

# Visual Foxpro 6 , Manual del Programador - Microsoft Press

**Título :** Visual Foxpro 6 , Manual del Programador

**Autor :** Microsoft Press

**ISBN :** 84-481-2089-2

**Editora :** Mc Graw Hill

**Nº Paginas :** 847



## CONTENIDO

### Introducción.

#### [PARTE 1. Programación con Visual FoxPro.](#)

- Capítulo 1. Introducción a la programación.
- Capítulo 2. Programar una aplicación.
- Capítulo 3. Programación orientada a objetos.
- Capítulo 4. Descripción del modelo de eventos.

#### [PARTE 2. Trabajar con datos.](#)

- Capítulo 5. Diseñar una base de datos.
- Capítulo 6. Crear bases de datos.
- Capítulo 7. Trabajar con tablas.
- Capítulo 8. Crear vistas.

#### [PARTE 3. Crear la interfaz.](#)

- Capítulo 9. Crear formularios.
- Capítulo 10. Usar controles.
- Capítulo 11. Diseñar menús y barras de herramientas.

#### [PARTE 4. Agrupar todos los elementos.](#)

- Capítulo 12. Agregar consultas e informes.
- Capítulo 13. Compilar una aplicación.
- Capítulo 14. Probar y depurar aplicaciones.

Capítulo 15. Optimizar aplicaciones.

#### [PARTE 5. Ampliar aplicaciones.](#)

Capítulo 16. Agregar OLE.

Capítulo 17. Programar para acceso compartido.

Capítulo 18. Programar aplicaciones internacionales.

#### [PARTE 6. Crear soluciones cliente-servidor.](#)

Capítulo 19. Diseñar aplicaciones cliente-servidor.

Capítulo 20. Upsizing de bases de datos de VisualFoxPro.

Capítulo 21. Implementar una aplicación cliente-servidor.

Capítulo 22. Optimizar el rendimiento cliente-servidor.

#### [PARTE 7. Crear archivos de Ayuda.](#)

Capítulo 23. Crear Ayuda gráfica.

Capítulo 24. Crear Ayuda de tipo .DBF.

#### [PARTE 8. Distribuir aplicaciones.](#)

Capítulo 25. Generar una aplicación para su distribución.

Capítulo 26. Crear discos de distribución.

#### [PARTE 9. Acceso a las bibliotecas API.](#)

Capítulo 27. Extender Visual FoxPro con bibliotecas externas.

Capítulo 28. Acceso a la API de Visual FoxPro.

#### [PARTE 10. Crear soluciones empresariales.](#)

Capítulo 29. Programar en equipo.

Capítulo 30. Soluciones empresariales de Visual FoxPro.

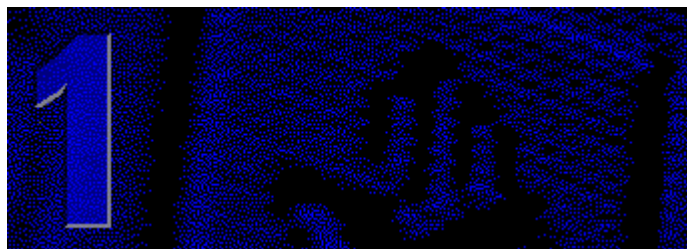
#### [PARTE 11. Lo nuevo en Visual FoxPro.](#)

Capítulo 31. Interoperabilidad e Internet.

Capítulo 32. Programación de aplicaciones y productividad del programador.

Capítulo 33. Mejoras para la programación.

# Manual del programador, Parte 1: Programación en Visual FoxPro



Visual FoxPro es una eficaz herramienta de administración de datos, pero además podrá beneficiarse de toda su eficacia para crear aplicaciones. Comprender las técnicas de programación orientada a objetos y el modelo controlado por eventos puede aumentar su productividad como programador.

## Capítulo 1 [Introducción a la programación](#)

Si está empezando a programar, aprenda el proceso y el método de programación en Visual FoxPro.

## Capítulo 2 [Programar una aplicación](#)

Cuando programe una aplicación, organice sus componentes con el Administrador de programas, una forma integrada de generar y probar su aplicación a medida que la cree.

## Capítulo 3 [Programación orientada a objetos](#)

Con la programación orientada a objetos, puede crear componentes de aplicación independientes que respondan a acciones del usuario y al sistema y que se puedan mantener y reutilizar fácilmente.

## Capítulo 4 [Descripción del modelo de eventos](#)

El modelo de eventos define cuándo y cómo tienen lugar las interacciones con el usuario y el sistema.

# Capítulo 1: Introducción a la programación

En Visual FoxPro funcionan juntas la programación por procedimientos y la programación orientada a objetos para permitirle crear aplicaciones potentes y flexibles. Conceptualmente, puede imaginarse que la programación consiste en escribir una secuencia de instrucciones con el fin de realizar tareas específicas. A un nivel estructural, la programación en Visual FoxPro precisa la manipulación de los datos almacenados.

Si no tiene experiencia en programación, este capítulo le ayudará a ponerse en marcha. Si ya conoce otros lenguajes de programación y desea compararlos con Visual FoxPro, vea el tema [Visual FoxPro y otros lenguajes de programación](#). Si desea una descripción de la programación orientada a objetos, consulte el capítulo 3, [Programación orientada a objetos](#).

