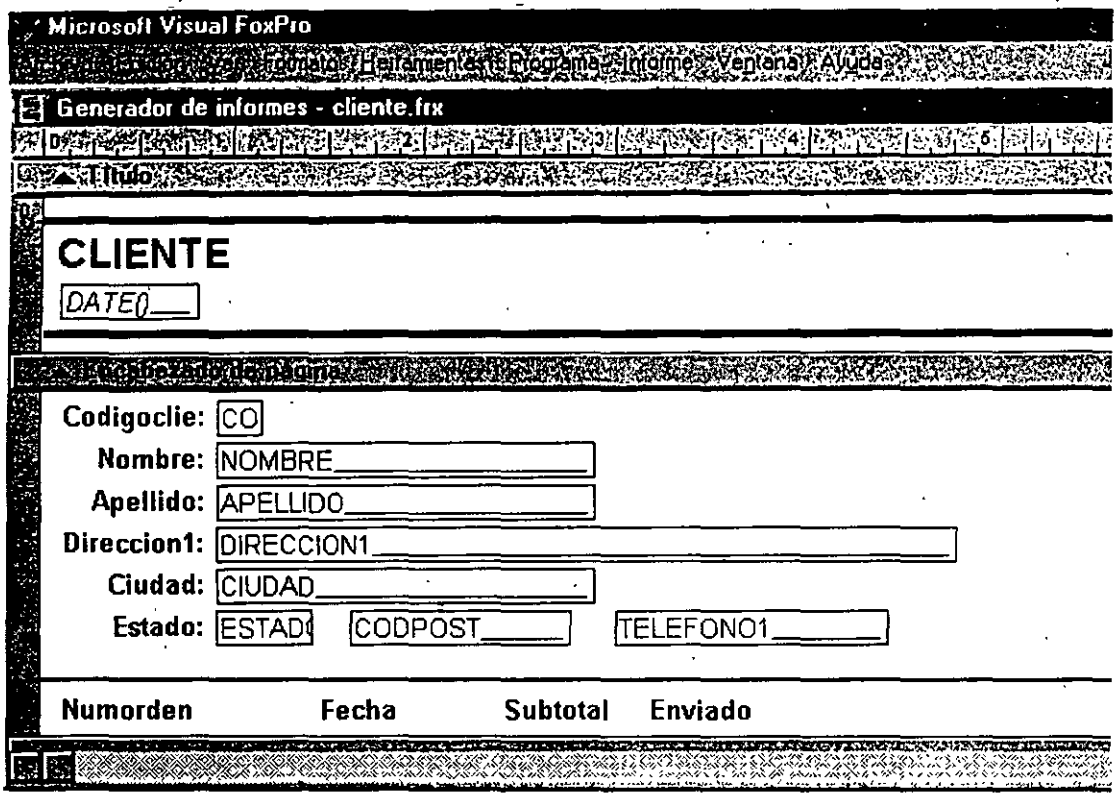
Alargar si hay desbordamiento

Los campos de texto, en especial los campos memo, pueden contener considerablemente más texto del que cabrá en la anchura proporcionada por un campo. En este caso, sólo marque la casilla de verificación y se agregarán filas adicionales hasta que se haya imprimido el contenido del campo (hasta el punto en que el resto del campo sólo consiste en espacios en blanco).

Cree un informe uno a varios con el Asistente para informes

Si comienza de nuevo el proceso de creación de un informe y selecciona Asistente para informes uno a varios de la lista, obtendrá un tipo de informe por completo diferente. Es evidente que debe tener dos archivos que están relacionados como tabla primaria y tabla secundaria. Para el archivo primario, utilice Cliente, y para el archivo secundario, emplee Pedidos. Se relacionan por medio del botón Agregar. Yo hice una vista previa de los tres estilos y Doble carta es la que tenía mejor apariencia. El resultado, guardado como CLIENTE.FRX, se muestra en la figura 10-16.

Figura 10-16



Informe uno a varios generado.

Como puede apreciar, la forma generada supone que la información de la tabla primaria (Cliente) va en una banda Grupo de inicio y la información de la tabla secundaria (Pedidos) va en la banda Detalle.

Parecía una buena idea mover el campo Total compras de la banda Encabezado de grupo 1 a la banda Pie de grupo 1. Recuerde, esto se hace desde un campo en el archivo Cliente. Si por alguna razón, el contenido de este campo no coincide con lo que en realidad se encuentra en el archivo Pedidos, se le presionará mucho para que lo explique. De modo que se realizaron tres cambios:

- Se extendió la banda Pie de grupo 1 al hacer clic y arrastrar hacia abajo la línea de separador por debajo de ésta.
- Se tomó la etiqueta Compras y el cuadro de texto, y se movieron hacia abajo al área de Pie de página de grupo 1 que se acabaron de extender.
- Se hizo doble clic en el cuadro de texto para hacer que aparezca la Expresión de informe y se cambió el nombre del campo fuente CLIENTE.Compras a ORDENES.TotalOrden. Luego se hizo clic en Cálculos y se seleccionó Suma. Restablecer, Código de cliente significa

que se debe restablecer el total en cero antes de comenzar a acumularlo para el cliente siguiente. También se cambió el tamaño de los recuadros que se encuentran alrededor tanto del Encabezado de grupo 1 como del Detalle ya que la cantidad de datos que se imprimía no ocupaba todo el ancho de la página. El resultado se presenta en la figura 10-17.

Los informes uno a varios pueden incluir datos de varias tablas. Por ejemplo, el área del Encabezado de grupo 1 puede contener campos del archivo Pedidos, con los campos de Clientes relacionados, en tanto que la banda Detalle tiene campos de DETALLES.DBF.

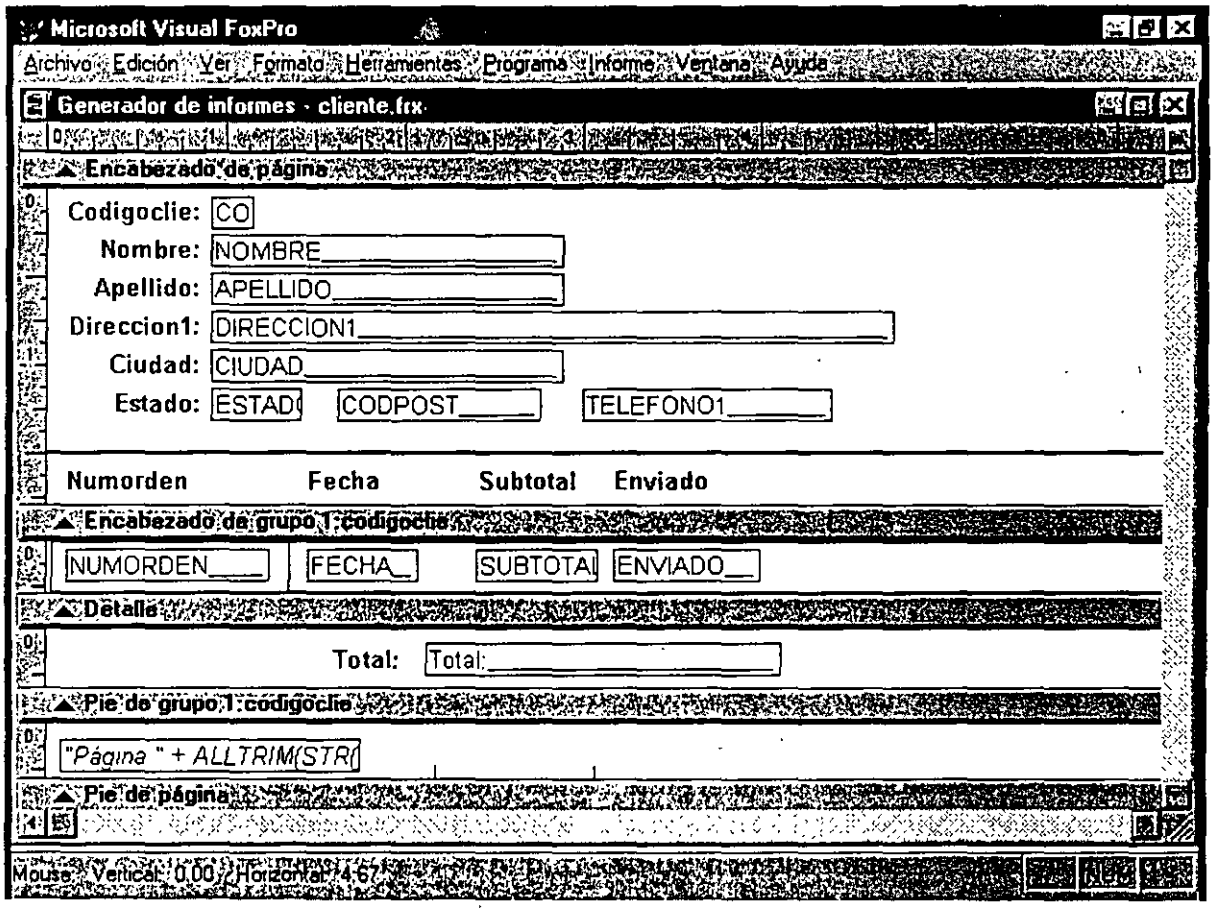


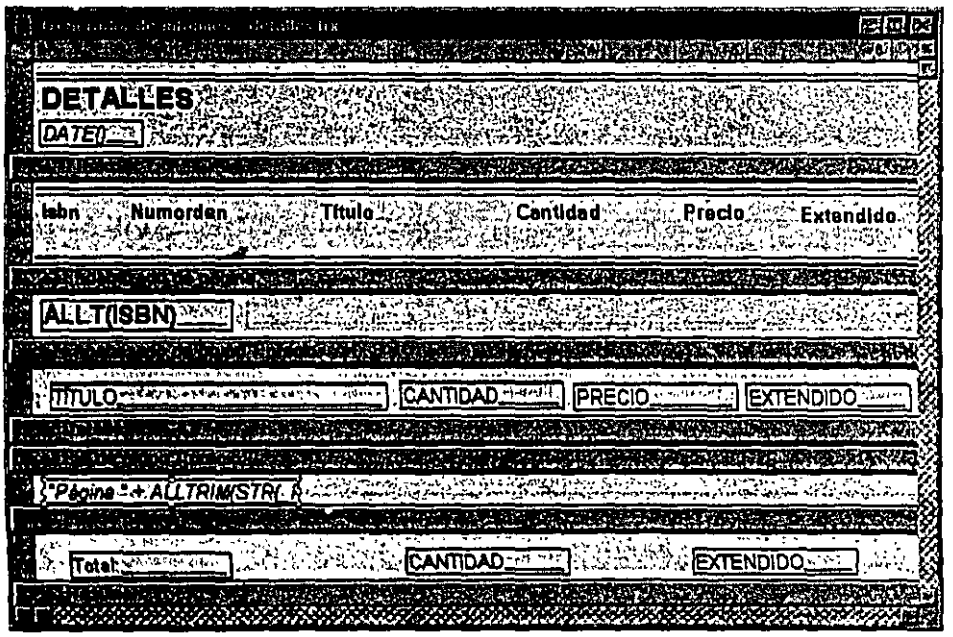
Figura 10-17

Informe modificado.

Cree un informe de grupos/totales con el Asistente para informes

El Asistente para informes genera informes de grupos/totales. Seleccionó este asistente, luego se escogió Detalles de la tabla de entrada. Los campos de interés eran Número de pedido, ISBN, Título, Cantidad, Precio y Extendido. Ésa también es la orden de despliegue. En el paso 2 del Asistente, donde pidió un campo de total, se seleccionó ISBN. El siguiente paso pedía una clave adicional de ordenación, pero no hay una. El informe generado se presenta en la figura 10-18, con una modificación. El asistente incluía Precio como uno de los campos de total general en la línea Resumen, pero sumar un total para todos los libros no tiene sentido. De modo que se hizo clic una vez sobre éste para resaltarlo, luego se suprimió.

Figura 10-18

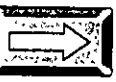


Informe de total/subtotal para DETALLES.DBF.

Los informes de subtotal sólo tienen una bandá Pie de grupo con los campos numéricos de su banda Detalle repetidos en las mismas posiciones de columna y su opción de cálculo establecida en Suma. Ya que se encuentran en la línea de Pie de grupo, acumulan la suma de los campos correspondientes hasta que el campo Encabezado de grupo cambia de valor; después se restablecen en cero.

El cuadro de diálogo Expresión de informe aparece si se hace doble clic en un campo, pero también se puede abrir al hacer clic en el objeto con el botón secundario del ratón y luego seleccionar Propiedades de un menú contextual. Es

interesante ver el Generador de expresiones, que se utiliza para modificar las características del campo, como una manera de establecer las propiedades de un campo Pie de grupo. Pero eso es lo que son las propiedades, características modificables.



El comando REPORT FORM (Formulario de informe)

El comando que se utiliza para ejecutar un informe es éste:

```
REPORT FORM NombreArchivo1 ¦ ?
  [ASCII]
  [ENVIRONMENT]
  [Scope] [For Expresion1] [WHILE Expresion2]
  [HEADING cTextoEncabezado]
  [NOEJECT]
  [NOCONSOLE]
  [NOOPTIMIZE]
  [PDSETUP]
  [PLAIN]
  [PREVIEW [NOWAIT] ]
  [TO PRINTER [PROMPT] ¦ TO FILE NombreArchivo2]
  [SUMMARY]
```

y los argumentos son los siguientes:

? Despliega el cuadro de diálogo Abrir, de donde se puede seleccionar un archivo de informe.

ASCII (sólo Visual FoxPro) Crea un archivo de texto en ASCII a partir del archivo de definición de informe. Cualesquier gráficos, líneas, rectángulos o rectángulos redondeados en la definición del informe no son salida. Utilice las variables de memoria del sistema _ASCIICOLS y _ASCIIROWS para establecer el número de columnas y filas en cada página del archivo de texto ASCII. Los valores predeterminados son 80 para _ASCIICOLS y 63 para _ASCIIROWS.

ENVIRONMENT Utiliza el entorno de datos guardados en el archivo de definición de informe, incluyendo los nombres de todas las tablas abiertas y archivos de índice, el orden del índice y cualquier vàriable SET RELATION (Establecer relación).

Scope ALL (Todos), NEXT (siguiente) # de registros, RECORD # (Registro #) y REST (resto).

FOR lExpresion1 Sólo incluye registros en el filtro.

WHILE lExpresion2 Es similar al argumento anterior, excepto que éste supone que su tabla está indexada y que acaba de realizar una operación SEEK (búsqueda). Los registros se imprimen en tanto que la expresión lógica lExpresion2 sea .T. (verdadera).

HEADING cTextoEncabezado Texto para un encabezado adicional en cada página del informe.

NOEJECT (sólo MS-DOS) Suprime una alimentación de forma a la impresora antes de imprimir el informe.

NOCONSOLE No despliega el informe en la pantalla al imprimirlo.

NOOPTIMIZE Impide la optimización Rushmore de REPORT (Informe).

PDSETUP Disponible en FoxPro sólo para MS-DOS.

PLAIN Imprime un encabezado de página sólo al inicio del informe.

PREVIEW Despliega el informe en modo de presentación preliminar en vez de enviar el informe a la impresora. Para imprimir un informe, debe utilizar REPORT con TO PRINTER (Hacia impresora).

NOWAIT Continúa el programa de llamada sin cerrar la ventana de presentación preliminar de la página.

TO PRINTER [PROMPT] Envía un informe a la impresora. PROMPT hace que aparezca un cuadro de diálogo de valores de la impresora antes de que comience la impresión.

TO FILE [nombrearchivo] (sólo DOS) Envía la salida a un archivo de texto con la extensión predeterminada .TXT.

SUMMARY Sólo se imprimen totales y subtotales (no líneas de banda Detalle).

El Generador de etiquetas

El Generador de etiquetas comparte gran parte de la sintaxis del Generador de informes y algunas de sus características. Las principales peculiaridades son que usa etiquetas Avery para dar formato, de modo que usted debe decidir en qué tipo de etiqueta Avery va a imprimir, y que las líneas individuales de la etiqueta se crean usando una interfaz de diseño única, que requiere algo de tiempo para acostumbrarse a su uso.

Haga clic en Archivo, Nuevo, Etiqueta, Asistente para obtener la pantalla que aparece en la figura 10-19. Seleccione Cliente para el archivo y haga clic en el botón Siguiente. Luego, escoja el tamaño de la etiqueta en que quiere imprimir. Esto determina el número de líneas disponibles para el Generador de etiquetas.

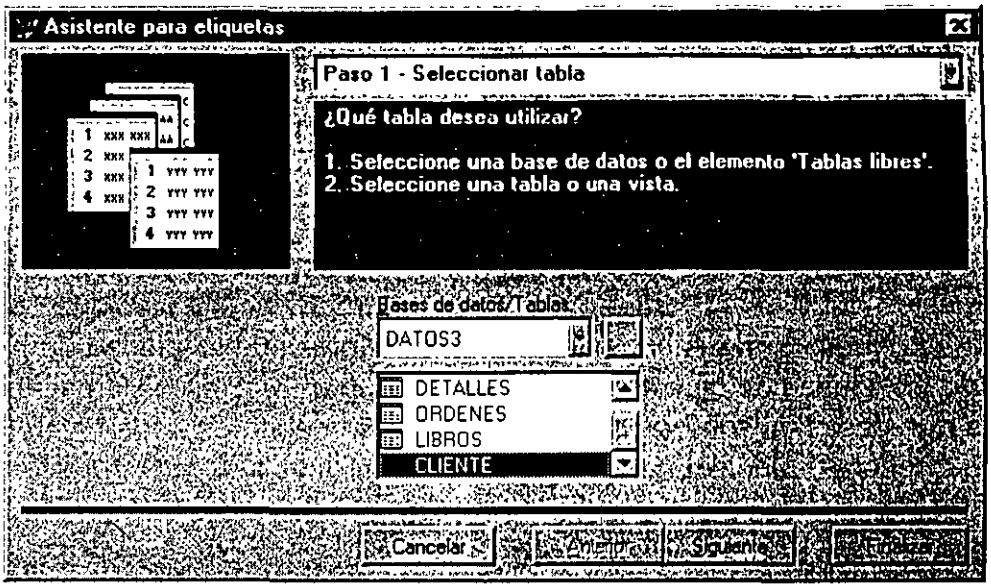


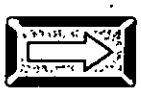
Figura 10-19

Cuadro de diálogo Asistente para etiquetas.

El paso 3 del Asistente para etiquetas es el de Distribución. Aquí es donde se diseñan las líneas individuales de la etiqueta. Una de las características de las etiquetas es que se pueden unir o concatenar varios campos, por ejemplo, apellido, una coma, un espacio y el nombre. El Generador de etiquetas eliminará los espacios restantes de los campos, pero aún necesita indicar qué va en dónde. Así, haga clic en Nombre_contrato, luego en el botón de coma, después en el botón de espacio, y por último, en el campo IDCliente. Haga clic en el símbolo de retorno de carro para bajar a la siguiente línea. Agregue los campos restantes, usando el botón de retorno de carro para moverse una línea hacia abajo.

Nótese que el botón de flecha izquierda, que se usa para eliminar conceptos escogidos de la lista Campos seleccionados, elimina los elementos seleccionados uno a la vez y que los conceptos seleccionados aún aparecerán en la lista Campos disponibles. Asimismo, nótese que puede teclear texto en el campo Texto y agregarlo al contenido de la etiqueta. Estas ligeras diferencias de la forma en que trabaja el Generador de informes pueden ser un tanto desconcertantes al principio, pero la mayoría de los programadores diseñan etiquetas con poca frecuencia, de modo que no será una situación cotidiana.

Haga clic en Finalizar y su etiqueta está hecha. Puede tener una presentación preliminar de la misma con el botón Presentación preliminar. Cuando proporciona un nombre, éste se guarda con el par de extensiones .LBX y .LBT. La sintaxis para imprimir etiquetas de hecho es idéntica a la que se usa para imprimir informes, excepto que se teclea LABEL FORM (Forma de etiqueta) en vez de REPORT FORM (Forma de informe).



Genere sus propios informes

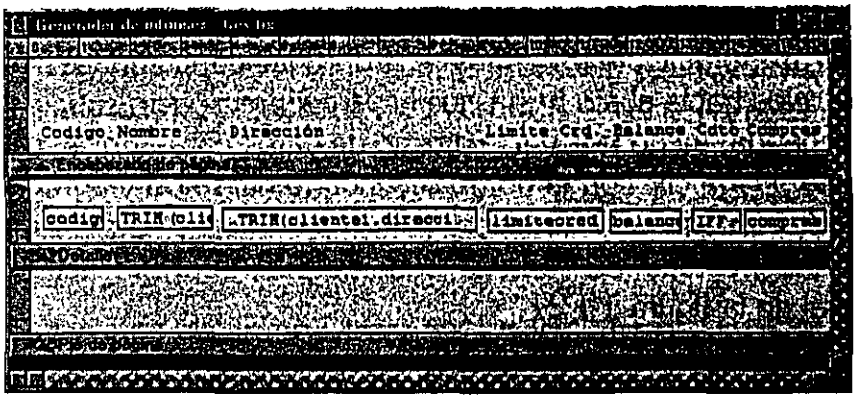
Siendo tan útiles como son estos tres asistentes, tal vez desee diseñar informes especiales. Suponga, por ejemplo, que quiere una lista de clientes por estado, ciudad y apellido del cliente. Esto implica que Alabama debe ir antes que Nueva York y que, en Nueva York, Albany debe aparecer antes que la Ciudad de Nueva York. Luego en Ciudad de Nueva York, se listan los nombres en orden alfabético. Y suponga que desea un subtotal de los saldos que se deben en cada nivel. Si no tiene dicha etiqueta de índice, no tiene que ser parte del CDX de producción. Sólo capture esto:

```
index on upper ( estado + ciudad + apellido ) to xyz
```

Ya que esto crea un .IDX en la operación, que puede suprimir al terminar, no cambia nada acerca del ambiente de producción. Después, puede crear un formulario de informe al teclear CREATE REPORT TRES. Obtendrá la pantalla que aparece en la figura 10-20.

No obstante, el Escritor de informes no sabe mucho sobre la nueva expresión de índice, mucho menos acerca de los tres campos, de modo que el informe no funcionará sin esto. Pero para Visual FoxPro, éstos son eventos no relacionados y usted es responsable de hacer que coincidan.

Figura 10-20

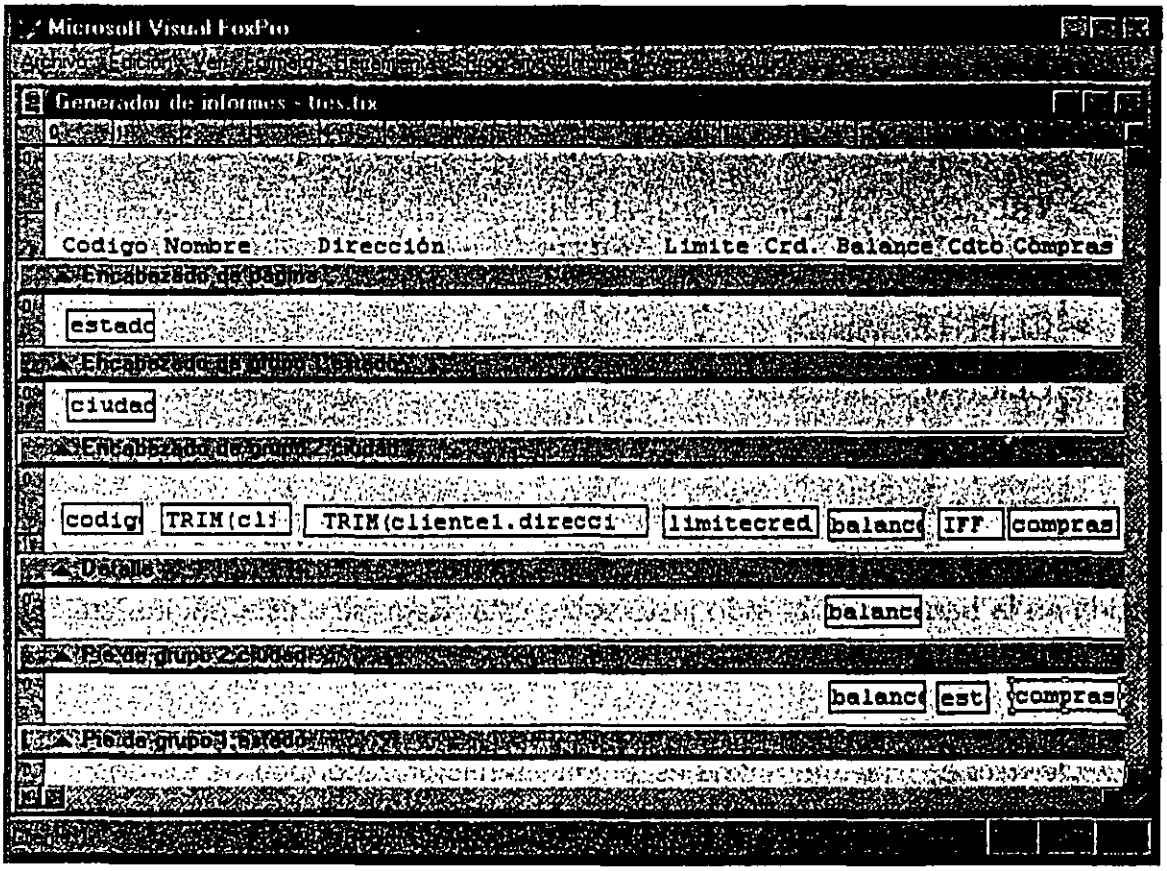


Pantalla del Generador de informes.

Añada bandas de subtotal de grupo

Haga clic en Informe, Agrupar datos. Introduzca Estado para el primer nivel y Ciudad para el segundo. El Apellido no es un nivel de grupo, aún si es una clave de ordenación, ya que no va a pedir el subtotal de todas las personas con el mismo apellido. Así, en general, tendrá un grupo menos que las claves de ordenación. Haga clic en Aceptar cuando termine.

Figura 10-21



Generador de informes con dos grupos añadidos.

La distribución del informe habrá cambiado cuando regrese a ella. Ahora tiene dos Encabezados de grupo adicionales y Pies de grupo que concuerdan. Tómelos con el ratón y arrástrelos hacia abajo hasta que haya hecho espacio suficiente para un campo en cada banda; luego use Ver, Entorno de datos para abrir la ventana Entorno de datos. Haga clic con el botón secundario del ratón y agregue el archivo Cliente, jalando la ventana de descripción de archivo lo suficiente para poder ver los campos Ciudad y Estado. Su ventana Generador de informes debe tener un aspecto similar al de la figura 10-21.

Ahora haga clic en la palabra *Estado* en la ventana Entorno de datos, mantenga oprimido el botón del ratón y arrástrelo sobre la primera banda Encabezado de grupo y suéltelo. El campo Estado aparecerá en la banda. Muévalo lo suficiente hacia la derecha como para insertar una etiqueta con la leyenda Estado: utilizando el icono A. Haga lo mismo para arrastrar y soltar el campo Ciudad en la banda Encabezado de grupo 2.

Agregue campos a la banda Detalle

Ahora necesita imprimir los siguientes campos en la banda Detalle:

- Código de cliente
- TRIM(CLIENTE.Nombre)+[ ]+TRIM(CLIENTE.Apellido)
- TRIM(CLIENTE.Direccion1)+CHR(13)+TRIM(CLIENTE.Direccion2)
- CLIENTE.LimiteCrd
- CLIENTE.Balance
- CLIENTE.Creditousa
- CLIENTE.Compras

Ponga los campos que desea ver en la banda Detalle, con una etiqueta que coincida en la banda Encabezado justo sobre ésta. Regrese a esta página en cuanto no tenga más espacio horizontal.

¿De regreso tan pronto? No toma mucho tiempo con campos de 40 caracteres de ancho. De hecho, ¿qué tan ancha es la concatenación recortada de los dos campos de dirección, con un CHR(13) para unirlos? ¿Cómo puede lograr que ur cliente imprima en un informe tan angosto?

201

Una solución es cambiar la orientación del papel de vertical a horizontal en la opción Preparar página del menú Archivo. Pero eso es muy sencillo y no le enseña nada acerca del estrechamiento vertical.

Todos los datos que caben en la impresión

Haga clic una vez en el campo Dirección, luego tome la marca de centro en el lado derecho y reduzca su anchura aproximadamente cinco centímetros. Después haga doble clic en el campo para desplegar el cuadro de diálogo Expresión de informe y haga clic en la casilla Alargar si hay desbordamiento en la esquina inferior izquierda. Edite los campos Direccion1 y Direccion2 del primer registro para que contengan algo como esto:

```
Direccion1: Cedro # 512
Direccion2: Col. Atlampa
```

Ahora haga una presentación preliminar del informe. Verá más o menos lo mismo que aparece en la figura 10-22, que responde varias preguntas. Recortar un campo de carácter elimina los espacios restantes, concatenar dos campos con CHR(13) fuerza un retorno de carro entre ellos y el estrechamiento vertical hace que los campos de texto envuelvan la línea siguiente.

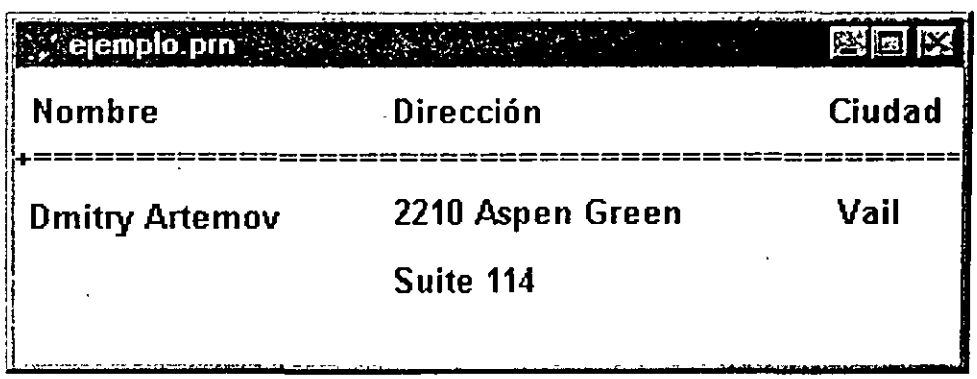


Figura 10-22

Efecto de estrechamiento vertical y un retorno de carro forzado entre dos campos concatenados.

Es evidente que puede agrupar simplemente campos y proporcionar campos lo suficientemente largos para contener la cantidad máxima posible de texto. Pero recortar espacios restantes, junto con campos más angostos y la opción Alargar si hay desbordamiento, brindan una solución clara para los campos de texto con longitud desconocida.

Agregue subtotales en las bandas Pie de grupo

Para obtener los subtotales, haga clic en el campo Saldo en la banda Detalle una vez con el botón secundario del ratón, seleccione Copiar, luego haga clic en Pegar. Tome el campo nuevo con el apuntador del ratón y muévalo a la banda Pie de grupo 1 justo abajo del campo. Haga doble clic y después haga clic en Cálculos, marque el botón Suma y guárdelo. Haga lo mismo con el campo Compras, después repita el proceso para ambos campos para copiarlos en el Pie de grupo 2. Nótese que el campo Restablecer en el cuadro de diálogo Campo calculado o de subtotal se establece automáticamente en el nombre del campo de grupo para la banda Pie de página en que está.

Si ha introducido algunos datos de prueba es fácil hacer el seguimiento, como una docena de registros, por ejemplo, con dos estados diferentes y dos registros cada uno para varias ciudades y un valor de 1 en cada campo numérico, puede hacer una presentación preliminar y verificar los resultados.

Elementos especiales del informe

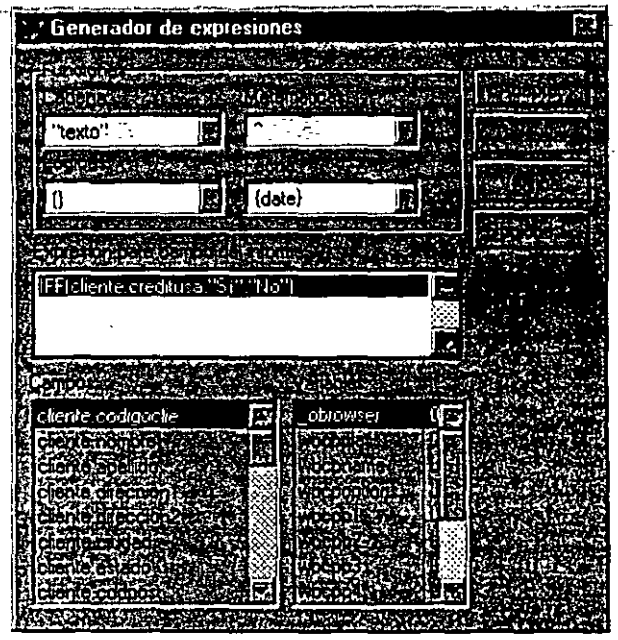
CreditoUsa, el sexto campo en la banda Detalle, es un campo lógico con valores posibles de .T. o .F. A los encabezados de propulsión no les interesa mucho esta clase de salida, pero la mayoría de los usuarios prefiere un simple sí o no. Para realizar esto, haga doble clic en el campo de la banda Detalle para CreditoUsa y teclee lo siguiente en la Expresión de informe:

```
IIF (CLIENTE.CreditoUsa, "Sí", "No" )
```

Los resultados, que aparecen en la figura 10-23, son mucho más fáciles de ver. Puede ampliar este ejemplo para recodificar información de campo en código críptico, como éste:

```
IIF (Codigo = 1, [Australia], ;
IIF (Codigo = 2, [Nueva Zelanda], ;
IIF (Codigo = 3, [Tahiti], ;
IIF (Codigo = 4, [Bora Bora], ;
[otros destinos])))))
```

Figura 10-23



Campo CreditoUsa combinado a Sí o No.

Agregue otros alias al informe

Es fácil agregar campos de un archivo relacionado. Suponga que tiene un archivo que contiene los nombres de los estados, así como sus abreviaturas. Si el archivo tiene un aspecto como éste:

```
ESTADOS.DBF

CODIGO Carácter    2    (Etiqueta de índice CODIGO)
NOMBRE Carácter   12
```

Todo lo que tiene que hacer para imprimir el nombre del estado en vez (o además) del código del estado es agregar estas cuatro líneas al código que ejecute antes del informe:

```
SELECT ESTADOS
SET ORDER TO ESTADOS
SELECT CLIENTE
SET RELATION TO ESTADO INTO ESTADOS ADDITIVE
```

La palabra clave ADDITIVE es necesaria en caso de que CLIENTE también esté relacionado con otros archivos. Ahora agregue ESTADOS a su Entorno de datos y agregue el campo ESTADOS.Nombre en su primera banda Encabezado de grupo 1. Estará en la lista de campos seleccionables en el cuadro de diálogo Expresión de informe, ya que lo ha agregado al Entorno de datos para el informe. ¡No olvide incluir las cuatro líneas de código requeridas para operar el informe!

El efecto de SET SKIP en las líneas de la banda Detalle

Si hace un informe simple de banda Detalle usando un archivo con 25 registros, tendrá 25 líneas de salida de la banda Detalle. Así que si imprime 10 pedidos que tienen un total de 25 líneas de detalle y el código de instalación contiene las líneas:

```
SELECT DETALLES ORDER OrdenNum
SELECT ORDENES
SET RELATION TO OrdenNum INTO DETALLES
SET SKIP TO DETALLES
```

también obtendrá 25 líneas de salida de banda Detalle. Pero eso es lo que implica SET SKIP (Dar salto). Cada vez que su programa o informe espera para avanzar al siguiente registro en la tabla seleccionada, actúa como si la tabla seleccionada fuera la tabla de SET SKIP; cuando la expresión SET RELATION (Establecer relación) deja de ser verdadera, hace un salto en la tabla primaria.

Imprima cien copias de una etiqueta o página

Una consecuencia natural de esta conducta es que Visual FoxPro imprime una banda Detalle para cada registro de Detalle, aun si la banda Detalle no contiene ningún dato del registro de detalle. Por ejemplo, si quiere imprimir cien etiquetas de dirección de remitente para un nombre del archivo de clientes, cree una etiqueta que indique su archivo de clientes. Luego ejecute este programa:

```
CREATE CURSOR CONTADOR (X C(1))
FOR I = 1 TO 100
   INSERT INTO CONTADOR (" ")
ENDFOR
LABEL FORM CLIENTE TO PRINT
USE IN CONTADOR
```

Así es cómo funciona: éste pequeño programa crea cien registros con un campo en blanco de un carácter en ellos, luego ejecuta ABEL FORM (Forma de etiqueta) que señala a CLIENTE.DBF, un campo no relacionado en absoluto. A Visual FoxPro no le importa; imprime 100 etiquetas y regresa. La línea del contador

205

CREATE CURSOR abre el área de trabajo siguiente disponible y hace un archivo temporal en la misma. Las tres líneas que siguen ocupan 100 registros en blanco, la siguiente imprime 100 etiquetas y la última borra el cursor.

Informes con columnas múltiples

Algunos tipos de información caben en un formato angosto, de modo que una página completa es más de lo que necesita. En tales casos, puede reducir los requerimientos de papel. Si puede salvar un árbol, es importante.

Se utilizará el archivo de clientes para diseñar el directorio telefónico de una compañía. Teclee CREATE REPORT TWOCOL (Crear informe con dos columnas), luego haga clic en Ver, Entorno de datos y agregue el archivo de clientes. Introduzca primero el código de cliente (IDCLIENTE), luego baje un poco e introduzca cuatro cuadros de texto, usando el cuadro de diálogo Expresión de informe para introducir lo siguiente:

```
TRIM (Nombre) + " " + TRIM(Apellido)
Direccion1
TRIM (Ciudad) + " , " + Estado + " " + CP
Telefono1
```

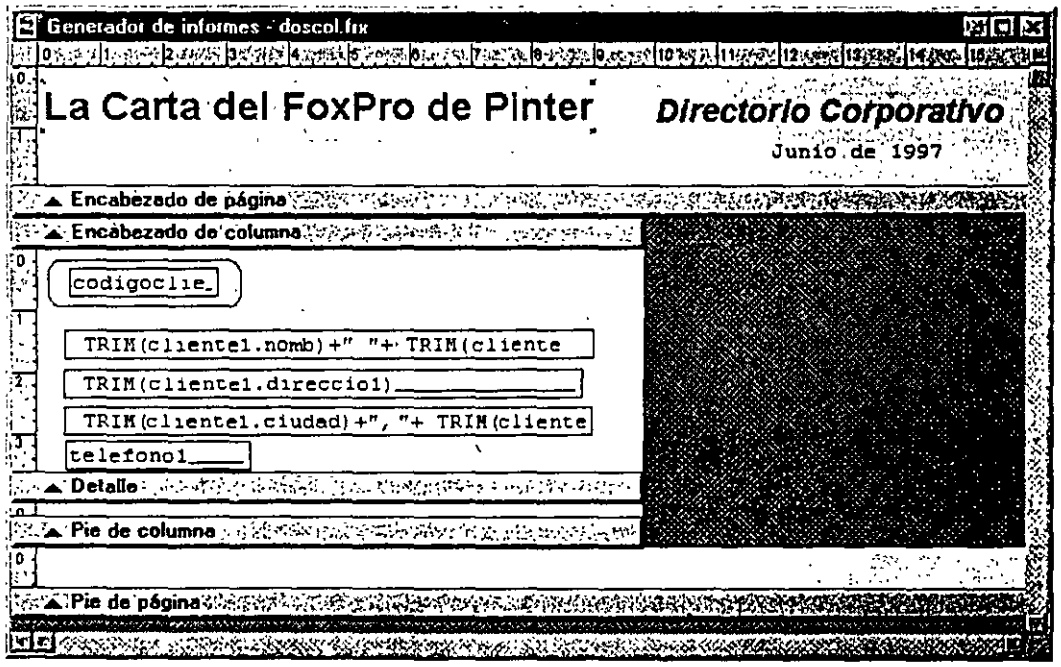


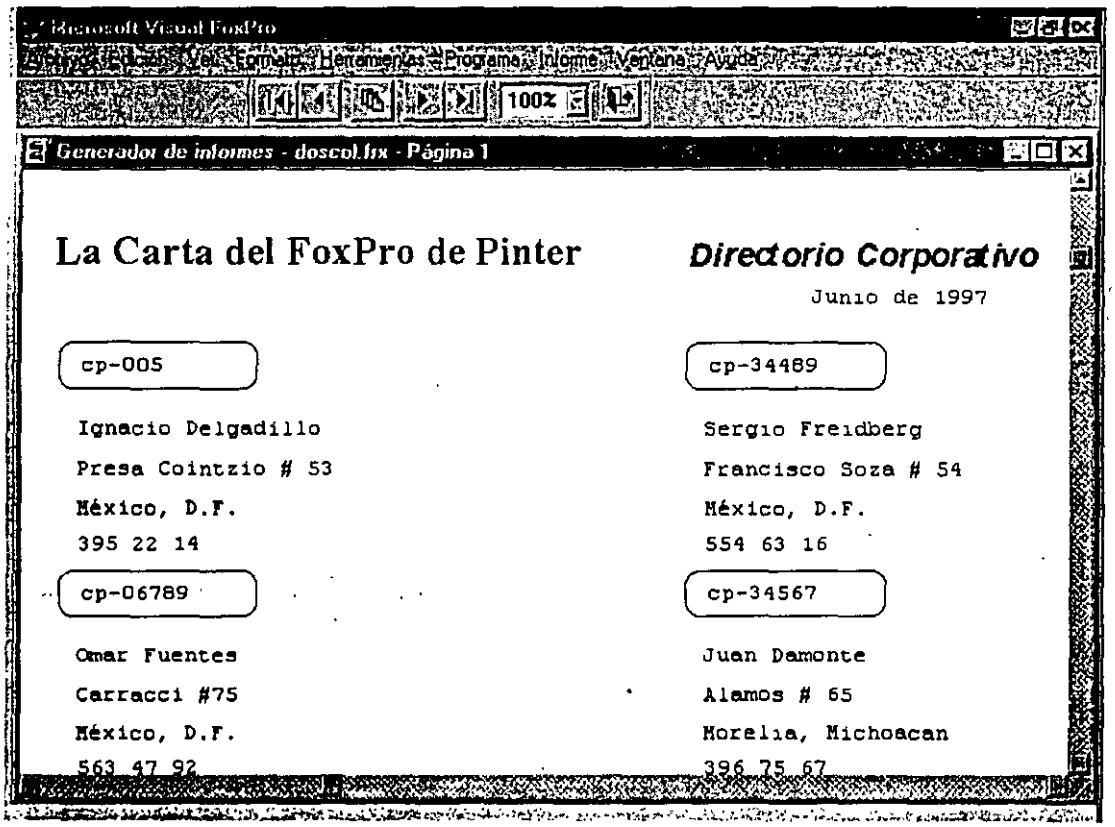
Figura 10-24

Despliegue del informe en dos columnas para el directorio.

Mejoré la distribución un poco, como se ilustra en la figura 10-24. Haga clic en Preparar página del menú Archivo e introduzca 2 para el número de columnas. Haga clic en Presentación preliminar y verá el informe que aparece en la figura 10-25.

Como lo indica el cuadro de diálogo Preparar página, hay varias opciones más que pueden afectar el formato final de su informe. El recuadro Área de impresión, que contiene las opciones Página imprimible o Toda la página, es útil si su impresora sólo puede imprimir a 5 milímetros del borde real de la página.

Figura 10-25



Directorio en dos columnas terminado.

11

Uso del SQL

CAPITULO 11

CUANDO apareció SQL por vez primera en una versión previa de FoxPro, yo no sabía qué hacer con él. Ahora que ha llegado la computación cliente/servidor lo he imaginado finalmente.

FoxPro heredó la sintaxis peculiar de acceso de datos de xBASE; SKIP, GO TOP, GO BOTTOM, GO &recno y otros comandos eran la forma de operar en una tabla. Con el paso del tiempo, nadie siguió a xBASE de la misma manera. En su lugar, se adoptó el lenguaje de consultas estructurado (SQL: *structured query language*) en el resto del mundo de la computación.

Por último, apareció una nueva vèrsión de FoxPro, la cual incluía tres comandos SQL: CREATE, SELECT e INSERT. Estos comandos funcionaban, pero también lo hacían todos los comandos originales. Luego vi el nuevo comando UPDATE en Visual FoxPro y todo cobró sentido. SQL podía hacer más que leer de las tablas de datos; podía escribir en ellas. Y los comandos son los mismos, ya sea que trabaje en su estación de trabajo, en una LAN o en el mainframe de la compañía.

No sé cuántas personas más tengan el mismo problema, pero con la llegada de Visual FoxPro es claro: se puede escribir en aplicaciones con acceso a enormes almacenes de datos valiosos en su pequeña computadora. Los trabajos que solían durar un año y costaban a las compañías cientos de miles de dólares en un mainframe ahora se pueden realizar en una PC con un programador de FoxPro por una parte del dinero, en sólo un mes o un periodo similar.

Acoplar un entorno de desarrollo muy eficiente con el poder de las bases de datos SQL de último nivel hace de Visual FoxPro un candidato importante para una gran cantidad de aplicaciones que en el pasado sólo podían efectuarse con herramientas caras que con frecuencia eran muy lentas. Una herramienta popular de desarrollo de último nivel cliente/servidor que se vende por cerca de 3,000 dólares tal vez no parezca mucho más cara que invertir 500 dólares en Visual FoxPro, pero agregue a esto el doble o triple de tiempo de programación y verá que usar el otro producto podría sumar decenas de miles de dólares al costo de un trabajo. Visual FoxPro es la posibilidad que las compañías esperaban para desarrollar aplicaciones económicas cliente/servidor. Y el mejoramiento del soporte de VFP en SQL es la clave.

Aun si no usa un servidor SQL para almacenar datos, usará los comandos SELECT de SQL en todas las pantallas que tengan un cuadro de lista o un cuadro combinado porque RowSourceType 3-SQL es una forma excelente de poblarlas. Sólo hay una complicación, la cual se abordará en breve.

En este capítulo, usted verá la sintaxis de los comandos SQL de FoxPro, así como algunas de las implicaciones de usar SQL para el acceso a archivos. Si usted

'iseña sus aplicaciones para acceso remoto, hay varias tareas que debe y no debe ,acer. Así que dé el primer paso.

Los comandos básicos

Aunque el SQL de cada vendedor tiene su propio conjunto de comandos, que con frecuencia incluyen los que no están respaldados por la mayoría de los otros vendedores, hay cinco comandos SQL básicos:

- SELECT
- INSERT
- DELETE
- UPDATE
- CREATE

Si escribe sus aplicaciones usando estos comandos en vez de sus contrapartes de xBASE, está en camino a crear aplicaciones cliente-servidor que sean independientes de su fuente de datos.

SELECT

Se utiliza SELECT para recuperar datos. La sintaxis básica de SELECT es:

```
SELECT [ALL ¦ DISTINCT]
  [Alias.] elemento [AS Nombrecolumna]
  [, [Alias.] elemento [AS Nombrecolumna] ...]
FROM [Basededatos!]Tabla [Alias_local]
  [, [Basededatos!]Tabla [Alias_local] ...]
[[INTO DESTINO]
  ¦ [TO FILE Nombredearchivo [ADDITIVE] ¦ TO PRINTER [PROMPT] ¦ TO SCREEN]]
[PREFERENCE Nombrepreferencia]
[NOCONSOLE]
[PLAIN]
[NOWAIT]
[WHERE CondicionUnion [AND CondicionUnion ...]
 [AND ¦ OR Condicionfiltro [AND ¦ OR Condicionfiltro ...]]]
[GROUP BY  campo_grupo [, campo_grupo ...]]
[HAVING Condicionfiltro]
[UNION [ALL] ComandoSELECT]
[ORDER BY campo_orden [ASC ¦ DESC] ...]]
```

La cláusula SELECT especifica los campos, constantes y expresiones que se despliegan en los resultados de la consulta. Los argumentos son los siguientes:

* DISTINCT

Excluye los duplicados de cualquier renglón de los resultados de la consulta. Sólo se tiene un argumento DISTINCT por cláusula SELECT.

* Alias

Califica nombres de elementos que coinciden. Cada elemento que especifica con campo genera una columna de los resultados de la consulta. Si dos o más variables tienen el mismo nombre, incluya el alias de la tabla y un punto antes del nombre del concepto para impedir que se dupliquen las columnas. Campo especifica un elemento que se debe incluir en los resultados de la consulta. Un elemento puede ser uno de los siguientes:

- ➢ El nombre de un campo de una tabla en la cláusula FROM.
- ➢ Una constante que especifica que el mismo valor constante debe aparecer en todos los renglones de los resultados de la consulta.
- ➢ Una expresión que puede ser el nombre de una función definida por el usuario.

* AS Nombrecolumna

Especifica el encabezado para una columna en la salida de la consulta. Es útil cuando el elemento es una expresión o contiene un campo de función y desea dar a la columna un nombre significativo. Nombrecolumna puede ser una expresión pero no puede contener caracteres (por ejemplo, espacios) que no están permitidos en los nombre de campo de la tabla.

* FROM [Basededatos!]Tabla [Alias_local] [, [Basededatos!]Tabla [Alias_local] ...]

Menciona las tablas que deben leerse durante la consulta. Una vez abierta, la tabla sigue así después de completar la consulta.

Ahora que tiene contenedores de base de datos, SQL de VFP se ha modificado para tomarlos en cuenta. Esto se realiza en una forma similar a la sintaxis de Nombredetabla.nombredecampo: un signo de exclamación separa el nombre DBC del nombre de la tabla. Esto es necesario sólo si usa una tabla de una base de datos distinta de la actual.

* Basededatos!

Especifica el nombre de una base de datos no actual que contiene la tabla. Debe incluir el nombre de la base de datos que contiene la tabla si no es la base de datos actual. Incluya el signo de exclamación (!) después del nombre de la base de datos y antes del nombre de la tabla.

* Alias_local

Especifica el nombre temporal para una tabla que se menciona en Tabla. Si especifica un alias local, debe utilizar el alias local en lugar del nombre de la tabla en la instrucción SELECT. El alias local no afecta el entorno de Visual FoxPro. Con frecuencia se usan alias locales que consisten en una sola letra con objeto de reducir la captura.

* INTO Destino

Especifica dónde se deben almacenar los resultados de la consulta. Si incluye la cláusula INTO y la cláusula TO en la misma consulta, se ignora la cláusula TO. Si no incluye la cláusula INTO, los resultados de la consulta se despliegan en una ventana Examinar. También puede emplear TO para dirigir los resultados de la consulta a la impresora o a un archivo. El destino puede ser una de las siguientes cláusulas:

ARRAY Nombredearreglo Almacena los resultados de la consulta en un arreglo (matriz) de variables de memoria. El arreglo no se crea si ningún registro coincide con la consulta.

CURSOR Nombrecursor Almacena los resultados de la consulta en un cursor (Current Set of Records) temporal. No utilice el nombre de una tabla abierta. Si ha desactivado la seguridad (SET SAFETY OFF), Visual FoxPro cerrará la tabla y creará un cursor con ese nombre. Una vez que utiliza USE IN nombredelcursor, Visual FoxPro suprime el cursor. Por esta razón, los cursores generalmente son preferibles que las tablas.

DBF Nombredetabla o TABLE Nombredetabla Almacena los resultados de la consulta en una tabla. Si nombra una tabla que ya está abierta, Visual FoxPro sobrescribirá la tabla. La tabla sigue abierta después de ejecutar SELECT.

* TO FILE Nombredearchivo [ADDITIVE] | TO PRINTER [PROMPT] | TO SCREEN

Si incluye una cláusula TO pero no una cláusula INTO, puede enviar la salida de la consulta a un archivo de texto ASCII llamado Nombredearchivo, a la impresora o

a la ventana principal (SCREEN) de Visual FoxPro. ADDITIVE agrega la salida de la consulta al contenido existente del archivo de texto especificadoen TO FILE Nombredearchivo. TO PRINTER dirige la salida de la consulta a una impresora. La cláusula opcional PROMPT despliega un cuadro de diálogo para cambiar los valores establecidos de la impresora.

* Nombres de columna

Cuando crea una salida de consulta, las columnas se nombran de acuerdo con las reglas siguientes: si un concepto seleccionado es un campo con un nombre único, si el nombre de la columna de salida es el nombre de un campo y si más de un concepto seleccionado tiene el mismo nombre. Por ejemplo, si una tabla llamada Cliente tiene un campo denominado CALLE y una tabla de nombre Empleados tiene un campo que se llama CALLE, las columnas de salida se llamarían CALLE_A y CALLE_B. Para un concepto seleccionado con un nombre de 10 caracteres, el nombre se corta de modo que quepan la raya y la letra. Si algunos conceptos seleccionados son expresiones, sus columnas de salida se llaman EXP_A, EXP_B, etcétera. Las funciones de campo como COUNT( ) y SUM( ) producen las columnas de salida con nombres como CNT_A y SUM_A.

* PREFERENCE Nombrepreferencia

Si los resultados de la consulta se despliegan en una ventana Examinar, puede incluir PREFERENCE para guardar los atributos y opciones de la ventana Examinar para usarlos más tarde. En aplicaciones programadas, por lo regular no empleo esta opción.

* NOCONSOLE

Impide que los resultados de la consulta se desplieguen en la ventana principal de Visual FoxPro. NOCONSOLE puede utilizarse ya sea que se incluya o no la cláusula TO. Sin embargo, si se incluye una cláusula INTO, se ignora NOCONSOLE.

* PLAIN

Evita que aparezcan encabezados de columna en la salida de consulta desplegada. Funciona de la misma manera que NOCONSOLE con respecto a las cláusulas TO e INTO.

* NOWAIT

Continúa la ejecución del programa después de abrir la ventana Examinar y los resultados de la consulta se dirigen hacia aquí. El programa no espera a que se cierre la ventana Examinar pero sigue la ejecución en la línea del programa

inmediatamente después de la instrucción SELECT. No he encontrado un uso para esta función.