En este capítulo se abordan los temas siguientes:

- [Ventajas de la programación](#)
- [La mecánica de la programación en Visual FoxPro](#)
- [Conceptos básicos de programación](#)
- [El proceso de la programación](#)
- [Usar procedimientos y funciones definidos por el usuario](#)
- [Pasos siguientes](#)

## Ventajas de la programación

Normalmente, cualquier función que pueda realizar con un programa podrá realizarla también a mano, si dispone de suficiente tiempo. Por ejemplo, si desea consultar información sobre un cliente en una tabla de clientes, como por ejemplo la empresa Ernst Handel, podría hacerlo manualmente si sigue una secuencia concreta de instrucciones.

### Para buscar manualmente un único pedido en una tabla

1. En el menú **Archivo**, elija **Abrir**.
2. En el cuadro **Archivos de tipo**, elija **Tabla**.
3. Haga doble clic en Customer.dbf en la lista de archivos.
4. En el menú **Ver**, elija **Examinar**.
5. Desplácese por la tabla, examinando el campo Company de los registros hasta encontrar “Ernst Handel”.

Mediante programación podría conseguir el mismo resultado escribiendo los siguientes comandos de Visual FoxPro en la [ventana Comandos](#):

```
USE Customer  
LOCATE FOR Company = "Ernst Handel"  
BROWSE
```

Cuando haya localizado el pedido de esta empresa, tal vez desee incrementar la cantidad máxima del pedido en un 3%.

### Para incrementar manualmente la cantidad máxima del pedido

1. Presione la tecla Tab para desplazarse hasta el campo max\_ord\_amt.
2. Multiplique el valor mostrado en el campo max\_ord\_amt por 1,03 y escriba el nuevo valor en el campo.

Para conseguir el mismo resultado mediante programación, escriba el siguiente comando de Visual FoxPro en la ventana Comandos:

```
REPLACE max_ord_amt WITH max_ord_amt * 1,03
```

Es relativamente sencillo cambiar la cantidad máxima del pedido para un cliente, ya sea manualmente o escribiendo las instrucciones en la ventana Comandos. Sin embargo, suponga que desea incrementar en un 3% la cantidad máxima de pedido de todos los clientes. Podría hacerlo manualmente, pero le llevaría mucho tiempo y es posible que cometiese errores. Si especifica las instrucciones correctas en un archivo de programa, Visual FoxPro podrá realizar esta tarea con rapidez y facilidad, sin cometer ningún error.

### Programa de ejemplo para incrementar las cantidades máximas de pedido de todos los clientes

Código	Comentarios
USE customer	Abre la tabla CUSTOMER.
SCAN	Examina todos los registros de la tabla y realiza todas las instrucciones comprendidas entre SCAN y ENDSCAN para cada registro.
REPLACE max_ord_amt WITH ; max_ord_amt * 1.03	Incrementa la cantidad máxima de pedido en un 3%. (El punto y coma (;) indica que el comando sigue en la línea siguiente).
ENDSCAN	Final del código que se ejecuta para cada registro contenido en la tabla.

La ejecución de un programa ofrece numerosas ventajas en comparación con la introducción de distintos comandos en la ventana Comandos:

- Los programas se pueden modificar y volver a ejecutar.
- Se pueden ejecutar programas desde los menús, formularios y barras de herramientas.
- Los programas pueden ejecutar otros programas.

En las siguientes secciones se describe la mecánica, los conceptos y los procesos que subyacen a éste y otros programas de Visual FoxPro.

## La mecánica de la programación en Visual FoxPro

Puede programar en Visual FoxPro escribiendo código: instrucciones en forma de comandos, funciones u operaciones que Visual FoxPro puede entender. Puede incluir estas instrucciones en:

- La [ventana Comandos](#).
- Archivos de programa
- Ventanas de código de eventos o de métodos en el [Diseñador de formularios](#) o en el [Diseñador de clases](#)
- Ventanas de código de procedimientos en el [Diseñador de menús](#)
- Ventanas de código de procedimientos en el [Diseñador de informes](#)

## Usar la ventana Comandos

Puede ejecutar un comando de Visual FoxPro si lo escribe en la ventana Comandos y presiona ENTRAR. Para volver a ejecutar el comando, lleve el cursor a la línea que contiene el comando y presione nuevamente ENTRAR.

Puede ejecutar varias líneas de código en la ventana Comandos como si constituyeran un programa.

### Para ejecutar varias líneas de código en la ventana Comandos

1. Seleccione las líneas de código.
2. Presione ENTRAR o elija **Ejecutar selección** en el [menú emergente](#).

Como la ventana Comandos es una ventana de edición, puede modificar comandos con las herramientas disponibles en Visual FoxPro. Puede modificar, insertar, eliminar, cortar, copiar o pegar texto en la ventana Comandos.

La ventaja que supone poder escribir código en la ventana Comandos radica en el hecho de que las instrucciones se ejecutan de inmediato. No es necesario guardar un archivo y ejecutarlo como un programa.

Además, las opciones que elige en los menús y los cuadros de diálogo aparecen en la ventana Comandos como comandos. Puede copiar y pegar estos comandos en un programa de Visual FoxPro y a continuación ejecutar el programa repetidamente, lo cual facilita la ejecución de miles de comandos, una y otra vez.

## Crear programas

Un programa de Visual FoxPro es un archivo de texto que contiene una serie de comandos. Puede crear un programa en Visual FoxPro de una de las siguientes maneras:

### Para crear un programa

1. En el [Administrador de proyectos](#), seleccione **Programas** en la ficha **Código**.
2. Elija **Nuevo**.  
—O bien—
  1. En el menú Archivo, elija Nuevo.
  2. En el cuadro de diálogo Nuevo, seleccione Programa.
  3. Elija Nuevo archivo.  
—O bien—

- En la ventana **Comandos**, escriba:

MODIFY COMMAND

Visual FoxPro abrirá una nueva ventana denominada Programa1. Podrá entonces escribir su programa en esta ventana.

## Guardar programas

Una vez creado un programa, asegúrese de guardarlo.

### Para guardar un programa

- En el menú **Archivo**, elija **Guardar**.

Si intenta cerrar un programa sin antes guardarlo, aparecerá un cuadro de diálogo en el que se le preguntará si desea guardar o descartar los cambios realizados en el mismo.

Si guarda un programa creado a partir del Administrador de proyectos, el programa se agregará al proyecto.

Si guarda un programa al que todavía no ha asignado un nombre, se abrirá el cuadro de diálogo Guardar como, en el que podrá especificar el nombre del programa. Cuando haya guardado el programa, podrá ejecutarlo o modificarlo.

## Modificar programas

Después de guardar el programa, podrá modificarlo. En primer lugar, abra el programa de una de las siguientes maneras:

### Para abrir un programa

- Si el programa forma parte de un proyecto, selecciónelo en el [Administrador de proyectos](#) y elija **Modificar**.

–O bien–

- En el menú **Archivo**, elija **Abrir**. Aparecerá un cuadro de diálogo en el que se muestra una lista de los archivos disponibles. En la lista **Archivos de tipo**, elija **Programa**. En la lista de archivos, seleccione el programa que desea modificar y elija **Abrir**.

–O bien–

- En la ventana **Comandos**, escriba el nombre del programa que desea modificar:

MODIFY COMMAND miprogram

–O bien–

- En la ventana **Comandos**, escriba:

MODIFY COMMAND ?

Cuando aparezca la lista de archivos, seleccione el programa que desea modificar y a continuación elija **Abrir**.

Después de abrir el programa, podrá realizar cambios en el mismo. Cuando haya terminado de introducir los cambios, asegúrese de guardar el programa.

## Ejecutar programas

Después de crear un programa, podrá ejecutarlo.

### Para ejecutar un programa

- Si el programa forma parte de un proyecto, selecciónelo en el [Administrador de proyectos](#) y elija **Ejecutar**.

–O bien–

- En el menú **Programa**, elija **Ejecutar**. Cuando aparezca la lista de programas, seleccione el programa que desea ejecutar y a continuación elija **Ejecutar**.

–O bien–

- En la ventana **Comandos**, escriba DO y el nombre del programa que desea ejecutar:

DO miprogram

## Escribir código en las herramientas de diseño de Visual FoxPro

El [Diseñador de formularios](#), el [Diseñador de clases](#) y el [Diseñador de menús](#) le permiten integrar fácilmente código de programas mediante la interfaz de usuario, de forma que el código apropiado se ejecute como respuesta a las acciones del usuario. El [Diseñador de informes](#) le permite crear informes complejos y personalizados integrando código en el archivo del informe.

Para aprovechar plenamente la eficacia de Visual FoxPro, debe utilizar estas herramientas de diseño. Si desea más información sobre el Diseñador de informes, consulte el capítulo 7, [Diseñar informes y etiquetas](#), del *Manual del usuario*. Para obtener información más detallada sobre el Diseñador de formularios, consulte el capítulo 3, [Programación orientada a objetos](#), de este manual. Para obtener información más detallada sobre el Diseñador de formularios, consulte el capítulo 9, [Crear formularios](#), y si desea más información acerca del Diseñador de menús, consulte el capítulo 11, [Diseñar menús y barras de herramientas](#).

## Conceptos básicos de programación



Cuando se programa, se almacenan datos y se manipulan mediante una serie de instrucciones. Los datos y los contenedores en los que se almacenan los datos constituyen la materia prima de la programación. Las herramientas utilizadas para manipular esta materia prima son comandos, funciones y operadores.

## Almacenar datos

Los datos con los que trabaja probablemente incluyan períodos de tiempo, dinero y elementos contables, así como fechas, nombres, descripciones, etc. Cada dato corresponde a un determinado tipo, es decir, pertenece a una categoría de datos que se manipula de maneras similares. Podría trabajar directamente con estos datos sin almacenarlos, si bien perdería la mayor parte de la flexibilidad y potencia que ofrece Visual FoxPro. Visual FoxPro aporta numerosos contenedores de almacenamiento con el fin de ampliar su capacidad para manipular fácilmente los datos.

Los tipos de datos determinan la manera en que se almacenan los datos y la forma en que se pueden utilizar tales datos. Puede multiplicar dos números, pero no puede multiplicar caracteres. Puede imprimir caracteres en mayúsculas, pero no puede imprimir números en mayúsculas. En la tabla siguiente se muestran algunos de los principales tipos de datos de Visual FoxPro.