* La cláusla WHERE

La sintaxis de la cláusula WHERE es:

```
WHERE CondicionUnion [AND CondicionUnion ...]
[AND ¦ OR Condicionfiltro [ AND ¦ OR Condicionfiltro ...]]
```

Hay dos maneras de filtrar la salida de una consulta. WHERE indica a Visual FoxPro que incluya sólo ciertos registros en los resultados de la consulta. También se requiere WHERE para recuperar datos de tablas múltiples, pero filtra la salida para incluir sólo los registros que satisfacen la condición CondicionUnion (JoinCondition). CondicionUnion especifica los campos que unen las tablas en la cláusula FROM. Si incluye más de una tabla en una consulta, debe especificar una CondicionUnion para todas las tablas después de la primera. Al crear condiciones de unión, recuerde lo siguiente:

- Si incluye dos tablas en una consulta y no especifica una condición de unión, todos los registros en la primera tabla se combinarán con todos los registros en la segunda tabla en tanto que se satisfagan las condiciones de filtro. Dicha consulta puede producir resultados extensos.
- Tenga cuidado al utilizar funciones como DELETED( ), EOF( ), FOUND( ), RECCOUNT( ) y RECNO( ), que respaldan un alias opcional o área de trabajo en condiciones de unión. Incluir un alias o área de trabajo en estas funciones podría generar resultados inesperados. SELECT no utiliza áreas de trabajo; es el equivalente de USE AGAIN. Las peticiones con una sola tabla que emplean estas funciones sin un alias opcional o área de trabajo darán resultados apropiados, pero las peticiones con tablas múltiples que usan estas funciones podrían producir resultados inesperados.
- Tenga cuidado al unir tablas con campos vacíos. Visual FoxPro ajusta los campos vacíos. Por ejemplo, si une CLIENTE.CP y PEDIDOS.CP y CLIENTE contiene 100 códigos postales vacíos y FACTURAS contiene 100 códigos postales vacíos, la salida de la consulta contendrá 10,000 registros.

Emplee el operador AND para conectar condiciones de unión múltiples. Cada condición de unión tiene la forma siguiente:

```
TABLA1.campo1 comparacion TABLA2.campo2
```

Los operadores de unión son los comunes: =, ==, >, >=, <, <=, #= y así sucesivamente. Cuando utiliza el operador = con cadenas, su comportamiento exacto depende de los valores establecidos de SET ANSI. Cuando SET ANSI está en el estado OFF, Visual FoxPro trata las comparaciones de cadena en la forma en que usted estaba acostumbrado, la expresión más corta se encuentra del lado derecho y las comparaciones emplean sólo el número de caracteres en la cadena de la derecha. Cuando SET ANSI está en el estado ON, no importa para la comparación de qué lado se encuentre la cadena corta, lo cual puede ser muy raro para la operación del antiguo xBASE. Sin embargo, si debe trabajar con fuentes SQL tal vez se acostumbre a ello.

* Funciones definidas por el usuario con SELECT

Si usa un FDV en su cláusula SELECT y su desempeño es inaceptablemente lento, considere emplear el API y escriba la función en C o en un lenguaje ensamblador. No obstante, tenga en cuenta que el entorno Visual FoxPro puede ser muy extraño durante la optimización cuando se utilizan FDV, de modo que prepárese para pasar algún tiempo eliminando errores.

* Funciones de campo

Las funciones de campo AVG( ), COUNT( ), MIN( ), MAX( ) y SUM( ) funcionan con expresiones que incluyen un campo. Pueden estar anidadas. Producen campos cuyos nombres empiezan con los tres primeros caracteres de los nombres de las funciones.

* Expresiones de filtro

CondicionFiltro puede tener cualquiera de las formas en los ejemplos siguientes:

- Cliente.IDCliente = Pedido.IDCliente
- Pagos.monto >= 1000
- ALL (SELECT compania FROM cliente WHERE pais = "MX")
- ANY | SOME (SELECT compania FROM cliente WHERE pais = "MX")
- cliente.cp [NOT] BETWEEN 90000 y 99999
- [NOT] EXISTS (SELECT * FROM pedidos WHERE clientes.cp = pedidos.cp)
- clientes.cp NOT IN ("98052", "98072", 98034")

- cliente.idcliente IN (SELECT pedidos.idcliente FROM pedidos WHERE ; pedidos.ciudad = "Monterrey")
- cliente.ciudad NOT LIKE "MX"

La condición LIKE busca todos los campos que concuerdan con la expresión. Puede emplear los signos por ciento (%) y raya (_) como parte de la expresión. La raya representa un solo carácter desconocido en la cadena.

* GROUP BY ColumnaGrupo [, ColumnaGrupo ...]

Las filas de los grupos en la consulta se basan en una o más de las columnas. ColumnaGrupo puede ser el nombre de un campo en una tabla, un campo que incluye una función de campo SQL o un número de columna en una tabla de resultados (el número de columna del extremo izquierdo es 1), por ejemplo, GROUP BY 3.

* HAVING Condicionfiltro

Especifica una condición de filtro que los grupos deben satisfacer para incluirse en los resultados de la consulta. Puede incluir tantas condiciones de filtro como quiera, conectadas con el operador AND u OR. La condición de filtro no puede contener subconsultas.

Una cláusula HAVING sin una cláusula GROUP BY actúa como una cláusula WHERE. Puede utilizar alias locales y funciones de campo en la cláusula HAVING. Utilice una cláusula WHERE para un desempeño más rápido si su cláusula HAVING no contiene ninguna función de campo.

* [UNION [ALL] ComandoSELECT]

Combina los resultados finales de un SELECT con los resultados finales de otro SELECT. UNION verifica los resultados combinados y elimina las filas duplicadas. Utilice paréntesis para combinar cláusulas UNION múltiples. ALL impide que UNION elimine las filas duplicadas de los resultados combinados. Las cláusulas UNION siguen estas reglas:

- No se puede usar UNION para combinar subconsultas.
- Ambos comandos SELECT deben tener el mismo número de columnas en su salida de consulta.
- Cada columna en los resultados de la consulta de un SELECT debe tener el mismo tipo de datos y anchura que la columna correspondiente en el otro SELECT.

- Sólo el SELECT final puede tener una cláusula ORDER BY, que debe referirse a las columnas de salida por número. Si se incluye una cláusula ORDER BY, esto afecta el resultado completo.

Un requerimiento común de los usuarios de bases de datos no está respaldado directamente por la implantación de SQL en Visual FoxPro, que se llama OUTER JOIN. Si tiene registros padre sin registros hijo, o registros hijo prohibidos sin registros padre, no se seleccionarían en forma ordinaria.

Puede usar la cláusula UNION para simular una unión externa. Cuando une dos tablas en una consulta, sólo los registros con valores coincidentes en los campos de unión se incluyen en la salida. Si un registro en la tabla padre no tiene un registro correspondiente en la tabla hijo, el registro en la tabla padre no se incluye en la salida. Una unión externa le permite incluir todos los registros en la tabla padre en la salida, junto con los registros coincidentes en la tabla hijo. Para crear una unión externa en Visual FoxPro necesita un comando SELECT anidado, como el siguiente:

```
SELECT  ;
    cliente.compania,                          ;
    pedido.idpedido,                           ;
    pedido.idemp                               ;
  FROM cliente, pedido                         ;
  WHERE                                        ;
      cliente.idcliente = pedido.idcliente     ;
UNION                                          ;
SELECT                                         ;
      cliente.compania,                        ;
      0,                                       ;
      0                                        ;
    FROM cliente                               ;
    WHERE cliente.idcliente NOT IN             ;
    (SELECT pedido.idcliente FROM pedido)
```

La instrucción SELECT entre paréntesis se procesa primero. Esta instrucción tiene como resultado una selección de todos los números de cliente en la tabla de pedidos.

La cláusula WHERE encuentra todos los números de cliente en la tabla de clientes que no están en la tabla de pedidos. Puesto que la primera sección del comando proporciona todas las compañías que tienen un número de cliente en la tabla de pedidos, todas las compañías en la tabla de clientes ahora se incluyen en los resultados de la consulta.

Dado que las estructuras de las tablas incluidas en un UNION deben ser idénticas, y dos lugares en la segunda instrucción SELECT para representar pedido.idpedido y pedido.idemp de la primera instrucción SELECT.

Las salidas de ambas partes de la instrucción UNION deben producir el mismo número de campos de resultado y tienen que ser del mismo tipo. A SQL no le agrada que los elementos correspondientes no coincidan.

* ORDER BY elemento1 [ASC | DESC] [, elemento2 [ASC | DESC] ...]

Ordena los resultados de la consulta con base en los datos de una o más columnas. Cada elementodeorden debe corresponder a una columna en los resultados de la consulta y puede ser ya sea un campo en una tabla FROM que también es un concepto seleccionado en la cláusula principal SELECT (no en una subconsulta), o una expresión numérica que indica la localización de la columna en la tabla de resultados. (La columna del extremo izquierdo es 1.)

ASC | DESC determinan que la salida se ordene en forma ascendente o descendente. ASC es el parámetro predeterminado. Si no da una cláusula ORDER BY, los resultados son una salida desordenada.

Comentarios

Puede crear una consulta SELECT desde:

- La ventana Comandos
- Un programa de Visual FoxPro (como con cualquier otro comando de Visual FoxPro)
- El Generador de consultas
- El Generador de vistas

Cuando emite SET TALK ON y ejecuta SELECT, Visual FoxPro despliega la longitud de tiempo que la consulta requirió para ejecutarse y el número de registros en los resultados. _TALLY contiene el número de registros en los resultados de la consulta.

Una subconsulta es un SELECT dentro de un SELECT y se debe encerrar entre paréntesis. Puede tener subconsultas múltiples en el mismo nivel (no anidadas) en la cláusula WHERE.

INSERT

Utilice INSERT para agregar registros a una tabla. La sintaxis del comando INSERT es:

```
INSERT INTO nombretabla [(nombrecampo1  [, nombrecampo2, ...])]
   VALUES (eExpresion1 [, eExpresion2, ...])
```

o bien:

```
INSERT INTO nombretabla FROM ARRAY nombrearreglo | FROM MEMVAR
```

También hay un comando INSERT BLANK en Visual FoxPro, que no está relacionado con el comando INSERT de SQL. Es la única manera de agregar una línea en blanco en un archivo en una localización precisa (por ejemplo, antes del registro 2) de modo que sea muy accesible, pero no es lo mismo. Los argumentos son:

```
INSERT INTO nombretabla
```

Puede poner un solo registro en nombretabla que puede ser una cadena o una variable de un carácter y también puede contener información de la ruta. El área actual de SELECT no cambia, aunque el archivo se abra y se deje abierto si no lo estuviera ya. Las cláusulas del comando INSERT son las siguientes:

* La cláusula FIELDS

La sintaxis para FIELDS es:

```
[(campo1 [, campo2 [, ...]])]
```

que menciona los campos en que deben insertarse los valores. No emplee la palabra clave FIELDS; no existe. En ocasiones tecleo la palabra FIELDS sólo para ver cómo desaparece. Supongo que era demasiado lógico incluir una palabra clave FIELDS...

Si va a proporcionar valores para todos los campos en la tabla, puede descartar los nombres de los campos. Por ejemplo, si tiene una tabla llamada Notas, con un solo campo de 30 caracteres, la sintaxis siguiente es correcta:

```
INSERT INTO notas VALUES ("Cierre de libros 11/30/95")
```

Sin embargo, tal vez sea una buena idea listar todos los nombres de los campos, sólo en caso de que posteriormente agregue más campos y el código original deje de funcionar.

* La cláusula VALUES

Su sintaxis es:

```
VALUES (eExpresion1 [, eExpresion2 [, ...]])
```

Tiene que incluir la fuente de los datos en una de tres maneras: la cláusula VALUES( ), FROM ARRAY o FROM MEMVAR. Si usa la cláusula VALUES, puede nombrar ya sea todos los campos de la tabla o sólo algunos de ellos. No obstante, los valores entre paréntesis, que pueden ser campos, variables de memoria o constantes, deben coincidir con los tipos de campos en la lista de campos entre paréntesis. Si no se nombra ningún campo, los valores deben coincidir exactamente con los de la tabla en el orden en que aparecen en ésta. Por ejemplo:

```
TABLA PEDIDOS:
idpedido    Carácter   10
Fecha       Fecha       8
Monto       Numérico   10,2

INSERT INTO PEDIDOS                                          ;
   VALUES (                                                  ;
      TRANSFORM ( CONTROL.ultimoid, "@L ######" ),           ;
      FECHA(),                                               ;
      PEDIDOS.pedidostotal
```

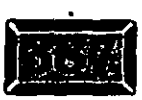
Otro ejemplo:

```
INSERT INTO PEDIDOS (idpedido) VALUES ( "123456" )
```

* FROM ARRAY nombrearreglo

Es útil si utilizó el comando SCATTER TO ARRAY. Si quiere transferir un registro de una tabla a otra que tiene exactamente la misma estructura (por ejemplo, un respaldo o un archivo de transferencia), utilice:

```
SELECT TRANSACC
SCAN
  SCATTER TO XYZ
  INSERT INTO TRANSFERENCIA FROM ARRAY XYZ
ENDSCAN
```



Nótese que los comandos no son simétricos. Emplee SCATTER TO xyz para crear cualquier arreglo llamado XYZ, pero no puede emitir el comando INSERT INTO TRANSFERENCIA FROM xyz; tiene que ser FROM ARRAY XYZ. De manera similar, puede usar GATHER FROM xyz, en lugar de APPEND FROM ARRAY xyz.

* FROM MEMVAR

Esta instrucción busca variables de memoria cuyos nombres coincidan con los campos en la tabla. Cualquiera que falte se deja vacío, a diferencia de FoxPro 2.6, que desaparecía si faltaba alguna variable de memoria.

DELETE

La sintaxis para este comando es como sigue:

```
DELETE FROM (NombredeBasedeDatos!] NombredeTabla
[WHERE CondicionFiltro1 AND | OR CondicionFiltro2 ...]]
```

Delete elimina registros de una tabla. En las tablas de Visual FoxPro, se marcan ' registros para borrarlos, para eliminarse después con el comando PACK. En SQ los datos se manejan en forma distinta, por lo que no hay un comando PACK o RECALL NEXT 1. Es evidente que si está trabajando con DBF, PACK y RECALL se siguen aplicando, pero no son comandos SQL.

* FROM [NombredeBasedeDatos!]NombredeTabla

Especifica la tabla en que se marcan los registros para suprimirlos. Si la tabla en cuestión no es el contenedor de bases de datos que ya está abierto, tiene que dar un prefijo al nombre de la tabla con su nombre de base de datos y un signo de admiración, por ejemplo, DATA2!FACTURAS.

* WHERE CondicionFiltro1 [AND | ORCondicionFiltro2 ...]

Marca sólo los registros que coinciden con la(s) condición(ones) de filtro que se deben suprimir. Puede conectar condiciones de filtro múltiples con AND u OR.

UPDATE

La sintaxis del comando UPDATE es:

```
UPDATE [NombredeBasedeDatos1!]NombredeTabla1
  SET NombreColumna1 = eExpresion1
  [, NombreColumna2 = eExpresion2 ...]
  WHERE CondicionFiltro1 [AND ¦ OR CondicionFiltro2 ...]]
```

* UPDATE [NombredeBasedeDatos1!] NombredeTabla1

Indica a Visual FoxPro qué tabla debe actualizar. Como es usual, si el contenedor de la base de datos al que pertenece la tabla no es el actual, tiene que dar un prefijo al nombre de la tabla de su DBC y un signo de exclamación. Una vez abierto, sigue así hasta que lo cierra con un USE IN alias o un comando CLOSE DATABASES.

* SET NombreColumna1 = eExpresion 1 [, NombreColumna2 = eExpresion 2]

Nombra las columnas que se deben actualizar y sus nuevos valores.

* WHERE CondicionFiltro1 [AND ¦ OR CondicionFiltro2 ...]]

Especifica los registros que se deben actualizar con los nuevos valores. Si omite la cláusula WHERE, todos los renglones en la columna se actualizan con el mismo valor, lo cual quizá no tenía en mente.

Si hace clic en Herramientas, Opciones y selecciona la página Datos remotos notará una selección llamada Método de actualización SQL. El cuadro combinado contiene dos opciones: SQL UPDATE y SQL DELETE + SQL INSERT.

La razón para estas opciones es que, dependiendo de la naturaleza de sus tablas y los tipos de ediciones que realiza, una podría tener un mejor comportamiento que otra. Pruebe ambas y compruébelo usted mismo.

CREATE TABLE/CURSOR

Puede usar el comando CREATE TABLE para crear una tabla desde adentro de su programa. Si quiere montar aplicaciones que elaboran sus propias tablas vacías cuando corren por primera vez, así es cómo se hace. Tal vez sea una buena idea incluir esta clase de opción en caso de que sus usuarios echen a perder algunos archivos. Y durante la prueba es particularmente útil. La sintaxis es:

```
CREATE TABLE ¦ DBF NombredeTabla1 [NAME NombredeTablaLargo] [FREE]
   (NombredeCampo1 TipodeCampo [(nAnchodeCampo [, nPrecision])]
       [NULL ¦ NOT NULL]
```

```
        [CHECK lExpresion1 [ERROR cTextodeMensaje1]]
        [DEFAULT eExpresion1]
        [PRIMARY KEY ¦ UNIQUE]
        [REFERENCES NombredeTabla2 [TAG NombreEtiqueta1]]
        [NOCPTRANS]]
    [, NombredeCampo2 ...])
        [, PRIMARY KEY eExpresion2 TAG NombreEtiqueta2
        [, UNIQUE eExpresion3 TAG NombreEtiqueta3]
        [, FOREIGN KEY eExpresion4 TAG NombreEtiqueta4
           REFERENCES NombredeTabla3 [TAG NombreEtiqueta5]]
        [, CHECK lExpresion2 [ERROR cTextodeMensaje2]]
¦ FROM ARRAY NombredeArreglo
```

* CREATE TABLE ¦ DBF NombredeTabla1

Da el nombre de la tabla que se va a crear. Puede usar TABLE o DBF en forma alternativa.

* NAME NombredeTablaLargo

Si tiene un contenedor de bases de datos abierto, puede proporcionar un nombre largo para una tabla. Puede especificar un nombre de tabla largo sólo cuando una base de datos está abierta porque los nombres están almacenados en contenedores de bases de datos. Los nombres largos pueden contener hasta 128 caracteres y se pueden emplear en lugar de los nombres de archivos cortos en la base de datos.

* FREE

Hace que la tabla sea una tabla FREE (libre), no un miembro de una base de datos abierta. Si no hay una base de datos abierta, no se requiere FREE.

* (NombredeCampo1 TipodeCampo [(nAnchodeCampo [, nPrecision])]

En cada nombre de campo utilice esta instrucción para especificar el nombre del campo; su tipo de campo y su precisión (número de lugares decimales). TipodeCampo es una sola letra que indica el tipo de datos del campo. Algunos tipos de datos de campo requieren nAnchodeCampo o nPrecision, o ambos.

El apéndice B lista los valores para TipodeCampo y especifica si se requieren nAnchodeCampo y nPrecision, los cuales se ignoran para los tipos D, T, Y, L, M, G y P. Los valores predeterminados de nPrecision son cero (sin lugares decimales) si no se incluye nPrecision para los tipos N, F o B.

* NULL

NULL sigue igual, a menos que incluya la cláusula PRIMARY KEY o UNIQUE, en cuyo caso el campo establece los valores predeterminados en NOT NULL.

* CHECK lExpresion1

Especifica una regla de validación para el campo. lExpresion1 puede ser una función definida por el usuario. Sólo se usa en BROWSE y EDIT.

* ERROR cTextodeMensaje1

Especifica el mensaje que Visual FoxPro despliega cuando la regla de campo genera un error. El mensaje se despliega sólo cuando los datos se cambian con una ventana Examinar o Editar.

* DEFAULT eExpresion1

Especifica un valor predeterminado para el campo. El tipo de datos de eExpresion1 tiene que ser el mismo que el tipo de datos del campo. Cuando se agrega un registro nuevo, éste es el valor que se incluye en el campo. Puesto que cero es el valor predeterminado para campos numéricos, los espacios para los campos de caracteres y una fecha nula para los campos de datos, el único uso que he dado a esta función es utilizar DATE( ) para los campos de datos a fin de insertar inmediatamente la fecha de hoy. Es obvio que esto es conveniente para algunas aplicaciones, pero por completo inútil para otras.

* PRIMARY KEY

Crea un índice principal para el campo. La etiqueta de índice principal tiene el mismo nombre que el campo.

* UNIQUE

Crea un índice candidato para el campo. La etiqueta de índice candidato tiene el mismo nombre que el campo. Nótese que los índices candidatos no son los mismos que los índices que se crean con la opción UNIQUE en el comando INDEX. Un índice que se crea con la opción UNIQUE en el comando INDEX permite duplicar claves índice; los índices candidatos no permiten duplicar claves índice. En otras palabras, puede utilizar una tabla con muchos registros que tienen el valor FL en el campo de estado. Si utilizara INDEX UNIQUE en ese campo, sólo aparecería un registro de Florida, pero puede tener muchos registros en el archivo. No obstante, una vez que tiene un índice candidato, no puede capturar valores duplicados de su clave en el archivo.

* REFERENCES NombredeTabla3 [TAG NombreEtiqueta1]

Especifica la tabla padre para la que se establece una relación persistente. Si omite TAG Nombre Etiqueta1, se establece la relación usando la clave del índice principal de la tabla padre. Sin embargo, quizá sea una buena idea asignar el nombre de la etiqueta cada vez.

* NOCPTRANS

Se refiere a soportar los idiomas diferentes del inglés, y con el riesgo de parecer provinciano, si no está creando software que se use en países en que se habla otro idioma además del inglés, ignore esto. En caso de que desee saberlo, impide la traducción de una página de código diferente para los campos de caracteres y memo. Si la tabla se convierte en otra página de código, cada campo que se haya creado como un campo NOCPTRANS no se traduce.

* PRIMARY KEY eExpresion2 TAG NombreEtiqueta2

Especifica un índice principal que se debe crear. eExpresion2 especifica cualquier campo o combinación de campos en la tabla y TAG NombreEtiqueta2 da el nombre. Una tabla sólo puede tener un índice principal.

* UNIQUE eExpresion3 TAG NombreEtiqueta3

Crea un índice candidato. eExpresion3 especifica cualquier campo o combinación de campos en la tabla. Una tabla puede tener múltiples índices candidatos.

* FOREIGN KEY eExpresion4 TAG NombreEtiqueta4

Crea un índice externo (no principal) y establece una relación con la tabla padre, lo cual es muy semejante a SET RELATION, excepto que Visual FoxPro sabe al respecto en el instante en que utiliza el comando OPEN DATABASE Nombre del contenedor de la Base de Datos. Expresion4 especifica la etiqueta clave de índice externo que se crea. Incluya NODUP para crear un índice externo candidato.

Puede crear índices externos múltiples para la tabla, pero las expresiones de índice externo tienen que especificar diferentes campos en la tabla. Una tabla puede tener múltiples índices externos.

* REFERENCES NombredeTabla2 [TAG NombreEtiqueta5]

Ésta es la otra mitad de la expresión FOREIGN KEY. Una vez más, puede omitir TAG NombreEtiqueta5 y dejar el valor predeterminado en cualquier aplicación que esté en funcionamiento, pero tal vez sea mejor ser explícito.

* CHECK eExpresion2 [ERROR cTextodeMensaje2]

Especifica la regla de validación de tablas. ERROR cTextodeMensaje2 especifica el mensaje de error que Visual FoxPro despliega cuando se ejecuta la regla de validación de tablas. El mensaje se despliega sólo cuando se cambian datos en una ventana Examinar o Editar.

* FROM ARRAY NombredeArreglo

Especifica el nombre de un arreglo existente cuyo contenido es el nombre, tipo, precisión y escala para cada campo en la tabla. Puede definir el contenido del arreglo con la función AFIELDS( ).

Utilice esta función al copiar la estructura de una tabla existente, en casos en que COPY STRUCTURE no sea muy apropiado. Por ejemplo, si quiere ejecutar un informe que usa algunos de los campos de un archivo, además de otro que va a crear y poblar con algún valor clave calculado antes de ordenar e imprimir el informe, puede hacer esto:

❶ Crear la estructura deseada en un arreglo.

❷ Agregar un campo clave nuevo.

❸ APPEND FROM de su tabla existente.

❹ REPLACE reemplaza el campo clave con cualquier cosa que sea apropiada.

❺ Ejecutar el informe.

* Observaciones

La tabla nueva se abre en el área de trabajo menor disponible y se puede acceder mediante su alias. Ya que nadie además de usted sabe acerca de la tabla nueva, el hecho de que la tabla nueva se abra exclusivamente, sin importar los valores establecidos actuales de SET EXCLUSIVE, no debe ser un problema.

Si una base de datos está abierta y no incluye la cláusula FREE, la tabla nueva se agrega a la base de datos. Pero si no hay un contenedor de bases de datos abierto cuando crea la nueva tabla, incluyendo las cláusulas NAME, CHECK, DEFAULT, FOREIGN KEY, FREE, PRIMARY KEY, REFERENCES o UNIQUE, se producirá un mensaje de error. Visual FoxPro no tiene dónde almacenarlas si no está abierto un contenedor de bases de datos.

La sintaxis de CREATE TABLE es un cuanto inconsistente dado que usa comas para separar ciertas opciones de CREATE TABLE. Asimismo, las cláusulas NULL, NOT NULL, CHECK, DEFAULT, PRIMARAY KEY y UNIQUE se deben ubicar entre el paréntesis que contiene las definiciones de columna. De modo que con frecuencia se requieren algunos intentos para arreglar esto.

La sintaxis de CREATE CURSOR es similar, pero más sencilla, puesto que los cursores, por definición, no pertenecen a los contenedores de bases de datos:

```
CREATE CURSOR NombredeTabla
  (NombredeCampo1 tipo [(precision [, escala])
       [NULL | NOT NULL]
       [CHECK lExpresion [ERROR cTextodeMensaje]]
       [DEFAULT eExpresion]
       [UNIQUE]
       [NOCPTRANS]]
   [, NombredeCampo2 ...]])
   | FROM ARRAY NombredeArreglo
```

Si está creando un archivo temporal para retener un contenido de BROWSE hasta terminar la transacción y agregar su contenido en una tabla, CREATE CURSOR tal vez sea lo que está buscando.

El Generador de consultas

Si emplea un contenedor de bases de datos o una fuente SQL, puede crear una consulta que describa los datos que quiere usar y guardarlos después. Si no hay ninguna base de datos abierta, aún puede crear y usar la consulta, pero no guardarla.

Cree una consulta

Para crear una consulta, teclee CREATE QUERY nombre y verá el cuadro de diálogo del Generador de consultas. La pantalla básica consiste en una Vista de tabla (con aspecto similar a la ventana Entorno de datos), que ocupa la mitad superior de la ventana y un marco de página con cinco fichas o páginas en la mitad inferior de la ventana.

* La Vista de tabla

Al hacer doble clic en cualquier parte en la mitad superior de la ventana, puede alternar entre las vistas de datos de media pantalla y de pantalla completa.

Vista de tabla muestra relaciones entre tablas seleccionadas. Si Agregar tabla o cuadro de diálogo Vista no están en la pantalla, haga clic en cualquier sitio de la parte superior de la ventana, luego seleccione Agregar tabla del menú contextual. Si está abierto un contenedor de bases de datos, sus tablas aparecerán en la lista de elección; si ningún DBC está abierto, haga clic en Otros para obtener una lista de las tablas libres en el directorio actual.

Puede seleccionar tablas de las que se listan en el contenedor de bases de datos o de las tablas libres. También puede elegir de las tablas que se mencionan en otra vista al hacer clic en el botón de opción Vistas.

Una diferencia importante entre la Vista de tabla y un Entorno de datos de un formulario es que aquí se arrastra y suelta cualquier nombre de campo para establecer cualquier relación que desee, ya sea que los campos sean etiquetas de índice o no. Ya que lo que está haciendo es crear una condición de unión (JOIN). Visual FoxPro no requiere que ya existan etiquetas de índice. Es evidente que si une campos que no tienen ninguna relación entre sí, los resultados no tendrán ningún sentido. Pero Visual FoxPro ciertamente relaja las reglas en esta situación.

* El Generador de vistas

La página principal Criterios de selección le permite crear expresiones que se van a usar en la cláusula HAVING. Hay un botón OR para permitirle crear cualquier clase de expresión lógica que necesite, pero no hay un botón AND porque AND es el conector predeterminado entre expresiones. Note que no puede cambiar las primeras condiciones.

* La página Campos

Haga clic en la ficha Campos y obtendrá la pantalla que se muestra en la figura 11-1. Puede incluir cualquiera de los campos en cualquiera de las tablas seleccionadas para la consulta en la lista de salida del campo.

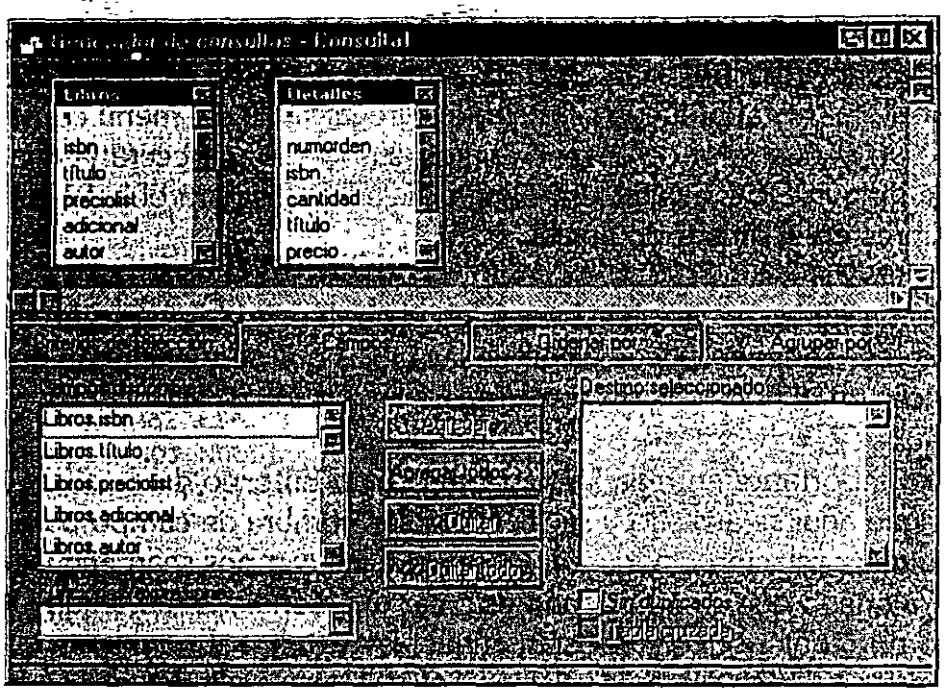
* Ordenar por

La página Ordenar por, que se muestra en la figura 11-2, le permite seleccionar sólo de los campos de salida hasta ese momento, puesto que así es cómo funciona SQL: no puede ordenar por con ORDER BY un campo que está en una tabla de salida pero no se incluye como salida. Es un pequeño requerimiento y no hay ninguna regla que establezca que tiene que imprimir todo en su salida SQL.

* Agrupar por

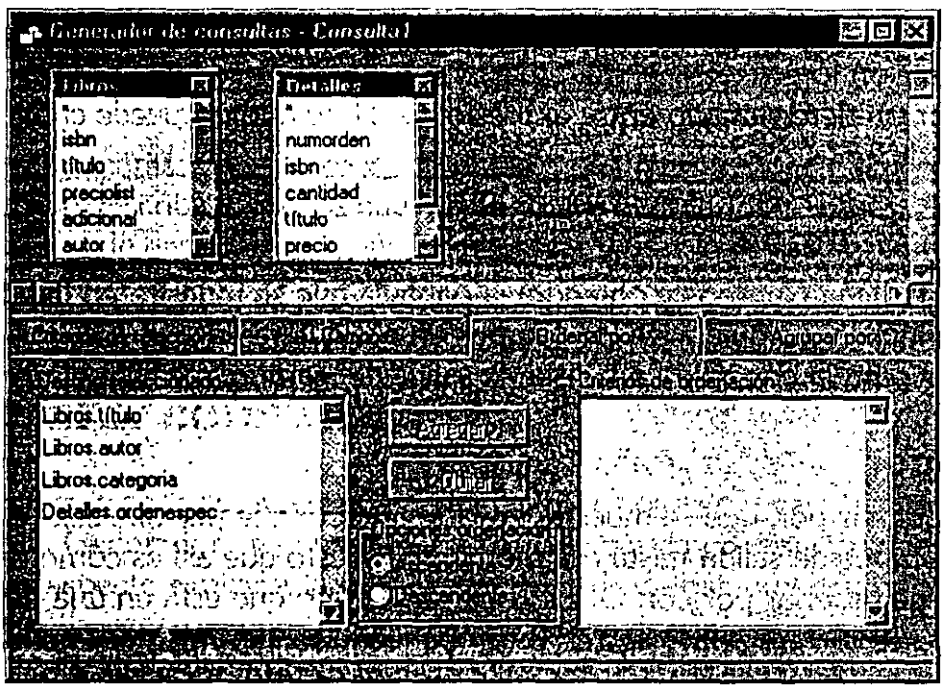
La página Agrupar por es casi idéntica, excepto que ésta se usa en la cláusula GROUP BY y permite emplear las funciones de campo SUM( ), AVG( ), etcétera.

Figura 11-1



Página Campos del Generador de consultas.

Figura 11-2



Página Ordenar por del Generador de consultas.

El menú Consulta

El menú principal de Visual FoxPro tiene un nuevo menú entre Programa y Ventana donde reside el Generador de consultas, denominado Consulta. Cuando lo despliega, obtiene las opciones siguientes:

Agregar tabla Agrega una tabla a la vista.

Eliminar tabla Elimina la tabla seleccionada de la vista; si no se selecciona ninguna, la opción se pierde.

Criterios de selección Abre la página Criterios de selección.

Campos de salida Abre la página Campos.

Ordenar por Abre la página Ordenar por.

Agrupar por Abre la página Agrupar por.

Criterios de actualización Abre la página Criterios de selección.

Vista SQL Muestra cómo se ve el SQL generado con base en las selecciones hechas hasta el momento.

Comentarios Es posible agregar un comentario a una vista y volverla a llamar en cualquier momento. Es útil para la documentación interna. Se pueden construir muchas vistas en un periodo muy corto y además es fácil olvidar qué era para qué.

Ejecutar consulta Ejecuta el SQL creado hasta el momento.

Aun si termina por escribir su propio SQL, ésta es una excelente herramienta de enseñanza.

La barra de herramientas del Generador de vistas

La barra de herramientas del Generador de vistas contiene cinco iconos:

Agregar tabla Agrega otra tabla a la Vista de tabla. Si se agregó un contenedor de bases de datos, puede seleccionar de la lista de tablas que contiene; si no, el botón Otros produce una lista de tablas libres de donde elegir.

Eliminar tabla Elimina una tabla de la Vista de tabla. Esto no tiene efecto alguno si no se resalta la tabla.

Agregar unión Agrega una condición de unión, que se convierte en una cláusula WHERE.

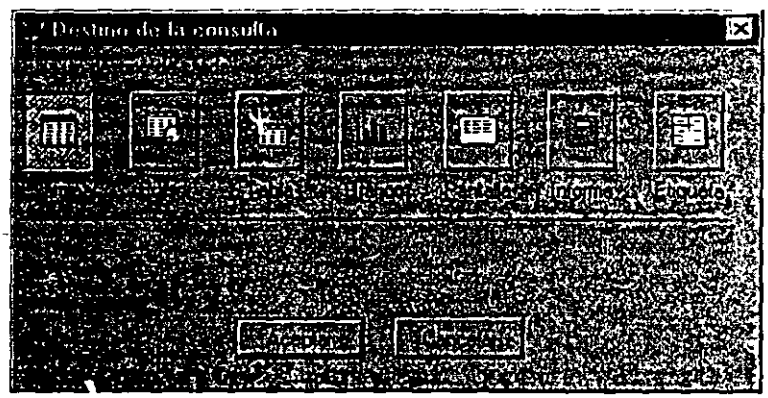
Mostrar SQL Muestra la instrucción SQL producida hasta el momento. Se puede hacer clic sobre éste en cualquier momento y forma un tutor privado de SQL.

Maximizar vista de tabla Hace lo mismo que cuando se hace doble clic en la mitad superior de la ventana; alterna el área de la Vista de tabla entre un despliegue de media ventana y de ventana completa. También puede seleccionar Maximizar el extremo superior (o Minimizar el extremo superior si ya está maximizado).

Destino de la salida

Seleccionar destino de la salida del menú del Generador de consultas o hacer doble clic en la barra de herramientas Consulta hace que aparezca el cuadro de diálogo que se muestra en la figura 11-3.

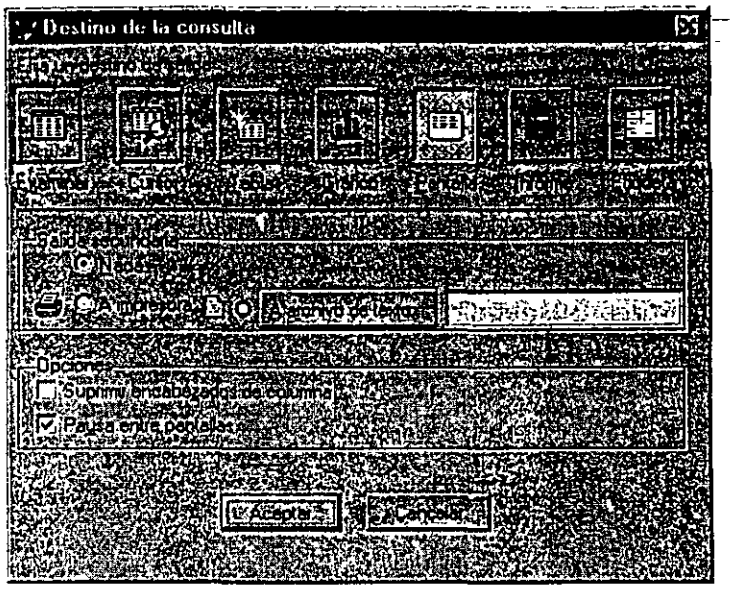
Figura 11-3



Cuadro de diálogo para indicar el destino de la salida.

Examinar y Gráfico no tienen opciones y Tabla y Cursor sólo piden un nombre para el destino. Pero seleccionar Pantalla produce la figura 11-4. Puede dirigir la salida a un archivo de texto o directamente a la impresora, además de lo que se

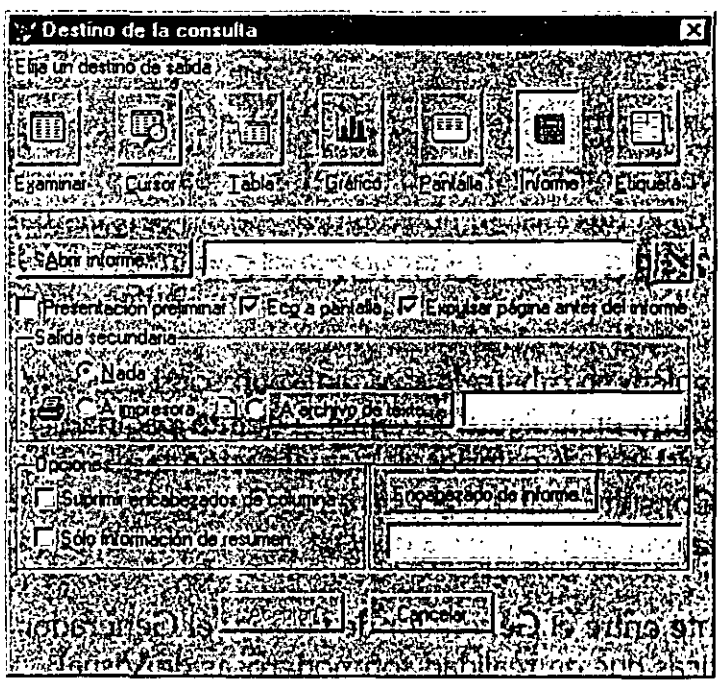
Figura 11-4



Seleccione Pantalla como el destino de salida.

envía a la pantalla. Esto es como SET ALTERNATE TO y SET ALTERNATE ON en FoxPro 2.6. También puede suprimir los encabezados de columna si está usando los datos en alguna otra aplicación que necesita acarrear texto de regreso delimitado como su entrada. (Le ahorra el paso de copiar primero a una tabla, al usar COPY TO archivo SDF.)

Figura 11-5



Cuadro de diálogo para indicar el destino de salida como informe.

Informe del destino de salida hace que aparezca la pantalla de la figura 11-5. Tiene varias opciones adicionales de informe y le permite enviar la salida del informe a un archivo modificando varios valores más establecidos en la operación. El Destino de salida de etiqueta, que no se muestra aquí, es similar.

El Generador de consultas puede enseñarle mucho acerca del SQL. Por otro lado, escribe archivos .QPR que contienen el código que generó y que pueden llamarse directamente en sus programas. Si se solicita una instrucción REPORT FORM, éste se crea justo en el archivo .QPR.

El Generador de vistas

El Generador de vistas es muy parecido al Generador de consultas, pero además de seleccionar tablas de un .DBC o de tablas libres, puede seleccionar datos de una fuente de datos remota. Asimismo, cuando modifica un cursor creado por el Generador de vistas, los cambios se envían de regreso a las tablas fuente.

Cree una vista

La creación de una vista puede ser tan simple como esto:

```
CREATE SQL VIEW misclientes AS SELECT * FROM DATA1!CLIENTE
```

Si tiene una conexión remota, la sintaxis es:

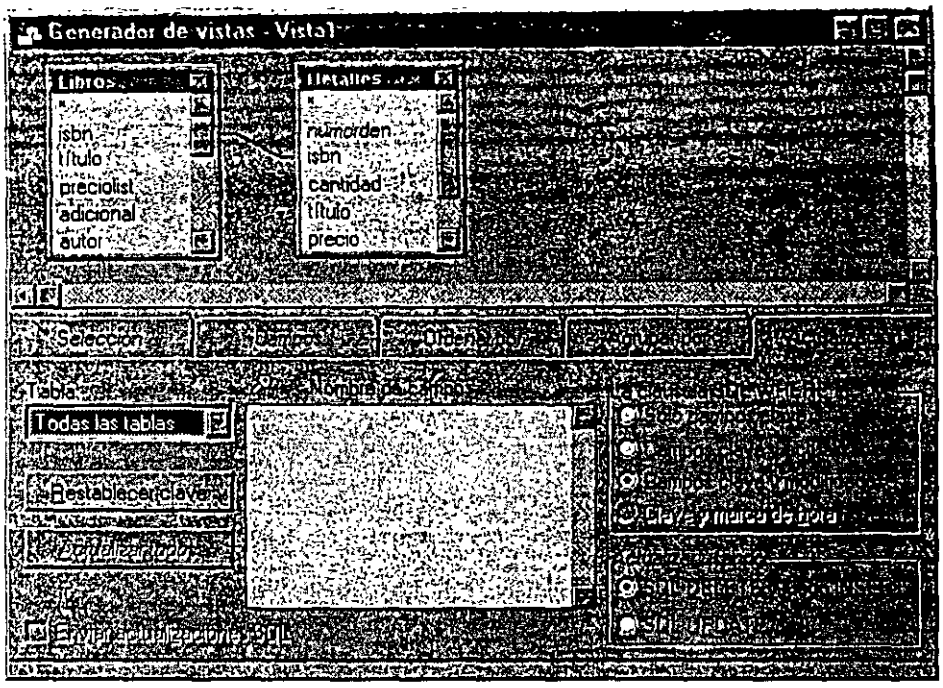
```
CREATE SQL VIEW nombre REMOTE CONNECTION conexion nombre SHARE
```

Diferencias adicionales del Generador de consultas

La página Actualización, que se muestra en la figura 11-6, es la quinta página del marco de página del Generador de vistas. Los criterios de actualización también son una nueva opción en el menú Consulta cuando se está en el Generador de vistas (a diferencia del Generador de consultas).

Esto revela otra diferencia importante entre el Generador de vistas y el Generador de consultas. Una de las características que en realidad son poderosas de Visual

Figura 11-6



Página para actualizar los criterios.

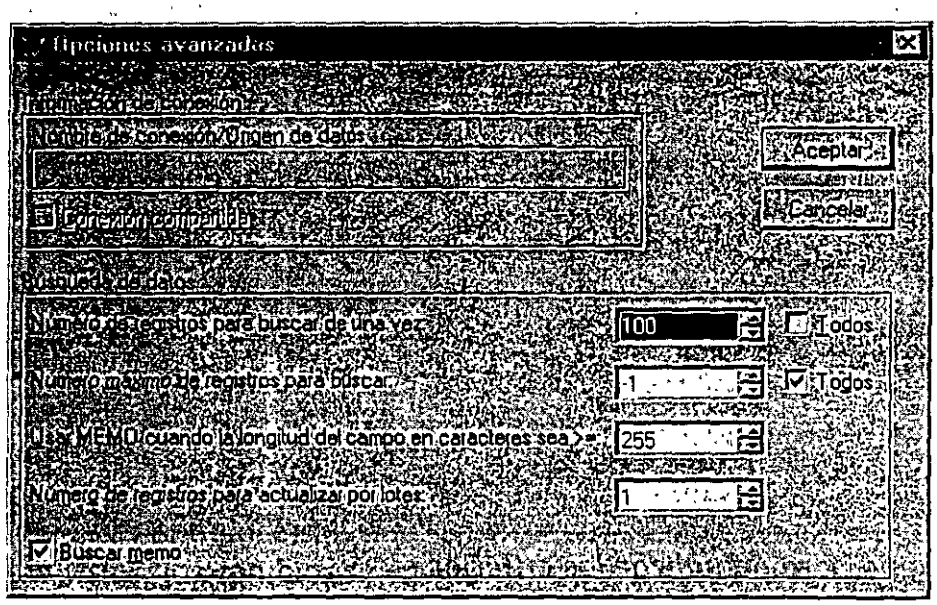
FoxPro es el hecho de que sus vistas, al cambiar, envían los cambios de regreso a las tablas fuente, aun si están en una conexión remota. Puede decidir qué tablas pueden actualizarse, el contenido de la cláusula WHERE de SQL (pruebe las tres opciones y vea el SQL generado) y si se usará para actualizar SQL UPDATE o SQL DELETE y luego INSERT. DELETE e INSERT podrían violar algunas reglas de integridad referencial de las bases de datos, de modo que tal vez no podría usarlas.

Hay dos columnas en la página Actualización, justo a la izquierda de la columna Nombre de campo. Puede designar campos clave y campos que pueden actualizarse al usar la marca de revisión en la columna izquierda o derecha, respectivamente. Debe marcar los campos clave y por lo menos un campo que pueda actualizarse.

Opciones hace que aparezca el cuadro de diálogo que se presenta en la figura 11-7. Las opciones de vista avanzada se usan con Vistas remotas para controlar la búsqueda progresiva, que es útil cuando hay muchos registros que recuperar. Su usuario puede ya sea hacer algo más mientras se buscan los datos en el respaldo, preguntar de vez en cuando si se ha regresado todo, o cancelar el proceso entero.

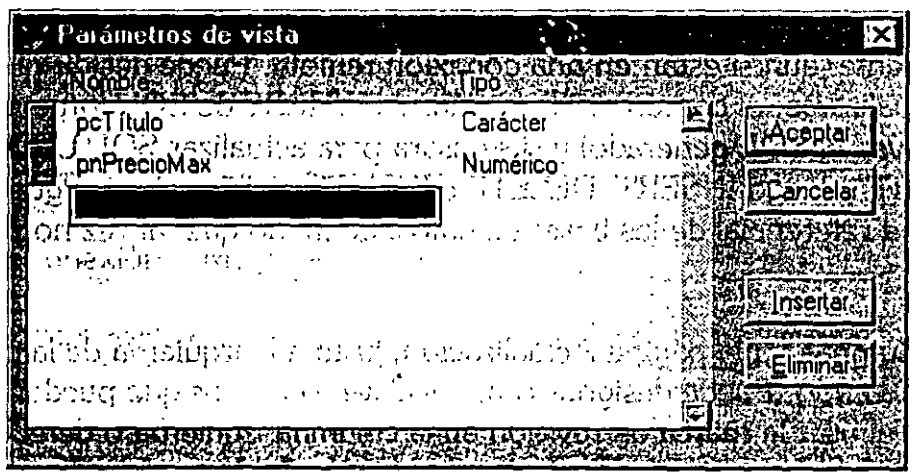
Puede pasar los parámetros a una vista y utilizarlos para seleccionar datos y Parámetros de vista hace que aparezca el cuadro de diálogo que se muestra en la figura 11-8.

Figura 11-7



Cuadro de diálogo Opciones avanzadas.

Figura 11-8



Cuadro de diálogo Parámetros de vista.

Qué falta del Generador de vistas

De la barra de herramientas Generador de vistas y el menú Consultas cuando el Generador de vistas está activo falta el icono Destino de salida. Ésa es la razón la que se usan vistas para modificar los datos originales, no para producir la sali...

Uso de salidas SQL para los informes

Recuerde los informes generados en el último capítulo con encabezados de grupo. Los informes con datos agrupados funcionan de manera apropiada si los datos agrupados se alimentan en el informe en la secuencia ordenada. Si no está convencido, ejecute algunos de sus informes agrupados con una etiqueta de índice errónea incluida. El informe no desaparecerá, pero los resultados tendrán un aspecto muy gracioso. De modo que el orden en que se alimentan los datos en el informe es lo que importa, no la etiqueta de índice. En tanto que los campos clave asciendan en el orden esperado, su informe no tiene idea de si agrega un índice o no. Ahí es donde interviene SQL.

Si su instrucción SQL incluye el parámetro SORT BY, sus datos ya están ordenados. La obtención de datos para un informe particular se puede manejar con facilidad por medio de SQL. En el caso de los informes que usan salida de consultas, los índices ciertamente no existen. Ésta es la razón por la cual usted guarda el SQL para una consulta particular. Para ejecutar el informe, debe ejecutar primero la consulta SQL para recrear el cursor que utilice.

Antes mencioné el producto Foxfire!, el cual en realidad crea el SQL y la forma de informe desde el inicio conforme genera el informe. .QPR, .FRX y .FRT se guardan como un requerimiento de informe. La próxima vez que ejecute el informe, Foxfire! primero ejecutará .QPR, luego REPORT FORM.

El Generador de conexiones

Si su fuente de datos es una fuente de datos ODBC, puede describirla como una conexión. Para agregar una conexión en su aplicación, teclee:

```
CREATE CONNECTION XYZ
```

Se debe abrir un contenedor de bases de datos para mantener sus especificaciones. Elija el nombre de la fuente ODBC y haga clic en Aceptar. Las tablas en la fuente de datos ODBC ahora están disponibles para su aplicación. Una vez que se ha creado una conexión, puede usarla como usaría cualquier otra fuente de datos.

Funciones relacionadas

Existen alrededor de 16 funciones que pueden emplearse con vistas SQL. Los argumentos se pasan entre paréntesis, mientras que el valor de regreso puede ya sea examinarse directamente o guardarse en una variable de memoria, por ejemplo:

```
IF funcion ( parametros ) > 0)
       WAIT WINDOW [ Registros Refrescados ] TIME 1
ENDIF
```

o bien:

```
num = REFRESH( parámetros )
```

* REFRESH( )

Regresa el número de registros refrescados. Los argumentos son:

nRegistros Especifica el número de registros que deben refrescarse. Si nRegistros es 0 u omite nRegistros, sólo se refresca el registro actual.

nDesplazamientoRegistro Especifica el número de registros desde el registro actual en donde comienza el refrescamiento. Por ejemplo, si el registro actual es el registro 10 y nDesplazamientoRegistro es -4, el refrescamiento del registro inicia con el registro 6. Si nDesplazamientoRegistro es 0 u omite nDesplazamientoRegistro, sólo se refresca el registro actual.

cAliasTabla Especifica el alias de la vista remota SQL en que se refrescan los registros.

nAreaTrabajo Especifica el área de trabajo de la tabla o el cursor en que se refrescan los registros. Si usted omite nAreaTrabajo y cAliasTabla, los registros se refrescan en la vista remota SQL en el área de trabajo seleccionada actualmente.

* CURSORSETPROP( )

Especifica los valores establecidos de propiedad para una tabla o cursor de Visual FoxPro. Use CURSORGETPROP( ) para regresar los valores actuales establecidos de propiedad para una tabla o cursor de Visual FoxPro o un cursor creado para una tabla. Los valores establecidos de propiedad se especifican para la tabla o el cursor abierto en el área de trabajo actual seleccionada si CURSORSETPROP( ) se emit sin los argumentos opcionales de cAliasTabla o nAreaTrabajo. Los argumentos son:

opiedad Especifica la propiedad de tabla o cursor que debe establecerse. Note Buffering es la única propiedad que puede especificar para una tabla de Visual FoxPro.

eExpresion Especifica el valor para la propiedad que se especifica con cPropiedad. Si se omite eExpresion, la propiedad se establece en su valor por omisión.

cAliasTabla Especifica el alias de la tabla o cursor para el cual se establece la propiedad.

nAreaTrabajo Especifica el área de trabajo de la tabla o cursor para el cual se establece la propiedad.

Por ejemplo, para especificar un buffering de filas optimista para el área de trabajo actual, use esto en su código de evento LOAD del formulario:

```
SELECT CLIENTE
=CursorSetProp("Buffering",3)
```

En la tabla 11-1 se listan las propiedades que puede especificar para cPropiedad y una descripción de los valores que Expresion puede asumir. Nótese que 'JRSORSETPROP( ) regresa a verdadero (.T.) si Visual FoxPro establece con ..ito la propiedad que usted especifica.

Propiedades que usted puede especificar para cPropiedad y una descripción de los valores que la expresión puede asumir Tabla 11-1

Propiedad	Valores de la expresión
Buffering	1= Desactiva el buffering de filas y tablas. El bloqueo de registros y la escritura de datos son idénticos a las versiones anteriores de Visual FoxPro. Hacer esto probablemente no es una buena idea excepto para archivos de sólo lectura. 2= Activa el buffering de filas pesimista. Esto es el caso de las "reservaciónes en líneas aéreas" donde usted bloquea el registro mientras decide qué asiento escoger para asegurarse que nadie más escoja un asiento mientras usted observa lo que está disponible. 3= Activa el buffering de filas optimista. Ésta es por mucho la opción más útil. El registro se bloquea en el instante necesario para escribir sus cambios y adiciones en el archivo y no se requiere de esfuerzo para determinar si algo ha cambiado. 4= Activa el buffering de tabla pesimista. Éste es quizá el peor de todos los mundos posibles.

Tabla 11-1 **Continuación**

Propiedad	Valores de la expresión
	5= Activa el buffering de tabla optimista. Debe estar activado SET MULTILOCKS para todos los modos Buffering excepto 1 (desactivado). El valor predeterminado es 1.
FetchMemo	.T. = Los archivos memo son desplegados con los resultados de la vista. Esto puede ser muy lento para vistas remotas. A menos que esté seguro de que desea establecerlo así, establezca esta propiedad en .F.
	.F. = Los archivos memo no son desplegados con los resultados de la vista.
FetchSize	El número de filas es desplegado progresivamente desde el grupo de tablas remotas. El valor predeterminado es 100 filas. Establecer FetchSize en -1 recupera el juego completo de resultados (limitado por el parámetro MaxRecords). Establezca esta propiedad con el número de filas dentro de su objeto cuadrícula si éste es el mecanismo de despliegue.
KeyFieldList	Lista de campos primario delimitado por comas para el cursor. Si utiliza esta opción tiene que listar los nombres de campos; no tiene ningún valor predeterminado.
MaxRecords	El número máximo de filas que se despliega cuando se regresan grupos de resultados. El valor predeterminado es -1 (todas las filas son regresadas). 0 significa que la vista es ejecutada pero no son despegados los resultados. Haga esto si quiere forzar a su programa para que trabaje, por ejemplo, 20 registros por vez en una iteración DO WHILE. Programe el comando para "cebar el cohete", luego vuelva a establecerlo en 20.
SendUpdates	.T. =Especifica que una consulta SQL actualizada sea enviada para actualizar tablas remotas. Cuando usted consulta un BROWSE y luego le permite a los usuarios cambiar los registros en la visualización, los cambios son normalmente rechazados. Si no quiere que los cambios sean rechazados utilice .F.
	.F. =Especifica que una consulta SQL actualizada no se envíe para actualizar tablas remotas.
Tables	Una lista de nombres de tablas remotas delimitada por comas. Si utiliza esta opción será el responsable de introducir los nombres.
UpdatableFieldList	Una lista de nombres de campos remotos y los nombres de campos locales, delimitada por comas que se asigna al cursor. Utilice esta opción para especificar los nombres de los valores de Visual Foxl para los campos en el cursor que tienen nombres de campos Visual FoxPro inválidos.

Propiedad	Valores de la expresión
UpdateType	1= Todos lo registros viejos son eliminados, después son agregados los registros de la vista. 2= En lugar de eliminar e insertar, Visual FoxPro utiliza lo que sabe acerca de los datos para agregar, eliminar y actualizar registros modificados. Debido a que esto es más eficiente, es el valor por omisión. Al afinar su aplicación encontrará que UpdateType = 1 es más rápido o menos propenso a causar conflictos con otros usuarios.
UseMemoSize	Si el resultado de una consulta resulta ser más grande que 225 bytes de longitud, Visual FoxPro crea un campo memo para la salida. Sin embargo, si el resultado es menor a 255 bytes, se convierte otra vez en un campo de tipo carácter. Debido a que es raro que los campos de tipo carácter sean tan largos, puede decirle a Visual FoxPro que si el resultado no es mayor que, digamos, 20 bytes, utilice campos memo para almacenar el resultado.
WhereType	Se usa para decirle a Visual FoxPro qué campos actualizar. Si introduce un código de 1, Visual FoxPro espera encontrar sólo los campos primarios especificados con la propiedad KeyFieldList en la cláusula WHERE. Si es un 3 entonces la cláusula WHERE consiste en los campos KeyFieldList así como en cualquier otro campo actualizado.

Los usos más comunes de esta función son al usar buffering para establecer bloqueos de filas o tablas en el código LOAD para una pantalla, al usar WhereType para determinar cómo se efectúan las actualizaciones en tablas remotas y FetchSize para pasar sobre la propiedad FetchSize SQLSETPROP( ) para un cursor, ya que por lo general se hereda del manejo de conexión del cursor.

* CURSORGETPROP( )

Regresa los valores establecidos actuales de las propiedades listadas sobre éste.

* SQLCANCEL(nControladorConexion)

Cancela una consulta en curso que se está buscando progresivamente. Regresará un 1 si se cancela con éxito y un -1 o -2 si hay un error de nivel de conexión o entorno, respectivamente.

* SQLCOLUMNS(nControladorConexion,cNombreTabla, [FOXPRO : NATIVE], NombreCursor)

Regresa una tabla de las columnas en la conexión actual cuyo número de manejo : pasa como el primer parámetro. Nótese que puede obtener los resultados de

regreso ya sea en el lenguaje estándar de FoxPro o en terminología SQL. Asegúrese de especificar NombreCursor para indicar a Visual FoxPro dónde colocar la salida.

* SQLCOMMIT(nControladorConexion)

Hace una actualización permanente SQL; el opuesto es SQLROLLBACK. Regresará una .T. si tiene éxito.

* SQLCONNECT(NombreOrigenDatos, IDUsuario, Contraseña, NombreConexion)

Regresa un número de manejo numérico que se puede usar después con estas funciones SQL. Los parámetros son:

NombreOrigenDatos Especifica el nombre de una fuente de datos como se define en su archivo ODBC.INI.

IDUsuario Especifica un identificador de usuario para explotar la fuente de datos.

NombreConexion Especifica una conexión nombrada que se creó con CREATE CONNECTION.

* SQLDISCONNECT(nControladorConexion)

Desconecta una fuente SQL.

* SQLEXEC(nControladorConexion, Comando, NombreConexion)

Envía un comando de paso SQL a una fuente de datos. La función regresa el número de conjuntos de resultados que deben enviarse. Si regresa más de uno, cada conjunto de resultados se envía a un nombre de archivo con base en el nombre de la conexión, con un sufijo aumentado. Si se opera en modo asíncrono, la función regresa 0 hasta que se realiza y luego regresa el número de conjuntos de resultados regresados.

* SQLGETPROP( )

Regresa el valor establecido actual de una de las propiedades de un cursor. Véase SQLSETPROP para códigos.

* SQLSETPROP (nControladorConexion | NombreOrigenDatos | NombreTabla, cConfiguracion[,Expresion])

Especifica los valores establecidos para una conexión activa, fuente de datos o tabla agregada. Nótese que algunas de las propiedades se pueden leer o escribir, en tanto que otras son de sólo lectura.

Especifica el valor para el parámetro establecido que designa con cConfiguracion. Si omite eExpresion, el valor predeterminado se reestablece para el parámetro establecido. SQLSETPROP( ) regresa un 1 si tiene éxito y regresa un -1 si ocurre un error en el nivel de conexión o un -2 si ocurre un error en el nivel de entorno. Los parámetros son:

nControladorConexion Especifica el manejo de conexión para la fuente de datos regresada por SQLCONNECT( ).

NombreOrigenDatos Especifica el nombre de una fuente de datos como se define en su archivo ODBC.INI.

NombreTabla Especifica el nombre de una tabla agregada abierta.

Configuración La tabla 11-2 lista los valores para Configuración.

* SQLSTRINGCONNECT(CadenaConexion)

Si la cadena de conexión está presente, se conecta a la plataforma ODBC nombrada. Si se omite, aparece el cuadro de diálogo Fuentes de datos SQL.

Valores de las variables para los parámetros Tabla 11-2

Parámetro	Descripción
Asynchronous	Especifica si los juegos de resultados son regresados sincrónicamente (0 es el valor predeterminado) o asincrónamente (1). Lectura-escritura.
BatchMode	Especifica si SQLEXEC( ) regresa de una sola vez todos los juegos de resultados (1 es valor predeterminado) o individualmente con SQLMORERESULTS( ) (0). Lectura-escritura.
ConnectBusy	Contiene verdadero (.T.) si una conexión compartida está ocupada; si no es así, contiene falso (.F.). Sólo lectura.
ConnectName	El nombre de la conexión. Sólo lectura.

Tabla 11-2 Continuación

Parámetro	Descripción
ConnectString	La cadena de inicio de la conexión. Lectura-escritura.
ConnectTimeOut	Especifica el tiempo de espera (en segundos) antes de regresar un mensaje de error de una conexión fuera de tiempo. Sí especifica 0 (el valor predeterminado), la espera es indefinida y el mensaje de error fuera de tiempo nunca es regresado. ConnectTimeOut puede ser de 0 a 600. Lectura-escritura.
DataSource	El nombre de la fuente de datos es definido en el archivo ODBC.INI. Lectura-escritura.
DispLogin	Contiene un valor numérico que determina cuando es desplegado un cuadro de diálogo Conexión ODBC. DispLogin asume los siguientes valores: 1 o DB_COMPLETE (de FOXPRO.H) es el valor predeterminado. Si se especifica 1 o DB_COMPLETE, Visual FoxPro despliega el cuadro de diálogo Conexión ODBC sólo si falta alguna información requerida. Si se especifica 2 o DB_PROMPT (de FOXPRO.H), el cuadro de diálogo ODBC siempre está desplegado, permitiéndole cambiar los parámetros antes de conectarse. Si es especificado 3 o DB_NOPROMPT (de FOXPRO.H), el cuadro de diálogo Conexión ODBC no se despliega y Visual FoxPro genera un error si la información requerida para la conexión no está disponible. Lectura-escritura.
DispWarnings	Especifica si los mensajes de error se despliegan (1) o no (0). El valor predeterminado es 0 (no son desplegados los mensajes de error). Lectura-escritura.
IdleTimeout	El intervalo de tiempo inactivo en segundos que causa un fuera de tiempo. Las conexiones activas son desactivadas después de un intervalo de tiempo especificado. El valor predeterminado es 0 (esperar indefinidamente). Lectura-escritura.
ODBChdbc	El manejo interno de la conexión ODBC que puede ser usado por bibliotecas de archivos externas (.FLL) para llamar a ODBC. Sólo lectura.
ODBChstmt	El manejo interno de las instrucciones ODBC que pueden ser usadas por bibliotecas de archivos externas (.FLL) para llamar a ODBC. Sólo lectura.
Password	La contraseña para la conexión. Sólo lectura.
QueryTimeOut	Especifica el tiempo de espera (en segundos) antes de regresar un error de fuera de tiempo general. Si usted especifica 0, la espera indefinida y nunca se regresa un mensaje de error fuera de tiempo. QueryTimeOut puede ser de 0 a 600. El valor predeterminado es 15. Lectura-escritura.

Parámetro	Descripción
Transactions	Contiene un valor numérico que determina cómo la conexión maneja las transacciones en la tabla remota. Transactions asume los siguientes valores: 1 o DB_TRANSAUTO (de FOXPRO.H) es el valor predeterminado. El proceso de transacciones para la tabla remota es manejado automáticamente. 2 o DB_TRANSMANUAL (de FOXPRO.H) significa que el procesamiento de transacciones es manejado manualmente a través de SQLCOMMIT( ) y SQLROLLBACK( ). Lectura-escritura.
UserId	La identificación de usuario. Lectura-escritura.
WaitTime	La cantidad de tiempo en milisegundos que transcurre antes de que Visual FoxPro revise si la instrucción SQL ha terminado su ejecución. El valor predeterminado es 100 milisegundos. Lectura-escritura.

* SQLTABLES(nControladorConexion, TipoTablas, NombreCursor)

Regresa una tabla de tablas conectadas que coinciden con la especificación de TipoTablas, por ejemplo, SYSTEM o VIEW. Se puede ejecutar en forma síncrona.

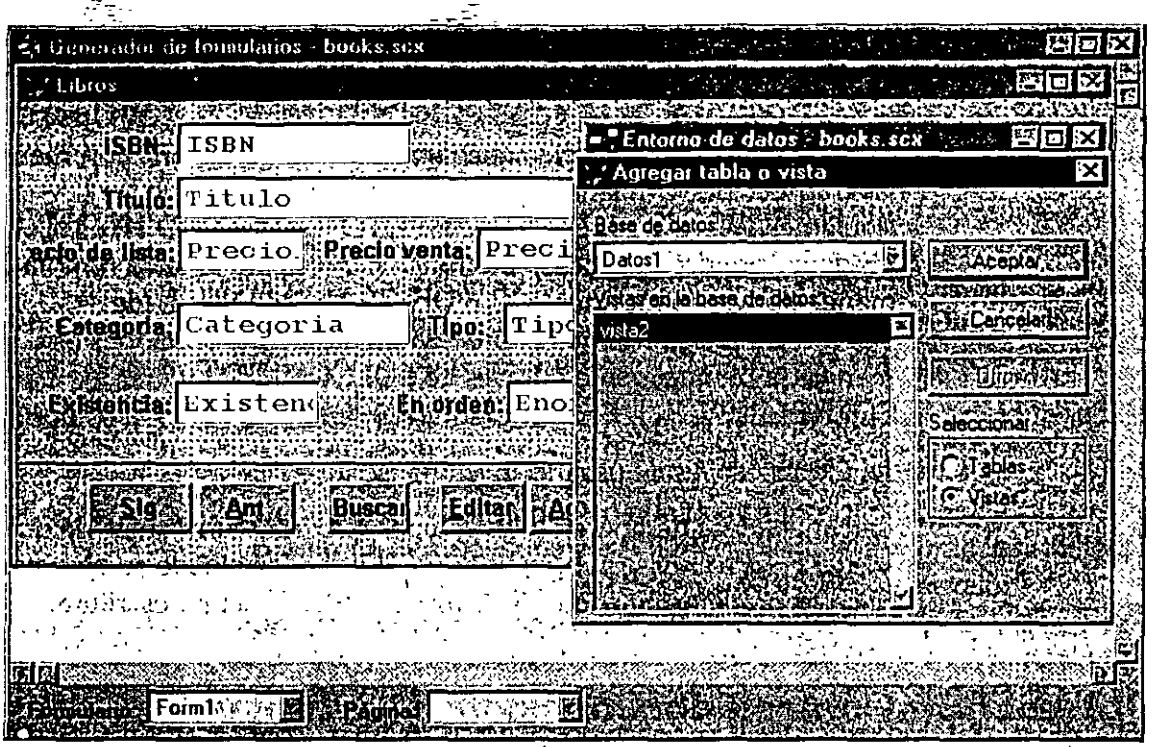
Estas 16 funciones proporcionan soporte completo para el ambiente cliente-servidor.

Uso de las vistas de datos en sus formularios en pantalla

Cuando está en el Generador de formularios, puede agregar una vista a su Entorno de datos, como se muestra en la figura 11-9. Así, Visual FoxPro sabe cómo conectar los datos que edita en su formulario de regreso a la fuente de datos. ¡Eso es todo al respecto!

Hay un uso importante de SQL que no tiene relación alguna con un servidor SQL. Siempre que utiliza un cuadro de lista o un cuadro combinado, tal vez desee usar una instrucción SQL como su RowSource. Esto es especialmente cierto si selecciona un subconjunto de registros con base en condiciones variables, a diferencia de todos los registros en una tabla o una lista fija de elementos que podrían cargarse una vez en un arreglo.

Figura 11-9



Adjunte una vista al Entorno de datos de un formulario.

En la figura 11-10, se capturó una instrucción SQL para poblar un cuadro combinado de códigos y nombres de clientes. ColumnCount en la página Distribución se estableció en 2, de modo que tanto el Código del cliente como el Nombre del cliente aparecerán en la lista desplegable. La lista para elegir que resulta se muestra en la figura 11-11. BoundColumn se establece en 1, de modo que el valor en la columna 1 del elemento seleccionado de la lista se capturará en el campo ControlSource en la tabla activa.

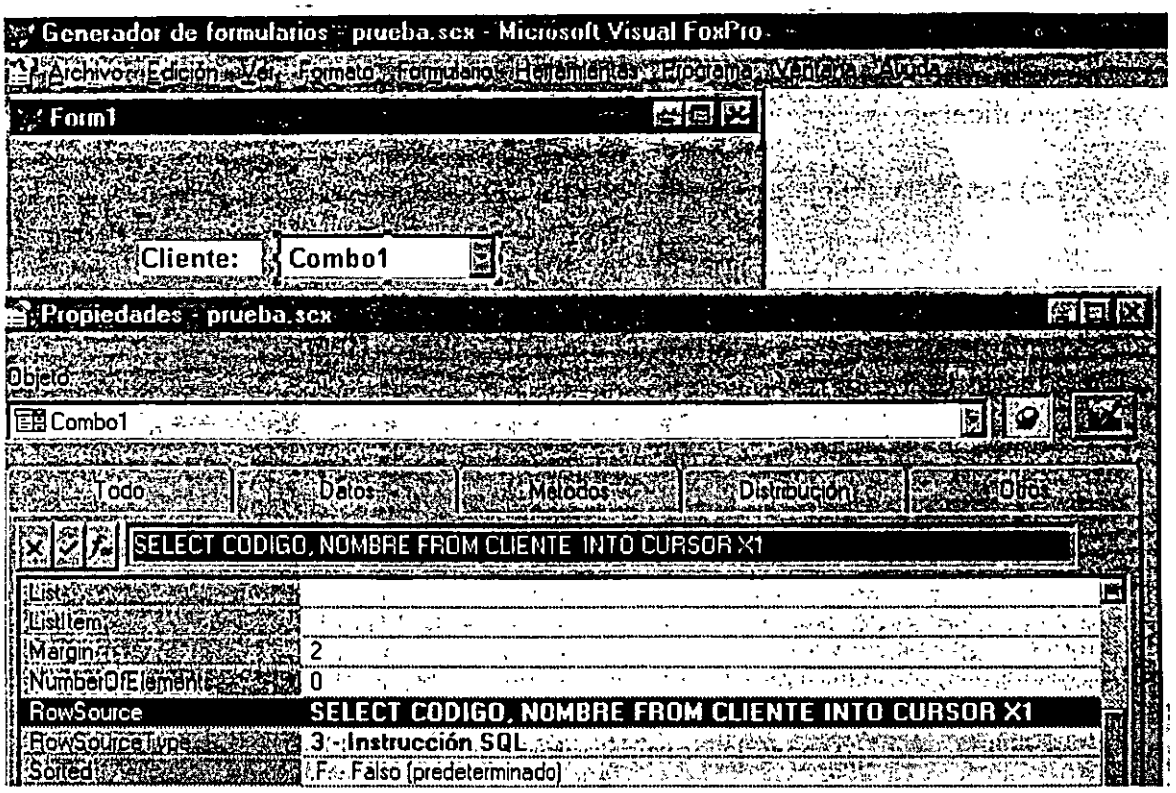
SQL es usualmente el RowSourceType cuando se emplea un parámetro para limitar la lista de valores seleccionados. El ejemplo no tiene un parámetro porque requiere cierta explicación. Cuando se comenzó a trabajar con VFP, se supuso que la instrucción SQL podría leerse como esto:

```
SELECT IDCliente.Nombre FROM CLIENTE WHERE Estado = THISFORM.Estado.
```

donde THISFORM.Estado era una propiedad de formulario que contenía el estado para el cual se debía seleccionar al cliente. Imagine la sorpresa cuando se obtuvo el mensaje:

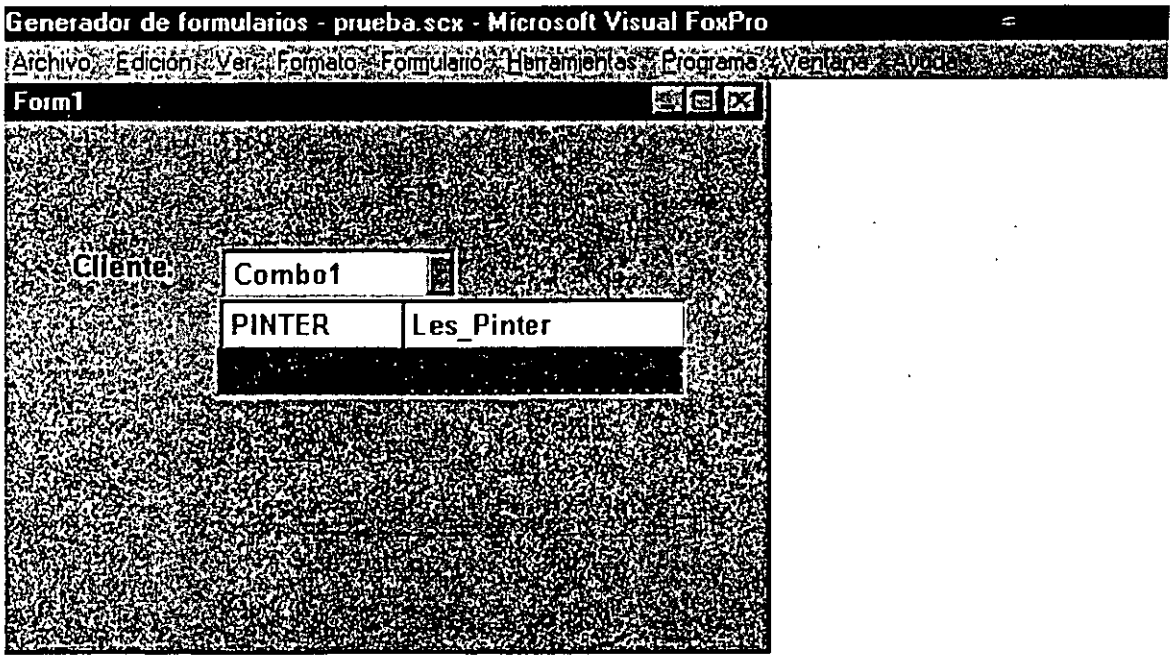
```
Error cargando archivo - RowSource - registro numero 12, THISFORM sol
puede utilizarse en un metodo
```

Figura 11-10



Una instrucción SQL para poblar un cuadro combinado.

Figura 11-11



Lista de selección resultante.

Puede entender su punto de vista, pero no hay ningún sitio para almacenar los criterios de selección actuales en un formulario. Aun así, no funciona y no funcionará, de modo que necesita declarar variables para este propósito. No puede especificar, por ejemplo:

```
SELECT IDCliente.Nombre FROM CLIENTE WHERE Estado = Pedidos.Estado
```

porque FoxPro hará una combinación (JOIN) y dará a todas las combinaciones ese ajuste. La solución desafortunada es declarar una variable PUBLIC para usarla como el parámetro. No es mucho, pero hará el trabajo. Recuerde liberar la variable cuando ya no la necesite.

Conclusión

SQL es una de las características más importantes de Visual FoxPro y la clave para las relaciones cliente-servidor. Los buenos empleados de Microsoft han hecho un esfuerzo para hacer un ejercicio simple el apuntar y hacer clic. Si no tiene un servidor, consiga uno. ¡Amará la computación cliente-servidor!

12

Controles de Visual FoxPro

CAPITULO 12

EN el entorno controlado por eventos y orientado a objetos de Visual FoxPro, el corazón de cualquier interfaz es el conjunto de controles (u objetos) que utiliza. Los controles estándar incluidos en Visual FoxPro constituyen un juego robusto, que permite diseños de formularios muy efectivos.

Sin embargo, el uso hábil del juego de herramientas completo requiere que usted entienda las capacidades y limitaciones de cada control. Algunos controles, como el botón de opción, pueden ser utilizados como un dispositivo de ingreso de datos o como un control. Y, ¿cuándo es mejor una casilla de verificación que un botón de comando? No existen reglas, pero hay pautas.

También es importante comprender cómo se personalizan los controles. Cada control tiene propiedades que determinan cómo se ve y cómo actúa. Saber cuándo especificar un comportamiento distinto al predeterminado puede transformar un control en algo radicalmente diferente.

Crear clases personalizadas es una manera de personalizar más allá los controles. Usted puede utilizar controles estándar como clases base, solos o en combinación, y añadir comportamientos para crear tipos de control completamente nuevos.

Aunque existen alrededor de 50 eventos, 48 métodos y 225 propiedades en el modelo evento/objeto de Visual FoxPro, en este capítulo solamente se tratarán los eventos, métodos y propiedades que usted utilizará con mayor frecuencia. Por supuesto, hay casos en los que cada uno de los eventos y métodos de Visual FoxPro vendrán bien, por eso están ahí. Pero si usted domina sólo esta lista "A", probablemente podrá manejar el 95% de todas las herramientas que llegará a necesitar.

Eventos

Las formularios y objetos de Visual FoxPro tienen alrededor de 50 eventos que ejecutan su código de evento, el código que usted escribe, cuando sus desencadenantes integrados se disparan. Sus nombres describen qué los dispara: DragOver, Click y así sucesivamente. Éstos son los eventos de la lista "A":

- Click
- Destroy
- DragDrop

- DragOver
- Error
- GotFocus
- Init
- LostFocus
- MouseDown
- MouseMove
- MouseUp
- ProgramaticChange
- RightClick
- Valid
- When
- Load
- Activate
- Unload
- InterActiveChange

Éstos se discutirán con detalle en las siguientes secciones. Los ejemplos son característicos de los usos que se ven en el propio trabajo de desarrollo.

Click

El evento Click ocurre cuando:

- Se hace clic en el objeto con el botón principal (generalmente el izquierdo) del ratón.
- El objeto se activa de manera interactiva, por ejemplo, cuando usted presiona la tecla de acceso asociada al control.
- Se cambia el valor del control, ya sea de manera interactiva o como una consecuencia de código.
- Se presiona la barra espaciadora cuando un botón o casilla de verificación están enfocados.

Este evento no se viene abajo con objetos del contenedor, por lo que un formulario obtendrá un evento Click sólo si usted hace clic en un área en blanco.

Ésta es la principal acción de evento en la mayoría de los controles, por lo general el control del botón de comando. Sea cual sea el código que usted desea disparar cuando se presiona un botón de comando, aquí es donde debe ponerlo. La siguiente línea de código en el evento Click de un botón de comando cerrará y liberará el formulario actual:

```
THISFORM.Release
```

También puede adjuntar código a los eventos Click de otros controles. Esto correrá el código y también efectuará la acción normalmente ejecutada por el control. Por ejemplo, una casilla de verificación se seguirá encendiendo o apagando sola, pero también correrá su código del evento Click.

Todos los objetos visibles tienen un evento Click, incluyendo imágenes, etiquetas y marcos de página, incluso líneas y formularios. Todos los objetos que responden al evento Click responden también a los eventos MouseDown, MouseMove y MouseUp. Todos, excepto los objetos Formulario y Barra de herramientas, responden también al evento RightClick. Usted puede utilizar estos otros eventos, descritos más adelante en esta sección, si necesita información más especifica acerca del ratón.

Uno de los usos más comunes de los botones, y especialmente del evento Click, es la navegación. A estos botones se les conoce como botones de navegación o botones VCR, que permiten al usuario avanzar o retroceder a través de los registros en una tabla. Aquí está el código para estos cuatro botones estándar:

```
* Evento Click de comando primero
GO TOP
THISFORM.Refresh

*  Evento Click de comando previo
IF NOT BOF()
     SKIP -1
     IF BOF()
          GO TOP
     ENDIF
ENDIF
THISFORM.Refresh

* Evento Click de comando siguiente
IF NOT EOF()
     SKIP
     IF EOF()
          GO BOTTOM
```

```
    ENDIF
ENDIF
THISFORM.Refresh

* Evento Click de comando último
GO BOTTOM
THISFORM.Refresh
```

Por brevedad, se han puesto los cuatro botones en un solo listado, pero cada sección debería ir en su propia ventana de código. También recuerde que éste es un ejemplo elemental. Sólo maneja errores obvios; no bloquea o desbloquea registros, no maneja formularios múltiples que utilizan la misma tabla, e ignora muchos aspectos específicos de las aplicaciones individuales. Pero resulta ser alrededor del 25% del código de una simple pantalla de visualización, por lo que vale la pena incluirlo.

Destroy

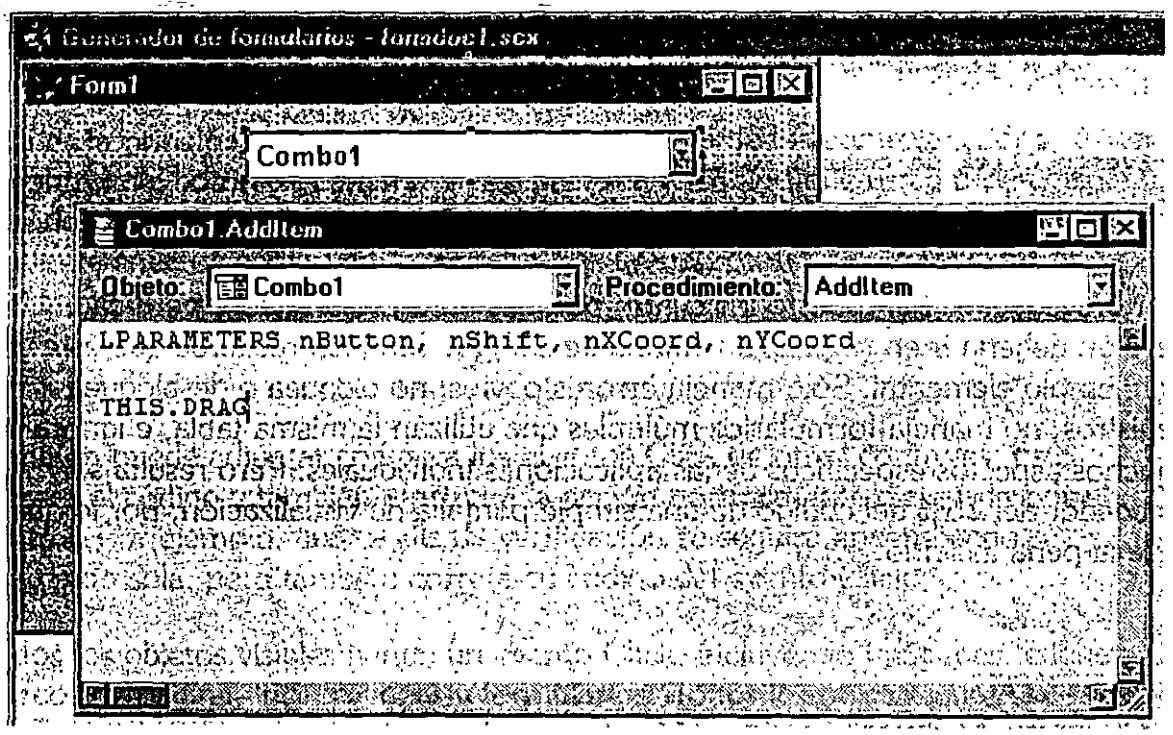
El evento Destroy ocurre cuando se libera de la memoria una instancia de una clase. Este evento sucede antes de que ningún objeto contenido en él sea destruido, por lo que el objeto puede referirse a objetos contenidos en él en su código Destroy. Se diferencia de los eventos Deactivate o LostFocus en que éstos indican sólo una desactivación temporal del control, mientras que el evento Destroy indica que un control se destruye de manera permanente. El único evento que sigue al evento Destroy es el evento UnLoad, que es el último evento de la vida de un objeto. Un formulario que corra con el comando DO FORM no crea una instancia de una clase, por lo que nunca recibe un evento Destroy.

DragDrop

El evento DragDrop sucede cuando se suelta un objeto en otro. Usted puede habilitar, arrastrar y soltar al invocar el método Drag en el evento MouseMove de un objeto (ver figura 12-1), y completar luego el movimiento al invocar el método Drag con un parámetro de 2 en el evento MouseUp. Mientras el control es arrastrado, usted puede cambiar el cursor del ratón utilizando la propiedad DragIcon. Por ejemplo, para arrastrar el valor de un cuadro combinado, añada la siguiente línea al evento MouseDown:

```
THIS.Drag
```

Figura 12-1



Contenido del cuadro combinado al ser arrastrado sobre un formulario.

y la siguiente línea al evento MouseUp:

```
THIS.Drag(2)
```

Cuando el evento DragDrop tiene lugar, pasa tres parámetros al código del evento: el objeto que fue soltado y las posiciones X y Y sobre las que se soltó. Éstas son oSource, nXCoord, y nYCoord, respectivamente.

La variable oSource es una referencia a un objeto. Lo que significa que usted puede utilizar la variable oSource como el nombre del objeto, teniendo acceso a métodos y propiedades directamente de la variable. oSource y Name, por ejemplo, le diría cuál es el nombre del objeto. Por lo tanto, usted tiene control absoluto sobre lo que desee hacer con el objeto una vez que haya sido soltado.

Para hacer que la propiedad Caption de cualquier objeto cambie al valor arrastrado sobre él, inserte el siguiente código en su evento DragDrop:

```
--DO CASE
CASE oSource.Type = "ComboBox"
     THIS.Caption = oSourceList[oCuadroCombo.Value]
CASE oSource.Type = "ComboBox"
     THIS.Caption = oSourcelist[oSource.Val]
CASE oSource.Type = "ComboBox"
     THIS.Caption = oSourceValue
OTHERWISE
     WAIT WINDOW "El objeto se soltó ilegalmente."
ENDIF
```

Esto muestra un uso sencillo de la capacidad DragDrop que tienen todos los objetos. Obviamente, cómo utiliza usted cada objeto después de que es soltado, depende del objeto. Probablemente deseará cambiar la funcionalidad del objeto una vez que algo ha sido soltado sobre él podría hacer que una imagen cambie si, por ejemplo, se suelta una palabra sobre ella, o que un botón de comando corra de manera diferente cuando se haga clic en él, dependiendo de su encabezado.

Con el método AddItem, usted puede utilizar DragDrop para arrastrar elementos dentro de una lista o más botones encima de una barra de herramientas.

DragOver

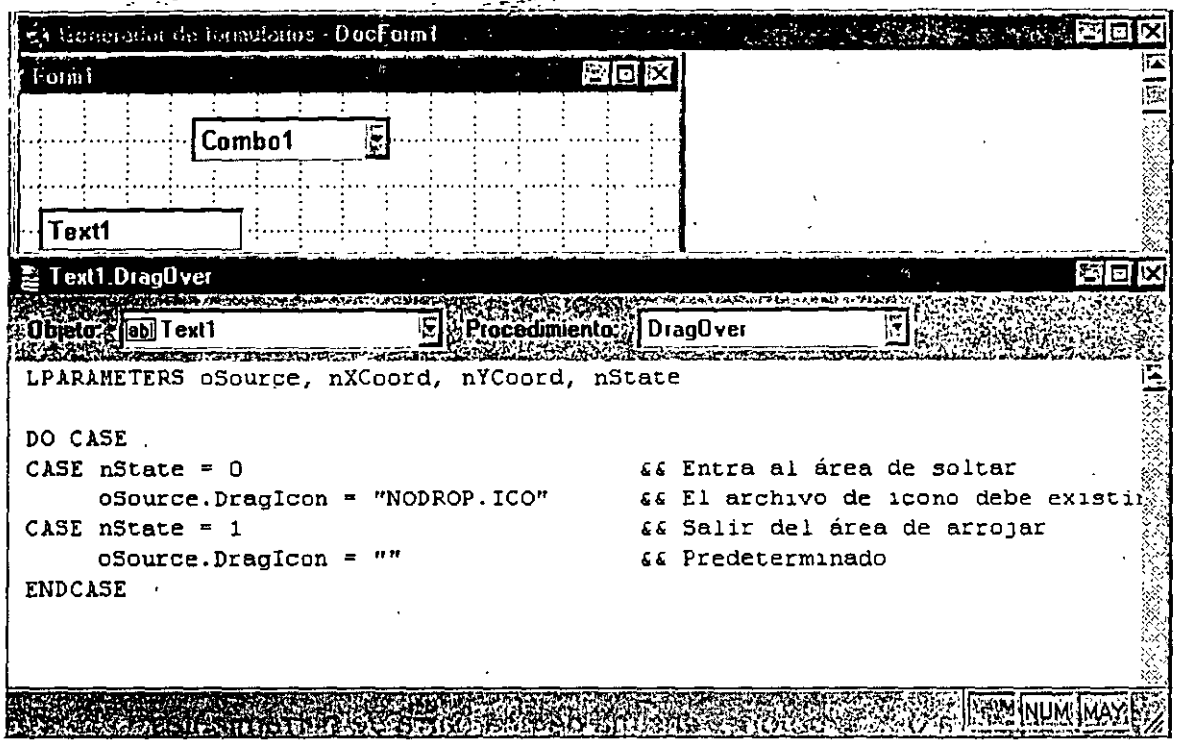
El evento DragOver tiene lugar cuando un objeto es arrastrado dentro del campo en que puede ser soltado sobre otro objeto, y luego se vuelve a llamar si existe un intervalo para soltar (ver figura 12-2). Pasa parámetros al objeto diciéndole qué tipo de objeto está siendo arrastrado para que usted pueda controlar el resaltado del objeto o cambiar el icono arrastrar basándose en si es posible o no soltar encima del objeto.

El siguiente código en el evento DragOver de un objeto hará que el icono arrastrar indique que soltar algo sobre este objeto no tendría ningún efecto, o que es ilegal:

```
DO CASE
CASE nSource = 0                        && Adentro del área
   oSource.DragIcon = "NODROP.ICO"      && El archivo icono deberá existir
CASE nState = 1                         && fuera del área
   oSource.DragIcon = " "               && por omisión
ENDCASE
```

Advierta que el icono NODROP.ICO debe existir en el directorio actual o del proyecto, o debe ser especificada una vía de acceso completa. Visual FoxPro viene con muchos iconos que usted puede utilizar para propósitos como éste.

Figura 12-2



Objeto arrastrado sobre un botón de comando; icono cambiado a un icono distinto.

Error

El evento Error tiene lugar cuando ocurre un error en el tiempo de ejecución en un método de un objeto. Tiene prioridad sobre la configuración actual ON ERROR, si la hay. Debería utilizarse para anular el procedimiento ON ERROR, si existe alguna razón para hacerlo al nivel de un objeto. La mayoría de los reportes de error deberían utilizar el comando ON ERROR con el fin de mantenerse centrados. Por ejemplo, para saltarse el procesamiento de errores si ocurre un error en un objeto en particular, puede simplemente añadir un comentario al evento Error. Esto ejecutaría el comentario en lugar del procedimiento ON ERROR, saltándoselo efectivamente.

Si desea cambiar el procesamiento de errores sólo de ciertos errores, o incluso un error determinado en una cierta línea de un cierto método, puede utilizar los parámetros del método Error para verificar cualquier condición que desee y manejarlos de manera específica, o pasar el control a la rutina ON ERROR si sus condiciones no son satisfechas.

Puede utilizar los parámetros de este método para dar información al usuario con respecto al lugar del error, por lo que las fallas pueden reportarse de manera más

efectiva. Aquí está un ejemplo de un manipulador de errores muy sencillo que demuestra esta técnica:

```
=MessageBox("Ocurrió el error " +;
   alltrim(str(nError) + ;
   "en la línea + ;
   alltrim(str(nLinea)) + ;
   " de " + cMetodo + "." + chr(13) + ;
   "Por favor reportelo inmeditamente al departamento MIS:", ;
   48,"Microsoft Visual FoxPro")r
```

La figura 12-3 es un ejemplo de este cuadro de mensaje. A estas alturas, dependiendo de la severidad del error, el programa podría bien cerrarse, o continuar normalmente. Podría escribirse en una bitácora de errores un registro del error, junto con cualquier otra información relevante, tal como el ID del usuario, el número y la tabla del registro, memoria disponible y cualquier otro factor que pueda ayudar en la depuración.

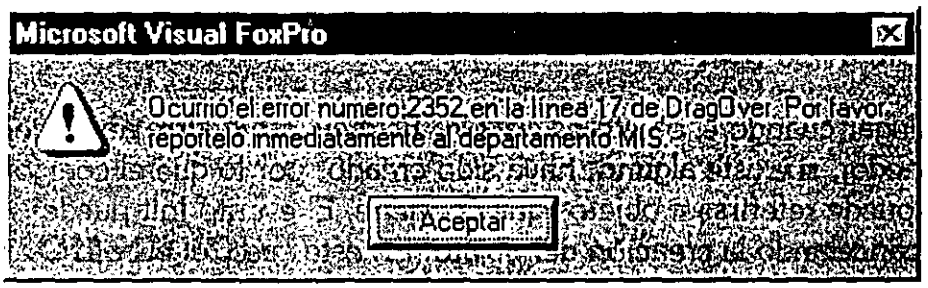


Figura 12-3

Ejemplo del cuadro de mensaje anterior.

GotFocus

El evento GotFocus se dispara cuando un objeto es el objeto activo. Esto sucede si se hace clic en un objeto, si es el siguiente elemento de tabulador y se presionan las teclas TAB o ENTER o se mueve con las flechas direccionales, si fue el último objeto activo en el formulario y el formulario está activado, o si se llama al método SetFocus. El evento When ocurre antes de que un objeto sea enfocado, por lo que para cuando el evento GotFocus tiene lugar, usted puede asumir que le ha sido "permitido" al objeto ser enfocado y obtener el código para verificar si éste debe ir en el evento When.

Un buen ejemplo de cuándo pueden ser utilizados GotFocus y LostFocus viene en la aplicación de muestra en el capítulo 9. Usted necesitaba cambiar el color de los campos mientras se movía de uno a otro, para mostrar al usuario qué campo

estaba en foco. Usted especificó negro sobre rojo cuando un objeto estaba en foco, y negro sobre blanco cuando no lo estaba. Para hacer esto, ponga el siguiente código en el evento GotFocus del objeto:

```
THIS.ForeColor = RGB(255,255,255)
THIS.BackColor = RGB(255,0,0)
```

y el siguiente en el evento LostFocus del objeto:

```
THIS.ForeColor = RGB(0,0,0)
THIS.BackColor = RGB(255,255,255)
```

Con pocas excepciones, los objetos que obtienen eventos GotFocus obtienen también eventos When. En los casos en que no existe esta relación, por lo general se trata de objetos contenedores, en cuyo caso los objetos miembros reciben el evento GotFocus y el contenedor obtiene el evento When.

Init

El evento Init tiene lugar cuando el objeto es creado. Este evento ocurre antes de que el objeto contenedor, si existe alguno, haya sido creado, por lo que el código Init del contenedor puede referirse a objetos que contiene. El evento Init puede regresar falso (.F.), cancelando la creación del objeto. En este caso, el evento Delete no se dispara.

LostFocus

El evento LostFocus ocurre cuando el objeto pierde el enfoque, lo que puede suceder si el usuario hace clic en otro objeto, mueve el tabulador lejos de un objeto, oprime ENTER para mover el foco al siguiente control o se aleja con las teclas de flechas. También sucederá si una llamada al método SetFocus mueve el control hacia otro objeto.

El evento When ocurre antes del evento LostFocus, lo que sucede a su vez antes del evento Deactivate. Si el código del evento When regresa falso (.F.), entonces no le sigue el evento LostFocus y el objeto conserva el foco. Para un ejemplo de cómo utilizar el evento LostFocus en conjunción con el evento GotFocus para cambiar el color de fondo del objeto enfocado, vea el evento anterior GotFocus.

Message

El evento Message exhibe un mensaje en la barra de estado al pie de la pantalla. Para mostrar este mensaje, el código del evento Message debería utilizar RETURN como cadena de texto, tal como la siguiente:

```
RETURN "Por favor inserte el apellido del cliente."
```

Una mejor manera de realizar esto es utilizar la propiedad StatusBarText, que se muestra en la figura 12-4. El evento Message está incluido para compatibilidad con versiones anteriores, y realiza la misma función que la propiedad StatusBarText. La propiedad StatusBarText tiene la ventaja adicional de ser modificada de manera más sencilla durante el tiempo de ejecución.

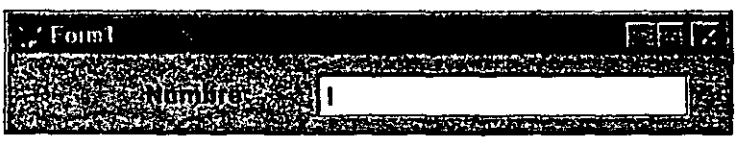


Figura 12-4

Ejemplo de texto de la barra de estado.

MouseDown

El evento MouseDown ocurre cuando se hace clic con un botón del ratón. Un evento distinto, MouseUp, sucede cuando se libera un botón. El código relacionado con la acción principal de una propiedad, tal como hacer clic en un botón de comando, debe estar vinculado al evento Click, porque se dispara sólo cuando se presiona el botón principal del ratón. El evento MouseDown ocurre cuando cualquiera de los botones del ratón (primario, secundario o central) se presiona. El evento MouseDown también contiene información sobre el estado de modificadores de teclado (MAYÚS, CTRL y ALT) cuando está presionado el botón, lo que no sucede en el evento Click.

Por ejemplo, si usted desea ejecutar un código sólo si se hiciera clic en un botón de comando con el botón central del ratón mientras que CTRL y ALT se oprimieran, usaría el siguiente código en el evento MouseDown:

```
IF nBoton=4 .AND. nDesplaz=6
   WAIT WINDOW "¡Usted encontró el huevo de pascua!"
ENDIF
```

El evento MouseDown también se utiliza para iniciar la función de arrastrar y soltar de los objetos. Si usted desea habilitar ésta para un objeto, simplemente introduzca la línea siguiente en el evento MouseDown:

```
THIS.Drag
```

y el siguiente código en el evento MouseUp:

```
THIS.Drag(2)
```

para finalizar el arrastre y hacer que un evento DragDrop ocurra en el objeto de destino. El objeto de destino tendrá entonces que ser programado para utilizar el objeto que fue soltado sobre él, cualquiera que fuera el propósito de arrastrarlo. Vea el ejemplo anterior en *DragDrop* para ver un ejemplo de comportamiento.

MouseMove

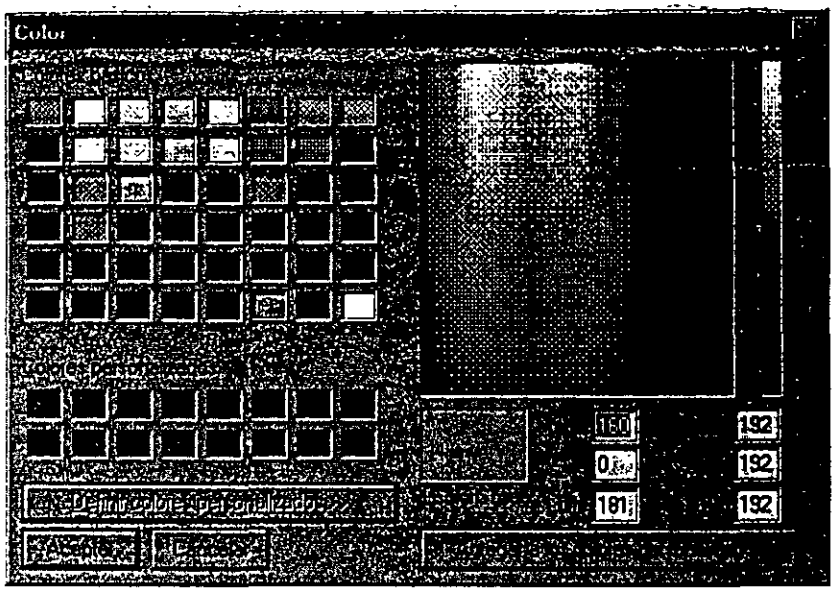
El evento MouseMove ocurre cada vez que el ratón está sobre un objeto y la posición del ratón cambia en relación con el objeto. Advierta que esto puede ocurrir si la ventana u objeto se mueven con el código, aunque el ratón no se mueva.

¡Tenga cuidado al usar este evento! Si usted observa con qué frecuencia ocurre (imprima un mensaje en la pantalla cada vez que suceda), verá que el evento se desencadena casi constantemente. Sea muy cuidadoso con el código que ponga aquí, pues será ejecutado hasta unas 20 veces por segundo.

Una buena demostración de este evento es el seleccionador de colores personalizados en el mismo Visual FoxPro, que se muestra en la figura 12-5. Ése es básicamente un mapa de bits de la paleta de color, la cual alimenta valores dentro de los cuadros de texto, basándose en el color sobre el que está rondando el ratón cada vez que se hace clic con el botón primario del mismo. Si esto estuviera escrito en Visual FoxPro, habría un mapa de bits en el formulario con el siguiente código en su evento MouseMove:

```
IF nBoton=1
   THISFORM.txtRed.Value      = nXCoord - THIS.Left
   THISFORM.txtGreen.Value    = nYCoord - This.Top
   * tercero (azul) esta dimensión sería manejada de forma separada
ENDIF
```

Figura 12-5



Cuadro de diálogo del seleccionador de color.

El motivo por el que usted utilizaría el evento MouseMove en lugar del evento MouseDown es que, mientras mueve el cursor del ratón por la pantalla, usted desea que los valores en los cuadros de texto sigan cambiando. El único evento que puede disparar esta respuesta es el evento MouseMove.

Otra demostración del evento MouseMove es cuando usted persigue un objeto en el formulario con el ratón, pero el objeto sigue intentando alejarse del mismo. Para implantar esto, simplemente especifique CREATE FORM CATCHME, y luego añada un solo objeto. Puede ser cualquier cosa, pero en mi caso es una forma de aproximadamente 75 x 75 pixeles, redondeado en las esquinas con una configuración de curvatura de 99 y amarillo, un FillStyle de 0 (sólido) y FillColor de RGB(255,255,0). Obviamente, usted puede cambiar cualquiera de estas configuraciones según lo considere conveniente. Sólo recuerde cambiar el código para que haga juego con el nombre del objeto.

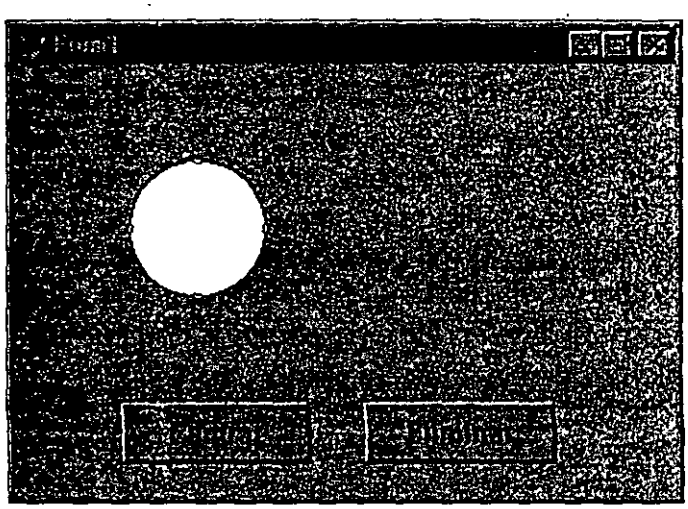
El paso siguiente es hacer doble clic en un área vacía del formulario para hacer aparecer la ventana del código. Seleccione el procedimiento del evento MouseMove desde el cuadro de lista en la parte superior de la ventana, y luego introduzca el código siguiente bajo la línea de parámetro:

```
IF nXCoord < THISFORM.Shape1.Left AND          ;
   nXCoord > THISFORM.Shape1.Left - 25 AND     ;
   nYCoord > THISFORM.Shape1.Top AND           ;
   nYCoord < THISFORM.Shape1.Top + THISFORM.Shape1.Height
   THISFORM.Shape1.Left = THISFORM.Shape1.Left + 10
ENDIF
```

```
IF nXCoord > THISFORM.Shape1.Left + THISFORM.Shape1.Width AND ;
   nXCoord < THISFORM.Shape1.Left + ;
      THISFORM.Shape1.Width + 25 AND ;
   nYCoord > THISFORM.Shape1.Top AND ;
   nYCoord < THISFORM.Shape1.Top + THISFORM.Shape1.Height
   THISFORM.Shape1.Left = THISFORM.Shape1.Left - 10
ENDIF

IF nYCoord < THISFORM.Shape1.Top AND ;
   nYCoord > THISFORM.Shape1.Top - 25 AND ;
   nXCoord > THISFORM.Shape1.Left AND ;
   nXCoord < THISFORM.Shape1.Left + THISFORM.Shape1.Width
   THISFORM.Shape1.Top = THISFORM.Shape1.Top + 10
ENDIF

IF nYCoord > THISFORM.Shape1.Top + THISFORM.Shape1.Height AND ;
   nYCoord < THISFORM.Shape1.Top + ;
      THISFORM.Shape1.Height + 25 AND ;
   nXCoord > THISFORM.Shape1.Left AND ;
   nXCoord < THISFORM.Shape1.Left + THISFORM.Shape1.Width
   THISFORM.Shape1.Top = THISFORM.Shape1.Top - 10
ENDIF
```

La figura 12-6 muestra el ejemplo CHASE.SCX. Cuando usted corra el formulario, el círculo amarillo evadirá sus intentos de tocarlo con el ratón. Si usted mueve el ratón suficientemente rápido, puede brincarse el foso de 25 pixeles y tocar el círculo. Jugar con este ejemplo le dará una idea del comportamiento del evento MouseMove.

Figura 12-6



Ejemplo del formulario CHASE.SCX.

MouseUp

El evento MouseUp ocurre cuando el botón del ratón es liberado de una posición presionada. Advierta que en la mayoría de las circunstancias este evento suele seguir al evento MouseDown durante una cantidad insignificante de tiempo. Si se le da la función de arrastrar y soltar a un objeto, el evento se utiliza para desencadenar la acción de soltar el objeto. Esto se logra de la siguiente manera:

```
THIS.Drag(2)
```

Vea el ejemplo que se da en la descripción del evento DragDrop anteriormente en el capítulo, para un ejemplo completo de cómo implantar la función de arrastrar y soltar.

ProgrammaticChange

El evento ProgrammaticChange ocurre si la propiedad Value del objeto cambia en el código, contrario a interactivamente, en otras palabras, si la línea:

```
<Object>.Value = <nada>
```

se ejecuta en cualquier parte de su programa. Esto es útil si, por ejemplo, usted desea desencadenar el evento Valid para validar cambios en el código. Si corre el código cada vez que cambia el valor, usted no tiene que duplicar el código en cada lugar en que se ha cambiado el mismo; simplemente ponga el código en este evento y Visual FoxPro lo ejecutará automáticamente, si es necesario.

Para desencadenar el código que se ejecutará cada vez que el valor de un control numérico cambia use el evento InteractiveChange en lugar de este evento o del evento Click.

RightClick

El evento RightClick ocurre cuando se hace clic con el botón secundario del ratón y se libera sobre un objeto. Advierta que el botón secundario puede cambiarse en la aplicación pequeña del panel de control del controlador (*driver*) del ratón, por lo que no es exacto referirse siempre a él como el botón secundario del ratón, aunque puede asumir que lo será el 99% de las veces.

El evento RightClick pasa los mismos parámetros que el evento Click, lo cual no es mucho decir. Se usa sobre todo de la misma manera en que se utiliza el evento Click, sin que se considere acompañar eventos del teclado, posición del ratón o temporización. Este evento se utiliza como un vínculo sencillo con el evento Click con el botón secundario del ratón.