### Tipos de datos

Tipo	Ejemplos
<a href="#">Numeric</a>	123 3,1415 – 7
<a href="#">Character</a>	“Prueba” “123” “01/01/98”
<a href="#">Logical</a>	.T. (verdadero) .F. (falso)
<a href="#">Date</a>	{^1998-01-01}
<a href="#">DateTime</a>	{^1998-01-01 12:30:00 p}

### Contenedores de datos

Los contenedores de datos le permiten realizar las mismas operaciones con varios datos. Por ejemplo, sumar las horas que ha trabajado un empleado, multiplicarlas por el salario por hora y restar los impuestos para determinar el sueldo que ha percibido el empleado. Deberá realizar estas operaciones para cada empleado y para cada período de pago. Si almacena esta información en contenedores y realiza las operaciones sobre éstos, bastará con sustituir los datos antiguos por los nuevos datos y volver a ejecutar el mismo programa. En la siguiente tabla se enumeran algunos de los principales contenedores de datos disponibles en Visual FoxPro:

Tipo	Descripción
<a href="#">Variables</a>	Elementos individuales de datos almacenados en la memoria RAM (memoria de acceso aleatorio) del PC.
<a href="#">Registros</a> de tabla	Varias filas de campos predeterminados, cada uno de los cuales puede contener un dato definido previamente. Las tablas se guardan en disco.
<a href="#">Matrices</a>	Varios elementos de datos almacenados en la memoria RAM.

## Manipular datos

Los contenedores y los tipos de datos le ofrecen los módulos que necesita para manipular los datos. Los elementos finales son los operadores, las funciones y los comandos.

### Usar operadores

Los operadores se utilizan para vincular los datos. A continuación se muestran los operadores utilizados habitualmente en Visual FoxPro.

Operador	Tipos de datos válidos	Ejemplo	Resultado
=	Todos	? n = 7	Imprime .T. si el valor almacenado en la variable es 7; de lo contrario, imprime .F.
+	Numeric, Character,Date, DateTime	? "Fox" + "Pro"	Imprime "FoxPro"
! or NOT	Logical	? !.T.	Imprime .F. (falso)
*, /	Numeric	? 5 * 5 ? 25 / 5	Imprime 25 Imprime 5

**Nota** Un signo de interrogación (?) situado delante de una expresión imprime el resultado de la expresión y un carácter de nueva línea en la ventana de salida activa, que es normalmente la ventana principal de Visual FoxPro.

Recuerde que debe utilizar el mismo tipo de datos con cada operador. Las siguientes instrucciones almacenan dos datos numéricos en dos variables. Los nombres de variable empiezan con la letra n, por lo que se puede determinar de inmediato que contienen datos numéricos, pero puede nombrarlas con cualquier combinación de caracteres alfanuméricos y caracteres de subrayado.

```
nPrimero = 123
nSegundo = 45
```

Las instrucciones siguientes almacenan dos datos de caracteres en dos variables. Los nombres de variable empiezan con la letra `c` para indicar que contienen datos de tipo character.

```
cPrimero = "123"  
cSegundo = "45"
```

Las dos operaciones siguientes, suma y concatenación, producen resultados distintos, ya que el tipo de datos es diferente en cada una de ellas.

```
? nPrimero + nSegundo  
? cPrimero + cSegundo
```

## Resultado

```
168  
12345
```

Puesto que `cPrimero` contiene caracteres y `nSegundo` contiene datos numéricos, se producirá un error de tipo de datos incorrecto si se intenta ejecutar el siguiente comando:

```
? cPrimero + nSegundo
```

Puede evitar este problema si utiliza funciones de conversión. Por ejemplo, [STR\(\)](#) devuelve el valor de tipo Character equivalente de un valor de tipo Numeric, mientras que [VAL\(\)](#) devuelve el equivalente numérico de una cadena de caracteres formada por números. Estas funciones y [LTRIM\(\)](#), que elimina los espacios iniciales, le permiten realizar las operaciones siguientes:

```
? cPrimero + LTRIM(STR(nSegundo))  
? VAL(cPrimero) + nSegundo
```

## Resultado

```
12345  
168
```

## Usar funciones

Las funciones devuelven un tipo específico de datos. Por ejemplo, las funciones `STR()` y `VAL()` utilizadas en la sección anterior devuelven valores de tipo Character y Numeric, respectivamente. Al igual que ocurre con todas las funciones, estos tipos devueltos están documentados con las funciones.

Hay cinco maneras de llamar a una función de Visual FoxPro:

- Asignar a una [variable](#) el valor que devuelve la función. La siguiente línea de código almacena la fecha actual del sistema en una variable denominada `dHoy`:

```
dHoy = DATE( )
```

- Incluir la llamada a la función en un comando de Visual FoxPro. El siguiente comando establece el directorio predeterminado como el valor devuelto por la función [GETDIR\(\)](#):

```
CD GETDIR( )
```

- Imprimir el valor devuelto en la ventana de salida activa. La siguiente línea de código imprime la hora actual del sistema en la ventana principal de Visual FoxPro:

```
? TIME( )
```

- Llamar a la función sin almacenar en ningún lugar el valor devuelto. La siguiente llamada de función desactiva el cursor:

```
SYS( 2002 )
```

- Incluir la función dentro de otra función. La siguiente línea de código imprime el día de la semana:

```
? DOW( DATE( ) )
```

A continuación se enumeran otros ejemplos de funciones utilizados en este capítulo:

Función	Descripción
<a href="#">ISDIGIT()</a>	Devuelve el valor verdadero (.T.) si el carácter situado al comienzo de una cadena es un número; de lo contrario, devuelve el valor falso (.F.).
<a href="#">FIELD()</a>	Devuelve el nombre de un campo.
<a href="#">LEN()</a>	Devuelve el número de caracteres de una expresión de caracteres.
<a href="#">RECCOUNT()</a>	Devuelve el número de registros de la tabla que está activa en este momento.
<a href="#">SUBSTR()</a>	Devuelve el número especificado de caracteres a partir de una cadena de caracteres, empezando en una posición especificada de la cadena.