Un uso muy común del botón secundario del ratón en aplicaciones recientes es como un atajo a un menú de acciones vinculadas a un objeto. Para implantar tal menú en Visual FoxPro, el menú debe definirse en el evento Init del formulario, o en el código de preparación de una aplicación de nivel más alto, si el menú es utilizado por múltiples formularios. Este código podría verse de la siguiente manera:

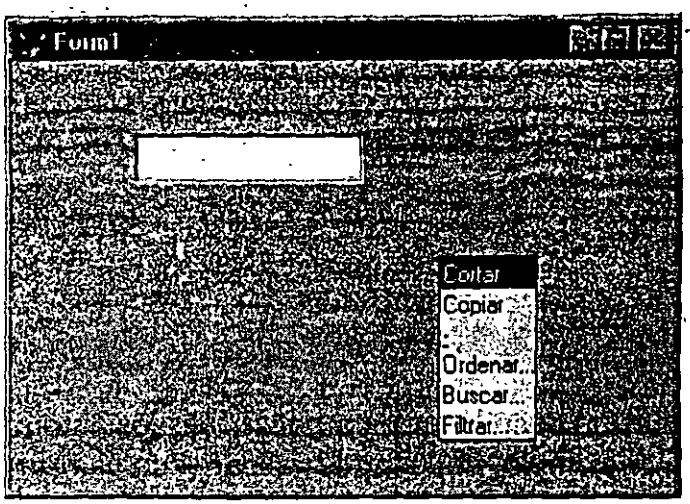
```
DEFINE POPUP popAccion IN WINDOW Form1   && no existe ubicación todavía
DEFINE BAR 1 OF popAccion PROMPT "Cortar"
DEFINE BAR 2 OF popAccion PROMPT "Copiar"
DEFINE BAR 3 OF popAccion PROMPT "\<-"
DEFINE BAR 4 OF popAccion PROMPT "Ordenar..."
DEFINE BAR 5 OF popAccion PROMPT "Buscar..."
DEFINE BAR 6 OF popAccion PROMPT "Filtrar..."
ON SELECTION POPUP popAccion DO popAccion.PRG
```

En el evento RightClick de cada objeto que utiliza este menú, el menú se movería primero a un lugar que lo asocie con el objeto, y luego se activaría.

```
MOVE POPUP popAccion TO 6,35
ACTIVATE POPUP popAccion
```

El código popAccion.PRG (o un procedimiento almacenado en un archivo de programa de nivel más alto, si el menú es utilizado por múltiples formularios) manejaría ya sea las rutinas de cortar y copiar o la ejecución de los formularios Ordenar, Buscar y Filtrar y el paso del parámetro para indicar qué objeto las invocó. El menú popAccion junto a un cuadro de texto se muestra en la figura 12-7.

Figura 12-7



Menú popAccion junto a un cuadro de texto.

Valid

El evento Valid es a donde debe ir la mayoría de su validación de datos, asumiendo que es una validación a nivel de campo. Después de verificar la validez del valor del control, usted debe regresar ya sea verdadero (.T.) o nada (verdadero es el valor predeterminado) si los datos son válidos, y falso (.F.) si no lo son. Si se regresa falso, entonces no se mueve el foco al siguiente objeto y se muestra el ValidationText en una ventana Wait.

Esto significa que, dependiendo de diversas razones en las que los datos podrían ser inválidos, usted puede desear cambiar la propiedad ValidationText antes de salir y regresar falso del evento Valid. Por otro lado, puede ignorar por completo el mensaje ValidationText fijándolo a una cadena vacía. Si hace esto, entonces el comportamiento del objeto cuando se regresa falso del evento Valid es simplemente que el objeto no será desenfocado. Podría desear sustituir su propio mensaje, por ejemplo, utilizando la función MessageBox.

Si está validando un campo de datos, puede requerir que la fecha sea igual o previa a la fecha de hoy. Pero también puede desear permitir al usuario ignorar esta protección, por ejemplo, si la fecha del sistema es equivocada y los datos deben introducirse sin detenerse a cambiarla (o si el usuario no sabe cómo hacerlo). Éste es un caso perfecto para utilizar la función MessageBox. El evento Valid para este cuadro de texto podría verse de la siguiente manera:

```
IF THIS.Value > Date()
   IF MessageBox("La fecha no puede estar en el futuro." + ;
      CHR(13) + "¿Aceptar de cualquier manera?", ;
      4+32+256, ;
      "Fecha inválida") = 7    && No
      THIS.ValidationText = " "
      RETURN .F.
   ELSE                        && Sí
      RETURN .T.
   ENDIF
ENDIF
```

El evento Valid también es un buen lugar para hacer cálculos automáticos de campos relacionados. Por ejemplo, en lugar de validar si un campo de fecha de nacimiento coincide con un campo de edad, calcule el campo de edad en el evento Valid del campo de fecha de nacimiento, cambie el campo de edad automáticamente, y no sólo le evitará al usuario el esfuerzo de introducir más datos, sino que le ahorrará molestos mensajes de "datos inválidos" si los datos que se introdujeron están equivocados. Es la misma cantidad de código, y una interfaz mucho más amigable.

When

El evento When se utiliza para la selección condicionada de barra de un objeto. Es como una propiedad Enable dinámica en que, cuando el usuario intenta seleccionar un objeto, el código de evento When puede decidir basándose en cualesquiera condiciones necesarias si el usuario es admitido o no en el campo. El objeto no parecerá estar inhabilitado, pero si la cláusula When regresa falso (.F.), el usuario no podrá seleccionarla de forma interactiva de ninguna manera, y las llamadas al método SetFocus no funcionarán.

Los usuarios de FoxPro 2.x están acostumbrados a utilizar la cláusula When para casi cualquier código de preselección. En Visual FoxPro, se han añadido tantos eventos nuevos que usted debe considerar cuidadosamente el verdadero propósito de cada evento. El evento When ahora se usa exclusivamente para determinar si un objeto puede ser enfocado o no. Si es así, entonces el evento GotFocus es el lugar para la mayoría del preprocesamiento del objeto. El resultado de esto es que el evento When ya no sirve como un evento que abarca todo, como sucedía en 2.x. Debe utilizarse solamente cuando existe una posibilidad de que al enfoque no le sea permitido descansar sobre el objeto. Si ésta es una conclusión previa, entonces el código pertenece al evento GotFocus. Y si usted puede predeterminar que el objeto no es seleccionable, entonces probablemente deba ser inhabilitado. Sea moderado con el uso del evento When.

Uno de los usos del evento When, por ejemplo, es pedir una contraseña antes de permitir la edición de un objeto. El evento When puede presentar una ventana de contraseña, verificar la misma y luego permitir o no la entrada. El siguiente código en el evento When de un objeto es un ejemplo de cómo podría ser implementarse esta función.

```
DO FORM clave
IF txtclave <> cDecripta(Usuarios.cClave)
   RETURN .T.
ELSE
    WAIT WINDOW "Clave incorrecta. Acceso denegado."
    RETURN .F.
ENDIF
```

Este código asume la existencia de un formulario llamado Clave que introduce txtclave, una tabla llamada Usuarios que contiene datos de contraseñas y una función llamada cDecripta que descifra los datos en la tabla.

Load

El evento Load es el primer evento que se desencadena cuando usted ejecuta DO FORM *nombredeformulario*. El propósito principal del código LOAD es asegurar que sean accesibles las cosas que necesitan los objetos de los formularios, especialmente tablas y parámetros para los comandos SQL en los cuadros de lista y cuadros combinados. Puesto que usualmente usted puede utilizar el entorno de datos para abrir las tablas necesarias, la única ocasión en que abrirá su propio código será para proveer pruebas aisladas de formularios modales que asuman que la tabla necesaria es utilizada. Por ejemplo, en el programa de búsqueda de la palabra clave, el código Load es:

```
IF NOT USED ( "LIBROS" )
   USE LIBROS IN 0
ENDIF
SELECT LIBROS
SET ORDER TO ISBN
```

Usted no debe abrir LIBROS en el entorno de datos porque tendrá su propio indicador de registro, y no puede utilizarlo para regresar con el indicador de registro apuntando al libro seleccionado.

Un segundo uso del evento LOAD es crear una variable pública que es necesaria como parámetro para una instrucción SQL SELECT utilizada como RowSource para un cuadro combinado o un cuadro de lista. Por ejemplo, si tiene una función

de búsqueda para encontrar el registro de un cliente, necesita el cuadro de lista de órdenes para que ese cliente se vuelva a cargar. No puede utilizar:

```
SELECT NumOrden, Fecha, Monto  FROM ORDENES, CLIENTE ;
   WHERE IDCliente = CLIENTE.IDCliente
```

porque producirá una combinación de los archivos ORDENES y CLIENTE. Y no puede utilizar:

```
SELECT NumOrden, Fecha, Monto FROM ORDENES, CLIENTE ;
   WHERE IDCliente = THISFORM.IDCliente
```

porque THISFORM no está permitido fuera de un METODO. Así que escriba lo siguiente en el código LOAD:

```
IF TYPE ( "mCliente" ) = "U"
   PUBLIC  mCliente
ENDIF
mCliente = Cliente.IDCliente
```

y cambie el valor de mCliente cada vez que cambie a los clientes: en el código Click de los botones Siguiente, Anterior y Buscar.

Activate

El código Activate es su oportunidad para asegurarse de que la tabla principal utilizada por un formulario particular sea seleccionada cada vez que el formulario se convierta en el formulario "a la cabeza". Puesto que las aplicaciones de ventanas múltiples son fáciles de crear en Visual FoxPro, espere que los usuarios pongan varias ventanas en la pantalla y seleccionen entre ellas. Ya que el código ACTIVATE del formulario corre cada vez que un formulario se vuelve el formulario colocado a la cabeza, puede simplemente incluir la línea:

```
SELECT CLIENTE
```

en el código ACTIVATE del formulario CLIENTE, y está listo.

Unload

Si desea que un formulario regrese un parámetro, tiene que hacer tres cosas:

- Hacerlo un formulario MODAL, utilizando WindowType = 1 - Modal.
- Incluir la línea RETURN *valor* en el evento UNLOAD.
- Llamar al formulario con la sintaxis:

```
DO FORM nombredeformulario TO nombredevariable
```

InteractiveChange

Si su diseño llama a dos cuadros de lista en una pantalla, donde todos los elementos en el segundo cuadro de lista son registros hijos de la fila actualmente resaltada en el primer cuadro de lista, InteractiveChange es para usted. Simplemente ponga el código SQL para poblar el segundo cuadro de lista en su RowSource, basándose en un parámetro (por ejemplo, mIDCliente), y luego escriba algo como esto en el código InteractiveChange del primer cuadro de lista:

```
* Código InteractiveChange:
mIDCliente = THIS.Value
THISFORM.List2.Requery
```

Métodos

Visual FoxPro tiene alrededor de 50 métodos que ejecutan su código, el código que usted escribe, sólo cuando los llama. Sus nombres describen lo que los desencadena, Move, Refresh, SetFocus, y así sucesivamente. Éstos son los métodos en la lista "A":

- Drag
- Move
- Refresh
- Release
- Requery

- SetAll
- SetFocus
- ZOrder

Probablemente usted no utilizará tantos métodos para todos los objetos, pero esto es porque los métodos tienden a ser altamente especializados para el objeto al que pertenecen. Aunque no se hablará de tantos métodos como se hizo con los eventos, o como se hará con las propiedades, son igualmente esenciales para el diseño de la aplicación. Sería difícil escribir una aplicación decente sin los métodos Refresh y SetFocus, y mientras que SetAll no es esencial técnicamente, puesto que su funcionalidad puede ser duplicada con rizos, es uno de los métodos accesibles más útiles.

Drag

El método Drag se utiliza para iniciar y finalizar la función de arrastrar y soltar objetos. Se le llama con uno de tres parámetros. El primero se pasa para poner el objeto en modalidad de arrastrar. Esto es también el modo predeterminado si no se pasa ningún parámetro. Este método se invoca usualmente desde el evento MouseDown. El segundo se pasa para finalizar la modalidad de arrastrar y soltar el objeto en cualquier objeto sobre el que está actualmente. Este objeto recibirá entonces un evento DragDrop, con el objeto como uno de los parámetros. Este método se invoca generalmente desde el evento MouseUp. El tercer parámetro se pasa para finalizar la modalidad de arrastrar y cancela la operación. Ningún objeto recibirá un evento DragDrop. Para un ejemplo de cómo implementar arrastrar y soltar, vea el evento DragDrop en *Eventos*, anteriormente en este capítulo.

Move

El método Move mueve el objeto a una nueva posición en el formulario y/o le vuelve a dar tamaño al objeto. Las coordenadas para el movimiento son la posición de la esquina superior a la izquierda del objeto relativa a la esquina superior a la izquierda del formulario (0,0). También puede volver a ajustarse el tamaño de un objeto con el método Move. Los parámetros tercero y cuarto del método Move son para el ancho y la altura del objeto.

Cuídese de utilizar el método Move en conjunción con el evento MouseMove. El evento MouseMove se desencadena cuando la posición del ratón cambia en relación con el objeto. Por lo tanto, si un evento MouseMove mueve el objeto, desencadenará otro evento MouseMove, lo que volverá a mover al objeto, y desencadenará otro evento MouseMove, esto eventualmente provocará una falla en la pila.

Refresh

El método Refresh se utiliza para actualizar el aspecto de los objetos. Si su aspecto o sus contenidos han cambiado y refrescar no es una función automática, usted debería llamar este método al objeto. Si se cambian objetos múltiples en un objeto contenedor (como un formulario) usted debe llamar al método Refresh del objeto contenedor.

La mayoría de los cambios a un objeto, tales como el color, la visibilidad o habilitar de las propiedades, y el valor, son refrescados automáticamente. Si usted realiza un cambio a un objeto y no refleja el cambio de manera visual, debe añadir una llamada al método Refresh.

Release

Utilice este método para eliminar un formulario.

Requery

El método Requery recarga un cuadro de lista.

SetAll

El método SetAll es un método muy poderoso. Se utiliza para cambiar de un solo golpe las propiedades de todos los objetos contenidos. Por ejemplo, usted puede inhabilitar todos los objetos de un formulario con un comando:

```
THISFORM.SetAll("Enabled", .F.)
```

Es tan sencillo como eso. No se necesita una línea de código para cada objeto, o rizos poco elegantes y consumidores de tiempo.

El método SetAll tiene un parámetro adicional que lo hace aún más poderoso. Puede fijar un parámetro de objetos de una clase dada a una configuración determinada. Es decir, si usted desea inhabilitar sólo los cuadros de texto en un formulario, puede correr el método con el nombre de clase como el tercer parámetro:

```
THISFORM.SetAll("Enabled", .F., "TextBox")
```

Sabiendo esto, puede crear una clase nueva de cuadro de texto que sea un modelo no modificado de su objeto cuadro de texto Ingreso, salvada con un nombre de clase distinto. Cuando su botón Editar habilita campos de ingreso, utilizando, por decir, SetAll ("Enabled",.T.,"Ingreso"), su clase sólo-despliegue permanece inhabilitada. No existe una manera más sencilla de poner campos SAY en un formulario.

SetFocus

El método SetFocus intenta mover el enfoque hacia el objeto sobre el que fue llamado el método. Para que este método tenga éxito, debe cumplirse un cierto número de condiciones, en el siguiente orden de importancia:

- El objeto destino debe ser habilitado =.t..
- El objeto destino debe ser visible=.t..
- El objeto actual debe retornar verdadero de su evento Valid.
- El objeto destino debe retornar verdadero de su evento When.

Si se cumplen todas estas condiciones, el método se realizará con éxito y el objeto actual recibirá un evento LostFocus antes de que pase el control y el objeto destino reciba un evento GotFocus. Llamar a este método es equivalente a cambiar el valor de _CUROBJ en FoxPro 2.x.

Cuando yo llamo a mi código del botón Editar, puedo mover el enfoque al primer campo de entrada utilizando esto:

```
cmdEditar::Click
ThisForm.IDCliente.SetFocus
```

ZOrder

El método ZOrder fija el orden gráfico de un objeto en el tiempo de ejecución. Esto no cambia el orden de tabulación de los controles, ni pondrá objetos gráficos (como cuadros, líneas y círculos) frente a los controles. Existen dos capas gráficas en un formulario: la capa de dibujo y, en frente de ésta, la capa del objeto. Los objetos sólo pueden ordenarse dentro de su capa específica.

Este método por lo general se utilizará sólo con objetos gráficos, puesto que los objetos de control rara vez se superponen entre sí. Ya que los objetos de control siempre se superponen sobre los objetos gráficos, no es necesaria ninguna manipulación ZOrder para hacer visibles los controles sobre objetos gráficos. La figura 12-8 muestra objetos gráficos en pila.

El orden sólo puede cambiarse trayendo un objeto al frente o moviéndolo hasta el fondo. Si un objeto es el dieciseisavo de veinte y usted desea convertirlo en el décimo tercero, tendrá que enviarlo al fondo con:

```
<Objeto>.ZOrder(1)
```

y luego enviar los siete objetos siguientes al fondo detrás de él.

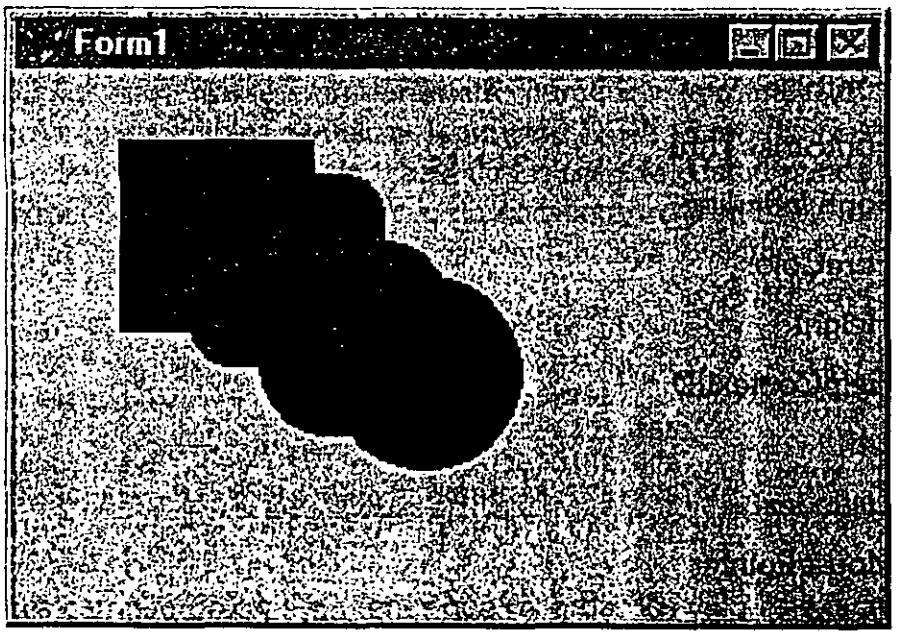


Figura 12-8

Objetos gráficos en pila.

Propiedades comunes

Visual FoxPro tiene alrededor de 250 propiedades que contienen información acerca de los formularios y los objetos. Los datos pueden considerarse variables locales de un objeto, lo que afecta automáticamente su aspecto y comportamiento, dependiendo de sus valores. Éstas son las propiedades de la lista "A":

- BackColor
- Caption
- Comment
- ControlSource
- DragIcon
- DragMode
- Enabled
- FontBold
- FontItalic
- FontName
- FontOutline
- FontShadow
- FontSize
- FontStrikeThru
- FontUnderline
- ForeColor
- Height
- HelpContextID
- Left
- ListIndex
- MousePointer
- Name
- RowSource

- RowSourceType
- SpecialEffect
- TabIndex
- TabStop
- ToolTipText
- Top
- Value
- Visible
- Width

Éstas son unas cuantas propiedades que se usan comúnmente por muchos objetos y que son muy útiles. En las siguientes secciones se describirán y se tratará de ofrecer ejemplos significativos de su uso.

BackColor

La propiedad BackColor fija el color del fondo del objeto. Si la propiedad BackStyle es 0 (transparente), se ignora. Si es 1 (opaco), entonces el color de fondo cambia al valor BackColor tan pronto como se establece.

Aunque BackColor debe fijarse a un parámetro numérico, la manera más sencilla de fijarlo en colores inteligibles es utilizar la función RGB( ) para fijar los componentes relativos de rojo, verde y azul de los colores. Usted puede desear almacenar colores comunes en un archivo de encabezado, y utilizar #include para ponerlo en sus aplicaciones. Por ejemplo, si usted incluye lo siguiente:

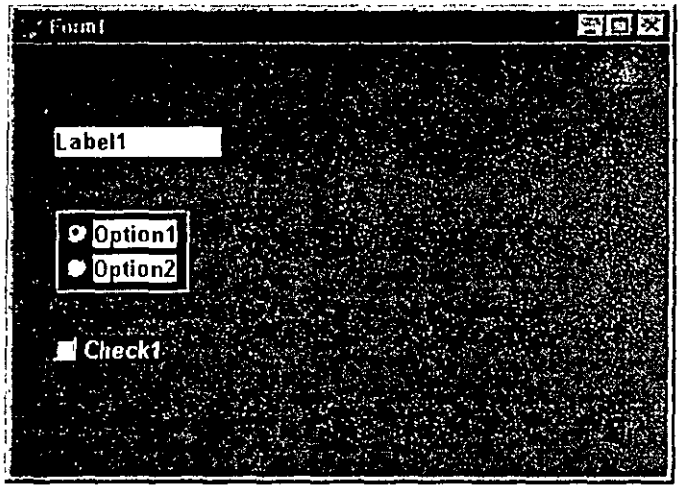
```
#define rgbRed                  RGB(255,0,0)
#define rgbBlue                 RGB(0,255,0)
#define rgbGreen                RGB(0,0,255)
#define rgbWhite                RGB(255,255,255)
#define rgbLightGray            RGB(192,192,192)
#define rgbDarkGray             RGB(128,128,128)
#define rgbBlack                RGB(0,0,0)
```

puede entonces cambiar colores sin tener que recordar los valores exactos de RGB:

```
<Objeto>.BackColor = rgbLightGray
```

Si desea poder cambiar el color de fondo de sus formularios fácilmente, se recomienda que todos los objetos en los formularios tengan un BackStyle transparente. De otra manera, necesitará una serie compleja de métodos SetAll en el formulario para hacer que los colores de los objetos hagan juego. La figura 12-9 muestra objetos opacos con distintos colores de fondo.

Figura 12-9



Objetos opacos con colores de fondo distintos a los del formulario.

La propiedad BackColor en conjunción con BackStyle y ForeColor puede utilizarse para indicar un estado al usuario, para dividir la funcionalidad en un solo formulario, o simplemente para resaltar un objeto de un formulario.

Caption

La propiedad Caption es probablemente una de las propiedades más importantes de Visual FoxPro. Lo que carece en complejidad lo suple por su visibilidad. Es la principal herramienta para comunicarse con funcionalidad con el usuario. La propiedad Caption es la descripción del texto que el usuario ve en cada objeto. Para los formularios es el título, para los botones es el texto en el botón, para las casillas de verificación y botones de opcion es el texto a la derecha del botón y para configuraciones de página en las áreas de tabulación de las páginas.

La capacidad para cambiar encabezados durante el tiempo de ejecución no es nueva para Visual FoxPro, y no se recomienda mucho, pero la propiedad Caption afecta a casi todos los objetos.

Comment

La propiedad Comment simplemente almacena información acerca de un objeto. Úsela para recordar por qué creó un objeto, qué características aún deben implantarse para el mismo, o por qué éste hace algo de una manera particular que usted probablemente no pueda recordar después.

ControlSource

La propiedad ControlSource determina la fuente de datos a la que está unido el control, si es que está unido a alguna. Si éste es un campo en una tabla, por ejemplo, el objeto unido tendrá el mismo tipo y valor del campo. Los controles pueden unirse a campos, variables o a nada en lo absoluto. Si no están unidos a nada, entonces actúan como variables por sí mismos.

DragIcon

La propiedad DragIcon es la propiedad que usted establece cuando un objeto está siendo arrastrado y desea cambiar el icono que muestra mientras se arrastra. Por ejemplo, en el evento DragOver de un objeto sobre el que no puede soltarse nada, usted podría sustituir un icono indicando que no es legal soltar. Entonces, cuando el evento DragOver indica que el objeto arrastrado está saliendo del espacio del objeto, el icono puede volverse a cambiar al icono predeterminado.

Si utiliza un icono no estándar para arrastrar el objeto cada vez, entonces puede simplemente fijar el icono en el momento de diseñar, y se utilizará en lugar del icono de error. Si desea cambiarlo dependiendo de las condiciones en el momento de arrastrar, simplemente fíjelo en el evento MouseDown, donde se llama al método Drag para el objeto.

DragMode

Fije la propiedad DragMode en 1 si el propósito principal de un objeto es arrastrar y soltar. Podría utilizarla como una interfaz donde, por ejemplo, el usuario selecciona acciones de un menú al arrastrar representaciones gráficas de opciones a un espacio de selección. El único propósito de las imágenes gráficas es que puedan arrastrarse y soltarse, por lo que si su propiedad DragMode se fija en automático (1), no necesita programar la función de arrastrar/soltar vía el método Drag en los eventos MouseDown y MouseUp.

Puede utilizar esta propiedad también para activar o desactivar la función de arrastrar y soltar de un objeto. Simplemente fije la propiedad DragMode en automático, y el objeto iniciará el modo arrastrar automáticamente cuando se haga clic con el ratón sobre él, y lo soltará cuando se suelte el botón del ratón.

Esto puede evitarle tener que programar la función de arrastrar y soltar de manera manual, pero con el costo de cierto control sobre el objeto. Por ejemplo, si se habilita el arrastrar y soltar automático, el evento GotFocus no se desencadena cuando se hace clic en el objeto. Si se implementa el arrastrar y soltar manual, el evento GotFocus tiene lugar cuando se hace clic en el objeto antes que el evento MouseDown pueda desencadenar arrastrar y soltar. Podría utilizar esto para habilitar arrastrar y soltar para un cuadro de texto sólo si sus contenidos cumplen una condición. De otra manera, al usuario se le permite editar el texto.

Enabled

La propiedad Enabled determina si un objeto puede ser enfocado o no. Si su propiedad Enabled resulta verdadera (la configuración predeterminada), el objeto se comporta normalmente. Si se fija en falso, se exhibe en un color distinto establecido para indicar su estado. La combinación de color predeterminado para objetos inhabilitados deja generalmente el color de fondo igual al color de fondo habilitado, y cambia el color del primer plano a una versión diluida del color del primer plano habilitado, gris oscuro en lugar de negro, por ejemplo.

Las propiedades para fijar los colores de fondo y de primer plano inhabilitados son DisableBackColor y DisableForeColor, respectivamente. Es importante probar todos los objetos en su aplicación en el modo inhabilitado para verificar que su combinación de colores funcione. Es una falla muy común tener un elemento inhabilitado y de pronto encontrar que su color de fondo se dejó en blanco cuando

la pantalla de fondo es gris. Una manera fácil de evitar este problema es hacer a todos los objetos BackStyle 0 (transparentes). Esto se aplica si el objeto está habilitado o inhabilitado, y tiene el efecto de ignorar el color de fondo (BackColor) del objeto y utilizar lo que sea que esté detrás suyo como fondo. A menos que desee que sus objetos tengan un color de fondo diferente a su propio color de fondo, BackStyle siempre debe ser transparente.

Si un objeto está inhabilitado, y no puede obtener el enfoque, nunca recibe eventos GotFocus. Tampoco puede recibir eventos del ratón u otros eventos interactivos. Aún recibe eventos Init y Destroy, pero no responderá de ninguna manera a la interacción del usuario. La figura 12-10 muestra ejemplos de objetos habilitados e inhabilitados, incluyendo uno con un problema de color de fondo.

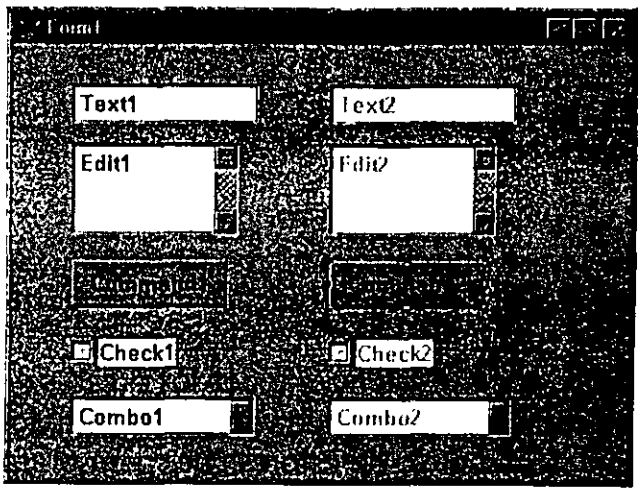


Figura 12-10

Ejemplos de objetos habilitados e inhabilitados, incluyendo uno con el problema del color de fondo.

Fonts

La familia de propiedades de fuentes consiste en lo siguiente:

- FontBold
- FontItalic
- FontName
- FontOutline

- FontShadow
- FontSize
- FontStrikeThru
- FontUnderline

Colectivamente, describen la fuente con que se está presentando el texto en el objeto. El texto afectado por diversos controles varía, pero es relativamente intuitivo. Donde pueden existir múltiples configuraciones de fuentes, tal como en el objeto Columna, existen propiedades específicas de los tipos de texto, llamados el juego de propiedades Fuentes Dinámicas.

En los formularios, las propiedades de fuentes describen la fuente en la que se muestran las nuevas etiquetas colocadas en el formulario, aunque todas las etiquetas que ya están en el formulario tienen sus propias propiedades de fuentes y no son afectadas por las configuraciones del formulario.

* FontBold

Esta propiedad indica si la fuente se presenta en el tipo de negritas, que generalmente tiene un grosor mayor. Una clave práctica muy importante para utilizar texto en negritas es que cualquier objeto que puede ser inhabilitado debe estar en negritas (y por lo tanto, por consistencia, todos los objetos de ese tipo en su apliación). Los botones de comando, casillas de verificación y botones de opción conforman este estándar. El motivo de esto es que, para presentar el texto como inhabilitado en algunas presentaciones, el color del texto debe ser difuso. El color difuso constituye simplemente un diseño de pixeles al estilo de un tablero de damas, utilizado para aproximarse a un color cuando la presentación no es capaz de mostrar el color verdadero. Por ejemplo, un diseño al estilo de un tablero de damas de pixeles azules y rojos puede utilizarse para simular un campo púrpura en un monitor con una gama de colores limitada. Si el texto es negro, se utiliza a veces un negro difuso para mostrar el texto si el objeto está inhabilitado. Si la fuente no es negrita, el ancho del trazo puede ser potencialmente (y por lo general lo es) de un pixel. El texto difuso con un grosor de un pixel casi siempre es ilegible, y por lo tanto presenta un problema. Utilizar fuentes en negritas mejora en gran medida la legibilidad cuando se utiliza un color difuso, y se usa generalmente para resolver este problema. Con la creciente proliferación de estándares más altos de color, este problema es menos grave, pero los programadores siempre deberían codificar para el común denominador más bajo.

* FontItalic

Esta propiedad determina si la forma cursiva de la fuente es utilizada. Las fuentes cursivas simplemente se inclinan para dar mayor énfasis, *como esto*. La propiedad FontName determina el tipo que se está usando.

* FontName

Las fuentes TrueType estándar incluidas en Windows son:

- ➢ Arial (parecida a la Helvetica)
- ➢ Courier New
- ➢ Times New Roman

Las fuentes TrueType son una tecnología similar a Postscript, cuyo propósito es asegurar la similitud del aspecto de las fuentes en múltiples plataformas —en este caso, la pantalla y la impresora. También pueden utilizarse en los formularios fuentes que no sean TrueType. Windows incluye las siguientes fuentes que no son TrueType:

- ➢ MS Sans Serif (parecida a la Arial)
- ➢ Courier
- ➢ MS Serif (similar a la Times Roman)

Estas fuentes se ajustan de manera manual para resoluciones de pantalla particulares, y pueden utilizarse en los formularios, pero evítelas si necesita imprimir formularios a alta resolución. El motivo de esto es que las fuentes que no son TrueType presentan irregularidades al ser impresas. Las irregularidades de trazo son ángulos visibles donde la resolución del dispositivo de salida no es suficiente para ocultar la existencia de pixeles. En impresoras láser, este efecto no se elimina excepto en impresiones de muy alta calidad, pero cuando se utilizan fuentes que no son TrueType, aún puede existir porque las fuentes están diseñadas para resoluciones de pantalla particulares. Así, la letra P puede ser de 28 x 24 pixeles en la pantalla, pero el mismo tamaño impreso puede traducirse como 140 x 120 pixeles. En lugar de interpolar líneas suaves, la impresora simplemente traduce cada pixel en bloques de 50 x 50 pixeles, presentando curvas escalonadas visibles y otros efectos irregulares.

También es importante elegir fuentes estándar para una aplicación, e intentar quedarse con ellas durante toda la aplicación. Quizá pueda utilizar un tipo secundario para los títulos o encabezados, y una fuente primaria con variedad de

tamaños dentro de la aplicación. Las interfaces pueden ser confusas si se utilizan demasiadas variaciones de fuentes.

* FontOutline

Esta propiedad determina si se presenta una fuente con forma de contorno. Las fuentes de contorno exhiben todas las demás propiedades, tales como negritas o cursivas, pero sólo muestran una línea trazada alrededor de su perímetro, mientras que el cuerpo de la fuente es hueco.

* FontShadow

Esta propiedad determina si aparece una sombra en la parte inferior a la derecha de una fuente. Esto es útil cuando los colores de la fuente y el fondo no proveen un contraste suficiente para que la fuente sea fácilmente legible.

* FontSize

Ésta determina el tamaño del punto de la fuente. Utiliza tamaños de entre 8 y 12 puntos para los elementos principales de la pantalla y los controles, ascendiendo hasta tamaños de 14 y mayores para títulos o textos importantes que requieren de mucho énfasis. Puede utilizar fuentes más pequeñas, pero no es recomendable excepto para claves redundantes o textos destinados a ser ignorados en gran parte.

* FontStrikeThru

Esta propiedad determina si se dibuja una línea recta horizontal a través del texto. Generalmente se utiliza para exhibir textos viejos cuando se ha hecho una revisión.

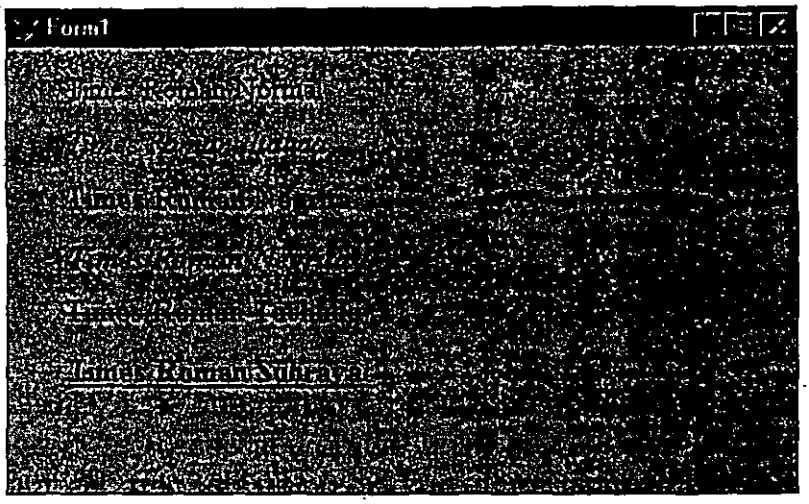
* FontUnderline

Esta propiedad determina si se dibuja una línea recta horizontal bajo el texto. Se utiliza para dar énfasis, aunque puede utilizarse para indicar encabezados de columnas para tablas que se presenten con etiquetas o cuadros de texto, al contrario de los controles de cuadrículas. La figura 12-11 muestra diversas combinaciones de configuraciones de fuentes.

ForeColor

La propiedad ForeColor fija el color del primer plano del objeto. En la mayoría de los controles, éste es el color del encabezado. En los formularios, es el color del contorno (el color del interior se fija con la propiedad FillColor).

Figura 12-11



Ejemplos de fuente Times Roman (o cualquier otra) con diversas combinaciones de configuraciones de fuentes.

Aunque BackColor debe fijarse en un parámetro numérico, la manera más fácil de fijarla a colores inteligibles es utilizar la función RGB( ) para fijar los componentes relativos de rojo, verde y azul. Para un ejemplo de cómo realizar esto, vea BackColor en este capítulo. Para fijar el color del texto de un botón de comando, utilice lo siguiente:

```
<Objeto>.ForeColor = RGB(128,0,0)     && Establecer la propiedad Caption a
                                      && rojo obscuro
```

Height

La propiedad Height determina la altura de un control. Si usted cambia esta propiedad mientras está ejecutando el formulario, el control cambiará su tamaño de forma dinámica. El borde superior del control se queda en su sitio a menos que usted cambie la propiedad Top. Las propiedades Left y Width controlan la posición horizontal y la dimensión.

HelpContextID

La propiedad HelpContextID es una herramienta para proveer ayuda sensible al contexto para sus aplicaciones. La ayuda sensible al contexto significa sencillamente que cuando usted presiona F1, no solamente aparece el archivo de

ayuda, sino que señala algo útil. Usted realiza esto proporcionando a tantos campos y controles como sea posible una identificación de contexto en el archivo de ayuda. Cuando usted escriba el archivo de ayuda, ponga rótulos de identificación de contexto en diferentes lugares del archivo, de manera que si se presiona F1 mientras el usuario está en, por ejemplo, el campo Tasa de interés, Visual FoxPro pueda hacer que corresponda la identificación de contexto para el campo Tasa de interés con la identificación de contexto en el archivo de ayuda y lo despliegue exactamente en el lugar adecuado.

Las identificaciones de contexto de ayuda no tienen que ser exclusivas de la aplicación Visual FoxPro en la que están siendo utilizadas. Campos múltiples pueden apuntar al mismo lugar en un archivo de ayuda, pero las identificaciones de contexto en un archivo de ayuda deben ser únicas.

Usted establece el archivo de ayuda para una aplicación con el comando SET HELP. Una vez hecho esto, aparece de manera automática cuando el usuario oprime F1, y si el control enfocado tiene un HelpContextID, ese tópico será desplegado en el archivo de ayuda.

Left

La propiedad Left determina la posición del control relativa al borde izquierdo del formulario. Si usted cambia esta propiedad mientras está ejecutando el formulario, el control se moverá inmediatamente. La posición vertical se controla vía la propiedad Top, y las propiedades Height y Width controlan el tamaño.

ListIndex

Esta propiedad establece el número de filas seleccionadas en un cuadro de lista.

MousePointer

La propiedad MousePointer le permite establecer el icono del cursor del ratón mientras está sobre un objeto. Adiverta que esto no lo cambia por un objeto que esté enfocado. Si el ratón pasa sobre un objeto con MousePointer establecido en 11, será un reloj de arena mientras esté sobre el objeto. Tan pronto como se mueve hacia un objeto distinto, vuelve al icono predeterminado. Los tipos de iconos que pueden establecerse utilizando esta propiedad son:

- ➢ Default
- ➢ Arrow
- ➢ Crosshairs
- ➢ I-beam
- ➢ Icon
- ➢ Size NSEW
- ➢ Size NE SW
- ➢ Size NS
- ➢ Size NW SE
- ➢ Size WE
- ➢ Up arrow
- ➢ Houtglass
- ➢ Nodrop

Un buen ejemplo de cuándo utilizar esta propiedad está en conjunción con arrastrar y soltar. Por ejemplo, si existen cuatro cuadros de texto y un cuadro de lista en un formulario, y usted desea poder arrastrar un elemento desde el cuadro de lista a sólo uno de los cuadros de texto, es tan sencillo como seguir el ejemplo dado para el evento DragDrop anteriormente en este capítulo; pero, ¿qué sucederá cuando los usuarios arrastren un elemento sobre uno de los otros cuadros de texto? Nada. No se les ha dado indicación de que no deban soltar el objeto cuando es ilegal. Sería bueno que el cursor del ratón cambiara a un icono que indique que sería una mala idea soltar el botón del ratón, y luego volverlo a cambiar cuando sea legal soltar el objeto.

Para realizar esto, establezca en 13 la propiedad MousePointer de todos los objetos en la pantalla, excepto aquel aceptable, tan pronto como se inicie el arrastre, y luego restablézcalos cuando se haya completado. Haga esto añadiendo unas cuantas líneas al evento MouseDown del objeto arrastrado:

```
THISFORM.SetAll('MousePointer',13)
THISFORM.txtEstadoID.MousePointer = 0
THIS.Drag
```

Luego modifique el código MouseUp para volver a establecer estas propiedades cuando se haya completado:

```
THISFORM.SetAll('MousePointer',0)
THIS.Drag(2)
```

Ahora, mientras se arrastra el objeto por la pantalla, muestra un icono que disuade a los usuarios de soltarlo cuando está sobre objetos que no están preparados para aceptarlo.

Name

La propiedad Name contiene el nombre de un objeto, el cual se utiliza para manipular el objeto mediante el código. Usted podría hacer algunas cosas muy interesantes manipulando nombres de objetos durante el tiempo de ejecución, pero también podría causar mucho daño. El uso más común de esta propiedad es identificar objetos. Por ejemplo, si se arrastra y suelta un objeto, usted podría identificarlo utilizando el parámetro oSource del evento Drop utilizando:

```
oSource.Nombre
```

RowSource

Esta propiedad controla la fuente de datos para un cuadro combinado o un cuadro de lista.

RowSourceType

La propiedad RowSourceType fija el tipo de la fuente de datos para el cuadro combinado o el cuadro de lista. Puede elegir de los siguientes:

0	Ninguno
1	Valor
2	Alias
3	Instrucción SQL
4	Consulta QPR
5	Arreglo o matriz
6	Campos

7 Archivos
8 Estructura
9 Emergente

SpecialEffect

La propiedad SpecialEffect controla el aspecto de la mayoría de los controles, lo que puede quitarles su glamour tridimensional y exhibirlos de diversas formas (ver figura12-12). Advierta que ésta es distinta de la propiedad Style, la cual determina más acerca del comportamiento que acerca de la forma en que se muestra un objeto (aunque también cambia su aspecto).

TabIndex

La propiedad TabIndex determina el orden de tabulación de un objeto en una lista. Si se cambia mediante el código, asegúrese de hacerlo con todos los demás objetos en la página o en el formulario, o Visual FoxPro los reordenará basándose en los ordenamientos internos (el orden en que fueron añadidos o modificados últimamente), lo que puede producir resultados inesperados.

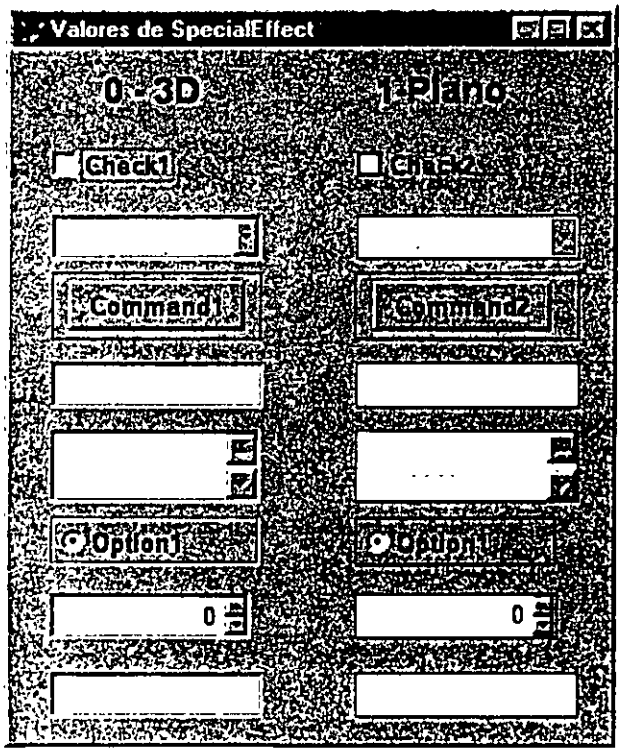


Figura 12-12

Ejemplo de 3-D versus efectos especiales sencillos en diversos controles del formulario.

TabStop

La propiedad TabStop determina si el objeto está de hecho en la lista del tabulador. Un objeto puede ser seleccionable, ya sea por el ratón o mediante código, pero no estar en la lista del tabulador. Para eliminar un control de la lista de tabulador, establezca la propiedad TabStop en falso (.F.).

ToolTipText

La propiedad ToolTipText contiene una descripción muy breve del texto del objeto. Debe utilizarse en botones que usan iconos u otros objetos representados gráficamente cuyo propósito podría ser vago para el usuario. Esto significa que un botón de comando Guardar no debería dejar que su texto de ayuda de herramientas se lea "guardar". De manera similar, los cuadros de texto y los cuadros de edición no deberían tener texto de ayuda de herramientas. Los usuarios dejarán descansar el ratón con frecuencia sobre un cuadro de texto preparándose a escribir en él, y esperar a que se despliegue un recuadro amarillo para decirles que deberían introducir el apellido del cliente puede ser muy molesto. Existen muchas capas de claves para el usuario, desde claves contextuales hasta etiquetas obvias, texto de barra de estado, texto de ayuda de herramientas para ayudar a los archivos vinculados por HelpContextID. ToolTipText es específicamente para cuando el contexto es insuficiente para proveer al usuario de claves en cuanto a las funciones de control. Esto significa usualmente barras de herramientas.

Para que ToolTipText sea exhibida, el formulario debe tener su propiedad ShowTips establecida en verdadero (.T.).

Top

La propiedad Top determina la posición del control relativa al borde superior del formulario. Si usted cambia esta propiedad mientras el formulario está corriendo, el control se moverá inmediatamente. La posición horizontal se controla vía la propiedad Top, y las propiedades Height y Width controlan el tamaño.

Value

La propiedad Value contiene los datos del control. Si es un cuadro de texto o un cuadro de edición, el valor es el texto del cuadro. Si es una casilla de verificación o un botón de opción, el valor es un cero si está desactivado y un uno si está activado. Si es un control numérico, el valor es el número al que esté establecido el control numérico. Si es un cuadro de lista, el valor es el índice del objeto de la lista seleccionado actualmente.

Visible

La propiedad Visible determina si un objeto es visible en el formulario o no. Esta propiedad puede establecerse directamente o a través de los métodos Show y Hide. Si un objeto es invisible, no es seleccionable. Se saltará su parada de tabulador y fallarán los intentos por programa de seleccionarlo mediante el código, como si el código del evento hubiera regresado falso.

Width

La propiedad Width determina el ancho del control. Si usted cambia esta propiedad mientras está ejecutando el formulario, el control cambiará su tamaño dinámicamente. El borde izquierdo del control permanece en su lugar a menos que usted cambie la propiedad Left. Las propiedades Top y Height controlan la dimensión vertical.

Controles OCX

Una de las grandes historias exitosas de la programación de Windows ha sido el uso de controles VBX en Visual Basic. Diseñados originalmente como una "puerta de servicio" para permitir a los programadores añadir controles nuevos en caso de que el producto no contuviera todo lo que los programadores pudieran desear, se multiplicó quizá como el mercado más dinámico en el entorno de la programación. Pero la aproximación VBX no estaba creada técnicamente para una expansión futura. La respuesta de Microsoft fue el modelo OCX. Afortunadamente para nosotros, Visual FoxPro apoya por completo los OCX.

Se comportan justamente como controles, utilizando propiedades y métodos y respondiendo a eventos. Usted puede registrarlos y seleccionarlos utilizando el icono seleccionador de la biblioteca de la clase, de la barra de herramientas Controles de formulario, y conectándose a sus aplicaciones con un clic de ratón.

Los controles OCX constituyen un mercado creciente de objetos genéricos, dan una variedad infinita a la gama de posibilidades. Se espera ver OCX que sean aún más capaces que muchas de las aplicaciones que se escriben en el curso normal de la programación. Por ejemplo, existen controles de hojas de cálculo que están virtualmente auto-contenidos; usted sencillamente los envuelve con una ventana y provee una fuente de datos. Sin lugar a dudas, controles OCX nuevos crearán oportunidades de mercado aún imprevistas. Los OCX son herramientas que todos los programadores profesionales de Visual FoxPro deberían tomar en cuenta.

El único control OCX que se ha utilizado en aplicaciones es el control de contorno (Outline), que funciona muy bien. Sólo tenga en cuenta que el nivel de tabulación del primer objeto es siempre cero, y que hay un error en la documentación del método Expand. Debe introducir la palabra *Object* y un punto antes de *Expand*. Gracias a Tom Retting por no ser demasiado tímido como para preguntar.

Conclusión

En este capítulo usted ha visto los eventos, métodos y propiedades más comunes y útiles de los controles estándar de Visual FoxPro. También ha visto ejemplos de cómo se utilizan, y en ocasiones de cómo no deben utilizarse.

Los eventos, métodos y propiedades descritos aquí dan razón de más del 80 por ciento de las combinaciones existentes con todos los objetos, y probablemente más del 95 por ciento de su uso real. Domine éstos, y luego utilice los eventos, métodos y propiedades restantes para las funciones de los objetos menos comunes y más especializados.

En el siguiente capítulo, se verá cómo crear objetos usuales para combinar las funciones de controles estándares individuales y cómo crear objetos mucho más útiles. También se verá el Generador de clases, y cómo diseñar una clase termómetro para indicar de manera dinámica el estado de un proceso que consume tiempo.

13

Clases y el Generador de clases

CAPITULO 13

¿SABE cómo conducir un Toyota Celica 1995? ¿Cómo lo sabe? La razón por la que puede estar razonablemente seguro de que puede conducir un automóvil que nunca antes ha conducido es porque todos los automóviles se manejan en una forma muy similar. Está familiarizado con la clase de objetos llamados automóviles. Tienen muchas propiedades en común: volantes, pedales de acelerador y freno, interruptores de encendido, velocímetros y espejos retrovisores.

Tal vez se haya preguntado, "¿Tiene transmisión automática o manual?". Si considera que los automóviles son automóviles, pero que algunos tienen transmisión automática y otros transmisión estándar, ya comprende las clases y la herencia. Todos los automóviles tienen muchas características en común, pero hay dos subclases de automóviles muy distintas que tienen características adicionales en común, sobre y por debajo de sus similitudes con todos los automóviles. Estas subclases son automóviles con transmisión automática y automóviles con transmisión manual. Éste es un ejemplo de *subclases*.

Algunos modelos agregan al modelo base algunas características como control de navegación y ventanas eléctricas. Pero todas las características estándar siguen siendo las mismas y siguen funcionando de la misma manera. En otras palabras, el nuevo modelo hereda (o deriva de) todas las características de la clase base y agrega algunas propias.

Visual FoxPro respalda el uso de clases y subclases de una manera análoga. Puede crear objetos, como formularios, controles o grupos de controles, y luego usarlos como la base para otros objetos. Sólo debe señalarlos en sus aplicaciones y éstas los usarán. Si los cambia, sus aplicaciones utilizarán las versiones modificadas automáticamente. Así, no tiene que pasar por todos sus programas para buscar copias de su código original. Se ha tomado todo en cuenta.

Las clases se ven justo como los formularios y los controles para formularios. El Generador de clases se ve exactamente como el Generador de formularios. La única diferencia es el lugar en que se almacenan los objetos que crea (como archivos .VCX) y cómo los emplea Visual FoxPro. De modo que ya sabe la mayor parte de lo que necesita saber para usar clases. Sólo debe pensar un poco diferente.

No es necesario que emplee clases en Visual FoxPro. En tanto que el modelo de evento y el modelo de objeto son inseparables del lenguaje, las clases son opcionales. De manera que si desea aprender la cantidad mínima de información necesaria para usar Visual FoxPro, puede omitir este capítulo. Pero si quiere

ıhorrarse mucho trabajo y ser mucho más eficiente, las clases son lo que usted ıecesita.

Clases en Visual FoxPro

En Visual FoxPro se diseña una clase de modo muy similar a como se diseña un formulario. No obstante, una vez que se guarda como clase, se puede seleccionar de la barra de herramientas Controles de formularios justo como se seleccionaría, por ejemplo, una casilla de verificación. El establecimiento de subclases consiste en entrar al Generador de clases, seleccionar una clase guardada con anterioridad del mismo modo que se seleccionaría un control de la barra de herramientas del Generador de formularios, hacer mejoras y modificaciones, y luego guardarlas en la clase nueva. La clase nueva es una subclase de la clase en que ésta se basa.

Cuando se crean subclases, heredan todo el comportamiento de sus padres, lo cual implica todo el código de evento, código de método y propiedades del padre. Entonces puede agregar o cambiar un código de método en la subclase. Este establecimiento de clases puede ser indefinido, pero los costos pronto superan los beneficios si las clases se subdividen demasiado.

Cuando crea una clase, ésta se almacena en una biblioteca de clases, que es un par de archivos .VCX/VCT. Cuando registra esa biblioteca de clases, Visual FoxPro la agrega a la lista de bibliotecas de clases de donde puede seleccionarla en el tiempo de diseño o también puede utilizar la página Herramientas, Opciones, Controles para registrar bibliotecas de clases, haciendo, por lo tanto, que estén disponibles para su selección, o agregarlas conforme las necesita.

Puede tener varias bibliotecas de clases registradas y seleccionar de entre ellas conforme sea necesario. Siempre que está en el Generador de formularios, puede ir a la barra de herramientas Controles de formularios y hacer clic en el icono de biblioteca de clases (el pequeño librero) y escoger una de las bibliotecas de clases registradas.

Sólo una de las bibliotecas de clases disponibles es la actual. Sus clases aparecen como iconos en la barra de herramientas Controles de formularios; la misma que utiliza para poner casillas de verificación y botones de comandos en sus formularios. (La biblioteca de clases estándar es la que normalmente ve cuando genera un formulario.) Para cada clase definida por el usuario, aparece un icono de FoxPro que refleja la clase base de la que se derivó su clase y que puede reemplazar con su propio icono. Puede agregar su propia clase definida por el

usuario a su formulario al hacer clic en el icono que representa la clase asociada y soltarlo en su formulario.

Es momento para un nuevo término, *instanciación*. Una instancia de su objeto de clase se agrega a su formulario. Una instancia no es una copia; se asemeja más a un apuntador para la clase. Una copia sería como una fotografía instantánea de la clase original. Una instancia es un respaldo de referencia para la clase en la biblioteca de clases nombrada. Es la razón de por qué cuando cambia la clase en su biblioteca de clases, los cambios se reflejan en cualquier formulario que contenga una instancia de esa clase. Visual FoxPro realiza una instancia de la biblioteca de clases nombrada en el tiempo de ejecución y copia cualquier cosa que se encuentre ahí en el momento en que se hace la instanciación. De modo que si mañana cambia la clase, la instanciación de la clase de mañana será diferente de la de hoy.

Suponga, por ejemplo, que crea una clase barra de termómetro, que despliega una gráfica de barras con movimiento conforme un proceso envía su actualización de acuerdo con su progreso. Entonces un día tiene una lluvia de ideas. ¿Qué tal un despliegue de estado que ocupe toda la pantalla desde la parte superior como una bañera que se llena con agua, en vez de que se llene de un lado a otro?

Puesto que diseñó su control de termómetro como una clase, tiene pocas opciones. ¿Quiere que su nueva clase dé su funcionalidad a todos los usos actuales de la misma? ¿O su clase nueva es una versión especializada de la clase de termómetro que desea que tenga pocas aplicaciones? En este caso, quiere compartir la funcionalidad con todos los usos actuales. Modificará la subclase en vez de hacer una subclase de ésta.

Seleccione Archivo/Abrir del menú de Visual FoxPro, luego seleccione MICLASE.VCX (o cualquier biblioteca de clases que esté almacenada) y edite su clase termómetro. Haga los cambios necesarios, agregando una propiedad Type, de modo que pueda seleccionar ya sea llenados verticales u horizontales y guarde la clase con el nuevo llenado vertical como el valor predeterminado. Entonces ejecute una aplicación que use la clase termómetro.

¡Eso es todo! Todas las barras de termómetro ya cambiaron y no se requiere ninguna programación adicional. Copie la clase nueva al directorio de proyectos en sus máquinas para usuarios y le llamarán genio una vez más.

Esto es tan fácil porque las clases funcionan como cajas negras. Una vez instaladas, sus aplicaciones no pueden cambiar la apariencia del termómetro por medio de la propiedad Type porque esa propiedad no existía cuando se utilizó la

clase por vez primera, pero toda la nueva funcionalidad está ahí sin tener que recodificar nada, ni cortar o pegar el código nuevo.

Si hubiera establecido una subclase del objeto, habría tenido el mismo efecto neto, excepto que usted habría retenido la versión más simple del termómetro, tal vez para establecer una vez más su subclase sin incluir la nueva capacidad de llenado vertical. Quizá quiera cambiarlo a un despliegue interactivo de uso de memoria y agregarle un control de temporización y capacidad de actualización interna. Es probable que no desee el despliegue con llenado vertical porque no tiene una resolución lo suficientemente fina para este tipo de funcionalidad.

La razón por la que todo esto surgió es que la programación estructurada, el último paradigma, no satisfacía las necesidades de los programadores. La programación estructurada recomendaba utilizar tantos procedimientos y funciones diferentes como fuera necesario para dividir todas las tareas en sus subtareas componentes. En el tiempo de compilación, se integraban todas las funciones. Si cada función era limpia, no afectaba nada más, y se limpiaba después por sí misma, ¿qué podía estar mal?

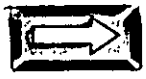
Se suponía que la clave era el establecimiento de estándares de programación. He oído sobre compañías que tratan de poner en práctica reglas que rigen el número máximo de líneas de código que un procedimiento puede tener, los anchos de la tabulación estándar, nomenclaturas de funciones y variables, y casi todos los aspectos del estilo de programación a fin de poner en vigor la modularidad. Se suponía que se debía crear un código con funcionalidad genérica que haría del mantenimiento del programa una tarea fácil.

Se oía bien, pero resulta que la programación modular estructurada era una teoría que no efectuó la transición a la práctica. El código nunca puede ser lo suficientemente genérico para emplearlo en todas partes. Hay una compensación constante ya sea que "clone o modifique" para cada situación ligeramente distinta o, si en realidad es dogmático, utilice una instrucción CASE para cada variable, que con el tiempo produce el código, el cual es mucho más difícil de mantener que su idea original, simple y elegante. Y si se descubrió un defecto fundamental en el código original o se necesita hacer un cambio que afecte todos los casos, tiene que hacer el seguimiento de todos los clones y hacerles los mismos cambios a todos.

Las bibliotecas de clases son la solución. Ponen en práctica la modularidad y reducen los gastos generales de administración de códigos. Si necesita agregar funcionalidad específica para un proyecto, puede establecer una subclase para éste. Entonces, al agregar funcionalidad a la clase base, la subclase la hereda en forma automática y no tiene que seleccionar el código, recortar y pegar todos los

cambios en sus ocho versiones clonadas. Las clases eliminan muchas de las inconveniencias de mantener la modularidad y, por tanto, fomentar su práctica.

Es evidente que aún puede anular todas las ventajas de las bibliotecas de clases al establecer demasiadas subclases. Si establece subclases siempre que piensa en una mejora para la clase, entonces termina por tener diez mejoras buenas, pero en diez espacios diferentes. El establecimiento de subclases debe efectuarse cuando varias adiciones o mejoras son mutuamente exclusivas, o cuando se agrega funcionalidad que usted desea de manera específica se excluya de otros usos de la clase.



Cree una clase

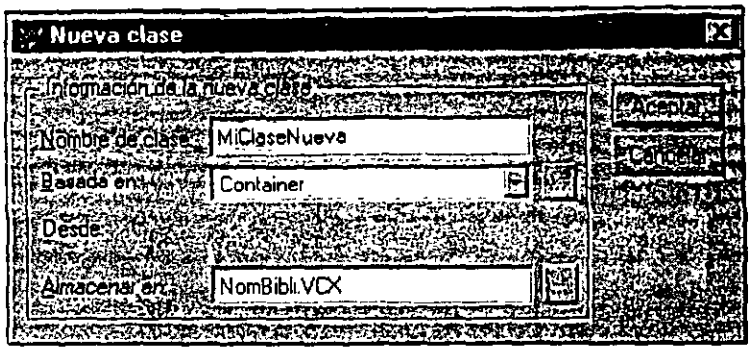
Hay cinco formas de crear una clase nueva:

- Seleccione Archivo/Abrir del menú. Haga clic en el botón de opción Clase y en el botón Nuevo.
- Use el comando de clase CREATE CLASS clase en la ventana de comandos.
- Haga clic en el icono Nuevo en la barra de herramientas.
- Use MODIFY CLASS clase OF biblioteca de clases en la ventana de comandos.
- Desde el Generador de formularios, ya sea sin control, con un solo control, o con controles múltiples seleccionados, seleccione Archivo/Guardar como clase del menú.

Las tres primeras de estas opciones hacen que aparezca el cuadro de diálogo Clase nueva, que se muestra en la figura 13-1. En este cuadro de diálogo, capture un nombre para su clase y la clase o el control en que se basa. Para crear una clase no visual, seleccione Personalizada de la lista desplegable. A fin de basar su clase en otra que ya creó, haga clic en el botón de puntos suspensivos (...) y seleccione la biblioteca de clases y la clase. Lo último que necesita indicar a Visual FoxPro es dónde desea almacenar su clase. Éste es el sitio en que selecciona o crea una biblioteca de clases.

Si selecciona Guardar como clase mientras está en el Generador de formularios, aparecerá el cuadro de diálogo Guardar clase (ver figura 13-2). En este cuadro de diálogo, puede seleccionar en qué conjunto de controles basa su clase, desde un

Figura 13-1



Cuadro de diálogo Nueva clase.

solo control hasta un conjunto entero de formularios. Luego dé un nombre a su clase, seleccione una biblioteca de clases para poner y capturar una breve descripción de texto de la misma. Esta descripción se desplegará, entre otros lugares, en la parte superior de la ventana Propiedades cuando una instancia de la clase es el objeto seleccionado actualmente.

Una vez que ha creado una clase, puede seguir adelante y usarla en un formulario o pasar al Generador de clases y refinarla.

El banco de trabajo del Generador de clases

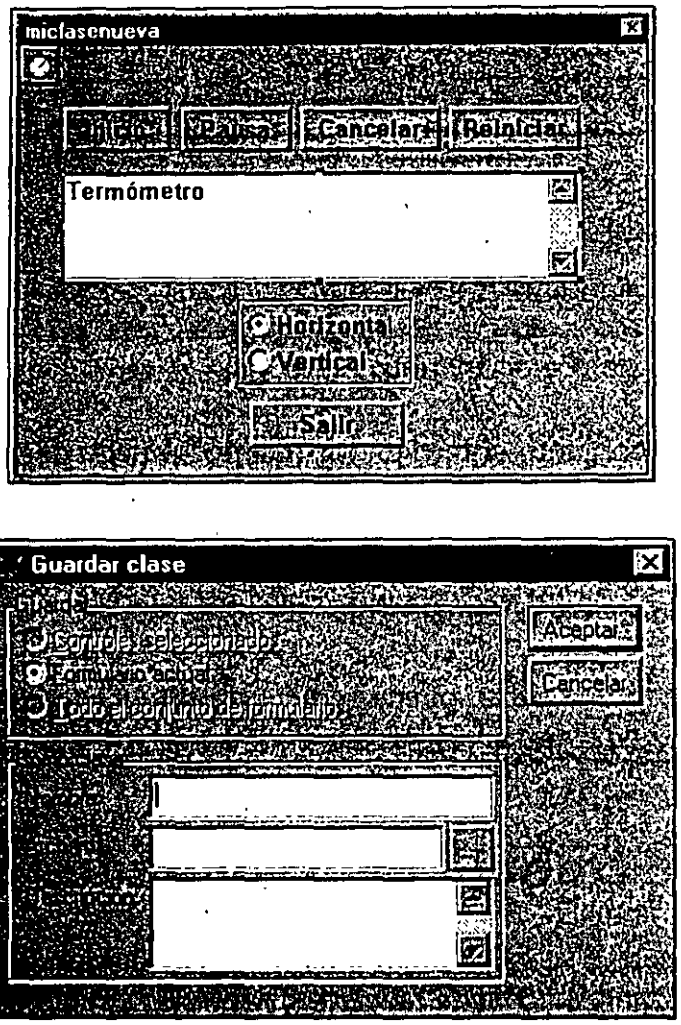
El Generador de clases se ve exactamente como el Generador de formularios y tiene la misma funcionalidad. Puede ubicar los controles en el área de diseño, moverlos, cambiar sus propiedades y agregar código a eventos. Una diferencia importante es que, mientras los formularios se almacenan en archivos con las extensiones .SCX y .SCT, las clases se almacenan en archivos .VCX y .VCT.

El menú Clase

El menú Clase contiene un subconjunto del menú Formulario con una opción adicional, Información de la clase... El cuadro de diálogo Información de la clase le permite seleccionar un icono para representar su clase, capturar una descripción, establecer el modo de escala y especificar cuáles de sus propiedades, métodos y eventos están protegidos. En la figura 13-3 se presenta el cuadro de diálogo Información de clase para una clase de ejemplo.

Como puede apreciar, se definió un icono y se hizo una descripción breve de la clase. Luego se diseñó la clase desde el borrador, incluyendo el icono. Si necesita que cualquiera de sus propiedades, métodos o eventos esté protegido, haga clic en

Figura 13-2



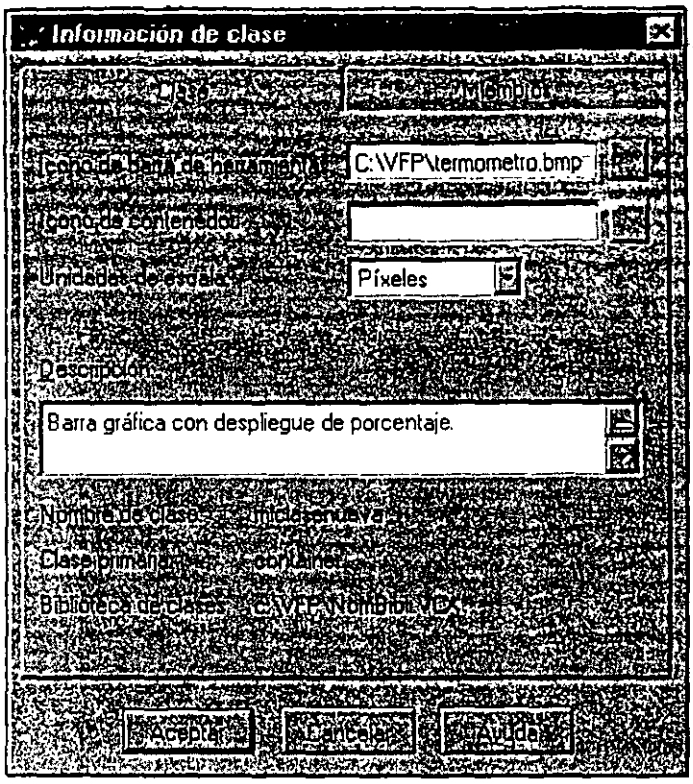
Cuadro de diálogo Guardar clase.

la ficha Miembros y en las casillas de verificación apropiadas. No se puede acceder a un miembro protegido cuando se crea una instancia de una clase. Si crea una propiedad llamada Contraseña, por ejemplo, no puede ir al Generador de formularios, hacer una instancia de la clase y luego examinar contrase.VALUE. De hecho, la única forma de conocer el valor de la propiedad es si usted proporciona un medio para accederlo. Por ejemplo, podría proporcionar un método Verify, que regresa verdadero (.T.) si su parámetro equivale al valor de la propiedad Contraseña:

```
PARAMETER cIntento
IF cIntento = THIS.cContra
   RETURN (.T.)
ELSE
   RETURN (.F.)
ENDIF
```

297

Figura 13-3



Ficha Clase del cuadro diálogo Información de clase.

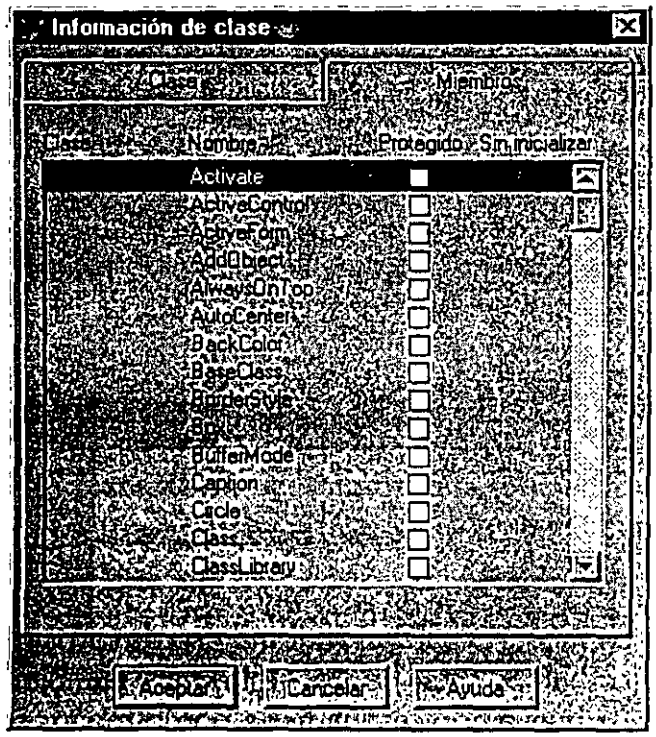
Podría emplear esta contraseña para permitir el uso de otros métodos sólo si una contraseña pasa en forma correcta, por ejemplo, para cambiar la contraseña. Lo siguiente verifica que la nueva contraseña tiene por lo menos seis caracteres de extensión:

```
PARAMETERS cContraAnte, cContraNueva
IF cContraAnte = THIS.cContra
  IF LEN (cContraNueva) >= 6
     THIS.cContra = cContraNueva
      RETURN(.T.)
  ELSE
      RETURN(.F.)
  ENDIF
ELSE
  RETURN(.F.)
ENDIF
```

La ficha Miembros de la clase de ejemplo se muestra en la figura 13-4. Los usuarios pueden cambiar la altura de un formulario o control al hacer clic y arrastrar. Si no protege las propieades Height y Width de sus clases, asegúrese de escribir su código para manejar cualquier cambio de tamaño de modo interactivo,

puesto que los usuarios de su clase podrían hacer esto. De manera similar, si el control del color es importante para la apariencia de su control, entonces proteja las propiedades del color. Si es necesario, puede proporcionar interfaces en el código alternativas para permitir su control interno.

Figura 13-4



Ficha Miembros del cuadro de diálogo Información de clase.

El menú contextual de la clase del botón secundario del ratón

Del mismo modo en que los menús Formulario y Clase comparten muchas características, los menús contextuales Formulario y Clase del botón secundario del ratón son muy similares. De nuevo, el menú contextual de la Clase del botón secundario del ratón es un subconjunto del menú contextual del botón secundario del ratón del formulario. El Entorno de datos, que se presenta en el menú contextual del botón secundario del ratón del formulario, falta en el Generador de clases.

Guarde clases desde el Generador de formularios

Para guardar una clase diseñada en el Generador de formularios, cree un conjunto de trabajo de controles en un formulario, luego seleccione el grupo de controles y guárdelos como una clase al escoger Guardar como clase desde el menú Archivo. Pueden eliminarse los errores de la clase por separado y ésta puede emplearse en cualquier otro formulario.

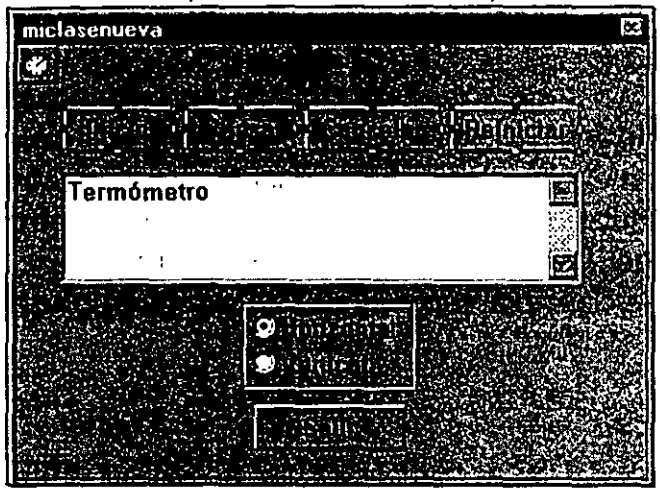
Por supuesto, sólo porque guardó los controles como una clase, no significa que se conviertan instantáneamente en una clase dentro de su formulario. Para usar su clase recién diseñada, suprima los controles y agregue una instancia de la clase que acaba de crear.

Cuando guarda un grupo de controles como una clase, aparece el cuadro de diálogo Guardar clase, como se muestra en la figura 13-2, permitiéndole capturar el nombre de la clase, la biblioteca de clases en que debe guardarse (o el nombre de una nueva biblioteca de clases) y una descripción. Otros atributos de la clase, como los iconos y la protección, pueden establecerse desde el Generador de clases.

Genere un control VCR

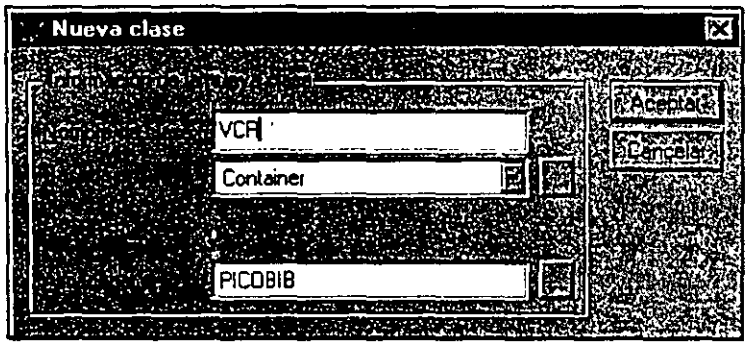
Uno de los conjuntos de controles más comunes que se usan en la programación de aplicación de bases de datos son los botones de VCR. Éstos son los botones que permiten a los usuarios controlar el apuntador de registro, yendo al primero o al último registro en la base de datos, o saltando uno hacia adelante y hacia atrás. Existen muchas representaciones visuales de estos botones, de modo que siéntase con libertad para cambiar sus versiones de los mapas de bits. Pero, en general, se ven como los símbolos en los botones correspondientes de su VCR (reproductora/grabadora de video), de ahí el nombre. Diseñará los suyos como un ejercicio, pero hay muchos archivos .BMP disponibles que simplemente puede copiar y usar. Un ejemplo de los que se diseñaron aparece en la figura 13-5. Ésta es la clase que diseñará paso a paso en esta sección.

Figura 13-5



Botones VCR.

Figura 13-6



Cuadro de diálogo Nueva clase.

Cree una clase

Para comenzar, teclee CREATE CLASS en la ventana Comandos. Aparecerá el cuadro de diálogo Nueva clase (ver figura 13-6). Tecle el nombre VCR, en qué clase se basa (es un contenedor para controles múltiples, de modo que seleccione contenedor) y en qué biblioteca de clases se va a almacenar. Escogimos Picobib, porque uno de nosotros tiene el sobrenombre Pico, pero puede seleccionar cualquier nombre que desee. Haga clic en el botón Aceptar y aparecerá la ventana Generador de clases con el título que indica que trabajan en PICOBIB.VCX (.VCR).

Defina la clase

El siguiente paso es crear varios botones de comando. Haga clic en el icono de comando Bloqueo de Generador en la barra de herramientas Generador de formularios, porque creará algunos controles y no quiere mantener un clic indefinido en el icono de control. Luego haga clic en el icono de botón de comando.

Es importante señalar que Visual FoxPro tiene un control de botón de comando llamado Grupo de comandos, que puede emplearse en este caso. Sin embargo, utilizar botones individuales en vez de un grupo de comandos tiene algunas ventajas. Los grupos de comandos requieren una instrucción CASE para determinar qué botón se oprimió. Cuando usted intente explicar cómo emplear el código para hacer que todo funcione, agregar una estructura DO CASE... ENDCASE en el código puede complicar la tarea. Asimismo, es más sencillo insertar otro control en un conjunto de botones de comando porque no se afecta ningún otro código del botón. No obstante, en gran medida es una cuestión de preferencia personal.

Una vez que seleccionó el icono Botón de comando, ponga cuatro botones en la ventana Generador de clases. Su ventana debe ser parecida a la que se presenta en la figura 13-7.

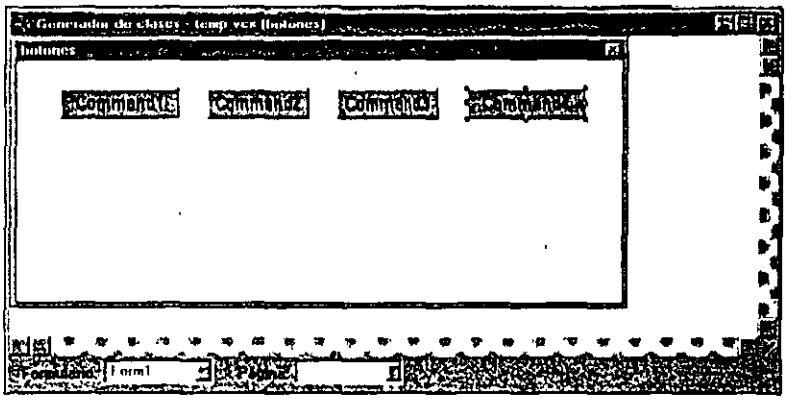


Figura 13-7

Cuatro botones de comando.

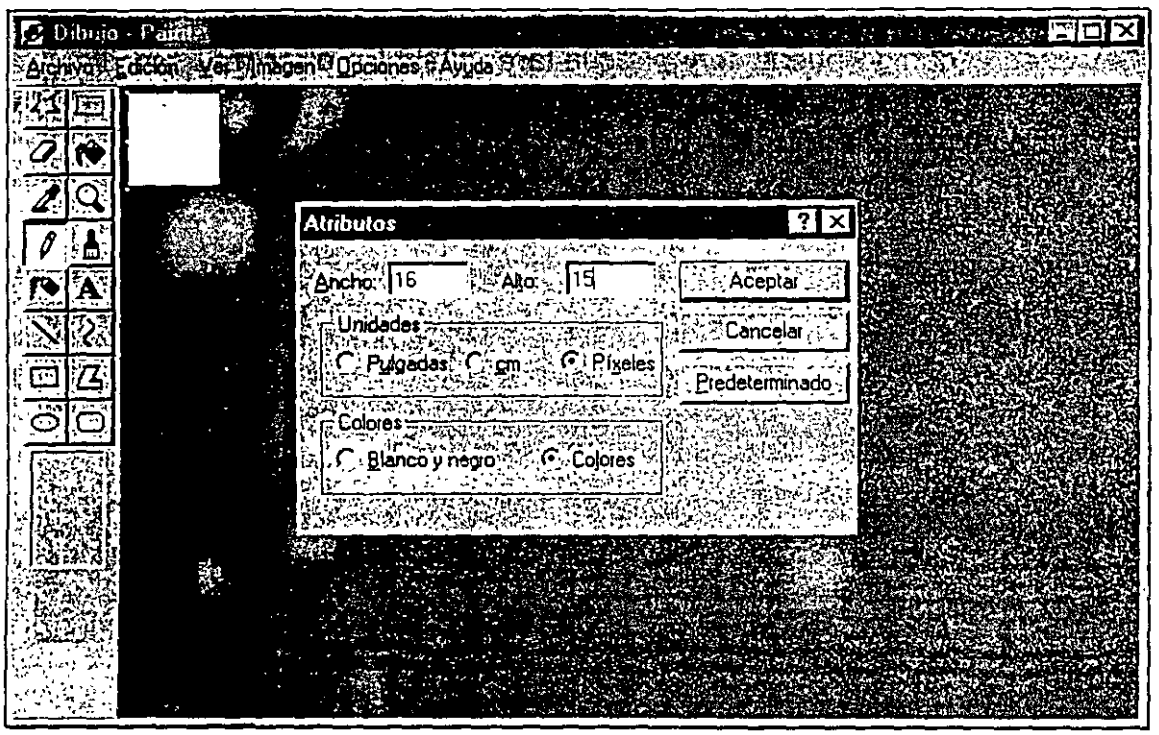
Me gusta usar nombres descriptivos para los controles. Puesto que es probable que cualquier código que escriba se refiera a sus controles por su nombre, es mejor nombrarlos antes de empezar a codificar. En este ejemplo, se nombraron los botones cmdPrimero, cmdAnterior, cmdSiguiente y cmdUltimo. Ahora realizará un ejercicio para diseñar sus propias gráficas de botón de comando.

Cree los archivos .BMP

Primero, abra la aplicación Paint (Paintbrush si tiene la versión 3x) de Windows (en su grupo Accesorios). Necesitará establecer el color del fondo en gris, de modo que haga clic en el recuadro de color gris con el botón secundario del ratón. También necesitará reducir el tamaño de la imagen, así que vaya a Opciones/ Atributos de imagen y haga clic en pixeles en el recuadro Unidades. (Pixel es la abreviatura del término *picture element:* elemento gráfico.) Luego introduzca un

ancho de 16 y una altura de 15; éstas son las dimensiones que se recomiendan para los botones de herramienta en la guía de diseño visual de Windows de Microsoft. El cuadro de diálogo del programa Paint debe verse como el de la figura 13-8.

Figura 13-8



Cuadro de diálogo Atributos de la imagen en Paint.

Cuando hace clic en Aceptar, la ventana de la imagen en realidad debe ser pequeña. La única forma de editar imágenes tan pequeñas es pixel por pixel. Para hacer esto, seleccione del menú Ver/Zoom para acercar y haga clic con el cursor dentro de cualquier parte en el área de la imagen gris. El área se acercará en una ventana de cuadrícula más grande, en la que puede pintar puntos negros con el botón primario del ratón y puntos grises con el botón secundario. Entonces se dibuja la imagen (ver figura 13-9).

Seleccione Ver/Zoom para alejar (para habilitar el menú), después Archivo/Guardar como. Guarde la imagen en su directorio de proyectos con algo descriptivo como DER15.BMP. Se utiliza DER15 a diferencia de DERECHO solamente, de modo que se pueden usar mapas de bits estándar de tamaños variables y volverlos a usar. El sufijo 15 expresa una altura de 15 pixeles.

Ahora repita los últimos pasos, comenzando con Vista/Zoom para acercar y cree los cuatro mapas de bits, asegurándose de guardarlos como IZQ15.BMP,

303

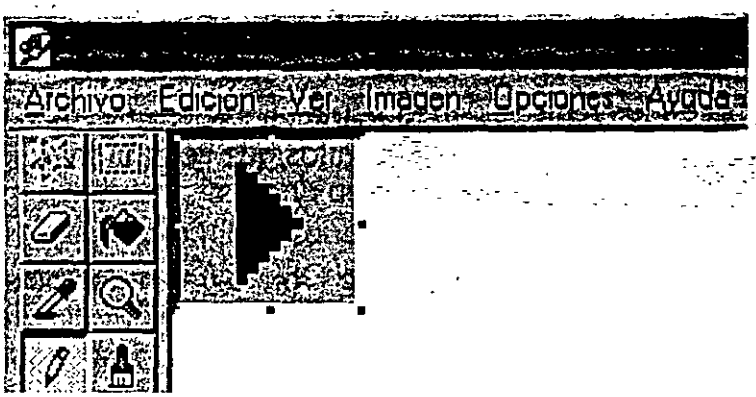


Figura 13-9

SIGUIENT.BMP

2IZQ15.BMP y 2DER15.BMP. El sufijo 2 significa que el control es una flecha doble.

Regrese a Visual FoxPro y seleccione el primer botón. En la ventana Propiedades, seleccione la propiedad Caption y anúlela; sólo resalte Command1 entero que es el título predeterminado y oprima el botón Suprimir. Haga lo mismo para cada uno de los otros tres botones.

Ahora regrese al primer botón y seleccione la propiedad Imagen. Teclee el nombre del archivo 2DER15.BMP. Debe ver un botón de gran apariencia con co que usted diseñó. Realice la misma operación para cada uno de los otros tres botones, usando los mapas de bits apropiados.

Programe la clase

Hasta el momento todo marcha bien, pero los botones nuevos aún no hacen nada. El paso siguiente es agregar el código. El código que se empleará quizá sea el código de programación más frecuente en el mundo de xBASE. Se usará para mover el apuntador de registro y luego volver a desplegar los datos en la pantalla usando los valores en lo que se ha convertido en el registro actual.

Cuando hace clic en el botón cmdPrimero, se abre el cuadro de código para el evento Click. Note que presionar REPÁG y AVPÁG hará un ciclo entre las ventanas del código a otros eventos para los que existe el código. Capture el código siguiente:

```
GO TOP
THISFORM.Refresh
```

Refresh es uno de los métodos integrados de Visual FoxPro, es necesario porque el solo hecho de que haya cambiado de registros no es suficiente para volver a desplegar los valores de campo de los registros en la pantalla. Tiene que mover los datos de los campos a los controles de cuadro de texto a los que se limitan. Eso es lo que hace la operación Refresh.

En la ventana de código, haga clic en la lista desplegable Objeto y verá los otros tres botones. Haga clic en ellos y capture el código apropiado en la ventana de código del evento Click.

```
IF NOT BOF()
  SKIP -1
  IF BOF()
      GO TOP
  ENDIF
ENDIF
THISFORM.Refresh

IF NOT BOF()
   SKIP -1
   IF BOF()
      GO TOP
  ENDIF
ENDIF
THISFORM.Refresh

GO BOTTOM
THISFORM.Refresh
```

Limpieza

Sólo queda un paso y es la limpieza de un bit. Tal vez haya realizado todo su diseño y codificación con los botones que aparecen en alguna parte en la mitad de la ventana. Si se hace una instancia en un formulario, el espacio vacío arriba y a la izquierda de los botones se duplicará exactamente. Necesita que el control (en este caso, por lo menos) encierre con firmeza los objetos contenidos. Agarre los botones, ya sea todos a la vez o en forma individual y arrástrelos a la esquina superior izquierda de la ventana de la clase, de modo que todos estén en línea cerca uno del otro, pero no haya espacio vacío entre ellos y ninguno a la izquierda del botón cmdPrimero.

Aquí es donde son útiles las herramientas de alineación en la barra de herramientas Distribución. Mientras mantiene oprimida la tecla de cambio a mayúsculas, haga clic de manera sucesiva en cada uno de los controles. Luego,

desde la barra de herramientas Distribución, haga clic en el tercer botón, Alinear
des superiores. En caso de ser necesario, repita el proceso con el octavo
trol, Misma altura.

Ahora reduzca la ventana de la clase a fin de que encierre exactamente los controles. Esto podría requerir ciertas operaciones matemáticas dado que visualmente es difícil hacerlo de modo correcto. En este caso, por ejemplo, los botones tienen justo 25 pixeles de altura, de manera que puede ir a la propiedad Height de la clase, introducir 26 y estará perfecto. Asimismo, si ve un botón individual, encontrará que su ancho es 25. Puesto que la cuadrícula predeterminada se establece para separar objetos en incrementos de 12 pixeles, sabe que se han colocado a 24 pixeles de distancia, de modo que el borde de cada uno se traslapa con el siguiente por un pixel (3 pixeles se traslapan en el grupo de 4). Esto implica que su ancho total es (25 x 4) - 3. Por tanto, establezca el ancho de la clase en ((25 x 4) - 3) + 1, o 98.

Cierre la ventana Generador de clases, guarde la clase y ya está preparado para usarla en un formulario.

Use su clase

Después de que ha creado una clase nueva, la única razón para suponer que funciona es su intuición como programador de que hizo todo lo correcto la primera vez. Pero todos sabemos que los programas rara vez son 100% perfectos la primera ocasión que se corren. Realice un formulario de muestra y pruebe los nuevos controles.

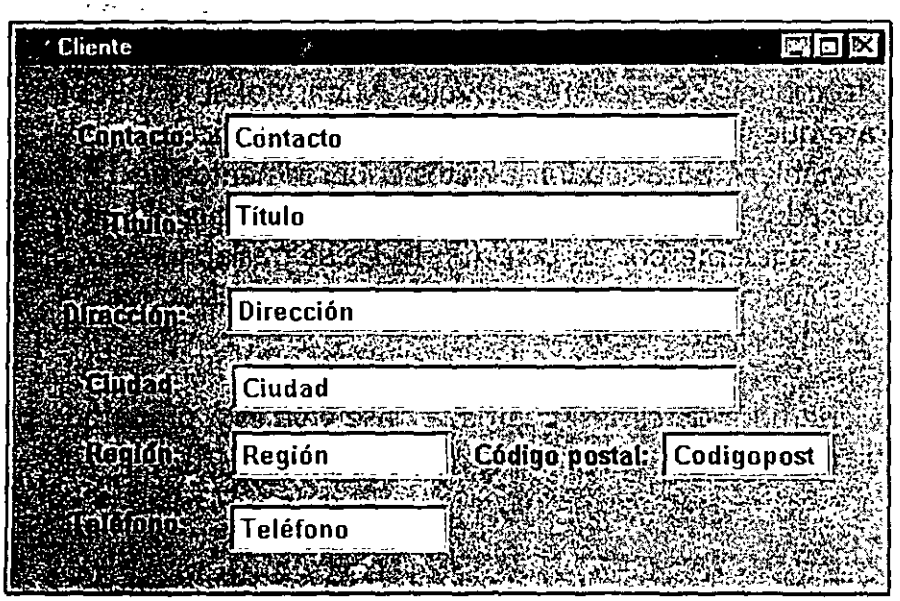
Cree el formulario

Para hacer una prueba simple del control VCR, necesita crear un formulario. Haga esto al teclear CREATE FORM PRUEBAVCR en la ventana Comandos. Aparecerá un nuevo formulario vacio.

Con objeto de simplificar la adición de campos en el formulario, agregue un nombre de tabla en el Entorno de datos Seleccione Ver, Entorno de datos, haga clic en cualquier parte en la ventana Entorno de datos con el botón secundario del ratón y escoja Agregar del menú contextual. Luego haga clic en el botón Otros para hacer que aparezca el cuadro de diálogo Abrir archivo. En este ejemplo, se

escogió CLIENTE en el subdirectorio SAMPLES/DATA en el directorio VFP. Desde el Entorno de datos, haga clic en el campo Contacto, luego arrástrelo sobre el formulario. Haga esto con los campos Título, Dirección, Ciudad, Región, Código Postal y Teléfono. En este punto, puede cerrar la ventana Entorno de datos. Su formulario debe verse como el que aparece en la figura 13-10.

Figura 13-10



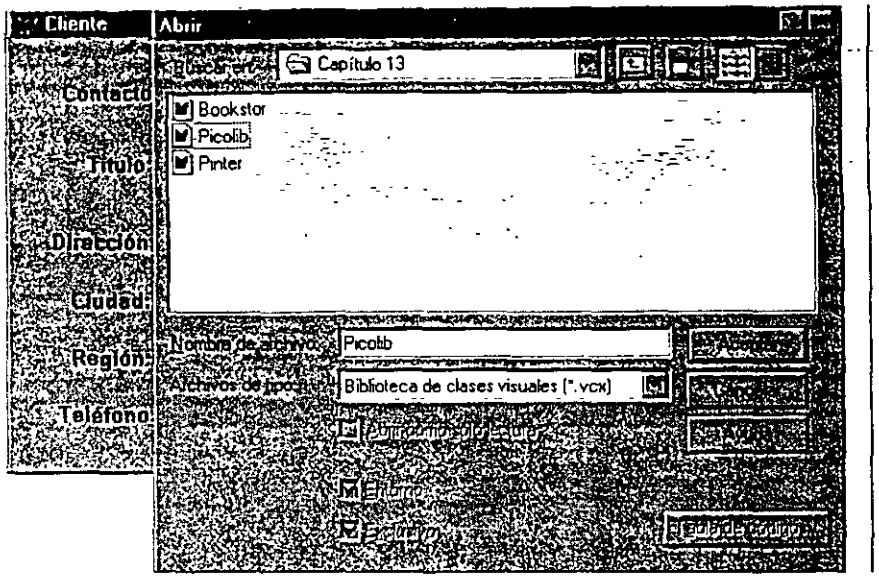
Formulario PruebaVCR: campos de datos.

Agregue la clase

Para agregar su clase, necesita hacer clic en el icono Ver clase en la barra de herramientas Controles de formularios. Esto hará que aparezca un menú. Haga clic en Agregar..., lo cual desplegará el cuadro de diálogo Abrir archivo, con la Biblioteca de clases visuales que se lista como el tipo de archivo predeterminado (ver figura 13-11). Seleccione la biblioteca de clases que acaba de usar para el control VCR. En este ejemplo, se puso en PICOBIB.VCX.

Una vez que ha hecho esto, puede seleccionar esta biblioteca de clases desde el icono Ver clases y hacer que aparezca una barra de herramientas que contiene todos los controles en su biblioteca de clases. Ya que aún no le ha asignado un icono, tendrá un icono de herramienta genérico. A fin de diferenciar entre los diversos controles de herramienta que podría tener en su biblioteca de clases, Visual FoxPro ofrece claves de herramientas que despliegan el nombre de cada control en la biblioteca de clases. Encuentre su control VCR (podría ser la única clase en su biblioteca hasta ahora) y haga clic en el icono.

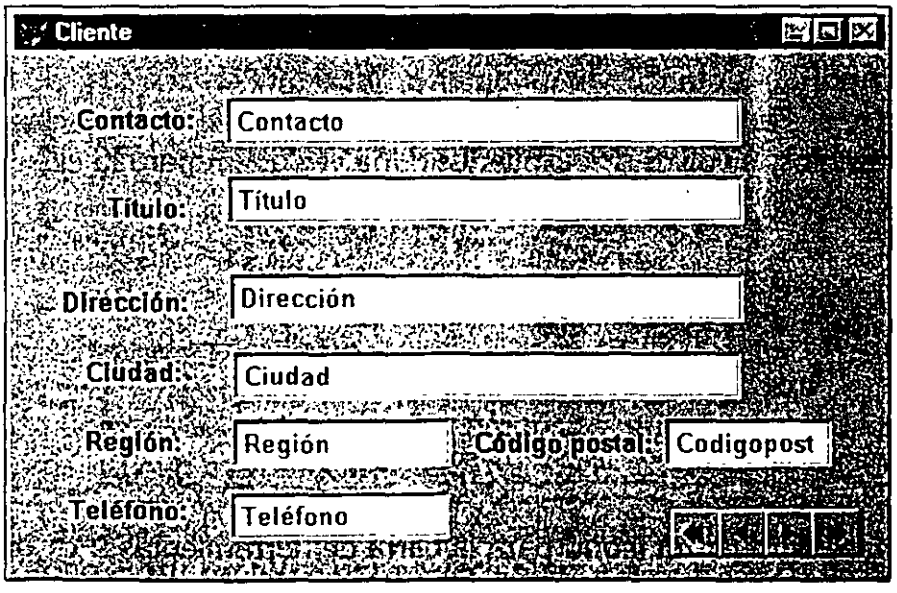
Figura 13-11



Agregue una biblioteca de clases a la barra de herramientas Controles de formularios.

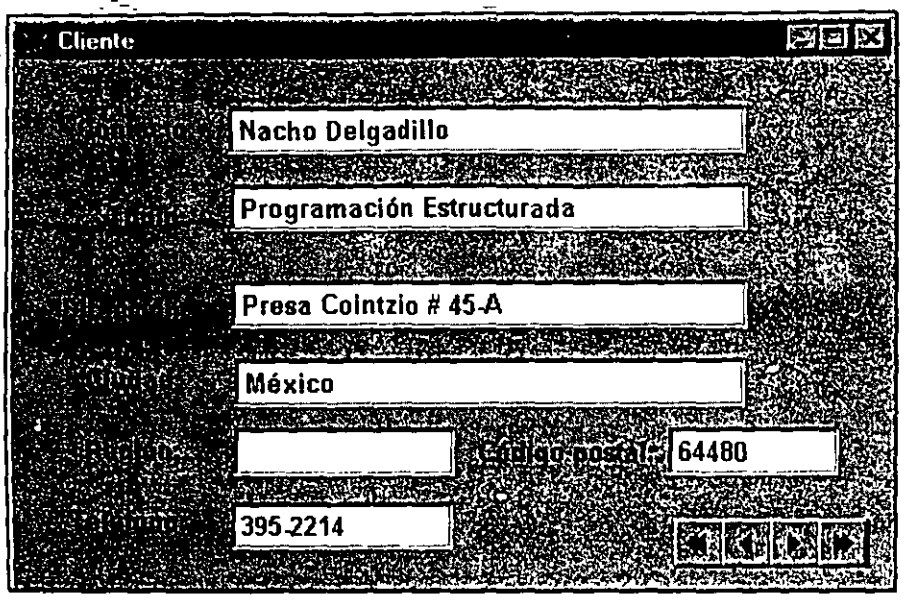
Ahora haga clic en el formulario en las áreas cercanas al lugar lógico para sus controles VCR y arrastre el apuntador del ratón hasta que cree un área por lo menos lo suficientemente grande para sus controles (ver figura 13-12). Como puede ver, crear un área más grande que el control no cambiará el tamaño de los controles. Por último, tome la esquina inferior derecha y reduzca el tamaño del área alrededor del control hasta que los controles llenen por completo el cuadro. Ejecute el formulario (ver figura 13-13), haciendo clic en cada uno de sus controles para verificar que en realidad funcionan.

Figura 13-12



Formulario PruebaVCR: campos de datos y controles VCR.

Figura 13-13



El formulario PruebaVCR en ejecución.

Eso es todo al respecto. Nunca tendrá que volver a escribir un código para un conjunto de botones VCR. Los tiene en su biblioteca de clases de uso y puede enviarlos a cualquier formulario en cinco segundos exactamente.

Modifique sus clases

Uno de los beneficios de usar clases, a diferencia de reutilizar el código de otras aplicaciones, es que tiene una ubicación central desde la cual puede hacer actualizaciones a su clase y todas las instancias existentes de la clase presentarán el nuevo comportamiento. Si repara un error incidente en la clase, no tiene que buscar entre sus aplicaciones para determinar cuáles utilizan esa clase, realizar los mismos cambios, y volver a compilarlos. Una vez que repara la clase, se arreglan las aplicaciones.

Edite la clase

Para modificar el control VCR, primero debe cargar el Generador de clases. Teclee MODIFY CLASS VCR OF PICOBIB (o como sea que la llame).

Tal vez desee agregar un icono para la barra de herramientas. Por omisión, el icono para cualquier clase nueva es un par de herramientas cruzadas; una llave de

tuercas y un martillo. Si agrega varias clases, todas tienen el mismo icono. Tiene
nover el ratón a la parte superior de cada botón de comando y esperar que el
ore aparezca para identificarlo. Asignar un icono a cada una de sus clases
hace que sea más fácil usarlas.

Cargue Pain o Paintbrush y cree la imagen de la figura 13-14, también como un mapa de bits de 16 pixeles de ancho y 15 pixeles de alto. Luego, en Visual FoxPro, seleccione Clase, Información de la clase del menú y haga clic en el botón de puntos suspensivos (...) para buscar el mapa de bits de su icono (ver figura 13-15). Asimismo, es probable que quiera agregar una descripción de la clase. Ocho caracteres no siempre son suficientes para describir por completo un objeto. El cuadro de diálogo Información de clase debe verse como el que se presenta en la figura 13-16 antes de que haga clic en Aceptar.

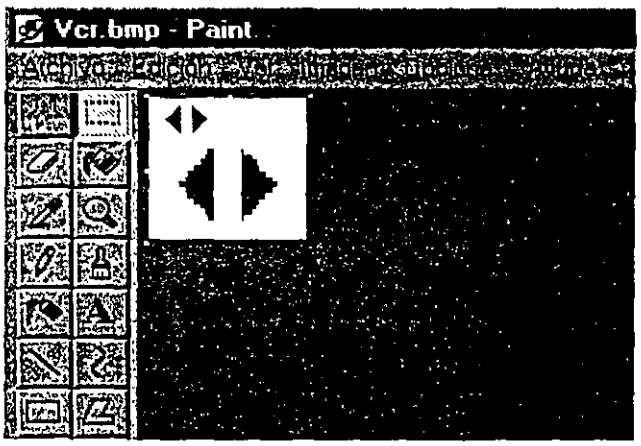


Figura 13-14

VCR.BMP.

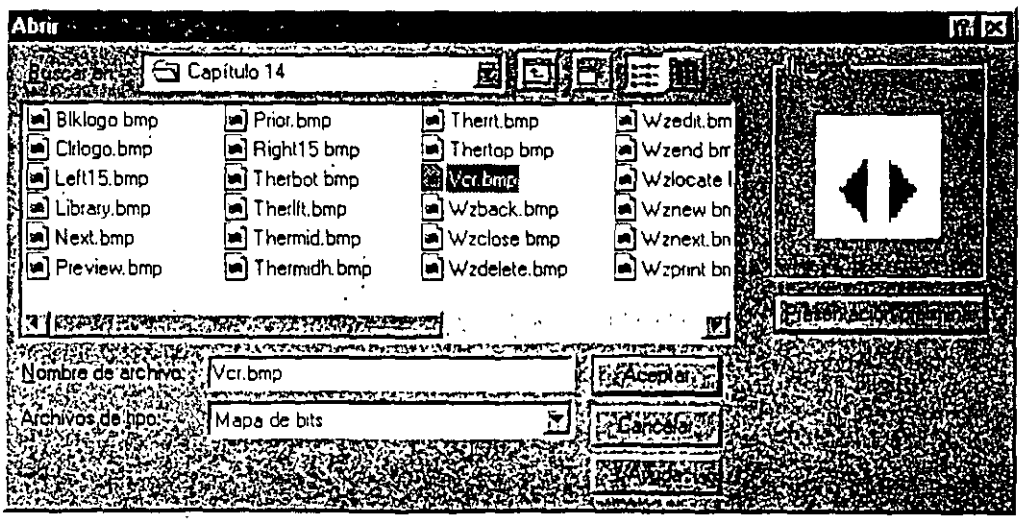
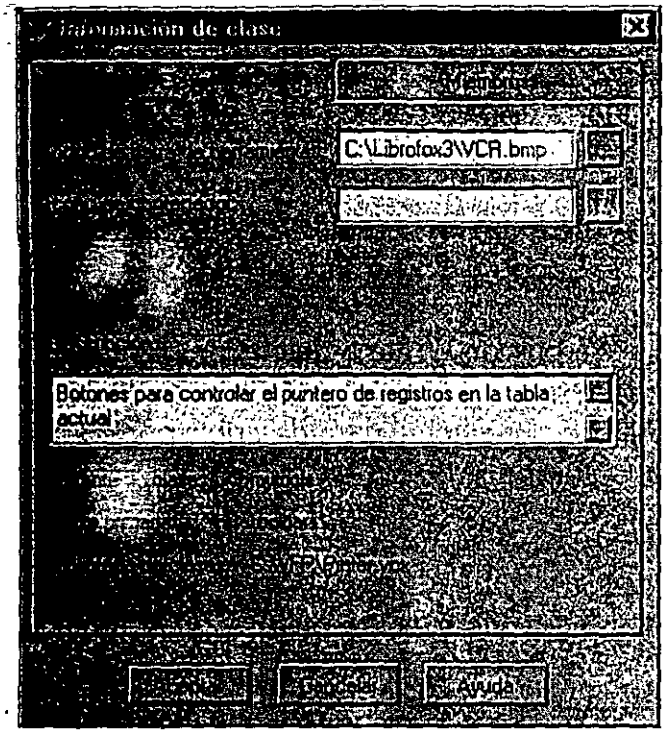


Figura 13-15

Cuadro de diálogo Abrir archivo, para seleccionar un mapa de bits.

Figura 13-16



Cuadro de diálogo Información de clase.

Realice cambios en la clase

Ahora piense cómo mejorar la clase. Se cambiará el botón de comando Siguiente, de modo que si su usuario hace clic en Siguiente mientras se encuentra en el último registro en el archivo, el programa ofrece envolver el primer registro en el archivo. Se podría agregar una capacidad igual al botón Anterior. El siguiente es el código modificado para el botón cmdAnterior:

```
IF NOT BOF()
  SKIP -1
  IF BOF()
    IF WAITWINDOW("Se encontró el final de la tabla."+CHR(13)+;
        "¿Regresar al principio?",4+32+256) = 6  && Sí
        GO BOTTOM
    ELSE                                  && 7 = No
        GO TOP
    ENDIF
  ENDIF
ENDIF
THISFORM.Refresh
```

Luego haga un cambio análogo en el botón cmdSiguiente:

```
IF NOT EOF()
  SKIP 1
  I    )F()
       WAITWINDOW("Se encontró el principio de la tabla."+CHR(13)+;
       "¿Regresar al final?",4+32+256) = 6  && Sí
       GO TOP
    ELSE                              && 7 = No
         GO BOTTOM
    ENDIF
  ENDIF
ENDIF
THISFORM.Refresh
```

Guarde sus cambios y está listo para probar sus modificaciones.

Pruebe las modificaciones

Ahora tiene un conjunto de botones de VCR un cuanto más inteligentes. Teclee DO FORM PRUEBAVCR, haga clic en el botón Ultimo y luego en el botón Siguiente, y aparecerá el cuadro de mensaje Regresar, como se muestra en la figura 13-17. Verifique que Previo también funcione y todo está listo.

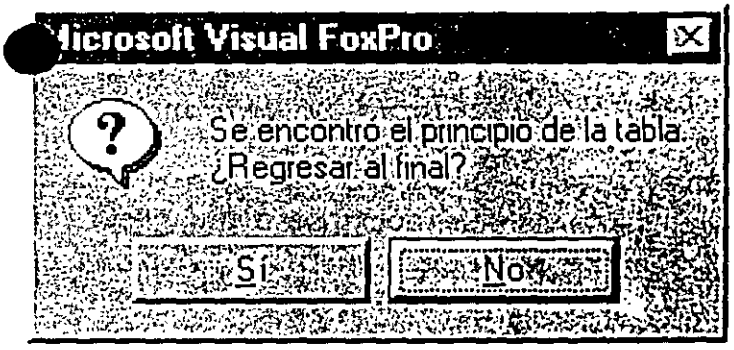


Figura 13-17

Despliegue del cuadro de mensaje.

Ahora que ha experimentado la creación y uso de una clase simple, tal vez desee intentarlo usted mismo. El ejemplo de VCR fue muy simple, de modo que definitivamente experimentará algunos desafíos conforme explore las complejidades del Generador de clases y sepa cómo crear clases complejas y sólidas.

El siguiente ejemplo es más complejo y trata sobre varios aspectos:

- Cambio de tamaño de las clases
- Creación de propiedades y métodos de uso

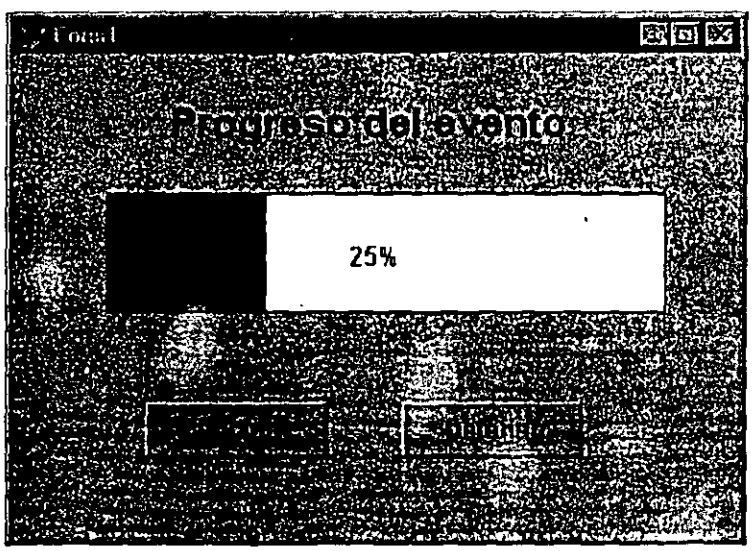
- Cambio de apariencia y comportamiento dependiendo de los valores establecidos de propiedad
- Uso de cronómetros
- Uso de requerimiento de prueba compleja

Una clase barra de termómetro

En éste y otros capítulos se ha hecho referencia a una clase barra de termómetro, la cual puede emplear para desplegar el porcentaje de terminación de un proceso que requiere tiempo. Ésta es la clase que creará en esta sección.

Primero, se dará una especificación más precisa. Desea una gráfica de barras que, con base en el porcentaje, se sombree en forma proporcional. También debe desplegar el porcentaje numéricamente y tener un mecanismo para actualizar el porcentaje sombreado. Tiene que verse algo parecido a la figura 13-18.

Figura 13-18



Prototipo de termómetro.