## Usar comandos

Un comando hace que se realice una determinada acción. Cada comando dispone de una sintaxis específica que indica lo que se debe incluir con el fin de que se ejecute correctamente el comando. Hay también cláusulas opcionales asociadas a los comandos que permiten especificar de forma más detallada la acción que se desea realizar.

Por ejemplo, el comando [USE](#) permite abrir y cerrar tablas:

Sintaxis de USE	Descripción
<a href="#">USE</a>	Cierra la tabla que aparece en el área de trabajo actual.
<code>USE customer</code>	Abre la tabla CUSTOMER en el área de trabajo actual y cierra cualquier tabla que ya esté abierta

	en el área de trabajo.
USE customer IN 0	Abre la tabla CUSTOMER en la siguiente área de trabajo disponible.
USE customer IN 0 ; ALIAS miCliente	Abre la tabla CUSTOMER en la siguiente área de trabajo disponible y asigna al área de trabajo el alias miCliente.

A continuación se muestran algunos ejemplos de comandos utilizados en este capítulo:

Comando	Descripción
<a href="#">DELETE</a>	Selecciona registros especificados de una tabla para su eliminación.
<a href="#">REPLACE</a>	Sustituye el valor almacenado en el campo del registro por un nuevo valor.
<a href="#">Go</a>	Coloca el puntero de registro en una posición específica de la tabla.

## Control del flujo del programa

Visual FoxPro incluye una categoría especial de comandos que "envuelven" a otros comandos y funciones, y determinan cuándo y con qué frecuencia se ejecutan. Estos comandos permiten realizar [bifurcaciones condicionales](#) y [bucles](#), dos herramientas de programación muy eficaces. El siguiente programa muestra el uso de las bifurcaciones y los bucles condicionales. Estos conceptos se describen de forma más detallada después del ejemplo.

Suponga que su empresa cuenta con 10.000 empleados y desea conceder a todos aquéllos que ganan 3.000.000 de pesetas o más un aumento salarial del 3%, y a todos los que ganan menos de 3.000.000 de pesetas un aumento del 6%. El siguiente ejemplo de programa le permite hacerlo.

Este programa presupone que en el área de trabajo actual está abierta una tabla que contiene un campo numérico denominado `salario`. Si desea obtener información sobre las áreas de trabajo, consulte "Usar múltiples tablas" en el capítulo 7, [Trabajar con tablas](#).

### Programa de ejemplo para aumentar el salario de los empleados

Código	Observaciones
SCAN	El código comprendido entre SCAN y ENDSCAN se ejecuta tantas veces como registros haya en la tabla. Cada vez que se ejecuta el código, el puntero de registro se desplaza al siguiente registro de la tabla.
IF salario >= 3000000 REPLACE salary WITH ; salario * 1,03	Para cada registro, si el salario es mayor o igual que 3.000.000, este valor se sustituye por un nuevo salario que es un 3% superior.

	El signo de punto y coma (;) que aparece después de WITH indica que el comando continúa en la siguiente línea.
ELSE REPLACE salario WITH ; salario * 1,06	Para cada registro, si el salario no es mayor o igual que 3.000.000, se sustituye este valor por un nuevo salario que es un 6% superior.
ENDIF ENDSCAN	Final de la instrucción condicional IF.  Final del código que se ejecuta para cada registro de la tabla.

Este ejemplo utiliza comandos de bifurcación y bucle condicional para controlar el desarrollo del programa.

### Bifurcación condicional

La bifurcación condicional permite someter a prueba condiciones y, a continuación, en función del resultado de la prueba, realizar distintas operaciones. Visual FoxPro ofrece dos comandos que permiten realizar una bifurcación condicional:

- [IF ... ELSE ... ENDIF](#)
- [DO CASE ... ENDCASE](#)

El código comprendido entre la instrucción inicial y la instrucción ENDIF o ENDCASE sólo se ejecuta si una condición lógica se evalúa como verdadera (.T.). En el programa de ejemplo, el comando IF se utiliza para distinguir entre dos estados: o el salario es de 3.000.000 pesetas o más, o no lo es. Se adoptan diferentes medidas, dependiendo del estado.

En el siguiente ejemplo, si el valor almacenado en la [variable](#) nTempAgua es menor que 100, no se realizará ninguna acción:

```
* definir una variable lógica como Verdadera si se cumple una condición.
IF nTempAgua >= 100
    lEbullición = .T.
ENDIF
```

**Nota** Un asterisco al principio de una línea de un programa indica que la línea es un comentario. Los comentarios ayudan al programador a recordar la función que debe realizar cada segmento de código, si bien Visual FoxPro los pasa por alto.

Si se desea comprobar varias condiciones posibles, un bloque DO CASE ... ENDCASE puede resultar más eficaz que varias instrucciones IF y además es más fácil realizar un seguimiento del mismo.

### Bucles

Un bucle le permite ejecutar una o más líneas de código tantas veces como sea necesario. En Visual

FoxPro hay tres comandos que permiten realizar bucles:

- [SCAN ... ENDSCAN](#)
- [FOR ... ENDFOR](#)
- [DO WHILE ... ENDDO](#)

Utilice SCAN cuando realice una serie de acciones para cada uno de los registros de una tabla, como en el ejemplo de programa descrito anteriormente. El bucle SCAN permite escribir el código una vez y ejecutarlo para cada registro a medida que el puntero de registro se desplaza por la tabla.