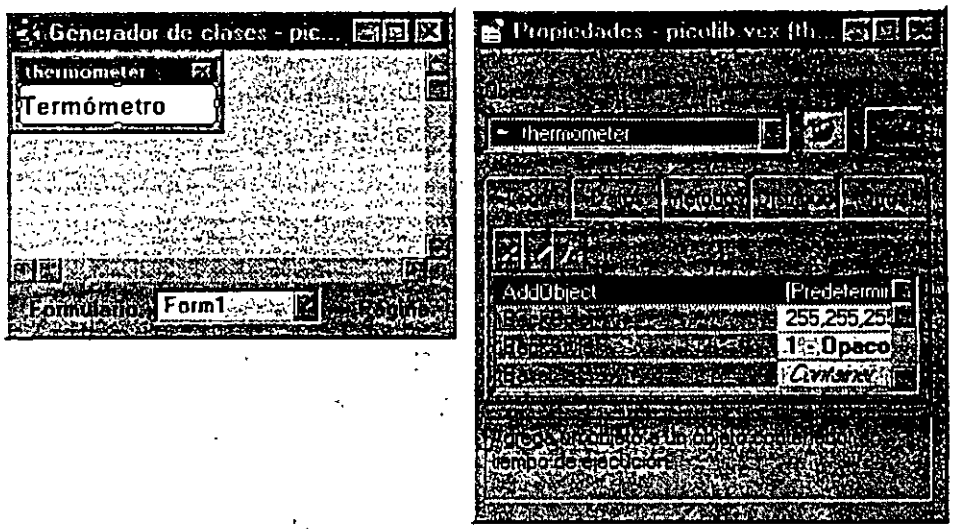
Cree la clase

Esta clase contiene dos objetos, una clase y una etiqueta. Cree una clase contenedora, llamada termómetro, y luego cree un formulario en la misma. Haga el formulario de un color claro (por ejemplo, verde claro) de modo que se pueda leer el texto sobre éste y determine su tamaño de tal forma que tenga la misma altura que la clase de termómetro con cualquier ancho. El ancho se establecerá interactivamente, de manera que no importa cuál sea durante el tiempo de diseño. Haga la etiqueta lo suficientemente ancha para que quepa el texto, 100%.

3/3

Inclusive, tal vez quiera que ése sea el valor inicial, de modo que se puede establecer su tamaño en forma horizontal exactamente para que quepa. De nuevo, se iniciará en cero y cambiará de manera interactiva, así que no tiene gran importancia cuál sea el valor en el tiempo de diseño. Inicie los suyos para tener una propiedad Caption de termómetro de modo que sea más descriptivo cuando se crea una instancia en un formulario. Ahora debe tener una clase similar a la que se muestra en la figura 13-19.

Figura 13-19



Elementos gráficos de la clase termómetro.

El siguiente paso obvio es agregar una propiedad de uso en la clase para hacer un seguimiento del porcentaje de la barra termómetro que está sombreada. Para hacer esto, seleccione Clase/Nueva propiedad del menú. Capture el nombre de la propiedad y una descripción. Ya que se usa para hacer un seguimiento del porcentaje de la barra que está sombreando, llamemos a la nueva propiedad Porcentaje. El cuadro de diálogo será similar al de la figura 13-20.

También necesita crear un método de uso a fin de actualizar el control. Denominémoslo Actualizar. Seleccione Clase, Nuevo método del menú y capture lo que se muestra en la figura 13-21.

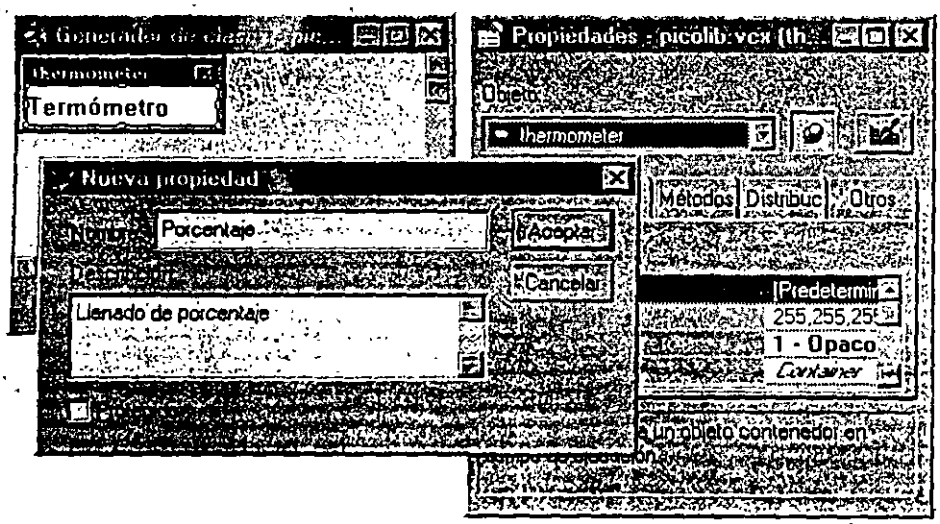
Ahora está listo para codificar. ¿Qué quiere que haga el método Actualizar? Asegúrese de que el valor se encuentre en el intervalo válido, actualice la propiedad Porcentaje, ajuste el ancho de la barra y actualice el despliegue de texto del porcentaje completado. Éste es el código:

```
PARAMETERS nPorcentaje
IF nPorcentaje < 0
   nPorcentaje = 0
ENDIF
```

```
IF nPorcentaje > 100
   nPorcentaje = 100
ENDIF

* Actualizar (Update)
THIS.Porcentaje         = nPorcentaje
THIS.shpBox.Width       = 106 * (nPorcentaje/100)
THIS.txtporcentaje.Caption = ALLTRIM(STR(nPorcentaje)) + "%"
THIS Refresh
```

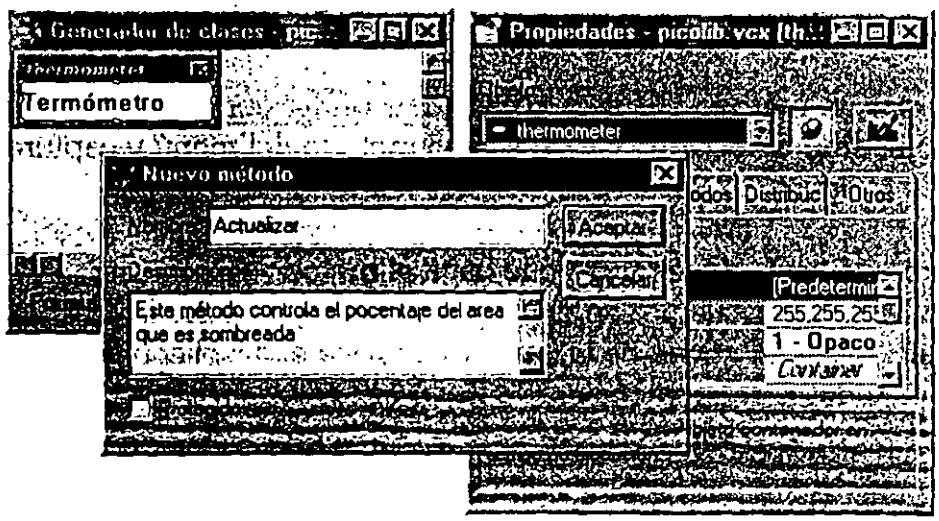
¡Cambie la constante 106 a cualquiera que sea el ancho de su clase, y su clase estará lista para usarse! No se necesita estrictamente la propiedad Porcentaje en la clase. Sólo necesita el valor en este método; ya esta ahí, en el parámetro

Figura 13-20



Nueva propiedad: Porcentaje.

Figura 13-21



Nuevo método: Actualizar.

315

nPorcentaje. Pero los controles que están fuera del termómetro tal vez deseen
ɔcer su estado y una buena forma en que pueden tener acceso es a través de
propiedad no protegida.

Tolerancia de un tamaño variable

Quizá se haya dado cuenta de que, cuando hace la instanciación de la clase en un formulario, la barra de estado sólo abre el área fija en los pixeles que designó en su clase. Nunca se especificaron dimensiones, pero en el ejemplo se utilizaron arbitrariamente 24 pixeles de alto por 106 de ancho. Puede crear una barra de termómetro que ocupa una ventana más ancha o más angosta, según se necesite. Asimismo, una barra de termómetro da la ilusión de una ejecución más rápida porque la velocidad absoluta del llenado es mayor, sin que tenga importancia la velocidad relativa de la tarea subyacente.

Hay dos objetos en la clase, de modo que debe enfocarse individualmente en los conceptos. Es sorprendente que la barra de termómetro en realidad sea la más simple de las dos.

Sólo se necesita cambiar una línea y agregar una línea para fijar la barra de
ado. El primer cambio es fácil. Busque constantes en el código que se proporcionó anteriormente. Las constantes a menudo pueden parametrizarse. Tal es el caso aquí, a excepción de 0 y 100, que son inherentes de la naturaleza del cálculo porcentual. Cambie el valor de la anchura para leer THIS.Width como sigue:

```
THIS.shpBox.Width = THIS.Width * (nPorcentaje/100)
```

Ahora el termómetro siempre cambiará de tamaño por sí mismo a la fracción apropiada del ancho del control, no importa de qué tamaño sea el control.

Necesita hacer lo mismo para la altura del termómetro. Ya que sólo requiere ajustarse una vez, en la inicialización, puede ponerlo en el evento Init de la clase termómetro. Teclee el renglón siguiente en este evento:

```
THIS.shpBox.Height = THIS.Height
```

Ahora debe pensar cómo asegurarse de que el texto siempre esté centrado. Establezca la propiedad Alignment de la etiqueta en 2-centro. De este modo, puede dejar constante el ancho de la etiqueta y no preocuparse por variar el

3/6

ancho del texto de las cadenas de texto, como 1%, 99% y 100%. El ancho se establece en 48, que dará espacio al 100%, y la altura se establece en 16.

Como fue el caso con la propiedad Height del cuadro de texto, este código puede localizarse en el código de evento Init para el control porque necesita correrse sólo una vez.

Con fines de claridad, se establecerá el valor de llenado porcentual en cero. Puede inicializar la barra de estado en cero con sólo hacer que el ancho sea 0 en la ventana de diseño. Éste es el código que debe ir en el evento Init del control del termómetro (incluyendo la inicialización tanto de la propiedad de texto como de Porcentaje en cero):

```
THIS.shpBox.Height         = THIS.Height

THIS.txtPorcentaje.Alignment = 2
THIS.txtPorcentaje.Width     = 48
THIS.txtPorcentaje.Left      = (THIS.Width - 48) / 2
THIS.txtPorcentaje.Top       = (THIS.Height - 16) / 2
THIS.txtPorcentaje.Caption   = "0%"

THIS.Porcentaje              = 0
```

Ahora, si hace la instanciación de una clase termómetro en un formulario, puede cambiar las dimensiones a cualquier tamaño y aún así funcionará. El texto se centrará y la barra de estado se llenará de izquierda a derecha con base en el porcentaje del ancho del control.

Variaciones sobre un tema

¿Qué le sucede al termómetro si lo hace una barra vertical? Aún se llena de izquierda a derecha. ¿No parece más natural llenarse verticalmente en un escenario como éste? ¿Por qué no modifica la clase para tener una propiedad Type que pueda emplearse para seleccionar un llenado de izquierda a derecha o de abajo hacia arriba? Inclusive puede permitir que el usuario cambie entre los dos en el tiempo de operación.

El primer paso es crear la propiedad. Haga clic en Clase, Nueva propiedad y capture el nombre Tipo. Después de que ha agregado la propiedad, sólo hay u cambio de código que debe efectuar. Este renglón actualiza el ancho de la barra:

```
THIS.shpBox.Width = THIS.Width * (nPorcentaje/100)
```

Sustitúyalo con el código siguiente:

```
L   ASE
CASE THIS.Type          = 0
   THIS.shpBox.Height = THIS.Height
   THIS.shpBox.Width   = THIS.Width * (nPorcentaje/100)
   THIS.shpBox.Top     = 0
   THIS.shpBox.Left    = 0
CASE THIS.Type          = 1
   THIS.shpBox.Height = THIS.Height * nPorcentaje/100)
   THIS.shpBox.Width   = THIS.Width
   THIS.shpBox.Top     = THIS.Height - This.shpBox.Height
   THIS.shpBox.Left    = 0
ENDCASE
```

Nótese que la línea de ajuste de ancho en sí no cambia, pero se movió hacia adentro de una de las instancias. Asimismo, nótese que el valor establecido de la propiedad Height se movió al código del método Update desde el evento Init. Esto se debe a que, si permite que Type cambie de manera interactiva, entonces todas las llamadas a Update podrían ser potencialmente un cambio en el tipo de llenado y por tanto un cambio en la altura de la barra. Además, las propiedades Top y Left de la barra se calculan ahora.

Si regresa a su formulario de prueba, verá todos los cambios incorporados
máticamente en la instancia existente. Sólo cambie la propiedad Type a 2
a ver si se llena de abajo hacia arriba.

Un formulario de prueba para la clase termómetro

Para probar este control, se crearon varios controles en el formulario además de la instancia del termómetro como tal. Se creó un control numérico y se redujo de modo que sólo las flechas fueran visibles, luego se ubicó junto a la barra de bloqueo de manera que la posición de la barra de bloqueo pudiera controlarse por el valor numérico. Se ve como la figura 13-22.

Los valores predeterminados de 0 y 100 para valores menores y mayores del control numérico fueron perfectos para el ejemplo, de manera que no es necesario tocarlos. La única línea nueva del código fue una llamada al método Update del termómetro en el evento InteractiveChange del contador, pasando el valor del Control numérico:

```
THISFORM.Termometro1.Update(THIS.Value)
```

Figura 13-22

Control numérico para verificar la clase termómetro.

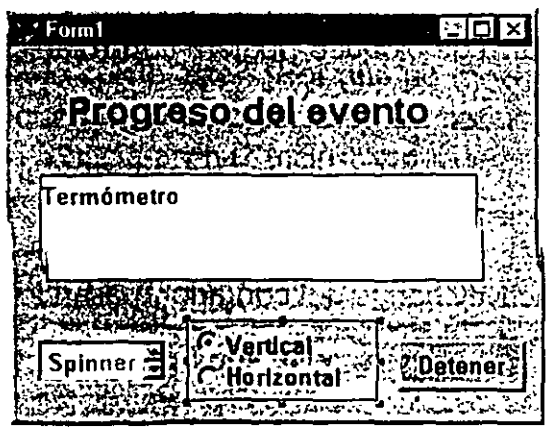
Con objeto de probar la capacidad de cambiar interactivamente entre los tipos de llenado vertical y horizontal, se agregó un grupo de opciones. Tiene un aspecto similar al de la figura 13-23. El código que afecta al termómetro también se agrega al evento InteractiveChange del grupo de opciones:

```
THISFORM.Termometro1.Type = THIS.Value-1
THISFORM.Termometro1.Update(THISFORM.Termometro1.Porcentaje)
```

La primera línea establece la propiedad Type para el valor del grupo de opciones, menos uno. Esto es porque se usó 0 y 1 como los valores de Type en vez de 1 y 2 para evitar agregar la línea:

```
THIS.Type = 1
```

Figura 13-23



Grupo de opciones para probar la clase termómetro.

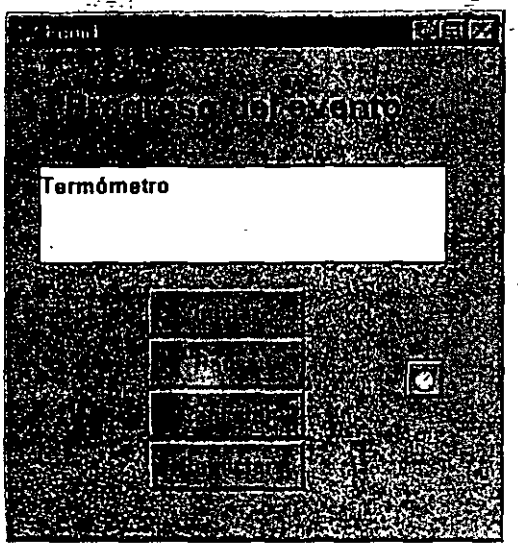
para el evento Init de la clase termómetro. En lugar de eso, aprovecha el hecho de que el valor predeterminado es 0. Ésta es una cuestión de estilo, así que hágalo como desee.

Una prueba más realista

También se quiso probar cómo se veía el control cuando se empezó a controlar su crecimiento por medio de una fuente automatizada, a diferencia del control de usuario directo como con el control numérico. Para hacer esto, se utilizó el control de temporización y cuatro botones (ver figura 13-24). El código para cada uno de los controles es:

```
THISFORM.Contador1.Interval    = 1
THISFORM.Contador1.Enabled     = .T.
THISFORM.Contador1.Reset

THISFORM.cmdIniciar.Enabled     = .F.
THISFORM.cmdPausa.Enabled       = .T.
THISFORM.cmdCancelar.Enabled    = .T.
THISFORM.cmdReiniciar.Enabled   = .F.

IF THIS Caption              = "Pausa"
  THISFORM.Contador1.Enabled  = .F.
  THIS.Caption                = "Continuar"
ELSE
  THISFORM.Contador1.Enabled  = .T.
  THIS.Caption                = "Pausa"
ENDIF

THISFORM.Contador1.Interval    = 0
THISFORM.cmdPausa.Caption   = "Pausa"

THISFORM.cmdIniciar.Enabled     = .T.
THISFORM.cmdPausa.Enabled       = .F.
THISFORM.cmdCancelar.Enabled    = .F.
THISFORM.cmdReiniciar.Enabled = .T.

THISFORM.Termometro1.Update(0)
THISFORM.Spinner1.Value    = 0

THISFORM.cmdIniciar.Enabled    = .T.
THISFORM.cmdPausa.Enabled      = .F.
THISFORM.cmdCancelar.Enabled   = .F.
THISFORM.cmdReniciar.Enabled   = .F.
```

Figura 13-24



Control de tiempo y botones para probar la clase termómetro.

Este código sincroniza la capacidad de opciones. La única llamada para el termómetro está en el botón Iniciar, para limpiarlo. La mayor parte del trabajo está hecho en el mismo control de tiempo:

```
IF THISFORM.Termometro1.Porcentaje < 100
  THISFORM.Termometro1.Update(THISFORM.Termometro1.Porcentaje + 1)
  THISFORM.Spinner1.Value = :THISFORM.Spinner1.Value + 1
ELSE
  THIS.Interval                  = 0
  THIS.Enabled                   = .F.
  THISFORM.cmdIniciar.Enabled    = .F.
  THISFORM.cmdPausa.Enabled      = .F.
  THISFORM.cmdCancelar.Enabled   = .F.
  THISFORM.cmdReiniciar.Enabled  = .T.
ENDIF
```

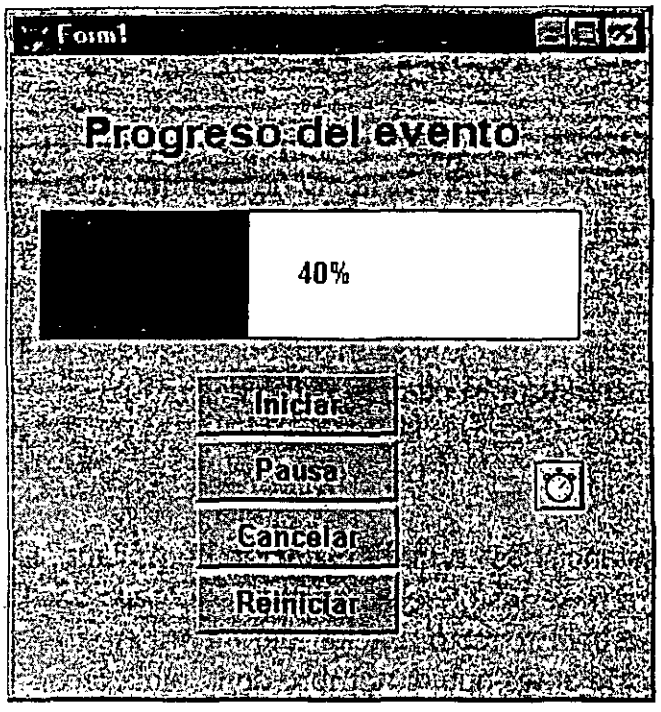
La instrución IF verifica si se permite la actualización. Si el valor del termómetro és mayor que 100, ya no se le envían valores. De otro modo, el valor actual se incrementa en uno.

La segunda línea actualiza el control numérico de modo que pueda usarlo para mover la barra de estado hacia atrás y hacia adelante, y el valor del contador siempre concuerda con el valor de la barra de estado.

Con los controles de prueba antes descritos, su formulario de prueba debe verse como la figura 13-25.

321

Figura 13-25



El formulario de prueba termómetro en ejecución.

Herencia: ¿mi código o el suyo?

Cuando basa una clase en una clase preexistente en lugar de hacerlo en una clase base, hereda todos sus valores establecidos de propiedad y todo el código en sus eventos y métodos. Su subclase tal vez necesite ejecutar el código ya sea en lugar o además del código previamente existente.

Con el fin de reemplazar el código previamente existente o escribir su propio código para ejecutarlo en vez del existente, sólo introduzca el código en el evento o método. En forma automática correrá en vez del código de la clase padre. Para ejecutar su nuevo código añadido al código padre, haga una llamada al método padre en su código. Por ejemplo:

```
Classname :: clic
THIS.MiPropiedad = 42
THISFORM.Refresh
```

Este código primero corre el código de la clase y luego corre el código adicional. Nótese que la llamada al método padre puede estar en cualquier parte en el código hijo; al inicio, al final o en cualquier parte en el centro del mismo.

Exclusión de clases de un proyecto

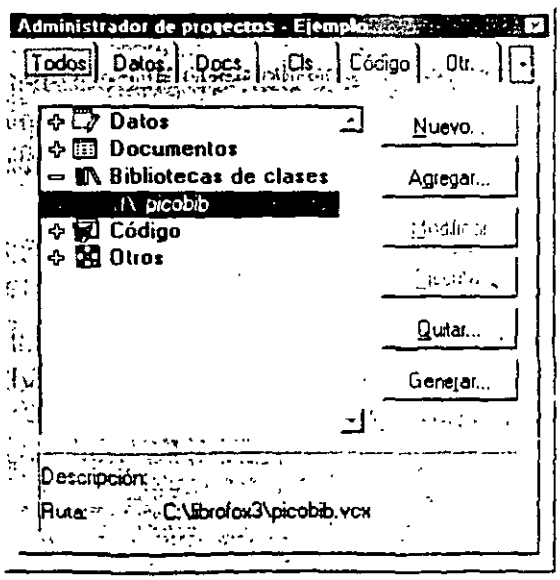
Uno de los beneficios más importantes de las clases es que puede modificarlas y todas las instancias de la clase que existan adoptan automáticamente los nuevos cambios. Pero a lo largo de este capítulo se hizo algo que asegura por completo que los cambios a las clases *no* se reflejen en sus aplicaciones: se compilaron.

Si ha creado una aplicación a partir de su proyecto y se incluyen clases en su proyecto, los cambios en esas clases no se usarán cuando usted ejecute su aplicación. Sólo se usarán las clases que se integren en la aplicación en el tiempo de construcción.

Para manifestar los casos en clases instanciadas en una aplicación compilada, se debe excluir la clase del proyecto. Para hacer esto, seleccione la biblioteca de clases de su lista de proyecto, luego escoja Proyecto, Excluir del menú. Aparecerá una pequeña circunferencia con un corte antes que el nombre de la biblioteca, indicando que no se compilará en el .APP o .EXE. Entonces puede distribuir la biblioteca de clases por separado. Cuando haga un cambio en la biblioteca de clases, ponga una copia de la nueva biblioteca de clases en el directorio con los archivos de aplicación y se usará cuando ejecute la aplicación.

La ventana de proyecto para un proyecto que utiliza PICOBIB pero que lo excluye de la construcción se ve como la figura 13-26. En nuestro foro de Compuserve, encontrará una versión de esta clase que se ve justo como un termómetro; sus usuarios podrían apreciar los gráficos atractivos.

Figura 13-26



Excluir una biblioteca de clases del proyecto.

Resumen

La adición del soporte de clases en Visual FoxPro representa un enorme progreso, que mejorará en gran medida la productividad y el profesionalismo de los programadores. Queda en usted mismo y en sus clientes y usuarios dar el paso siguiente y aprender cómo usar clases en el desarrollo de aplicaciones.

En el capítulo siguiente, volverá a escribir la aplicación básica usando lo que ha aprendido en los últimos capítulos. Seguramente apreciará la elegancia y simplicidad de Visual FoxPro cuando lo ponga a prueba.

14

Uso de clases para volver a generar una aplicación de ejemp

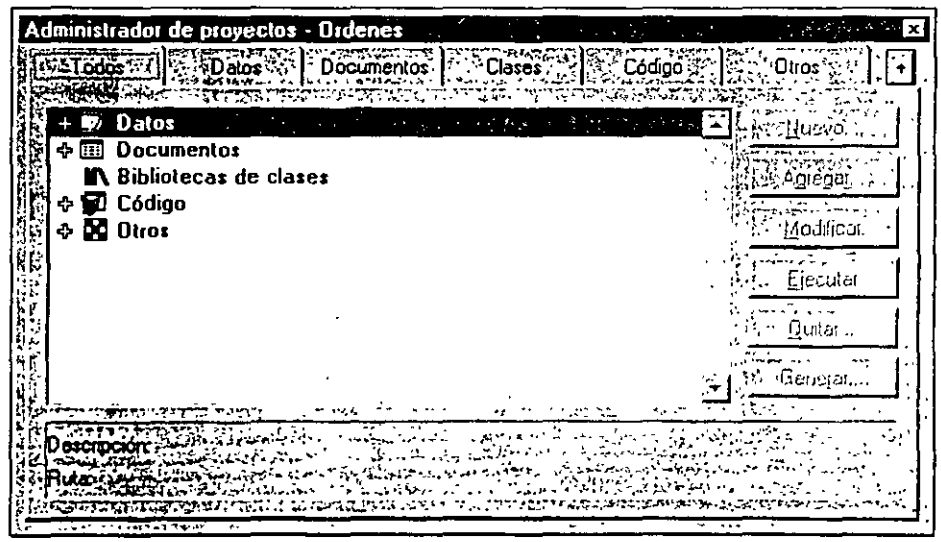
CAPITULO 14

EN el capítulo 3 usted desarrolló una aplicación simple. En el capítulo 9, utilizó habilidades adicionales desarrolladas en los capítulos intermedios para mejorar la aplicación. Ahora se tomarán las pantallas del capítulo 9 y se mostrará cómo puede arrastrarlas rápida y fácilmente con clases.

Cree un archivo de proyecto

Como es usual, empiece por crear un archivo de proyecto Ordenes. Conéctese a CompuServe o teclee GO PINTER para obtener algunos archivos de muestra de este capítulo, en el subdirectorio CHAP14. Los cargos por hacerlo son mínimos. Si contruye su propio archivo, escriba el comando MD/PINTER/CAP14 para crear el directorio, y luego CD/PINTER/CAP14 para llegar al directorio. Haga clic en Archivo, Nuevo, seleccione Proyecto y haga clic en Aceptar. Escriba ORDENES sobre el nombre predeterminado PROJECT1.PJX y presione ENTER, como se muestra en la figura 14-1.

gura 14-1



Cree del archivo de proyecto Ordenes.

El entorno de datos

Se utilizaron las mismas seis tablas que se usaron en la aplicación del capítulo 9, pero usted no puede simplemente copiarlas del directorio CAP09. Como las tablas fueron registradas con el entorno de datos del proyecto, sus encabezados contienen una liga de regreso hacia el DBC. Usted no puede usarlas en un DBC diferente.

326

Pero no tiene que volver a introducir las definiciones de las tablas. Usted puede usar cualquiera de dos sencillas técnicas. La primera es cambiarse al directorio CAP09. Para hacer eso:

```
OPEN DATABASE DATA1
USE CONTROL
COPY TO ../CAP14
USE ORDENES
COPY TO ../CAP14 WITH CDX
USE DETALLE
COPY TO ../CAP14 WITH CDX
USE CLIENTE
COPY TO ../CAP14 WITH CDX
USE LIBROS
COPY TO ../CAP14 WITH CDX
USE CLAVE
COPY TO ../CAP14 WITH CDX
```

Luego simplemente introduzca el proyecto, cree un contenedor DATOS1 (o cualquier otro nombre), résaltelo y haga clic en el botón Modificar para desplegar el Generador de bases de datos. Luego utilice el botón secundario del ratón para desplegar el menú contextual y utilice Agregar tabla, para agregar cada tabla. También puede usar el icono Agregar tabla de la barra de herramientas Generador de bases de datos.

Pero hay una forma aún más sencilla. En el directorio VFP/TOOLS/GENDBC, encontrará un programa llamado GENDBC.PRG. Yo lo copié en el directorio /CAP09 y luego lo ejecute. Aquí está lo que se produjo:

```
* ************************************************************
* *
* * 05/10/96              DATOS1.DBC              20:19:37
* *
* ************************************************************
* *
* * Descripción:
* * Este programa lo ha generado automáticamente GENDBC
* * Versión 1.4
* *
* ************************************************************
CLOSE DATA ALL
CREATE DATABASE 'DATOS1.DBC'
***** Configuración de tabla para CONTROL *****
CREATE TABLE 'CONTROL.DBF' NAME 'CONTROL' (UltimaORDEN N(6, 0) NOT NULL, ;
                      TIENDANOMB C(40) NOT NULL, ;
                      TIENDADIR1 C(40) NOT NULL, ;
                      TIENDADIR2 C(40) NOT NULL, ;
                      TIENDADIR3 C(40) NOT NULL, ;
```

```
                        MENSAJE1 C(40) NOT NULL, ;
                        MENSAJE2 C(40) NOT NULL)

***** Crear cada índice para CONTROL *****
***** Cambiar las propiedades (si hay alguna) para CONTROL *****

***** Configuración de tabla para CLIENTE *****
CREATE TABLE 'CLIENTE.DBF' NOMBRE 'CCLIENTE' (IDCLIENTE C(10) NOT NULL, ;
                        NOMBRE C(20) NOT NULL, ;
                        APELLIDO C(20) NOT NULL, ;
                        DIR1 C(40) NOT NULL, ;
                        DIR2 C(40) NOT NULL, ;
                        CIUDAD C(20) NOT NULL, ;
                        ESTADO C(3) NOT NULL, ;
                        CP C(10) NOT NULL, ;
                        PAIS C(10) NOT NULL, ;
                        TEL1 C(14) NOT NULL, ;
                        TEL2 C(14) NOT NULL, ;
                        LIMITECREDITO N(6, 0) NOT NULL, ;
                        BALANCE N(7, 2) NOT NULL, ;
                        CREDITOUSADO L NOT NULL, ;
                        NUMEROTARJETA C(16) NOT NULL, ;
                        FECHAEXP C(5) NOT NULL, ;
                        COMPRAS N(7, 2) NOT NULL)

***** Crear cada índice para CLIENTE *****
INDEX ON IDCLIENTE TAG IDCLIENTE
INDEX ON UPPER(APELLIDO+NOMBRE) TAG NOMBRE
INDEX ON DELETED() TAG DELETED

***** Cambiar las propiedades (si hay alguna) para CLIENTE *****

***** Configuración de tabla para LIBROS *****
CREATE TABLE 'LIBROS.DBF' NOMBRE 'LIBROS' (ISBN C(10) NOT NULL, ;
                        TITULO C(60) NOT NULL, ;
                        PRECIOLISTA N(7, 2) NOT NULL, ;
                        ADICIONAL M NOT NULL, ;
                        AUTOR C(20) NOT NULL, ;
                        PRECIOVENTA N(7, 2) NOT NULL, ;
                        EXISTEN N(4, 0) NOT NULL, ;
                        ORDENADOS N(4, 0) NOT NULL, ;
                        ORDENESP N(4, 0) NOT NULL, ;
                        LLEGARAN D NOT NULL, ;
                        IDESTADO C(3) NOT NULL, ;
                        ID_PUB C(10) NOT NULL, ;
                        CATEGORIA C(10) NOT NULL, ;
                        TIPO C(10) NOT NULL, ;
                        UBICACION C(10) NOT NULL)

***** Crear cada índice para LIBROS *****
INDEX ON ISBN TAG ISBN
INDEX ON ID_PUB TAG ID_PUB
```

328

```
INDEX ON UPPER(IIF(LEFT(UPPER(TITULO),2)="UN
" ,SUBSTR(TITULO,3),IIF(LEFT(UPPER(TITULO),3)="EL
",SUBSTR(TITULO,4),IIF(LEFT(UPPER(TITULO),4)="LOS",SUBSTR(TITULO,5),
TITULO)))) TAG TÍTULO
INDEX ON CATEGORIA TAG CATEGORIA
INDEX ON TIPO TAG TIPP
INDEX ON DELETED() TAG DELETED

***** Cambiar las propiedades (si hay alguna) para LIBROS *****
***** Configuración de tabla para ORDENES *****
CREATE TABLE 'ORDENES.DBF' NAME 'ORDENES' (NUMORDEN C(10) NOT NULL, ;
                      FECHA D NOT NULL, ;
                      IDCLIENTE C(10) NOT NULL, ;
                      NOMBRE C(40) NOT NULL, ;
                      DIR1 C(40) NOT NULL, ;
                      DIR2 C(40) NOT NULL, ;
                      CIUDAD C(20) NOT NULL, ;
                      ESTADO C(3) NOT NULL, ;
                      CP C(10) NOT NULL, ;
                      PAIS C(10) NOT NULL, ;
                      SUBTOTAL N(7, 2) NOT NULL, ;
                      IVA N(7, 2) NOT NULL, ;
                      TOTALORDEN N(7, 2) NOT NULL, ;
                      ENVIADO L NOT NULL)

***** Crear cada índice para ÓRDENES *****
INDEX ON IDORDEN TAG NUMORDEN
INDEX ON IDCLIENTE TAG IDCLIENTE
INDEX ON ENVIADO TAG ENVIADO
INDEX ON DELETED() TAG DELETED

***** Cambiar las propiedades (si hay alguna) para ORDENES*****
***** Configuración de tabla para DETALLE *****
CREATE TABLE 'DETALLE.DBF' NAME 'DETALLE' (NUMORDEN C(10) NOT NULL, ;
                      ISBN C(10) NOT NULL, ;
                      CANTIDAD N(1, 0) NOT NULL, ;
                      TITULO C(40) NOT NULL, ;
                      PRECIO N(7, 2) NOT NULL, ;
                      EXTENDIDO N(7, 2) NOT NULL, ;
                      ORDENESPE L NOT NULL)

***** Crear cada índice para DETALLE *****
INDEX ON ISBN TAG ISBN
INDEX ON DELETED() TAG DELETED
INDEX ON NUMORDEN TAG NUMORDEN

***** Cambiar las propiedades (si hay alguna) para DETALLES *****

***** Configuración de tabla para PALABRAS CLAVE *****
CREATE TABLE 'CLAVE.DBF' NAME 'CLAVE' (CLAVE C(20) NOT NULL, ;
                      KEYISBN C(10) NOT NULL)
```

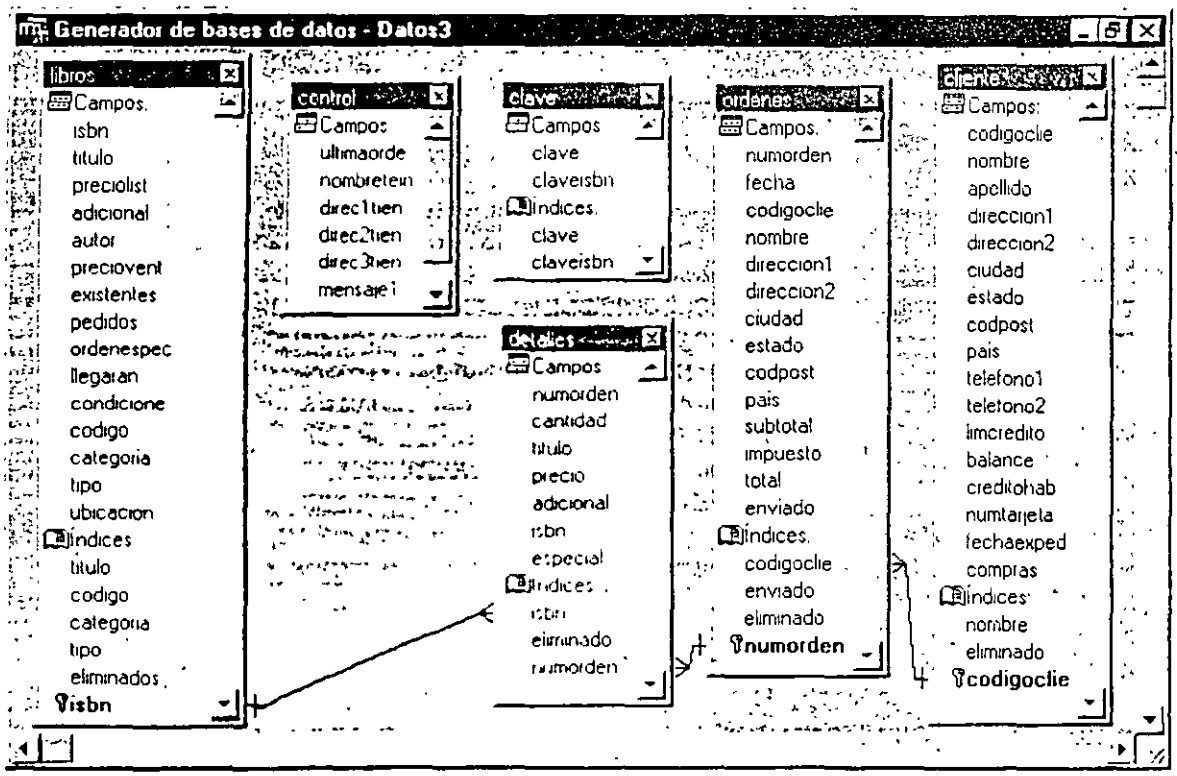
```
***** Crear cada índice para PALABRAS CLAVE *****
INDEX ON CLAVE TAG CLAVE
INDEX ON CLAVEISBN TAG CLAVEISBN

***** Cambiar las propiedades (si hay alguna) para CLAVE *****
```

Para utilizar este programa, cópielo al directorio CAP14 y ejecútelo. Esto volverá a crear el entorno de datos con ligas de regreso para el contenedor de bases de datos del directorio. Para continuar trayendo sus datos del directorio CAP09, haga esto para cada archivo, cambiando los nombres de archivos como sea necesario:

```
USE CLIENTE
APPEND FROM ../CAP09/CLIENTE
```

Pero antes de que usted empiece a escribir la aplicación, hay un pequeño cambio que se debe hacer al entorno de datos. Se agregó un nuevo campo, IVA, a la tabla CONTROL para que los usuarios pudieran cambiar la tasa de impuestos a la de su localidad. La figura 14-2 muestra el entorno de datos para la aplicación.

Figura 14-2



Entorno de datos para un ejemplo de aplicación.

Los programas

Usted puede copiar los archivos PRINCIP.PRG y el menú que desarrolló en el capítulo 9 a su directorio CAP14; no se usarán los otros archivos .PRG. Por ejemplo, ATRAPARERROR.PRG será reemplazado con un formulario. La función Add Keywords se moverá también del archivo .PRG al formulario, y la clase termómetro del capítulo 13 reemplazará las funciones INITTERMO, MOSTERMO y QUITERMO. Así que el único programa que queda es PRINCIP.PRG, como se muestra en la figura 14-3.

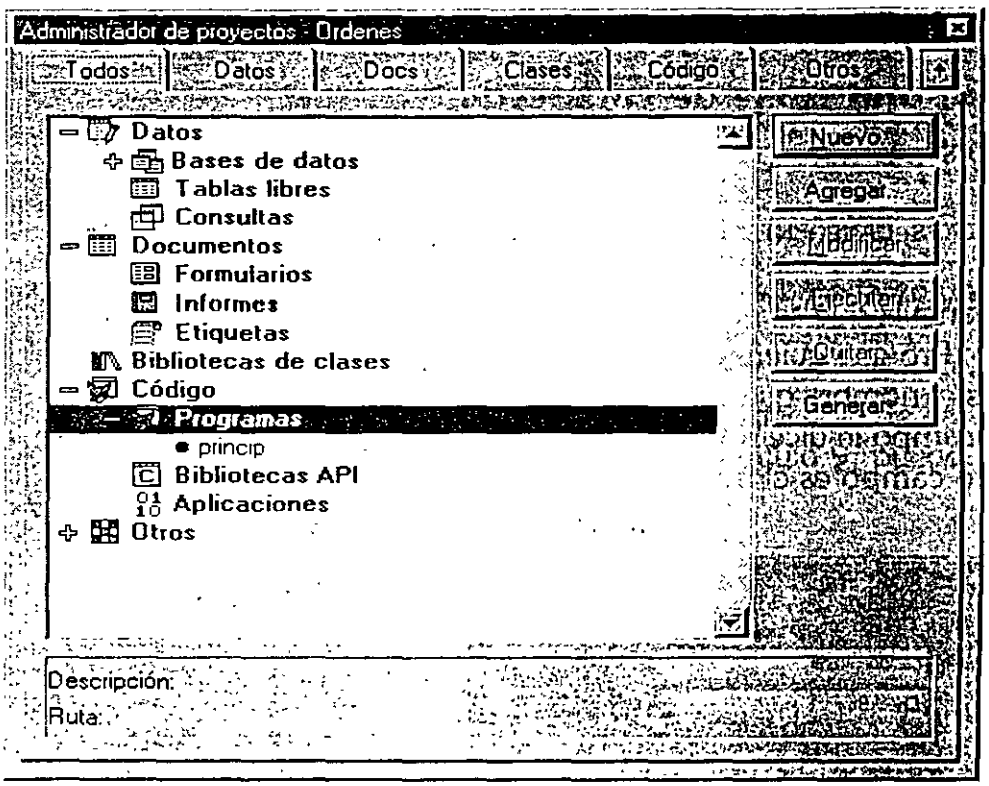


Figura 14-3

La sección Código del proyecto.

Los formularios

En el capítulo 9, usted generó los formularios Libros, Cliente y Ordenes con una cierta apariencia. Utilizó un color de fondo gris, campos de entrada con efectos 3-D con una fuente Courier. Agregó código a los eventos GetFocus y LostFocus, lo cual hizo que el campo de entrada utilizado tuviera un color diferente que los otros campos de entrada, y utilizó casi los mismos controles para los formularios Libros y Cliente. La pantalla Libros también contenía un botón Visualizar similar al del formulario Ordenes.

Esta vez, usted utilizará clases para generar los campos de entrada y las etiquetas, con las propiedades deseadas y el código de evento almacenado en la misma clase. Luego es cuestión de usar simplemente las clases en los formularios.

Genere la biblioteca de clases

Primero deseará generar el tipo de objetos que utilizará en los formularios y ponerlos dentro de una biblioteca de clases. Luego cuando seleccione un control de la barra de herramientas Controles, tendrá las propiedades que usted quiera y los métodos necesarios estarán ya incluidos.

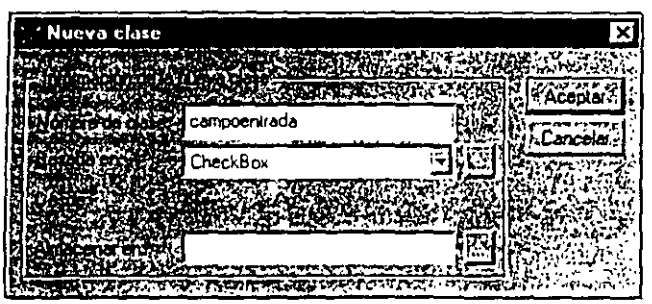
* Agregue una clase de campo de entrada

Escriba lo siguiente:

```
CREATE CLASS CAMPOENTRADA
```

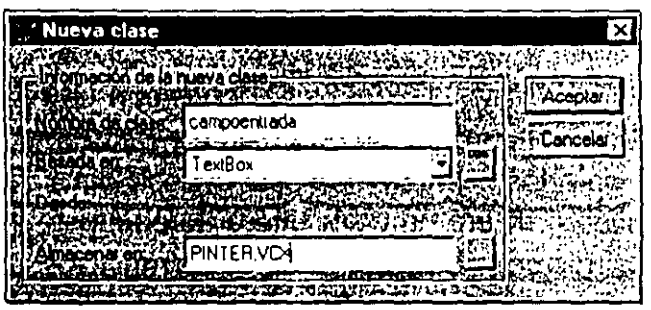
Verá la pantalla que se presenta en la figura 14-4. Aquí es donde usted le dice a Visual FoxPro que quiere crear una clase basada en un cuadro de texto. El primer campo es el nombre que usted proporcionó en el comando CREATE CLASS, el segundo campo le dice a Visual FoxPro qué clase debe usar -TextBox en este caso- y el último campo es el nombre del archivo .VCX para almacenarla en PINTER.

Figura 14-4



Creación de una clase.

Figura 14-5



Introducción del nombre de la clase y el nombre de la biblioteca de clases.

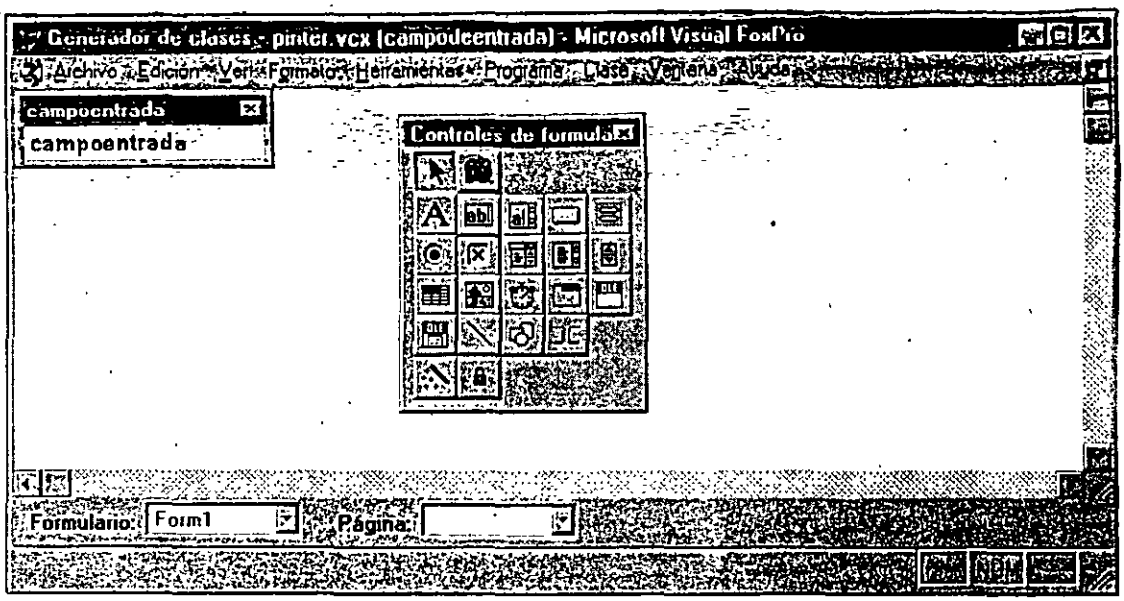


Figura 14-6

Clase campoentrada en PINTER.VCX.

Una vez que usted ha confirmado que desea crear una clase basada en un cuadro de texto (ver figura 14-5), obtendrá la pantalla mostrada en la figura 14-6.

Campoentrada es el campo de entrada estándar para introducción de datos. Viendo hacia atrás en los formularios que generó en el capítulo 9, puede seleccionar las propiedades de TextBox predeterminadas:

```
Enabled = .F.
FontBold = .T.
FontName = Courier New
FontSize = 12
Height = 25
Width = 100
```

La figura 14-7 muestra cómo son establecidas las propiedades para una clase. Es exactamente igual a establecer las propiedades para un objeto formulario.

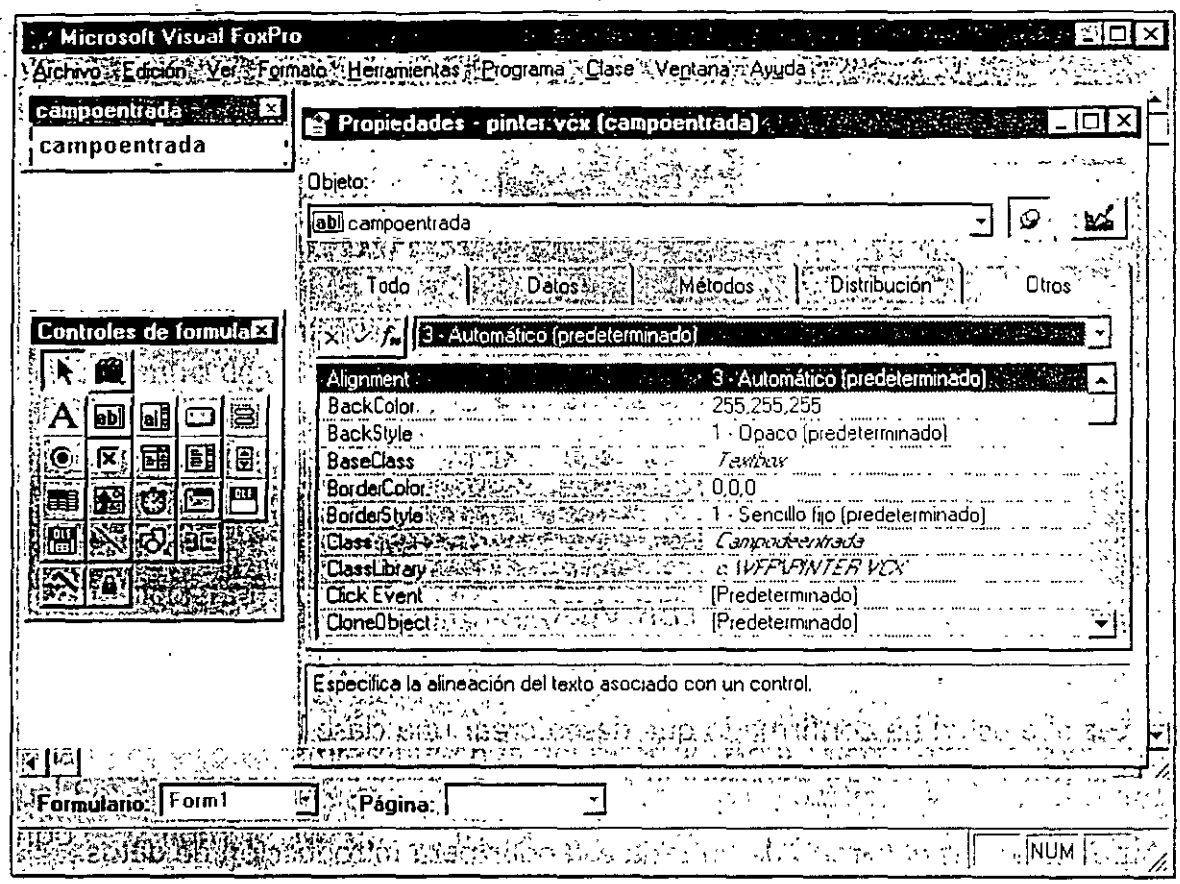
* Agregue una etiqueta de clase

Mientras esté creando clases, agregue una clase para hacer las etiquetas a la izquierda de cada campo de entrada. Escriba:

```
CREATE CLASS mietiqueta
```

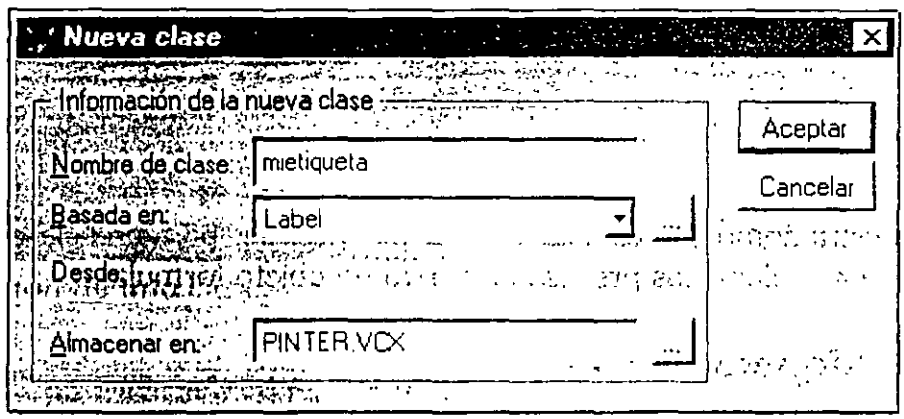
y obtendrá la pantalla mostrada en la figura 14-8. Para etiquetas, use Alignment 1 (derecha), Arial de 9 puntos y negrita para el título, BackStyle 0 (transaparente),

Figura 14-7



Establecimiento de las propiedades de la clase.

Figura 14-8

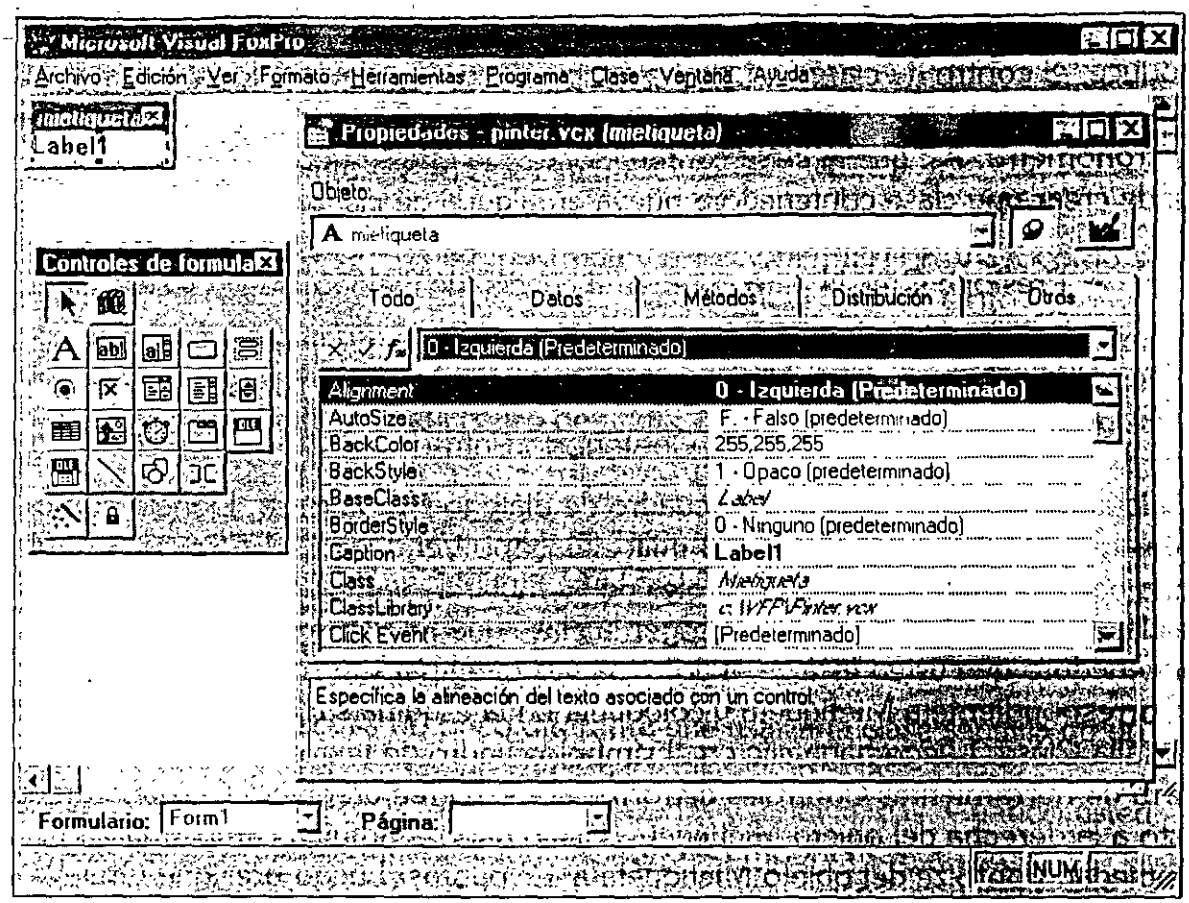


Creación de una clase de etiqueta.

una altura de 20 y un ancho de 100. El resultado se muestra en la figura 14-9. Puede evitar el cuadro de diálogo escribiendo:

```
CREATE CLASS mietiqueta AS label OF Pinter
```

Figura 14-9



Clase mietiqueta.