Utilice FOR cuando sepa cuántas veces debe ejecutarse la sección de código. Por ejemplo, sabe que una tabla contiene un número específico de campos. Puesto que la función FCOUNT( ) de Visual FoxPro devuelve este número, puede utilizar un bucle FOR para imprimir los nombres de todos los campos de la tabla:

```
FOR nRecuento = 1 TO FCOUNT( )
    ? FIELD(nRecuento)
ENDFOR
```

Utilice DO WHILE cuando desee ejecutar una sección de código mientras cumpla una determinada condición. Tal vez no sepa cuántas veces debe ejecutarse el código, pero sí sabe cuándo debe detenerse la ejecución. Por ejemplo, supongamos que dispone de una tabla en la que figuran los nombres y las iniciales de una serie de personas y desea utilizar las iniciales para consultar los nombres de las personas. Surgiría un problema la primera vez que intentase agregar una persona cuyas iniciales fuesen las mismas que las de otra persona contenida en la tabla.

Para resolver este problema, podría agregar un número a las iniciales. Por ejemplo, el código de identificación de Miguel Suárez podría ser MS. La siguiente persona cuyas iniciales fuesen las mismas, Margarita Sánchez, sería MS1. Si a continuación anexase María Sanz a la tabla, su código de identificación sería MS2. Un bucle DO WHILE permite localizar el número correcto que se debe adjuntar a las iniciales.

### Programa de ejemplo que utiliza DO WHILE para generar un número de identificación único

Código	Comentarios
nAquí = RECNO( )	Guardar la posición del registro.
cIniciales = LEFT(nombre,1) + ; LEFT(apellido,1) nSufijo = 0	Obtener las iniciales de la persona a partir de las primeras letras de los campos nombre y apellido.  Si es necesario, establecer una variable que contenga el número que se debe agregar al final de las iniciales de una persona.
LOCATE FOR id_persona = cIniciales	Comprobar si hay otra persona en la tabla cuyas iniciales son las mismas.
DO WHILE FOUND( )	Si en otro registro de la tabla hay un valor id_persona que coincide con cIniciales, la

función [FOUND\(\)](#) devolverá el valor verdadero (.T.) y se ejecutará el código contenido en el bucle DO WHILE.

Si no se encuentra ninguna coincidencia, la siguiente línea de código que se ejecute será la línea que figura a continuación de ENDDO.

---

```
nSufijo = nSufijo + 1
cIniciales = ;
  LEFT(cIniciales,2);
  + ALLTRIM(STR(nSufijo))
```

---

Preparar un sufijo nuevo y anexarlo al final de las iniciales.

---

CONTINUE

---

[CONTINUE](#) hace que se vuelva a evaluar el último comando [LOCATE](#). El programa comprueba si el nuevo valor contenido en cIniciales ya existe en el campo id\_persona de otro registro. Si es así, FOUND( ) seguirá devolviendo el valor .T. y se volverá a ejecutar el código contenido en el bucle DO WHILE. Si el nuevo valor contenido en cIniciales es efectivamente único, FOUND( ) devolverá el valor .F. y la ejecución del programa continuará con la línea de código que figura a continuación de ENDDO.

---

ENDDO

---

Final del bucle DO WHILE.

---

```
GOTO nAquí
REPLACE id_persona WITH cIniciales
```

---

Volver al registro y almacenar el código de identificación único en el campo id\_persona.

Puesto que no hay manera de saber de antemano cuántas veces se encontrarán los códigos de identificación coincidentes que ya se están utilizando, se utiliza el bucle DO WHILE.

## El proceso de la programación

Cuando entienda los conceptos básicos, la programación será un proceso reiterativo. Los pasos se repiten numerosas veces, perfeccionándose el código a medida que se avanza. Al principio, someterá el código a prueba frecuentemente mediante un sistema de prueba y tanteo. Cuanto más conozca el lenguaje, mayor será la rapidez con que pueda programar y podrá realizar más pruebas preliminares mentalmente.

Entre los pasos básicos de la programación cabe citar los siguientes:

- Definir el problema.
- Desglosar el problema en elementos discretos.
- Construir los elementos.
- Comprobar y perfeccionar los elementos.
- Ensamblar los elementos.



- Comprobar el programa en su conjunto.

A continuación se enumeran algunos aspectos que debe tener presentes al empezar a programar:

- Defina claramente el problema antes de intentar resolverlo. Si no lo hace, acabará por realizar numerosos cambios, desechará código, tendrá que empezar de nuevo o bien terminará con un resultado que no es realmente lo que deseaba.
- Desglose el problema en pasos manejables, en lugar de intentar resolver todo el problema de una sola vez.
- Pruebe y depure secciones de código a medida que desarrolla el programa. Compruebe que el código hace lo que quiere que haga. La depuración es el proceso de encontrar y solucionar problemas que impiden que el código se ejecute correctamente.
- Perfeccione los datos y el almacenamiento de datos para facilitar la manipulación de estos datos a través del código del programa. Esto suele implicar estructurar las tablas de forma adecuada.

En el resto de esta sección se describen los pasos que debe seguir para construir un pequeño programa Visual FoxPro.

## Definir el problema

Antes de poder resolver un problema, debe formularlo claramente. Algunas veces, si ajusta la formulación del problema podrá ver más opciones para resolverlo.

Suponga que obtiene muchos datos de distintos orígenes. Si bien la mayoría de los datos son estrictamente numéricos, algunos valores contienen guiones y espacios en blanco además de números. Suponga que quiere eliminar todos los espacios en blanco y los guiones de dichos campos y guardar los datos numéricos.

En lugar de intentar eliminar los espacios en blanco y los guiones de los datos originales, podría formular el objetivo del programa como:

**Objetivo** Reemplazar los valores existentes de un campo por otros valores que contengan todo lo que contenían los valores originales, excepto los espacios en blanco y los guiones.

Esta formulación evita la dificultad que supone manipular una cadena de caracteres cuya longitud sigue cambiando a medida que trabaja con ella.

## Descomponer el problema

Puesto que tiene que indicar instrucciones específicas a Visual FoxPro en términos de operaciones, comandos y funciones, debe descomponer el problema en pasos discretos. La tarea más discreta para este problema sería examinar cada carácter de la cadena. Hasta que pueda examinar un carácter individualmente, no podrá determinar si desea guardarlo.

Una vez que examine un carácter, deberá comprobar si se trata de un guión o de un espacio en blanco. En este momento, quizá desee refinar la declaración del problema. ¿Y si obtuviera más adelante datos que contienen paréntesis de apertura y de cierre? ¿Y si desea deshacerse de los símbolos de moneda, las comas y los puntos? Cuanto más genérico pueda hacer el código, más

trabajo se ahorrará de ahora en adelante; lo principal es ahorrar trabajo. He aquí una formulación del problema válida para una variedad mucho mayor de datos:

**Objetivo refinado** Reemplazar los valores existentes en un campo por otros valores que contengan únicamente los caracteres numéricos de los valores originales.

Con esta formulación, ahora puede volver a plantear el problema a nivel de carácter: si el carácter es numérico, guardar el carácter; si el carácter es no numérico, pasar al siguiente carácter. Cuando haya construido una cadena que sólo contenga los elementos numéricos de la cadena inicial, podrá reemplazar la primera cadena y pasar al siguiente registro hasta que haya terminado con todos los datos.

Para resumir, el problema se descompone en los siguientes elementos:

1. Examinar cada carácter.
2. Decidir si el carácter es numérico o no.
3. Si es numérico, copiarlo a la segunda cadena.
4. Cuando haya terminado con todos los caracteres de la cadena original, reemplazar la cadena original con la cadena que sólo contiene valores numéricos.
5. Repetir estos pasos para todos los registros de la tabla.

## Construir los elementos

Cuando sepa qué debe hacer, puede empezar a formular los elementos en términos de comandos, funciones y operadores de Visual FoxPro.

Como los comandos y funciones se usan para manipular datos, tiene algunos datos de prueba para trabajar con ellos. Los datos de prueba sirven para simular los datos verdaderos lo mejor posible.

Para este ejemplo puede almacenar en una variable una cadena de prueba introduciendo el siguiente comando en la ventana Comandos:

```
cTest = "123-456-7 89 0"
```

## Examinar cada carácter

En primer lugar desea buscar un único carácter en la cadena. Para obtener una lista de funciones que se pueden utilizar para manipular cadenas, vea [Funciones de carácter](#).

Verá tres funciones que devuelven determinadas secciones de una cadena: [LEFT\(\)](#), [RIGHT\(\)](#) y [SUBSTR\(\)](#). De estas tres funciones, SUBSTR( ) devuelve caracteres de cualquier parte de la cadena.

SUBSTR( ) usa tres argumentos o parámetros: la cadena, la ubicación inicial dentro de la cadena y el número de caracteres que se deben devolver de la cadena, empezando por la ubicación inicial. Para comprobar si SUBSTR( ) va a hacer lo que usted quiere, podría escribir los siguientes comandos en la

ventana Comandos:

```
? SUBSTR(cTest, 1, 1)
? SUBSTR(cTest, 3, 1)
? SUBSTR(cTest, 8, 1)
```

## Resultado

```
1
3
-
```

Puede ver que en la ventana principal de Visual FoxPro se muestran el primer, el tercer y el octavo caracteres de la cadena de prueba.

Para hacer eso mismo varias veces, utilice un bucle. Puesto que la cadena de prueba tiene un número determinado de caracteres (14), puede utilizar un bucle FOR. El contador del bucle FOR se incrementa cada vez que se ejecuta el código del bucle, por lo que puede utilizar el contador de la función SUBSTR( ). Puesto que en la ventana Comandos no puede comprobar las construcciones de bucles, deberá probar la siguiente sección de código en un programa de ejemplo.

## Para crear un programa nuevo

1. Escriba el siguiente comando en la ventana **Comandos**:

```
MODIFY COMMAND numonly
```

2. En la ventana que aparecerá, escriba las siguientes líneas de código:

```
FOR nCnt = 1 TO 14
    ? SUBSTR(cTest, nCnt, 1)
ENDFOR
```