Ahora usted tiene un campo de entrada y una etiqueta, los cuales pueden ser colocados dentro de los formularios sin modificaciones posteriores, excepto para establecer la propiedad ControlSource para el campo de entrada.

* Cree una clase compuesta

Pero vaya un paso más adelante al construir una clase que agregue tanto la etiqueta como el campo de entrada al mismo tiempo. En lenguaje estándar OOP, esto se llama *clase compuesta*.

Con el fin de almacenar este bicho compuesto, usted debe escoger un tipo diferente de clase para mantener la etiqueta y el cuadro de texto, debido a que no es ni una clase de etiqueta ni una clase de cuadro de texto. Hay tres candidatos para encerrar el par de etiqueta y cuadro de texto:

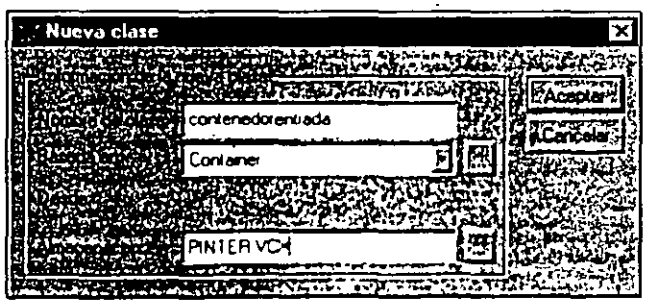
- ➢ Clases contenedoras
- ➢ Clases control
- ➢ Clases personalizadas

Todas ellas pueden contener otros controles, así que ¿cuál es la mejor? Como se produce, el contro de clases no puede ser cambiado -ni siquiera los valores de las propiedades- y las clases personalizadas se utilizan para objetos no visuales, como un cronómetro. Así que la clase contenedora es lo que se necesita aquí. Usted puede crear una clase contenedora nueva en alguna de las tres formas siguientes:

- Presione CTRL-N para obtener el cuadro de diálogo Nuevo y seleccione Clase.
- Seleccione Crear nuevo, del menú Archivo y luego Clase.
- Escriba CREATE CLASS ContenedorEntrada AS CONTAINER.

Cualquiera de éstas desplegará el cuadro de diálogo Nueva clase. Llénelo como se muestra en la figura 14-10. Luego utilice el icono Ver clases (el pequeño estante) en la barra de herramientas Controles de formulario, para cambiar a la biblioteca de clases PINTER y que pueda escoger la clase que irá dentro de la clase contenedora que usted está creando. Una vez que aparece, haga clic en el símbolo A correspondiente a Mietiqueta y colóquela en la esquina superior izquierda de la pantalla Clase. Luego haga clic en el símbolo cuadro de texto Campoentrada de la barra de herramientas Controles de formularios, y cree un objeto Campoentrada justo a la derecha del objeto Mietiqueta. Usted querrá también cambiar la propiedad FontSize del objeto Mietiqueta a 12 puntos. Luego presione CTRL-W o utilice Archivo, Cerrar del menú para guardar la clase.

Figura 14-10



El cuadro de diálogo Nueva clase llenado para crear una clase contenedora llamada ContenedorEntrada.

Ahora pruebe el objeto contenedor que usted acaba de crear. Escriba MODI FORM PRUEBA y verifique que ha seleccionado PINTER como la biblioteca de clases. Abra el Generador de bases de datos para PRUEBA.SCX y agregue la tabla CLIENTE. Luego haga clic en el objeto ContenedorEntrada y colóquelo dentro del formulario.

Si usted utiliza la página Propiedades para examinar el objeto que puso sobre el formulario, encontrará que tiene el nombre Contenedorentrada1, con los objetos

componentes Mietiqueta1 y Campoentrada1. Estos nombres fueron dados mientras la clase fue creada, pero nombres más lógicos serían Textos y Datos. Además, el contenedor mismo tiene un color de fondo blanco opaco y un borde. Vaya al Generador de clases, y arregle esto.

Escriba MODIFY CLASS ContenedorEntrada y seleccione PINTER.VCX como su biblioteca. Luego escójala de la lista de clases disponibles. Cambie tanto la propiedad BackStyle como la de BorderWidth a 0 (transparente). También puede cambiar el título de la etiqueta a Texto para que coincida el nombre, el cual hace una imitación de la conducta del cuadro de texto. No es muy importante, pero se ve mejor.

Ahora guarde la clase, regrese al formulario Prueba y agregue un ContenedorEntrada. Así está mejor, ¿no le parece?

* Utilice la clase

Usar una clase compuesta es un poco diferente que usar una clase sencilla de un componente. Por una razón, cuando usted hace clic en una clase, puede cambiarse inmediatamente a la página Propiedades y modificarlas. Cuando usted agrega una clase compuesta, el contenedor, no los componentes que la constituyen, es seleccionado. Así que cuando agrega una clase compuesta, tiene que dar un paso adicional para poder cambiar sus propiedades.

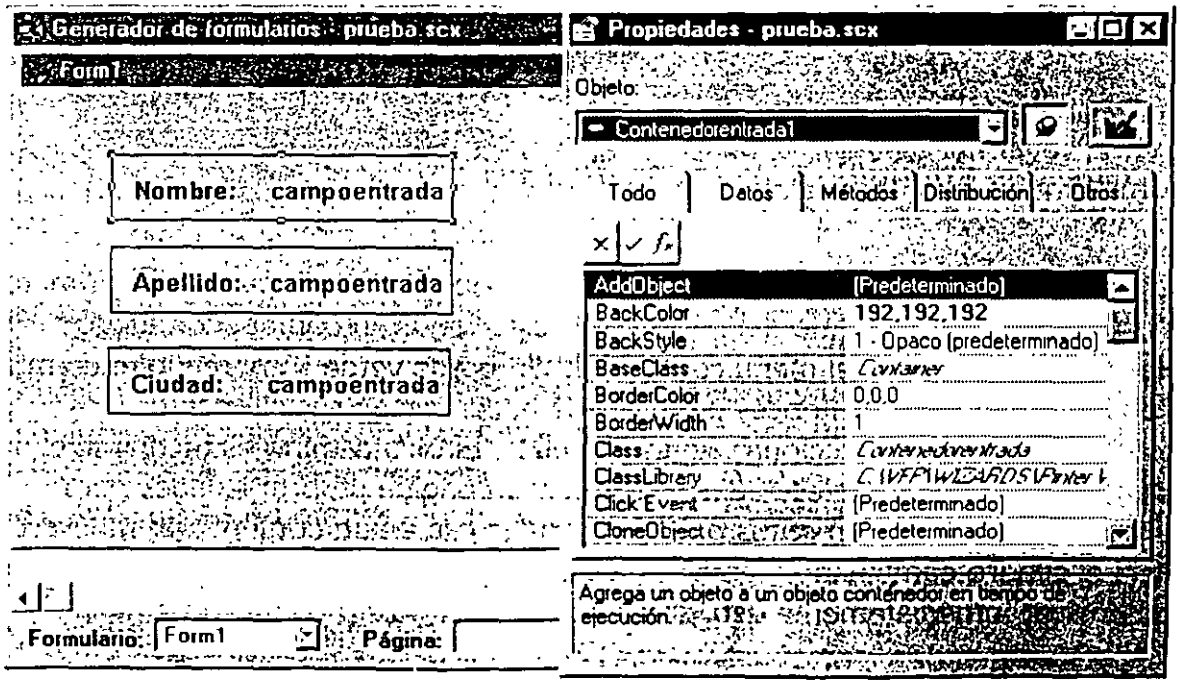


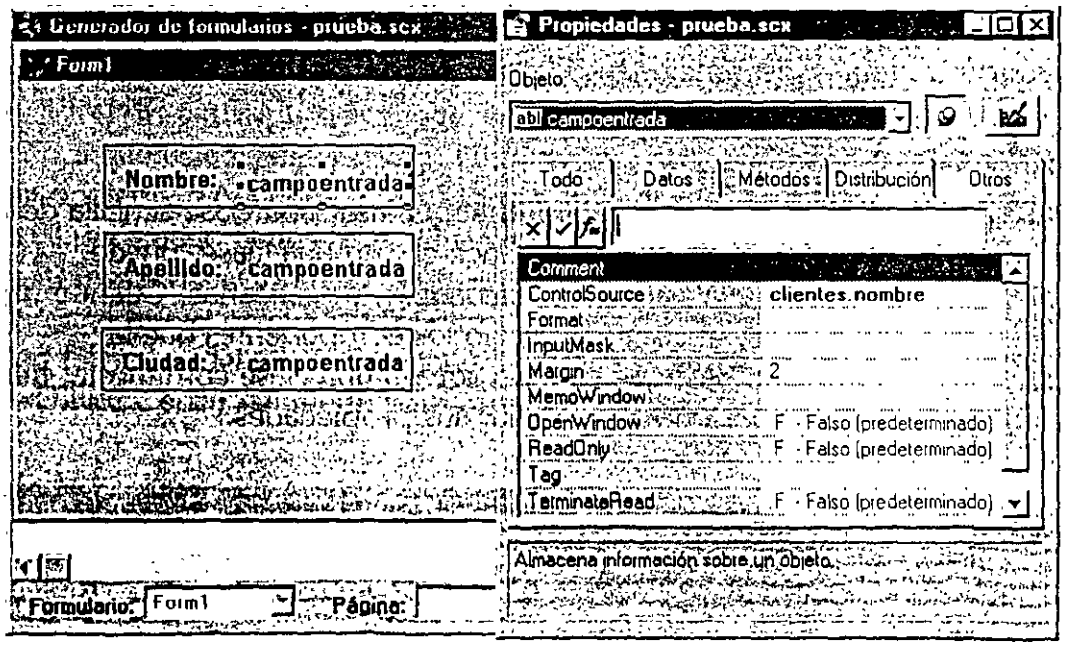
Figura 14-11

Una clase compuesta con varios elementos.

En la pantalla que se exhibe en la figura 14-11, he agregado un ejemplo de la nueva clase compuesta. La página de propiedades a la derecha muestra el nombre del elemento seleccionado, con sus componentes debajo de él. Note que, si la clase fue nombrada ContenedorEntrada, el nombre del primer ejemplo será Contenedorentrada1, el segundo Contenedorentrada2, y así sucesivamente.

Para cambiar el Título de la etiqueta, usted tiene que ir al componente Mietiqueta1, luego a la página Otras propiedades, para encontrar Caption y cambiarla, como se ve en la figura 14-12. De igual forma, debe seleccionar el elemento Campoentrada (hacerlo el objeto seleccionado en ese momento) antes de que pueda cambiar su propiedad ControlSource.

Figura 14-12



Cree una ControlSource para Mietiqueta1.

Hay dos formas de seleccionar un elemento de un objeto compuesto. El primero es usar el botón secundario del ratón para seleccionar un contenedor, luego seleccionar Modificar del menú contextual. Un borde verde sombreado encerrará el objeto contenedor. Entonces usted puede hacer clic en sus elementos constituyentes para seleccionar alguno de ellos. El otro camino es mediante el cuadro combinado de selección de objetos en la página de propiedades para moverse a través de elementos parecidos. Haga clic en un contenedor, muévase hacia el cuadro combinado y ábralo, y usted verá los nombres de los elementos del contenedor inmediatamente debajo del nombre del objeto contenedor.

Una característica útil de la página de propiedades es que, si Caption está resaltado en página Otros de la página principal Propiedades, se mantiene

338

resaltado cuando usted selecciona otro objeto del mismo tipo del cuadro combinado en la página de propiedades. Así que es relativamente fácil cambiar, por ejemplo, todos los títulos para un tipo de objeto en particular, mientras usted no salga de la página de propiedades. Después de cambiar la propiedad para un elemento, simplemente despliegue hacia abajo el cuadro combinado y seleccione su próximo objetivo.

Intente un experimento sencillo. Cree un formulario de prueba y agréguele cuatro ContenedorEntrada que sean Nombre, Apellido, Direccion1 y Ciudad. Seleccione el primer objeto ContenedorEntrada, vaya a la página de propiedades, y encuentre su componente Mietiqueta de la lista desplegable. Luego seleccione la página Otras propiedades, resalte Caption y cámbiela a Nombre:.

Ahora, sin hacer clic en el formulario, utilice la lista desplegable para seleccionar la próxima aparición de un objeto Mietiqueta. Como la página de propiedades no pierde el enfoque, Caption aún está seleccionado. Sólo escriba el nombre del título del siguiente campo, Apellido:. Haga lo mismo con Direccion1 y Ciudad. Repita el proceso para el elemento Data. Este procedimiento es especialmente útil con fuentes de datos, debido a que escoge automáticamente el próximo campo en el archivo.

Yo he encontrado que este enfoque, a diferencia de hacer clic en el objeto mismo para hacer cambios personalizados, funciona mejor con clases compuestas.

Agregue clases de botones de comandos

Agregar botones de comandos es más fácil, debido a que para la mayor parte usted puede usar exactamente los mismos botones que utilizó en el formulario Cliente en el capítulo 9. Escriba:

```
MODI FORM../CAP09/CLIENTE
```

Luego tome cada uno de los botones de navegación y haga clic en Archivo, Guardar como clase del menú. Guarde cada uno bajo un nombre parecido a cmdSiguiente, cmdAnterior, etcétera, en PINTER.VCX. Tendrá que hacer algunos cambios para los botones Modificar y Agregar. En los ejemplos del capítulo 9, la última línea del código de cmdEditar.Click fue:

```
THISFORM.Apellido.SetFocus
```

y para cmdAgregar.Click fue:

```
THISFORM.IDCliente.SetFocus
```

Obviamente, un botón genérico Agregar o Modificar no se puede referir al nombre de un objeto específico en una pantalla específica. De esta forma el código sale y es reemplazado por:

```
cmdEditar::Click
THISFORM.apellido.SetFocus   && o IDCliente para cmdAgregar
```

La línea cmdEditar::Click es necesaria debido a que, si el código se encuentra en cualquier lugar en la jerarquía de métodos de clases, Visual FoxPro deja de buscar la cadena para ejecutar código adicional. Usted tiene que llamar explícitamente el código de la clase Click, luego continuar la ejecución del código que usted agregó al objeto del método Click.

Subclases

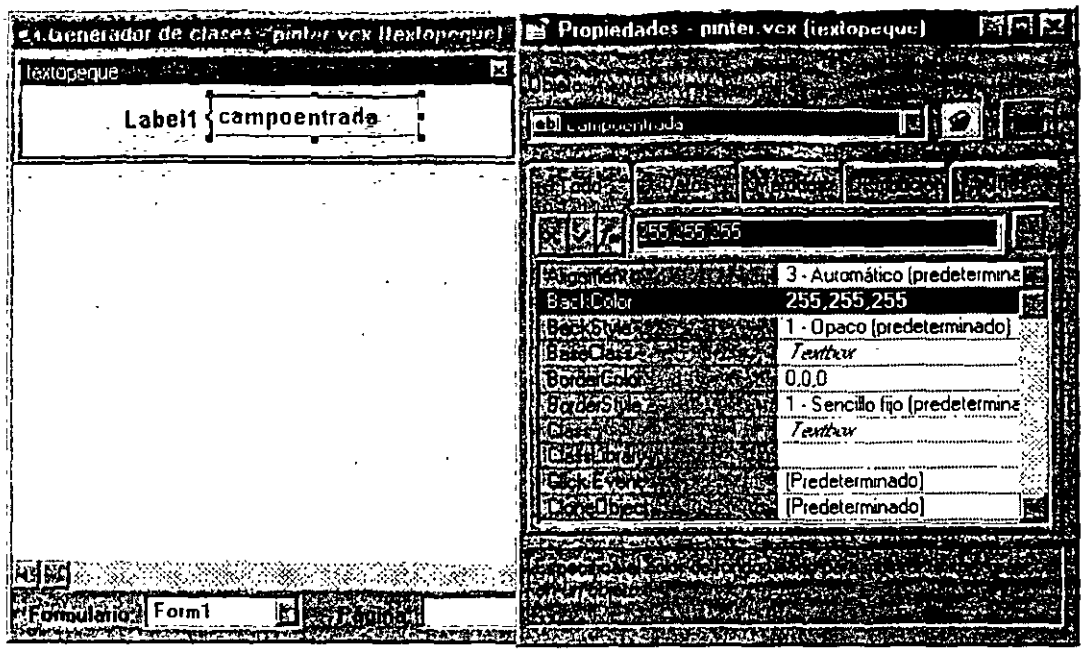
Subclase significa simplemente usar una clase existente como un punto de partida para una clase nueva. Esto es un poco diferente de la clase contenedora que usted acaba de crear. Aquí sólo necesita crear una copia idéntica de otra clase y cambiar una propiedad. La razón es que el formulario Ordenes estará ajustado de espacio, así que tendrá que crear una versión con fuentes más pequeñas de ContenedorEntrada. Escriba:

```
CREATE CLASS Textopeque OF PINTER AS ContenedorEntrada FROM PINTER
```

para obtener la pantalla mostrada en la figura 14-13. Usted también puede escribir sólo CREATE CLASS textopèque y hacerlo visualmente, pero ésta es la forma "correcta" de hacerlo todo con un solo paso.

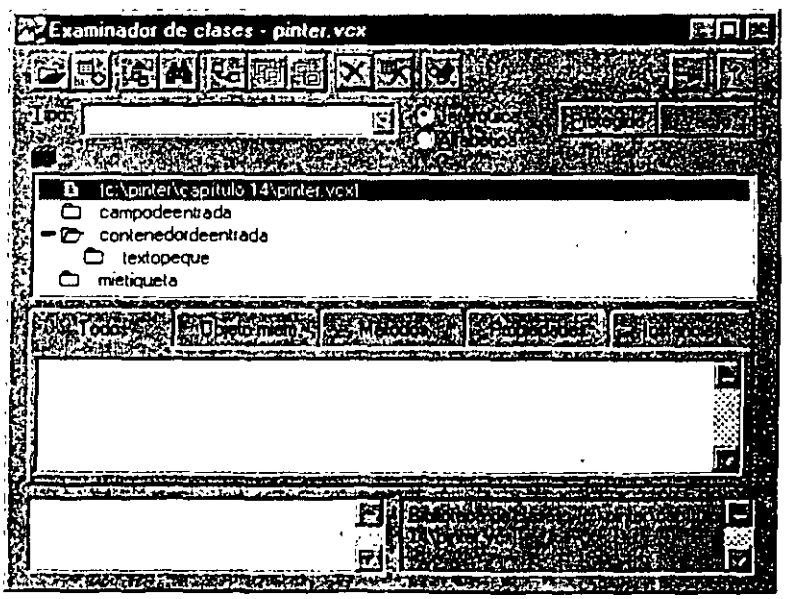
Luego, cambie el tamaño de la fuente a 9 puntos, reduzca la altura a 20 pixeles, y guarde la clase en ClaseLib PINTER.VCX. Ahora, cuando usted agregue un campo de texto al formulario Ordenes usando esta clase, el valor predeterminado del tipo de letra será Courier New negritas de 9 puntos. El resultado se muestra en la figura 14-14.

Figura 14-13



Una subclase Textopeque de ContenedorEntrada.

Figura 14-14



Complemento de la biblioteca de clases PINTER.

34

Agregue una plantilla de formulario

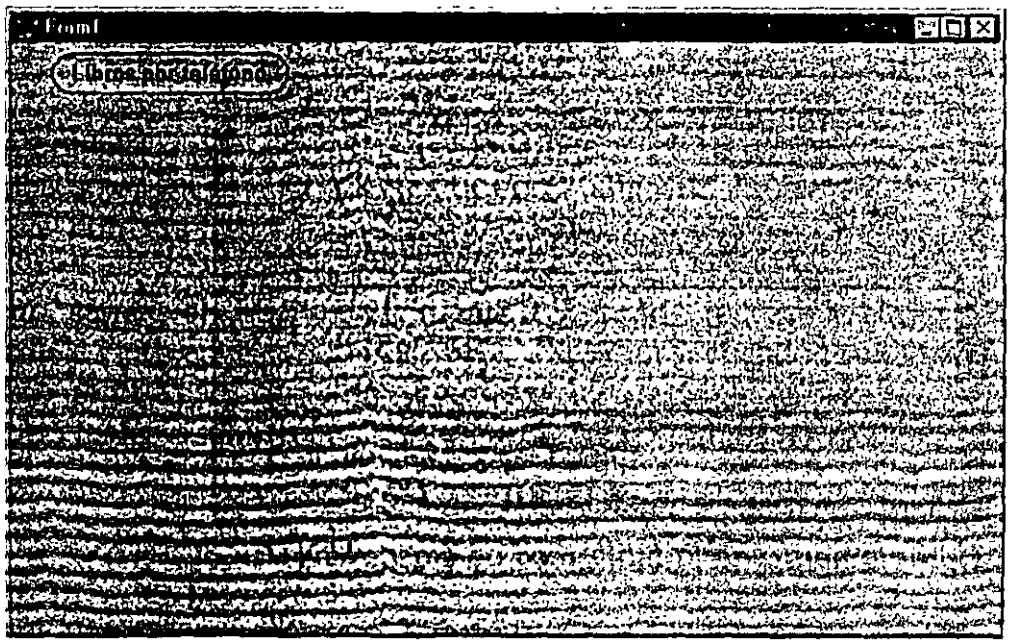
A usted le hace falta una clase final: la plantilla que será el patrón para las otras pantallas. Los modestos requerimientos son color de fondo gris y un tamaño razonable, así que cree un formulario maestro -llámelo Maestro- y establezca sus propiedades Height, Width, AutoCenter y BackColor como lo desee. Aquí se utilizaron estos parámetros:

```
Height     = 329
Width      = 600
BackColor  = RGB ( 192,192,192)
AutoCenter = .T.
```

Finalmente, añada un modesto logotipo en la esquina superior izquierda de la pantalla. En este ejemplo se utilizó un rectángulo con una propiedad Curvature de 25, lo cual produce un óvalo agradable, y una etiqueta con el título "Libros por teléfono". Como forma parte de la plantilla, aparecerá automáticamente en cada formulario, dándole una agradable consistencia profesional sin ningún esfuerzo adicional.

Por último, seleccione Guardar como clase del menú Archivo y nómbrelo MAESTRO.VCX. Luego haga clic en Herramientas, Opciones, Formularios y utilice MAESTRO.VCX como el nombre de la plantilla de formularios. El resultado se muestra en la figura 14-15.

Figura 14-15



Plantilla maestra.

342

Pero no he terminado todavía. Usted también puede utilizar la plantilla Maestro para almacenar funciones llamadas por los botones, ellas estarán ahí en cada uno de los formularios que se hayan obtenido de la plantilla. Como los métodos utilizan muchas propiedades de formularios, por ejemplo, Adicionando y GuardaReg, también podría agregarlos a la plantilla. Una vez que se han agregado a la plantilla, quedan disponibles para cualquier ejemplo de formulario que se base en la plantilla.

Primero utilice la selección de menú Clase, Nuevo método, para agregar los nombres de los métodos BotonesSi, BotonesNo, EnableTodo, DisableTodo y Botones. Copie el código de método correspondiente de ..\CAP09\CLIENTE.SCX utilizando CTRL-C y CTRL-V.

Unos cuantos cambios menores son necesarios. Recuerde que EnableTodo se utiliza para activar los campos de cuadros de texto para entrada de datos. Anteriormente usted tenía cuadros de texto para campos de entrada, así que el código para EnableTodo era como esto:

```
THISFORM.SetAll ( "Enabled", .T., "TextBox" )
```

Cuando usted agrega una clase contenedora, el código correspondiente es:

```
THISFORM.SetAll ( "Enabled", .T., "ContenedorEntrada" )
```

Sin embargo, si prueba este código, encontrará que activar InputContainers no activa sus elementos constituyentes. Así que el elemento TEXTO de TextBox seguirá sin aceptar entradas. La solución es activar todo. Usted puede seguir usando SetAll:

```
THISFORM.SetAll ( "Enabled", .T. )
```

El método DisableTodo requiere de cambios ligeramente distintos en su código. Los botones de comando son ahora instancias de los nombres de clases cmdSiguiente, cmdAnterior y así por el estilo. No puede seguir identificándolos como botones de comandos, ya que ellos están ejemplificados dentro de los formularios como ejemplos de sus nombres de clases. Pero sus nombres les dan una forma de identificarlos, como cmdSiguiente1, cmdAnterior1, etcétera, empiezan con los tres caracteres prefijos cmd. Así que haga que el código BotonesSi se vea como esto:

```
FOR I = 1 TO THISFORM.ControlCount
   IF UPPER(THISFORM.Controls(I).Name) = [CMD]
      THISFORM.Controls(I).Enabled = .T.
   ENDIF
ENDFOR
```

De forma parecida, el código para BotonesNo es éste:

```
FOR I = 1 TO THISFORM.ControlCount
   IF UPPER(THISFORM.Controls(I).Name) = [CMD]
      THISFORM.Controls(I).Enabled = .F.
   ENDIF
ENDFOR
```

Si su formulario no tiene botones de comando excepto los controles de navegación, usted puede usar simplemente:

```
THIS FORM. SetAll ("Enabled", .F., "CommandButton")
```

Se agregó el código UPPER(...[CMD] después de encontrar que Visual FoxPro ha puesto las clases en mayúsculas en lugar de minúsculas, Cmd. O quizá no fue él. ¿Quién sabe? Sólo siga así mientras funcione...

Finalmente, el método Buttons se utiliza después de que Siguiente, Anterior o Buscar movieron el puntero de registros a un registro distinto para activar o desactivar los botones VCR apropiadamente (por ejemplo, de esta forma Siguiente no estará activado si usted ya está en el último registro de la tabla). Aquí está el código:

```
GuardaReg = RECNO()
IF EOF()
   THISFORM.cmdSiguiente.Enabled = .F.
  ELSE
   THISFORM.cmdSiguiente.Enabled = .T.
   SKIP
   IF EOF()
      THISFORM.cmdSiguiente.Enabled = .F.
      GO BOTTOM
   ENDIF
ENDIF
IF BETWEEN ( GuardaReg, 1, RECCOUNT() )
   GO ( GuardaReg )
ENDIF
IF BOF()
   THISFORM.cmdAnterior.Enabled = .F.
  ELSE
   THISFORM.cmdAnterior.Enabled = .T.
   SKIP -1
   IF BOF()
   THISFORM.cmdAnterior.Enabled = .F.
      GO TOP
   ENDIF
ENDIF
```

```
IF BETWEEN ( GuardaReg, 1, RECCOUNT() )
   GO ( GuardaReg )
ENDIF
```

Esto lleva a la nota final en el uso de las clases para agregar botones de comando. Note que el código en el método Buttons necesita referirse específicamente a los nombres de los botones Siguiente y Anterior. Si usted agregó botones de comando de una biblioteca de clases, Visual FoxPro agregará un sufijo de 1 para el primero, 2 para el siguiente, y así sucesivamente. Desafortunadamente, el botón Siguiente es cmdSiguiente y no cmdSiguiente1. Por esta razón, usted tiene que eliminar los prefijos de numeración ascendente de los finales de los nombres de todos los botones de comandos.

El último método que hay que agregar es el código para el botón Buscar, el método EncontrarReg, que es como esté:

```
GurdaReg = RECNO()
IF NOT "JKEY" $ SET("LIBRARY")
   SET LIBRARY TO JKEY ADDITIVE
ENDIF
IF EMPTY ( ORDER() )
   SET ORDER TO 1
ENDIF
DEFINE WINDOW BROWSER FROM 3,3 TO 40, 90 FONT "Courier", 12
_JExitKey = 13
=JKeyInit("U")
BROWSE WINDOW BROWSER
=JKeyCanc()
RELEASE WINDOWS BROWSER
IF LASKEY() = 27
   IF BETWEEN ( GuardaReg, 1, RECCOUNT() )
      GO ( GuardaReg )
   ENDIF
ENDIF
THISFORM.Refresh
```

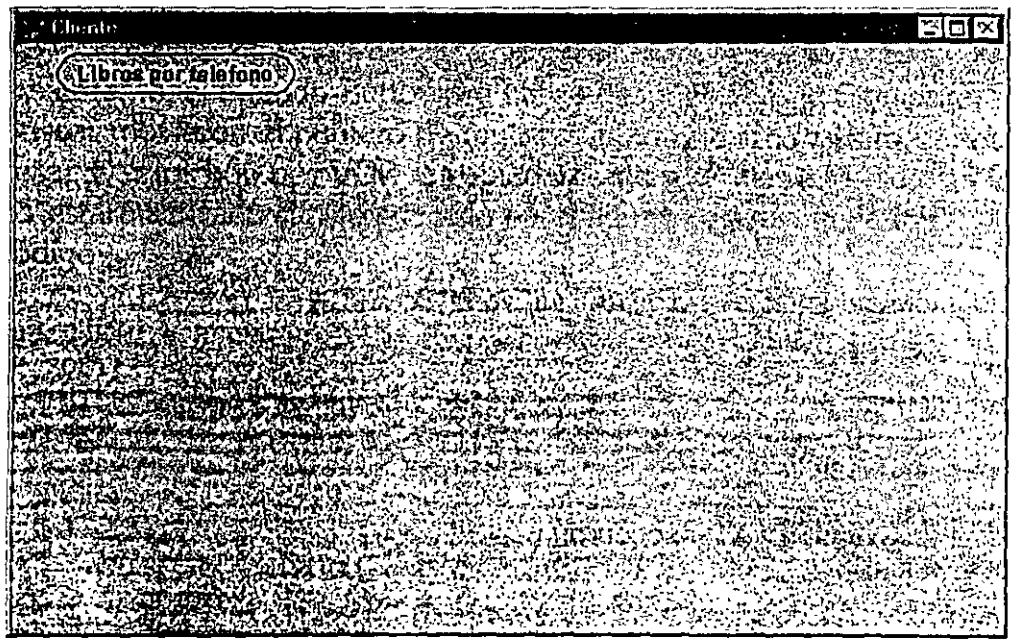
Esto va en la biblioteca de clases PINTER.VCX. EncontrarReg utiliza la utilería JKEY de Joe Gotthelf, la cual convierte a BROWSE en una búsqueda con incrementos basada en el orden actual del índice. Aquí se incluyó el código SET ORDER TO 1, que significa la clave principal, si no se había establecido un orden cuando BROWSE fue llamado. Hay tablas en las que la selección del índice deseado es mucho más importante de lo que es en un caso sencillo, pero esto fue hecho para ilustrar el uso de JKEY. (Si usted no tiene JKEY, puede bajarlo del foro Pinter de CompuServe por unos cuantos centavos -sólo escriba GO PINTER.)

Puede definir una ventana Examinar con el fin de desplegar registros en una ventana más grande; la ventana predeterminada es muy pequeña en Windows. La función JKeyInit("U") convierte lo que el usuario haya escrito a mayúsculas, y JKeyCanc( ) desactiva JKEY cuando usted ha terminado. La línea:

```
_JExitKey = 13
```

ocasiona que JKEY regrese cuando la tecla ENTER (código ASCII 13) se oprime. JKEY tiene algunas otras opciones útiles. Yo entiendo que Joe finalmente ha hecho caso al consejo que se le daba a menudo, e hizo a JKEY un producto shareware. Eso significa que si usted quiere, puede mandarle unos cuantos dólares. Por favor asegúrese de registrar su copia de JKEY. Viene con documentación completa y llena de ejemplos. Yo no escribo una aplicación FoxPro sin ella.

Figura 14-16



El nuevo formulario Cliente.

Cree un formulario Cliente utilizando clases

Ahora usted está listo para empezar a usar las clases que ha diseñado. Haga clic en Formulario, Nuevo, y seleccione Nuevo formulario (no Asistente para formularios). *Voilà*, ahí está el formulario Maestro. Dé al formulario vacío el título CLIENTE así que se verá como la pantalla de la figura 14-16.

Tiene que agregar la tabla Cliente al entorno de datos si usted espera conseguir que Visual FoxPro realice la mayor parte del trabajo más tarde, así que seleccione Ver, Entorno de datos, haga clic en el icono Entorno de datos, en la barra de

346

herramientas Generador de formularios, o utilice el botón secundario del ratón para desplegar el menú contextual y seleccionar Entorno de datos. Luego utilice el botón secundario del ratón para agregar Cliente a la pantalla gráfica.

* Agregue controles al formulario Cliente

Ahora usted está listo para usar sus nuevas clases. Despliegue la barra de herramientas Controles de formularios, si es que aún no está desplegada, y haga clic en el icono de los libros. Seleccione Agregar, luego escoja Pinter como la biblioteca de clases a usar. La barra de herramientas Controles de formularios se verá como en la figura 14-17. Si usted mueve el puntero del ratón hacia cada uno de los controles de enmedio, sus cuadros de consejos mostrarán sus nombres: Mietiqueta1 y Campoentrada1.

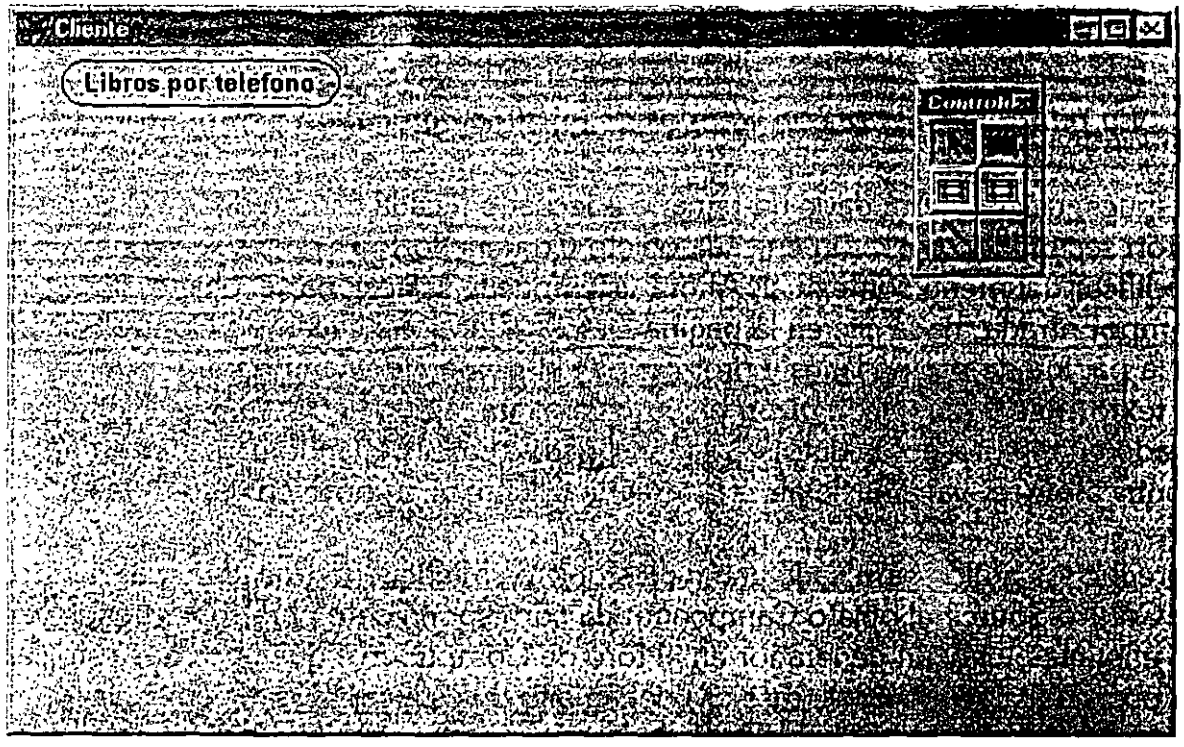
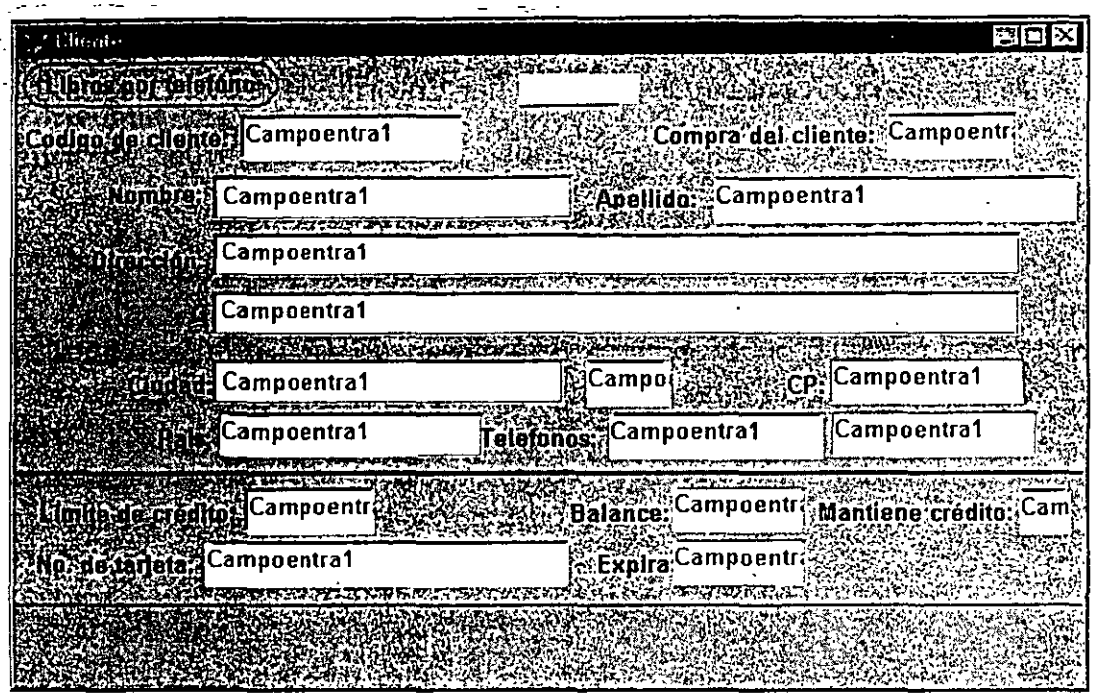


Figura 14-17

La barra de herramientas Controles de formulario con la biblioteca de clases PINTER.VCX.

Agregue un objeto ContenedorEntrada para cada uno de los campos en el formulario Cliente del capítulo 9. (Presione CTRL-F2 para regresar a la ventana Comandos, luego escriba MODI FORM ../CAP09/CLIENTE, después utilice CTRL-F1 para cambiarse dentro de los dos formularios hasta que los haya dispuesto como se muestra en la figura 14-18.) El bloqueador de botones se vuelve útil aquí.

Figura 14-18



La pantalla Cliente con campos de entrada agregados.

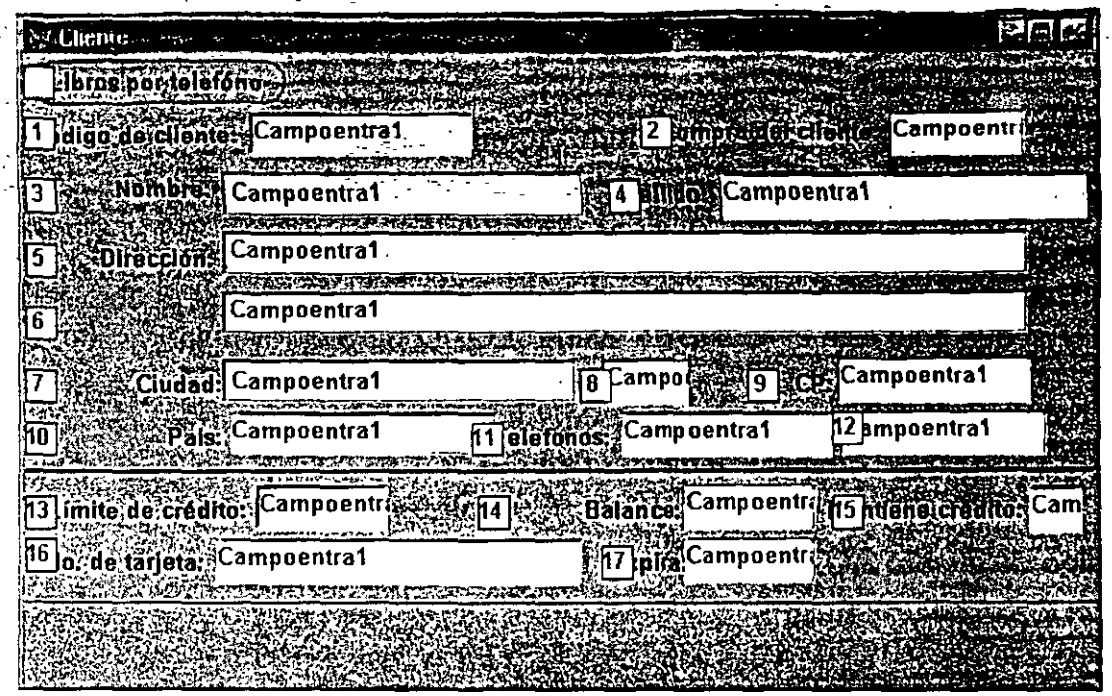
Después, usted querrá darle un tamaño adecuado a cada campo. Haga clic con el botón secundario del ratón en cada objeto ContenedorEntrada y seleccione Modificar del menú contextual. Ahora usted puede seleccionar el objeto CampoEntrada, haga clic en la página Datos, seleccione el campo ControlSource en la página de propiedades y proporcione el nombre de campo para cada cuadro de texto. Recuerde que la razón para que esto trabaje es debido a que Cliente está listado en el entorno de datos. Así que si nada aparece cuando haga clic en ControlSource, vaya hacia atrás y agregue Cliente al entorno de datos.

Agregue los títulos para cada uno de los objetos Mietiqueta. Puede buscar dentro de ControlSource el objeto Campoentrada correspondiente para saber dónde van las etiquetas. Hay un generador en el foro de CompuServe que hace que esto sea un poco más fácil, pero va más allá del ámbito de este capítulo. Además, la mayoría de la gente aprende haciendo las cosas, así que este pequeño ejercicio es realmente educativo. ¿No es grandioso no tener que establecer la alineación y el BackStyle para cada una de estas creaciones? El resultado se muestra en la figura 14-19.

Ahora añada el grupo de botones de navegación al final de la pantalla. Usted debe añadir sus propios botones Agregar, Editar y Encontrar ya que son un poco distintos en cada pantalla. Aun así, despúes de todo el trabajo que ha guardado, un poco de código probablemente será bienvenido. Para finalizar añada un par de líneas para separar los botones de comando y los campos de entrada al final de la pantalla, se verá como luce la figura 14-20.

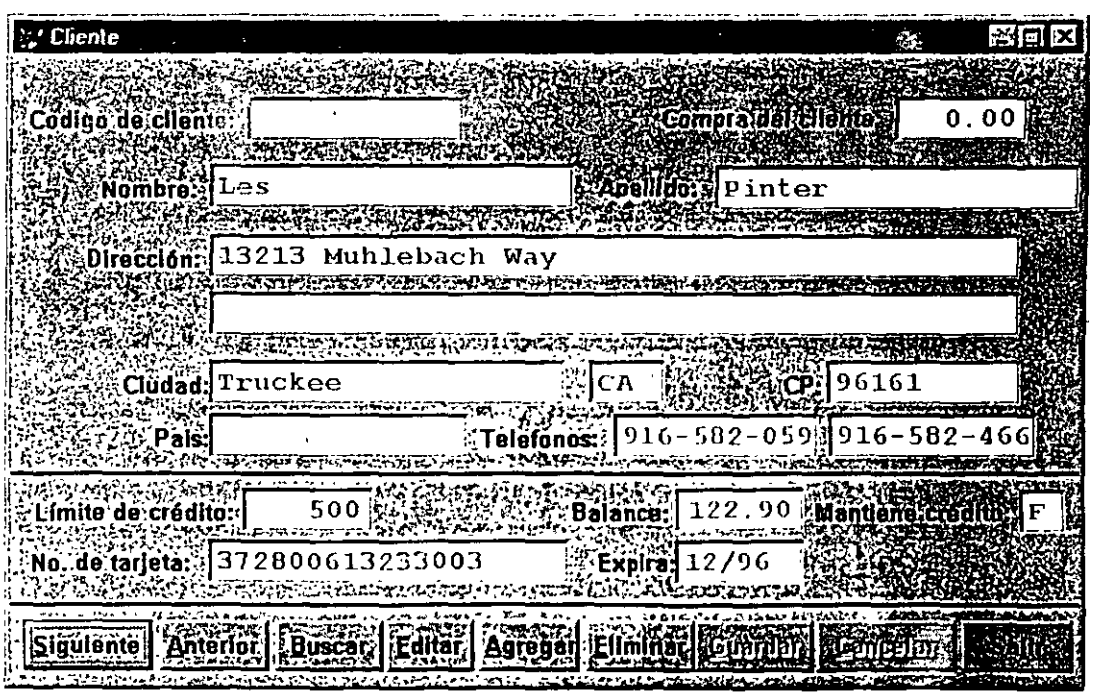
348

Figura 14-19



El formulario Cliente con títulos de etiqueta agregados.

Figura 14-20



Formulario Cliente terminado.

Es la hora del show. Haga clic en Formulario, Ejecutar del menú, luego seleccione Modificar. Notará que los campos de entrada se comportan exactamente como deberían.

* Agregue el código de los eventos LOAD y ACTIVATE

Generalmente se encuentra que, en el caso de Visual FoxPro, lo mejor es simplificar el uso del entorno de datos del formulario para manejar el abrir y cerrar tablas. Así que aquí está el código de evento LOAD para el formulario Cliente:

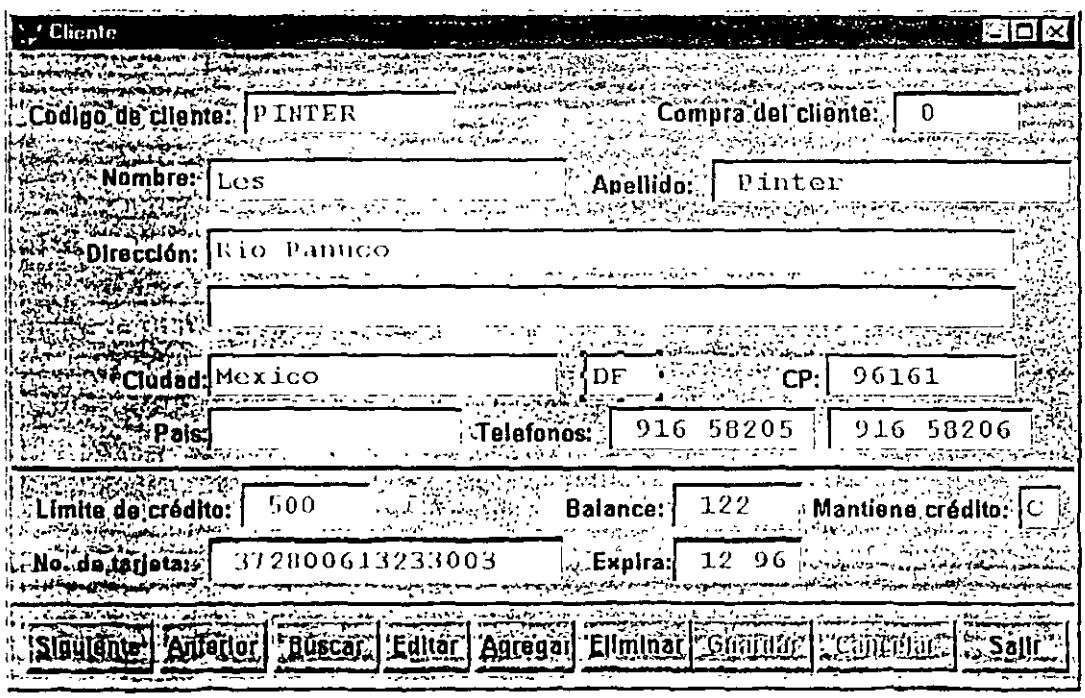
```
SELECT Cliente
SET ORDER TO IDCliente
=CURSORSETPROP("Buffering",3)
```

El código de evento ACTIVATE para el formulario Cliente es aún más corto:

```
SELECT Cliente
```

Ahora utilice la opción de menú Formulario, Ejecutar formulario para probar su trabajo. Deberá ver la pantalla de la figura 14-21. El proceso entero de generar un formulario creando sus clases debe tomar sólo tres o cuatro minutos. El resultado es un formulario de apariencia profesional que habría tomado mucho más tiempo para escribirse en FoxPro 2.6. Sin tomar en cuenta cuánto tiempo tomaría en código C o Pascal.

Figura 14-21



El formulario Cliente, terminado en acción.

350

Genere el formulario Libros

Generar el formulario Libros es simple ahora que usted ha creado sus clases. Repita el proceso de agregar todos los campos de la tabla hasta que usted tenga el formulario mostrado en la figura 14-22. El único objeto que no es una clase es la región Modificar, la cual se usa sólo en el formulario Libros. Usted puede crear una clase Modificar con el código GotFocus y LostFocus de su clase TextBox como un ejercicio.

El formulario Libros necesita los cambios apropiados al código de evento LOAD. Como ahora usted tiene un nombre de tabla diferente abra la ventana Propiedades y escriba en el código del evento LOAD y ACTIVATE (las siguientes dos secciones). Luego haga clic en Formulario, Ejecutar y pruébelo.

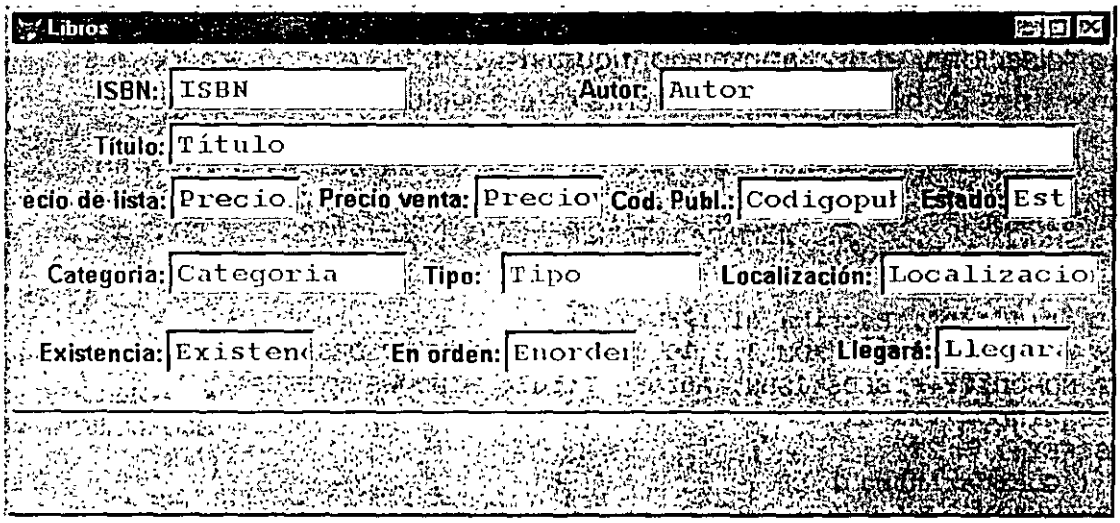


Figura 14-22

El formulario Libros.

```
IF NOT USED ( "LIBROS" )
   USE LIBROS IN 0
ENDIF
SELECT LIBROS
SET ORDER TO TITULO
=CURSORSETPROP("Buffering",3)

SELECT LIBROS
```

Usted utilizará los mismos botones de comando en Libros que los que utilizó en Cliente, con una excepción. En Libros se puede hacer una búsqueda por título, autor, ISBN o palabra clave. En los tres primeros se usan directamente JKEY

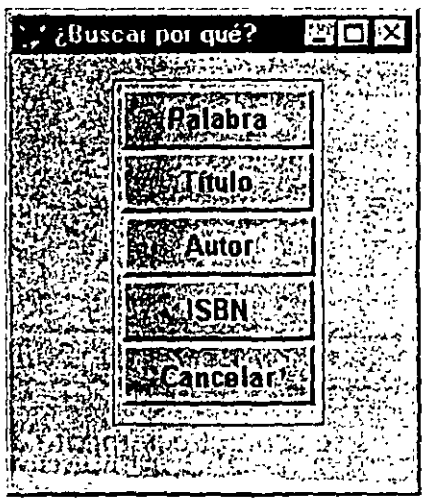
después de establecer el orden del índice apropiadamente, pero a qué se agregó la pantalla TipoBus, la cual es como esto:

```
THISFORM.GuardaRegNo = RECNO()
Llave= []
DO FORM TipoBus
SELECT LIBROS
SET ORDER TO TITULO
IF LASTKEY() <> 27
   SEEK Llave
  ELSE
   IF BETWEEN ( THISFORM.GuardaRegNo, 1, RECCOUNT() )
      GO ( THISFORM.GuardaRegNo )
   ENDIF
ENDIF
THISFORM.Refresh
THISFORM.Buttons
```

El formulario TipoBus despliega los tipos de búsqueda disponibles, y le permite al usuario seleccionar el tipo apropiado (figura 14-23). El código para los tres primeros tipos de búsqueda simplemente está basado en JKEY y establece el orden apropiado del índice. Una búsqueda por ISBN se ve como esto:

```
SELECT LIBROS
GuardarAntes = RECNO()
SET ORDER TO ISBN
=JKeyInit("U","","Escriba el ISBN:  ")
DEFINE WINDOW VIEWER FROM 3,3 TO 33, 90 SHADOW IN DESKTOP
BROWSE NOMODIFY FIELDS ISBN, Autor, Titulo WINDOWS VIEWER
RELEASE WINDOW VIEWER
IF LASTKEY() = 27
   GO ( GuardarAntes )
ENDIF
Llave = ISBN
RELEASE THISFORM
```

Figura 14-23



El formulario Tipo de búsqueda.

352

y una búsqueda por autor es similar:

```
SELECT LIBROS
GuardarAntes = RECNO()
SET ORDER TO Autor
=JKeyInit("U","","Escriba el autor:  ")
DEFINE WINDOW VIEWER FROM 3,3 TO 33, 90 SHADOW IN DESKTOP
BROWSE NOMODIFY FIELDS Autor, Titulo WINDOWS VIEWER
RELEASE WINDOW VIEWER
IF LASTKEY() = 27
   GO ( GuardarAntes )
ENDIF
Llave = ISBN
RELEASE THISFORM
```

Para la búsqueda por título recordará que se deben separar las palabras *un, una* y *el* del principio del título de la etiqueta del índice TITLE, así que un mensaje apropiado sería útil para el usuario:

```
SELECT LIBROS
GuardarAntes = RECNO()
SET ORDER TO Titulo
=JKeyInit("U","","Escriba el título - omita LOS, UN o EL:  ")
DEFINE WINDOW VIEWER FROM 3,3 TO 33, 90 SHADOW IN DESKTOP
BROWSE NOMODIFY FIELDS Titulo, Autor WINDOWS VIEWER
RELEASE WINDOWS VIEWER
IF LASTKEY() = 27
   GO ( GuardarAntes )
ENDIF
Llave = ISBN
RELEASE THISFORM
```

Sin embargo, las palabras clave son una desviación más grande de este simple enfoque. Las palabras clave están localizadas en un archivo separado con dos campos: CLAVE e ISBN. Usted cuenta el número de palabras que coinciden y lo despliega. Los usuarios deciden cuándo hacen una tabla y despliegan los títulos que coincidan.

La razón para esto se aclarará al final de este capítulo. Si usted hace esto dentro de una aplicación remota cliente-servidor, la parte más importante de la optimización necesaria para proporcionar un desempeño aceptable incluye las búsquedas por palabra clave. Así que el código del método Click de la búsqueda por palabra clave es como esto:

```
Llave = []
DO FORM BuscarLibro
IF LastKey() <> 27
```

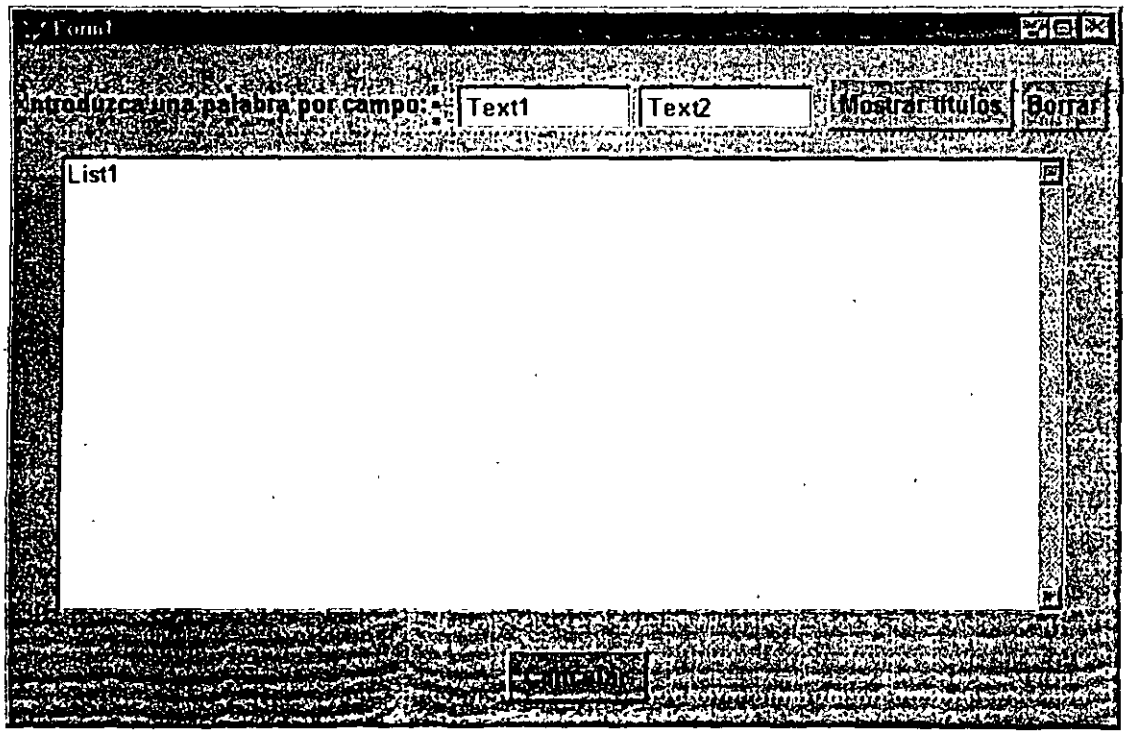
```
    SELECT Libros
    SET ORDER TO ISBN
    SEEK Llave
ENDIF
RELEASE THISFORM
```

Ahora usted necesita modificar la pantalla BuscarLibro en varias formas.

Agregue el formulario BuscarLibro

La pantalla BuscarLibro se muestra en la figura 14-24. Los dos botones de comando principales son Text1 y Text2 (los dos cuadros de texto), los cuales cuentan y despliegan el número de títulos que contiene la palabra clave que fue introducida en el cuadro de texto respectivo.

Figura 14-24



Pantalla BuscarLibro.

Los conteos son desplegados en etiquetas que aparecen directamente debajo de Text1, Text2 y cmdContar. Usted simplemente construye un título para cada palabra clave que haya sido contada:

```
Parameters nCodigoLlave, nMayusAltCtrl

IF BETWEEN ( nCodigoLlave, ASC("A"), ASC("Z") ) ;
OR BETWEEN ( nCodigoLlave, ASC("a"), ASC("z") )
   UltimoCar = UPPER(CHR(nCodigoLlave))
  ELSE
   UltimoCar = []
ENDIF
* Esto pasa ANTES de que la última tecla sea procesada, así la simula
Llave = TRIM(THISFORM.Text1.Value)
IF nCodigoLlave = 127
   a = 1
   Llave = IIF(LEN(Llave)=0,Llave,LEFT(Llave,LEN(Llave)-1))
  ELSE
   Llave = TRIM(THISFORM.Text1.Value) + UltimoCar
ENDIF
SELECT CLAVE
SEEK Llave
COUNT TO THISFORM.c1 WHILE Clave = Llave
THISFORM.Count1.Caption=ALLTRIM(STR(THISFORM.c1))+[ Encontro ]+Llave

Parameters nCodigoLlave, nShiftAltCtrl
IF BETWEEN ( nCodigoLlave, ASC("A"), ASC("Z") ) ;
OR BETWEEN ( nCodigoLlave, ASC("a"), ASC("z") )
   UltimoCar = UPPER(CHR(nCodigoLlave))
  ELSE
   UltimoCar = []
ENDIF
Llave= TRIM(THISFORM.Text2.Value)
IF nCodigoLlave = 127
   a = 1
   Llave = IIF(LEN(Llave)=0,Llave,LEFT(Llave,LEN(Llave)-1))
  ELSE
   Llave = TRIM(THISFORM.Text2.Value) + UltimoCar
ENDIF
SELECT CLAVE
SEEK Llave
COUNT TO THISFORM.c2 WHILE Clave = Llave
THISFORM.Count2.Caption=ALLTRIM(STR(THISFORM.c2))+[ Encontro ]+Llave
```

Mostrar títulos (objeto cmdMostrar) encuentra los títulos que coinciden con las dos palabras clave. En la versión cliente-servidor de este programa, este conteo se hace utilizando un procedimiento almacenado en el servidor. El usuario es interrogado antes de que sean desplegados los títulos, debido a que éstos deben ser enviados a través del módem.

```
Llave1 = TRIM(THISFORM.Text1.Value)
Llave2 = TRIM(THISFORM.Text2.Value)
IF NOT EMPTY ( Llave2 )
   DO CASE
      CASE THISFORM.c1 < THISFORM.c2
         Key1 = TRIM(THISFORM.Text1.Value)
```

```
            Key2 = TRIM(THISFORM.Text2.Value)
         OTHERWISE
            Key1 = TRIM(THISFORM.Text2.Value)
            Key2 = TRIM(THISFORM.Text1.Value)
      ENDCASE
ENDIF
SELECT CLAVE
SEEK Llave1
Encontrados = 0
SCAN WHILE CLAVE.CLAVE = Llave
      SELECT LIBROS
      SEEK CLAVE.CLAVEISBN
      IF       EMPTY(Llave2)          ;
      OR ( NOT EMPTY(Llave2)          ;
           AND Llave2 $ UPPER(Titulo) )
          Encontrados = Encontrados + 1
          THISFORM.List1.AddItem ( LIBROS.ISBN + [- ]'+ LIBROS.Titulo )
      ENDIF
      SELECT CLAVE
ENDSCAN
THISFORM.Cont3.Caption = ALLTRIM(STR(Encontrados)) + [ Encontrados ]
THISFORM.List1.SetFocus
```

Clear quita todas las entradas del cuadro de lista de títulos, de forma que pueden hacer la búsqueda de nuevo. Si no se aplica Clear, la lista continúa creciendo.

```
THISFORM.List1.Clear
THISFORM.List1.Refresh
THISFORM.List1.SetFocus
THISFORM.Text1.SetFocus
IF THIS.ListItemID > 0
   SELECT LIBROS
   SET ORDER TO ISBN
   Llave = ThisList(THIS.ListItemID)
   Llave = LEFT(Llave,10)
   THISFORM.DataSessionID = 1
   IF USED ( "LIBROS" )
         SELECT LIBROS
         SET ORDER TO ISBN
      SEEK Llave
   ENDIF
   THISFORM.Release
  ELSE
   THISFORM.cmdCancelar.Click
ENDIF
Llave = THISFORM.GuardarLlave
THISFORM.release
```

De esta forma el formulario da como resultado precisamente un ISBN de un libro, o ninguno si se hace clic en Cancelar.

356

Construya llaves

La figura 14-25 muestra el nuevo formulario ConstruirLlaves. En el ejemplo se utilizaron los archivos .BMP del subdirectorio VFP/SAMPLES/GRAPHICS/BMPS para crear un pequeño termómetro en la pantalla. La barra vertical de enmedio que crece para reflejar el progreso del proceso es la clase termómetro que se vio en el capítulo 13.

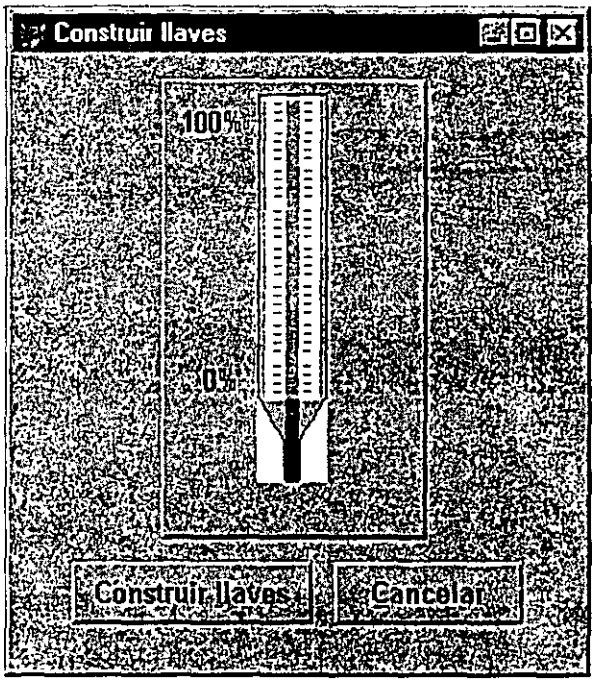


Figura 14-25

Formulario ConstruirLlaves.

Crear una tabla de palabras clave utiliza la misma lógica que antes, pero aquí se agregó un termómetro basado en la clase termómetro del capítulo 13 para desplegar el progreso, debido a que la construcción de la palabra clave puede tomar un rato.

El código del método Click para el botón ConstruirLlaves vacía el archivo de palabras clave, luego llama un método de formulario llamado ConstruirLlaves:

```
IF FILE ( "CLAVE.DBF" )
   IF USED ( "CLAVE" )
      USE IN CLAVE
   ENDIF
   DELETE FILE CLAVE.DBF
ENDIF
IF FILE ( "CLAVE.CDX" )
   DELETE FILE CLAVE.CDX
ENDIF
```

```
CREATE TABLE CLAVE ( CLAVE C(20), LlaveISBN C(10) )
ThisForm.ConstruirLlaves
THISFORM.Release
```

El método ConstruirLlaves es similar a la versión anterior, excepto que ésta llama a la clase termómetro:

```
* Genera el archivo de palabras clave del título
PRIVATE i
DIMENSION palabra(30)
STORE SPACE(10) TO palabra
npalabras = 1
prohibido = ;
   " Y / O / EL / DE / PARA / EN /  POR / "
IF USED ( "LIBROS" )
   SELECT LIBROS
  ELSE
   SELECT 0
   USE LIBROS
ENDIF
SELECT LIBROS
THISFORM.Termometro1.ShpBox.FillColor = RGB ( 255, 0, 0 )
THISFORM.Termometro1.Tipo = 1   && Llenar de abajo hacia arriba...
SCAN
   pct = ( recno()/ RECCOUNT() ) * 100
   THISFORM.Termometro1.Actualiza( pct )
   THISFORM.Parser
   I = 1
   DO WHILE i <= npalabras
      IF NOT (palabra(I) $ prohibido ) AND NOT EMPTY ( palabra(I) )
         INSERT INTO CLAVE VALUES ( palabra(i), LIBROS.isbn )
      ENDIF
      I = I + 1
   ENDDO
   SELECT LIBROS
ENDSCAN
DEFINE WINDOW PROGRESS FROM 1,45 TO 5, 80 DOUBLE SHADOW ;
 TITLE [ Creando índices de llave ]
ACTIVATE WINDOW PROGRESS
SET TALK ON WINDOW PROGRESS
SELECT CLAVE
INDEX ON CLAVE TAG CLAVES
INDEX ON LLAVEISBN TAG LLAVEISBN
SET TALK OFF
RELEASE WINDOW PROGRESS
```

Por último, debe internalizar la rutina Parser como un método dentro del formulario ConstruirLlaves:

```
STORE " " TO palabra
tcadena = UPPER ( TRIM ( LIBROS.TITULO ) )
```

```
nlon = LEN(TRIM(tcadena))
i = 1
cadena = ""
DO WHILE i <= nlon
   cadena = cadena + ;
      IIF ( SUBSTR(tcadena,i,1) $ [#&(:),.] , [-], SUBSTR(tcadena,i,1) )
   i = i + 1
ENDDO
npalabras =
DO WHILE .T.
   finpalabra = AT ( " " , cadena) - 1
   IF finpalabra = -1
      finpalabra = LEN ( cadena )
   ENDIF
   npalabrass = npalabras + 1
   palabra (npalabras ) = SUBSTR ( cadena, 1, finpalabra )
   IF ( (finpalabra+2) > LEN(cadena) ) OR ( LEN(cadena) = 0 )
      EXIT
   ENDIF
   cadena = SUBSTR ( cadena , finpalabra + 2 )
   cadena = LTRIM(cadena)
ENDDO
```

Como el archivo de las palabras clave adjunta los archivos de palabras clave y de libros para sus propios propósitos, hay un método especial UNLOAD:

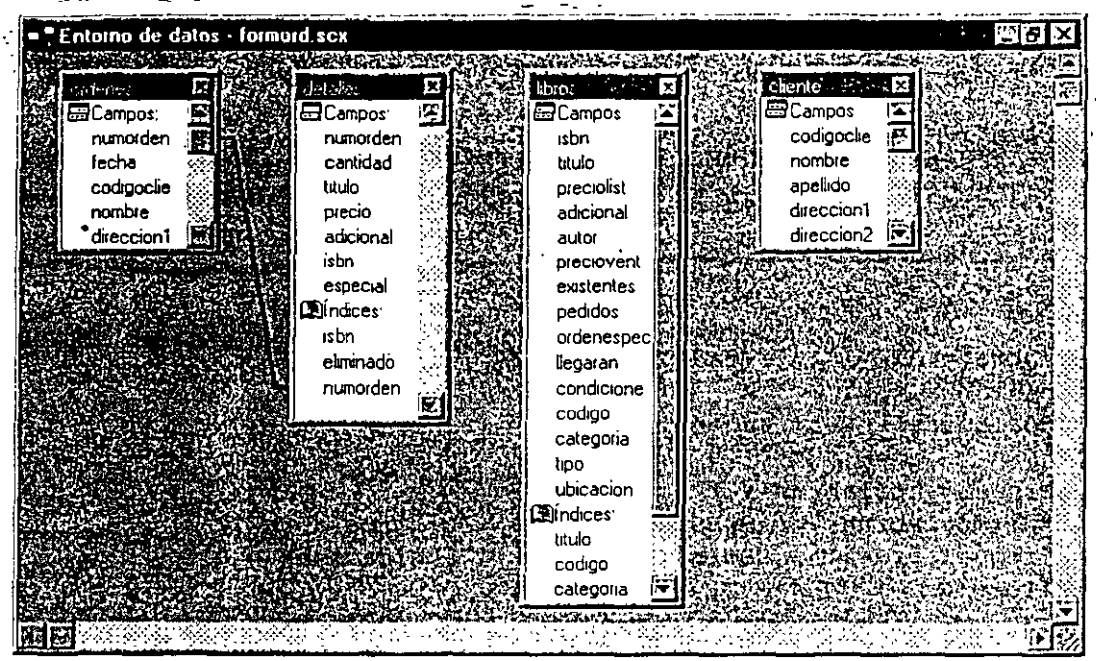
```
IF USED ( "CLAVES" )
   USE IN CLAVES
ENDIF
IF USED ( "LIBROS" )
   USE IN LIBROS
ENDIF
```

Agregue el formulario Ordenes

Haga clic en Formulario, Nuevo y obtendrá la plantilla. Cambie el título del formulario a Ordenes, y ponga las tablas Cliente, Ordenes, Detalles y Libros dentro del entorno de datos como se muestra en la figura 14-26.

Como en el capítulo 9, el formulario Ordenes consiste en campos de una tabla principal Ordenes y una cuadrícula que utiliza registros de una tabla hija llamada Detalles. El entorno de datos manejará la mayoría de la administración por usted. Intente hacer esto de la manera más sencilla posible.

Figura 14-26



El entorno de datos para el formulario Ordenes.

* Agregue los campos de la tabla principal

Como este formulario es un poco corto a diferencia de otros, se utilizará el objeto contenedor textopeque para agregar etiquetas y campos para todo excepto el objeto cuadrícula.

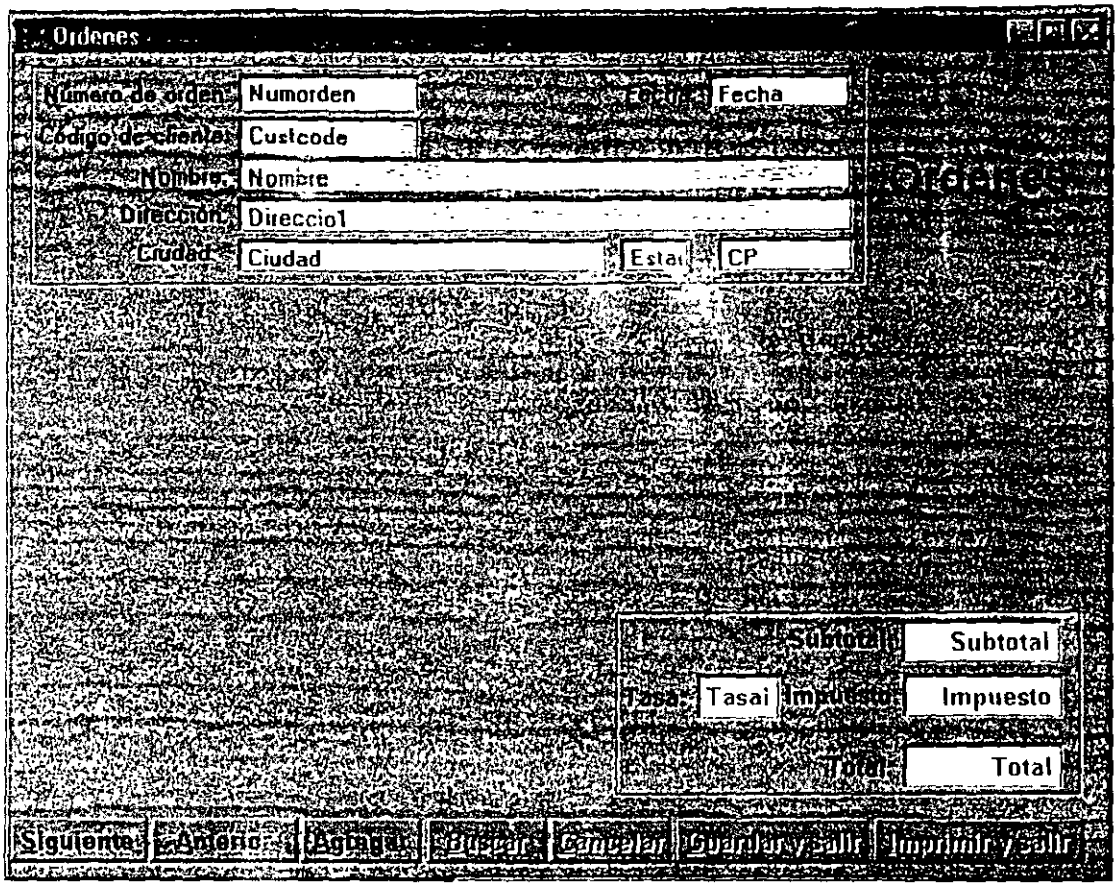
Los primeros dos campos son número de orden y fecha. El ControlSource para número de orden es ORDENES.NumOrden, mientras que ORDENES.Fecha está ligado al cuadro de texto Fecha. Ambos son de sólo lectura, debido a que son generados internamente. Cuando se agrega un nuevo formulario, se incrementa el orden de los números en la tabla CONTROL.DBF antes de introducir en la tabla Ordenes, e insertar la fecha actual al mismo tiempo.,

IDCliente, Nombre, Dirección, Ciudad, Estado y CP están todos ligados a sus campos correspondientes en ORDENES.DBF, como lo están los cuatro campos numéricos al final de la pantalla: Subtotal, PorcenIVA, IVA y TotalOrdenes. Tómelos de la biblioteca de clases PINTER.VCX y haga los cambios necesarios a los títulos y fuentes de control. El formulario terminado antes de agregar la cuadrícula se verá como en la figura 14-27.

* Agregue la cuadrícula para los registros hijos

Con el fin de que trabaje el objeto cuadrícula, usted tiene que proporcionar tablas y relaciones. En el capítulo 9, manejó la coordinación de tablas por usted mismo. Dé al objeto Cuadrícula una oportunidad para hacer todo el trabajo por usted.

Figura 14-27



El formulario Ordenes sin el objeto cuadrícula.

Agregue un objeto cuadrícula en el centro del formulario Ordenes, luego seleccione el marco de página Datos de la página de propiedades e introduzca lo siguiente:

```
ChildOrder - NumOrden
LinkMaster - Ordenes
RecordSource - Detalles
RecordSourceType - 1-Alias
RelationExpr - NumOrden
```

Con el fin de coordinar estos dos archivos para un objeto cuadrícula padre-hijo, el archivo hijo debe tener un índice ordenado en NumOrden. Aquí Detalles es el archivo hijo. Ordenes está ordenado por NumOrden, pero sólo por coincidencia. Usted utilizó un pequeño código para manejar este proceso en el capítulo 9, pero el administrador de datos integrado del objeto cuadrícula lo hará todo por usted.

Código de soporte

El código para la aplicación es muy parecido a lo que usted vio en el capítulo 9. El botón Agregar incrementa el orden de los números y rellena la tabla Detalles con 10 registros en blanco:

```
*(1) Activa los campos de entrada; note que sólo se utilizan 3
THISFORM.IDCliente.Enabled    = .T.
*(2) Desactiva todos los botones de comandos, luego cambia BuscarLibro,
* Guardar, Imprimir y Cancelar a activados
THISFORM.CmdAgregar.Enabled   = .F.
THISFORM.CmdSiguiente.Enabled = .F.
THISFORM.CmdAnterior.Enabled  = .F.

THISFORM.CmdEncontrar.Enabled = .T.
THISFORM.CmdGuardar.Enabled   = .T.
THISFORM.CmdImprimir.Enabled  = .T.
THISFORM.CmdCancelar.Enabled  = .T.

*(3) Incrementa el orden de numeración next
SELECT CONTROL
REPLACE NEXT 1 CONTROL.UltimaOrden WITH CONTROL.UltimaOrden + 1

*(4) Prepara el número de transacción para next
THISFORM.NuevaTransac = ;
   TRANSFORM ( CONTROL.UltimaOrden WITH CONTROL.UltimaOrden, "@L ######" )

* Ahora estamos listos para empezar la transacción
BEGIN TRANSACTION

* (5) Inserta el orden de los registros con dos valores ya introducidos
SELECT ORDENES
INSERT INTO ORDENES ( NumOrden,              Fecha   )  ;
           VALUES ( THISFORM.NuevaTransacNo,   DATE())

* No intente asignar el valor a THISFORM.ProcenIVA,
* el cual es el objeto mismo
THISFORM.PorcenIVA.Value = CONTROL.Default.PorcenIVA

*(6) Inserta 10 registros de detalles en blanco con el número
* de orden actual ya introducido para coordinación de la cuadrícula.
SELECT DETALLES
SET DELETED OFF
FOR I = 1 TO 10
   LOCATE FOR DELETED()
   IF NOT FOUND()
      INSERT INTO DETALLES ( NumOrden ) ;
     VALUES ( THISFORM.NuevaTransacNo )
   ELSE
```

```
      REPLACE NEXT 1 NumOrden WITH THISFORM.NuevaTransacNo
   ENDIF
ENDFOR
SELECT ORDENES
THISFORM.Refresh

*(7) Va al primer campo de entrada
THISFORM.IdCliente.SetFocus
```

Los botones Siguiente y Anterior tienen que desactivar SET SKIP de forma que SELECT ORDENES...SKIP realmente va al orden siguiente:

```
* Código del evento cmdSiguiente.Click:

SELECT ORDENES
SET ORDER TO NUMORDEN
SET SKIP TO
IF NOT EOF()
   SKIP
   IF EOF()
      GO BOTTOM
   ENDIF
ENDIF
SET SKIP TO DETALLES
THISFORM.Refresh

* Código del evento cmdAnterior.Click:

SELECT ORDENES
SET ORDEN TO NUMORDEN
SET SKIP TO
IF NOT BOF()
   SKIP -1
   IF BOF()
      GO TOP
   ENDIF
ENDIF
SET SKIP TO DETALLES
THISFORM.Refresh
```

El botón Buscar está activo sólo durante la introducción de la orden:

```
* Código del evento cmdBuscar.Click:

CLAVE = []
DO FORM TipoBus
SELECT LIBROS
SET ORDER TO ISBN
SEEK CLAVE
SELECT DETALLES
```

```
IF LastKey() <> 27
   SCAN WHILE NOT EMPTY(Titulo) AND NumOrden = THISFORM.NuevaTransacNo
   && Encuentra el siguiente registro en blanco
   ENDSCAN
   REPLACE NEXT 1                                          ;
      DETALLES.ISBN     WITH LIBROS.ISBN           , ;
      DETALLES.Titulo     WITH LIBROS.Titulo       , ;
      DETALLES.Cantidad WITH 1                     , ;
      DETALLES.Precio     WITH LIBROS.PrecioVenta  , ;
      DETALLES.Extendido WITH LIBROS.PrecioVenta   , ;
      THISFORM.Calctotal
      THISFORM.Refresh
ENDIF
```

Se agregó una forma más para obtener el orden del formulario mientras se agrega una orden. Los usuarios pueden presionar la tecla Esc en el proceso de introducir un código de cliente con el fin de cancelar:

```
LPARAMETERS nclavetecla, nShiftAltCtrl
IF nclavetecla = 27
   WAIT WINDOW [ Transaccion cancelada (presione ENTER)]
   ROLLBACK
   THISFORM.Release
ENDIF
```

La validación de IDCliente, hecha en el evento LostFocus, trabaja así:

```
SELECT CLIENTE
SET ORDER TO IDCliente
SEEK THISFORM.IDCliente.Value
IF NOT FOUND()
   ON KEY LABEL ENTER KEYBOARD CHR(23)
   BROWSE
   ON KEY LABEL ENTER
ENDIF
SELECT ORDENES
REPLACE NEXT 1 :
 ORDENES.IDCliente     WITH CLIENTE.IDCliente,   ;
 ORDENES.Nombre        WITH TRIM(CLIENTE.Nombre)+[ ] + CLIENTE.Apellido, ;
 ORDENES.Direccion1    WITH CLIENTE.Direccion1, ;
 ORDENES.Ciudad        WITH CLIENTE.Ciudad,      ;
 ORDENES.Estado        WITH CLIENTE.Estado,      ;
 ORDENES.CP            WITH CLIENTE.CP
THISFORM.Refresh

THISFORM.Grid1.Enabled          = .T.
THISFORM.Grid1.Column1.Enabled  = .T.
THISFORM.Grid1.Column3.Enabled  = .T.
THISFORM.PorcenIVA.Enabled      = .T.

THISFORM.Grid1.SetFocus
```

Si los usuarios introducen un ISBN, usted tiene que buscar el libro e introducir su información dentro del primer registro de detalle en blanco:

```
IF EMPTY ( THISFORM.GRID1.Column1.Text1.Value )
  - RETURN
ENDIF
SELECT LIBROS
SET ORDEN TO ISBN
SEEK THISFORM.GRID1.Column1.Text1.Value
IF NOT FOUND()
   WAIT WINDOWS [ ISBN not found - use LOOKUP button ] TIME 1
   SELECT DETALLES
   RETURN .F.
ENDIF
SELECT DETALLES
REPLACE NEXT 1              ;
 ISBN     WITH LIBROS.ISBN,      ;
 Titulo     WITH LIBROS.Titulo,        ;
 Cantidad    WITH 1,             ;
 Precio     WITH LIBROS.PrecioVenta,   ;
 Extendido    WITH LIBROSS.PrecioVenta
THISFORM.GRID1.Refresh
THISFORM.CalculoIVA
```

Calcula llama a CalculoIVA, la cual también es llamada si la tasa de impuestos se cambia:

```
REPLACE NEXT 1 ORDENES.IVA    ;
    WITH ROUND( ( THISFORM.PorcenIVA.Value / 100    ;
                * ORDENES.SubTotal), 2 ),          ;
           ORDENES.TotalOrden
    WITH ORDENES.SubTotal + ORDENES.IVA
THISFORM.Refresh
```

Si los usuarios hacen Guardar, Salir, tengo que eliminar los registros de detalles que no se utilizaron:

```
END TRANSACTION
SELECT DETALLES
SCAN FOR NumOrden = ORDENES.NumOrden
     IF EMPTY ( Titulo )
        =RLOCK()
        BLANK
        UNLOCK
        DELETE NEXT 1
       ELSE
        RECALL NEXT 1
      ENDIF
ENDSCAN
```

```
SET DELETED ON
THISFORM.Release
```

Si ellos hacen Imprimir y Salir, de todas formas tengo que eliminar los registros de detalle en blanco antes de imprimir la factura:

```
END TRANSACTION
SELECT DETALLES
SCAN FOR NumOrden= ORDENES.NumOrden
     IF EMPTY ( Titulo )
        =RLOCK()
        BLANK
        UNLOCK
        DELETE NEXT 1
       ELSE
        RECALL NEXT 1
      ENDIF
ENDSCAN
SET DELETED ON
SELECT ORDENES
SET ORDENES TO NumOrden
SEEK ORDENES.NumOrden
REPORT FORM ORDENES NOCONSOLE TO PRINT FOR ORDENES.NumOrden =
THISFORM.NuevaTransac
THISFORM.Release
```

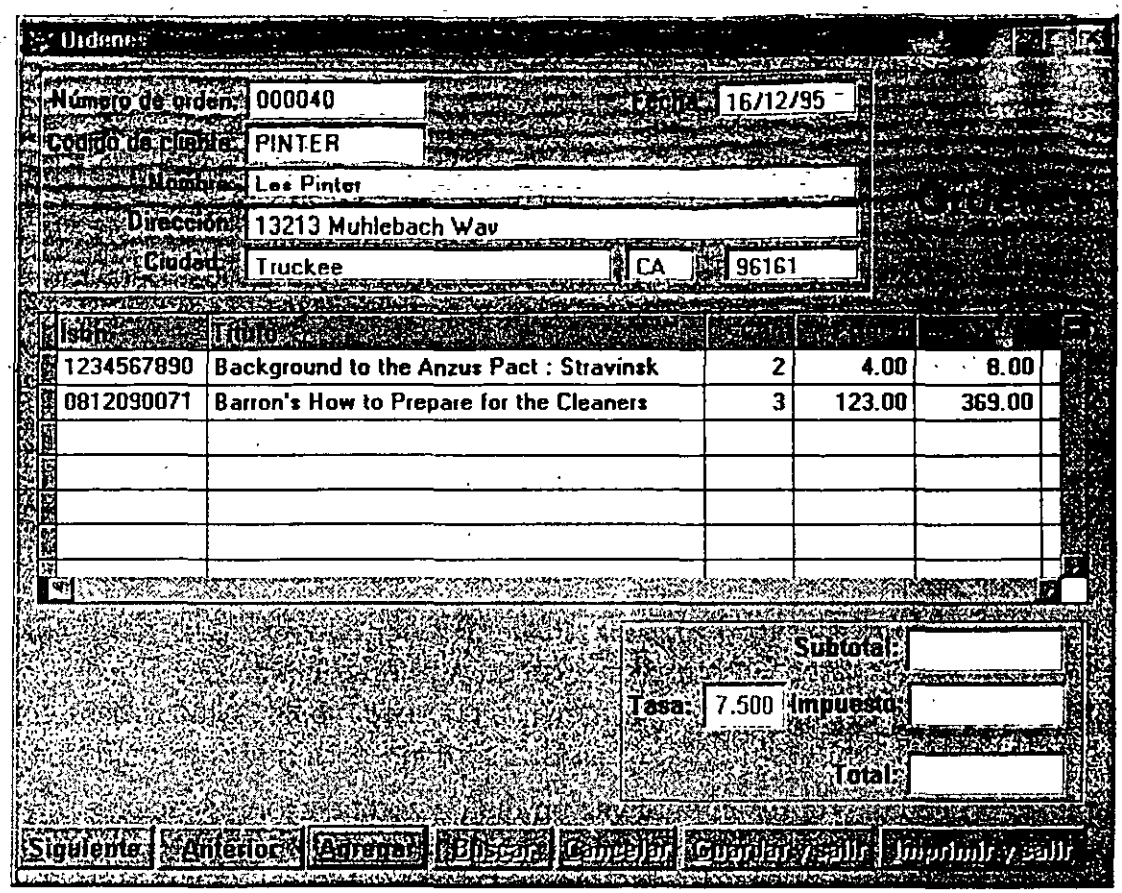
Si los usuarios cancelan es más fácil:

```
ROLLBACK
THISFORM.Release
```

Imprima la factura

Visual FoxPro hace que imprimir facturas sea un trabajo particularmente agradable. Aquí se diseñó una, mostrada en la figura 14-28, y se agregó a la aplicación. Como el Report Writer soporta pies de página, usted puede utilizar también formularios antes de imprimir; los totales se imprimirán en el lugar correcto.

Figura 14-28



Factura impresa.

Acceso remoto

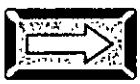
Uno de los propósitos originales de este ejercicio fue permitir búsquedas y entradas de órdenes remotas. La forma de hacer esto es tan sencilla que se le va a decir cómo hacerlo en los escasos siguientes párrafos. De hecho, le diré cómo en sólo tres letras: RAS (*remote access services*), lo cual significa servicios de acceso remoto. Windows NT 3.5 viene con esta característica, la cual consiste en un icono que espera una llamada. Windows para Trabajo en Grupo también contiene el icono RAS.

Para usarlo, introduzca el número telefónico del servidor en su libreta de teléfonos WFW RAS. Así, siempre que usted haga un doble clic en él, RAS marca a su servidor. Hasta este punto, el servidor es su unidad F: (o cualquiera que usted elija). Proporciona a su aplicación búsquedas de sus datos en la unidad F:, lo cual es fácil hacer utilizando el entorno de datos, es igual que si fuera una estación de trabajo en su servidor, a pesar de que es más lenta. Bueno, está bien, mucho más lenta. Es por eso que se utilizó la técnica de contar los títulos coincidentes antes de

mostrarlos en la pantalla BuscarLibro. En cualquier esquema de comunicaciones remotas, usted preferirá enviar la menor cantidad de datos posibles a través de la línea.

Si usted quiere ir un paso más adelante, puede utilizar SQL y almacenar procedimientos para expeditar la búsqueda de títulos aún más. No se hizo eso aquí debido a que conectarse a SQL es un tema separado y hay muchos, pero muchos otros aspectos que están involucrados en optimizar aplicaciones de servidor SQL. Aunque ciertamente es mucho más fácil de lo que era antes.

El desarrollo de esta aplicación, incluyendo acceso remoto, es mucho más fácil de lo que sería mediante la utilización de la tecnología del año pasado, de la cual yo siento que es justo decir que es todo un mundo nuevo. En algunas formas para mí es difícil trabajar con FoxPro 2.6 ahora que sé qué hay en el futuro.



Conclusión

Esta aplicación, la cual representa tal vez un día de codificación, es adecuada para muchas necesidades de empresas. En el pasado, esto habría representado semanas de programación, o aún más. Visual FoxPro ofrece bajar el costo de desarrollar software del mundo real al grado de que las empresas que piensan que no podrían pagar por tener aplicaciones construidas al gusto del cliente pueden ahora realizar ese sueño. ¿Es esto un gran trabajo, o qué?

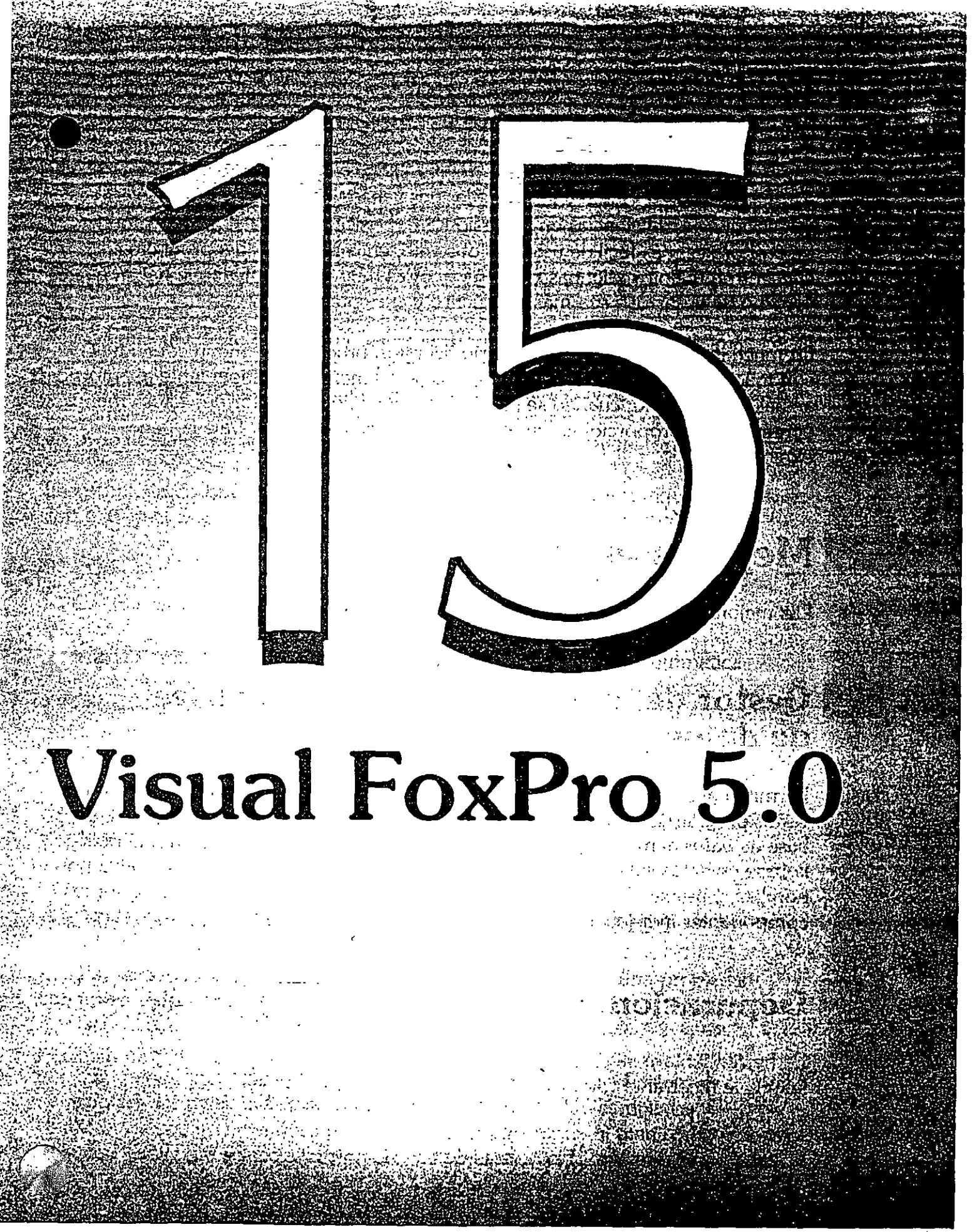
Yo espero que usted esté alentado por el sencillo enfoque que tomó este libro para crear aplicaciones utilizando las técnicas básicas. Decidimos escribir este libro con el fin de proporcionar una forma sencilla, de rápido acercamiento para desarrolladores.

Hay una variedad de poderosas extensiones para programación orientada a objetos, muchas de las cuales no se tocan en este texto introductorio. De hecho, hay un número importante de características de diseño que no se utilizaron aquí -no porque fueran demasiado complicadas, sino simplemente por falta de tiempo. Los marcos de página son por lo menos un elemento de diseño tan importante como los objetos cuadrícula, aunque no encontramos ninguna excusa conveniente para usar una. Las barras de herramientas son utilizadas completamente en muchos diseños tan comunes como los botones VCR para pantallas similares (FoxExpress tendrá bases más apegadas en su filosofía de diseño a barras de herramientas compartidas), pero nosotros no incluimos ni siquiera una en los ejemplos. SQL desde luego es el corazón de muchos esfuerzos de desarrollo, y

sólo se ha palpado su superficie. De hecho, Microsoft SQL 6.0 deberá salir cerca del momento en que este libro esté en los estantes, y se rumora que tendrá mejoras considerables sobre la versión actual. Los diseñadores harán nuestro trabajo aún más fácil, pero yo decidí no incluir un capítulo acerca de ellos (tenemos un puñado en nuestro foro de CompuServe, que pueden bajarse sin cargo). Y la programación orientada a objetos que utiliza código para desarrollar objetos personalizados no visuales es una parte muy importante de muchos esfuerzos de desarrollo.

Así que yo le invito a ver fuentes adicionales de información sobre Visual FoxPro. Nuestro foro en CompuServe le proporcionará muchos ejemplos, freeware y shareware para expandir su colección de herramientas -sólo escriba GO PINTER. Y mi boletín prácticamente no tiene competidor, así que nosotros estamos disponibles para reaccionar a nuevas tendencias en el mundo de FoxPro.

Sobre todo, experimente. Con alrededor de 2,000 propiedades y métodos de Visual FoxPro, le tomará un tiempo estar informado acerca de lo que hay en ellos. Usted está listo para ser sorprendido gratamente. Y si piensa que algo está faltando, envíe una nota a Microsoft. Después de todo, habrá nuevas versiones de FoxPro.


15
Visual FoxPro 5.0

CAPITULO 15

ESTE libro fue escrito usando la versión beta de Visual FoxPro 3.0. Desde entonces, Microsoft ha emitido una versión subsiguiente, con el nombre de "Visual FoxPro Versión 5.0".

Afortunadamente, los cambios entre las dos versiones son a grandes rasgos correcciones a deficiencias en la versión 3.0. Básicamente es el mismo producto, pero ahora es más fácil de usar. No obstante, en cada rincón del producto se encuentran mejoras -algunas pequeñas y otras muy importantes- que hacen que el entorno de desarrollo sea más productivo y eficiente.

En muchas áreas, el botón secundario del ratón produce un "menú contextual" con muchas de las selecciones disponibles en el menú correspondiente al entorno actual. Por ejemplo, cuando se diseña un formulario o un informe, estos "menús contextuales" proporcionan un atajo a la operación deseada. Vale la pena explorar.

Mejoras en la versión 5.0

Las mejoras se pueden agrupar en 8 áreas:

Gestor de proyectos y Gestor de bases de datos

Permiten el uso de SourceSafe para controlar el desarrollo de un proyecto entre varios programadores o versiones distintas. Múltiples usuarios pueden modificar la base de datos al mismo tiempo. Se puede limitar el muestreo de componentes de la base de datos solamente a vistas o tablas; puede buscarse una vista o tabla por nombre, y usarse el teclado para abrir y cerrar el bosquejo del proyecto y ver los componentes individuales.

Depuración mejorada

La herramienta para detectar y corregir errores ha sido mejorada en gran medida; se puede mostrar el valor actual de una variable poniendo el cursor encima de la

variable o campo en la ventana de código. Existen 5 ventanas de depuración: una para el código actual, otra para variables locales, una para la cadena de ejecución (por ejemplo, A corre B, B ejecuta C, etc.), otra más para muestreo de datos del nuevo comando DEBUGOUT, y la última para ver el valor actual de cualesquiera de los objetos, campos o variables.

Diccionario de datos y diseño de tablas

Es más fácil añadir índices y el acceso a las propiedades de las tablas es más directo. La mejora más importante en la versión 5.0 es que se puede asignar una clase de las bibliotecas de clases para cada tipo de campo, y así crear objetos en las pantallas por omisión usando clases propias, arrastrando los campos del entorno de datos de un formulario y dejándolos caer encima del formulario.

Las propiedades InputMask y Format funcionan con todos los controles, inclusive con el ComboBox.

Generadores de vistas y consultas

pueden crear Outer Joins, Left Joins y Right Joins en forma automática. Asimismo se pueden seleccionar sólo los primeros o últimos N registros, o los primeros o últimos N%; y se puede especificar un InputMask, formato o una clase predeterminada para ser usado cuando se añade un campo de una vista de Entorno de datos de un formulario.

Aun cuando una vista contiene un filtro, se puede hacer una referencia de manera subsecuente a la misma mediante la palabra clave NOFILTER.

Manejo de objetos en los formularios

Por fin, se puede seleccionar un grupo de controles y cambiar una propiedad común en todos los controles de una vez.

Existe un nuevo Generador de menús de métodos abreviados que se pueden detectar con el nuevo método RightClick. Se puede limitar el muestreo de la Hoja de propiedades a solamente las propiedades que hayan sido cambiadas; el Redactor de código tiene decenas de mejoras, por ejemplo, el menú Atajo ntiene listas para seleccionar nombres de objetos en el formulario. En vez de

escribir THISFORM.MarcoPagina1.Pagina1.Cuadricula1.Columna1.Texto1, se requiere un simple clic para escoger el nombre de una lista.

Las herramientas de alineación están muy mejoradas, se puede usar la tecla CTRL conjuntamente con REPAG, AVPAG, INICIO y FIN para efectuar cambios en el tamaño de objetos seleccionados.

Más y mejores asistentes

Hay nuevos asistentes (por ejemplo, para generar un programa que convierte sus datos en formato Oracle). Los asistentes originales tienen mayor funcionalidad, como el Asistente para formularios que puede generar formularios de manera opcional con múltiples marcos de página.

El nuevo Asistente para páginas Web genera páginas HTML para consultar y publicar el contenido de sus bases de datos.

Mejor integración de los controles ActiveX

Es más fácil usar los controles ActiveX, ya que vienen varios controles nuevos incluidos con la versión 5.0; se puede construir un servidor ActiveX de una aplicación VFP, al cual usted puede tener acceso desde otras aplicaciones mediante la interfaz ActiveX.

Ejemplos y clases

VFP viene acompañado de decenas de ejemplos, los cuales demuestran muchas técnicas interesantes; mediante una simple interfaz común, se puede ver funcionar un ejemplo e inmediatamente inspeccionar el código para ver cómo funciona. Incluye 6 bibliotecas de clases (VFP\SAMPLES\CLASSES), las cuales contienen muchas herramientas que se pueden usar de inmediato en las aplicaciones.

Cabe mencionar que, a diferencia de la versión 3.0, VFP 5.0 sólo funciona bajo Win95, NT 3.51 y NT 4.0. Debido al bajo costo de la memoria hoy en día, y a la omnipresencia de Win95, recomiendo sin reservas que todo programador Fox se traslade a la versión 5.0. El incremento en la productividad del programador pagará cualesquiera de los costos de migración en cuestión de semanas, si no días.

Implicaciones de la versión 5.0 para el lector de este libro

Debido a las correcciones y mejoras en VFP desde la versión que se usó para escribir el libro hace casi dos años, cabe mencionar que los ejemplos, de haber sido escritos hoy, serían bastante más sencillos de escribir y de entender. Por esta razón, se han corregido los ejemplos para reflejar la versión 5.0. Se pueden obtener los ejemplos corregidos en el sitio web *www.pinter.com*.

Por ejemplo, la cuadrícula empleada en la factura desarrollada en el capítulo 11 del libro podría ser bastante más sencilla usando las Vistas revisables disponibles en la versión 5.0. Asimismo, como ya no se produce un error fatal al intentar usar el método SetAll con el nombre de una clase que no aparece en un formulario, se recomienda un nuevo método genérico para el código Click de los botones Añadir, Editar, Guardar y Cancelar.

También se ha estandardizado el uso de Click, Valid y LostFocus de los controles. Por esto, algunas de las técnicas destinadas a evitar problemas relacionadas con deficiencias en la versión 3.0 ya no son necesarias. Las simplificaciones resultantes en el código también se encuentran en el sitio web.

Finalmente, se ha añadido al ejemplo unos formularios nuevos, los cuales muestran técnicas para que los usuarios mismos puedan crear consultas e informes nuevos. Éstos no aparecieron en el libro, no por faltas en la versión anterior, sino por falta de tiempo e imaginación mía, por lo que pido la indulgencia de mis amables lectores.

A

Sintaxis y parámetros del cuadro de mensajes

```
MESSAGEBOX(cTextodeMensaje [, nTipoCuadroDialogo [, cTextoBarraTitulo]])
```

nTipoCuadroDialogo es la suma de los valores para el grupo de botones del cuadro de diálogo, el icono de presentación y el valor predeterminado.

Valor	Botones del cuadro de diálogo
0	Sólo el botón Aceptar
1	Botones Aceptar y Cancelar
2	Botones Anular, Reintentar e Ignorar
3	Botones Sí, No y Cancelar
4	Botones Sí y No
5	Botones Reintentar y Cancelar

Valor	Icono
16	Señal de alto (punto)
32	Signo de interrogación
48	Signo de exclamación
64	Icono de información (i)

Valor	Botón predeterminado
0	Primer botón
256	Segundo botón
512	Tercer botón

cTextoBarraTitulo es el texto que aparece en la barra de título del cuadro de diálogo. Si usted omite cTextoBarraTitulo, el título "Microsoft Visual FoxPro" aparecerá en la barra de título. El valor MessageBox( ) indica qué botón fue elegido en el cuadro de diálogo. En cuadros de diálogo con un botón Cancelar, el oprimir Esc para salir del cuadro de diálogo regresa el mismo valor (2) que elegir Cancelar.

Valor de retorno	Botón
1	Aceptar
2	Cancelar
3	Anular
4	Reintentar
5	Ignorar
6	Sí
7	No

B

Tipos de campos para las tablas de Visual FoxPro

TipodeCampo	nAnchodeCampo	nPrecision	Descripción
C	n	-	Campo de caracteres de ancho *n*
D	-	-	Fecha
T	-	-	Fecha-hora
N	n	d	Campo numérico de ancho *n* con lugares decimales *d*
F	n	d	Campo numérico flotante de ancho *n* con lugares decimales *d*
B	-	d	Doble
Y	-	-	Moneda
L	-		Lógico
M	-	-	Memo
G	-	-	General
P	-	-	Imagen

Epílogo

Espero que este libro sea útil para mis colegas en España -los miembros del Grupo de Usuarios Fox de Madrid, quienes me han invitado varias veces a dar conferencias allá- y para mis hermanos y hermanas lectores en América Latina, especialmente en México. Tuve el enorme placer de vivir en la Ciudad de México durante los años sesenta, cuando éramos sólo 5 millones de habitantes, y aún se veían los volcanes desde el Periférico por las mañanas. En el México de aquel entonces, se contemplaba un bello futuro para el país. Todos hemos pasado por tiempos difíciles desde entonces, pero yo sueño con un futuro México "lindo y querido", a donde pienso ir algún día, para pasar el resto de mis días, y el resto de la eternidad.

Hasta pronto.
Les Pinter
San Mateo, Alta California

Índice

379

B

380

C

381

D

382

E

F

G

H

N

O

P

390

T

394

U

W

Z

Acerca de los autores

Les Pinter es presidente de Pinter Consulting, una empresa especializada en aplicaciones de FoxPro. Es editor de *Pinter FoxPro Letter*, una publicación mensual reconocida internacionalmente para programadores profesionales de FoxPro. Se le invita frecuentemente a participar en conferencias para especialistas en FoxPro en todo el mundo.

John Pinter se asoció a Microsoft como especialista en el equipo Excel después de licenciarse en Ciencias de la Computación en la Universidad de Berkeley en 1992. Ahí pasó un año antes de convertirse en un consultor de tiempo completo, especializándose en el desarrollo de FoxPro y C/C++.

396