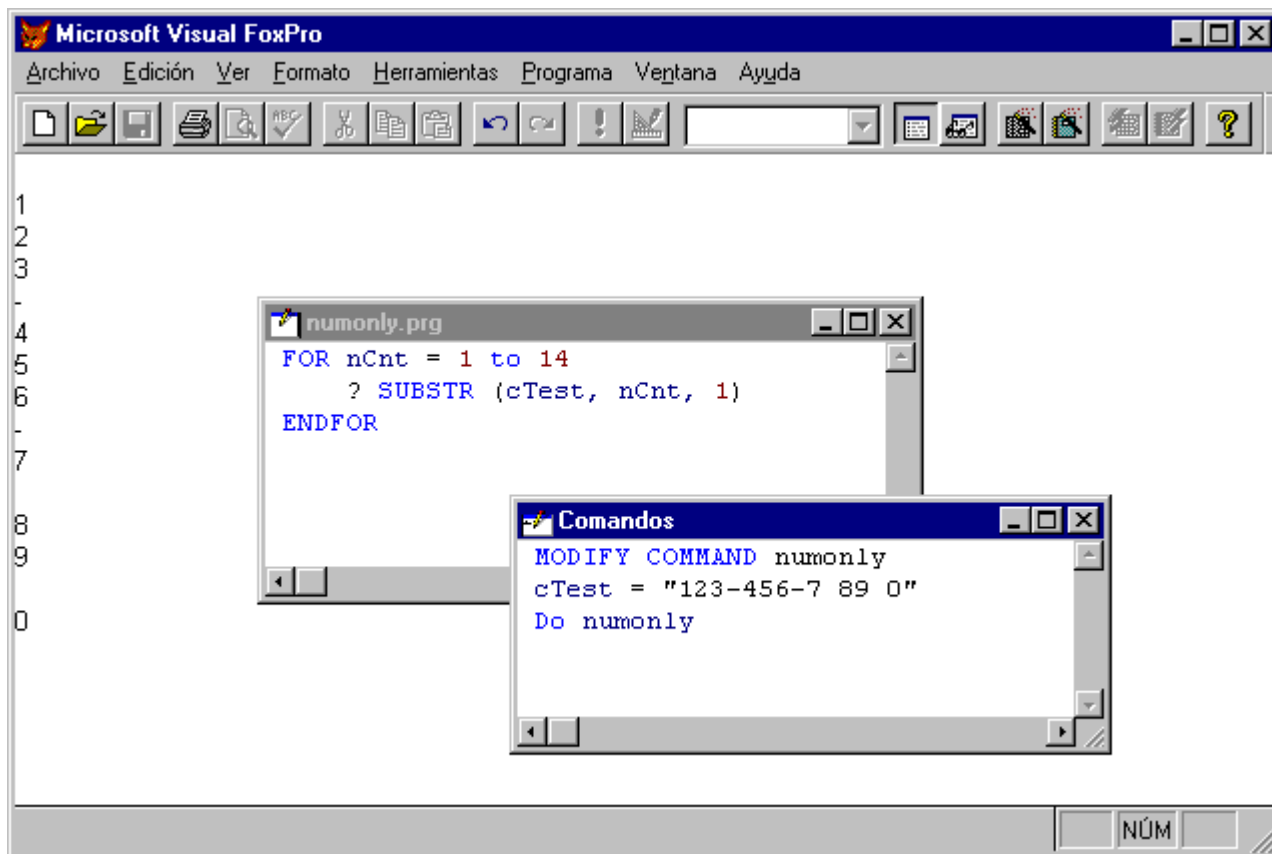
Ahora que ha creado un programa, ya puede ejecutarlo.

## Para ejecutar un programa

1. En la ventana del programa abierto, presione CTRL+E.
2. Si aparece un cuadro de diálogo **Guardar**, elija **Aceptar**.

Cuando ejecute este programa, los caracteres individuales de la cadena de prueba se imprimirán en líneas distintas en la ventana principal de Visual FoxPro.

## Comprobar parte del programa



Ya ha completado la primera tarea. Ahora puede examinar cada carácter de la cadena.

### Decidir si el carácter es numérico

Cuando tenga un único carácter de la cadena, debe saber si se trata de un número. Puede hacerlo con [ISDIGIT\(\)](#).

Puede probar los siguientes comandos en la ventana Comandos:

```
? ISDIGIT('2')  
? ISDIGIT('-')  
? ISDIGIT(SUBSTR(cTest, 3, 1))
```

### Resultado

```
.T.  
.F.  
.T.
```

De este resultado se desprende que '2' es un número, '-' no es un número y el tercer carácter, 3, es un número.

### Si el carácter es numérico, copiarlo a la segunda cadena

Ahora que puede examinar los caracteres y determinar si son o no numéricos, necesita una [variable](#)

para almacenar los valores numéricos: `cNumOnly`.

Para crear la variable, debe asignarle un valor inicial, una cadena de longitud cero:

```
cNumOnly = ""
```

A medida que el bucle FOR recorre la cadena, es conveniente crear otra variable para almacenar temporalmente cada carácter de la cadena a medida que ésta se manipula:

```
cCharacter = SUBSTR(cTest, nCnt, 1)
```

**Sugerencia** Normalmente es mejor almacenar en una variable de memoria el resultado de un cálculo, evaluación o función. Entonces puede manipular la variable en lugar de tener que repetir el cálculo o la evaluación.

La siguiente línea de código puede utilizarse cada vez que se encuentra un número para sumar el número a la segunda cadena:

```
cNumOnly = cNumOnly + cCharacter
```

Hasta ahora, el programa es el siguiente:

```
cNumOnly = ""
FOR nCnt = 1 TO 14
    cCharacter = SUBSTR(cTest, nCnt, 1)
    IF ISDIGIT(cCharacter)
        cNumOnly = cNumOnly + cCharacter
    ENDIF
ENDFOR
```

## Prueba de los elementos

Si agrega un par de comandos al final para imprimir las cadenas y ejecutar el programa, podrá ver que el programa funciona con la cadena de prueba:

```
cNumOnly = ""
FOR nCnt = 1 TO 14
    cCharacter = SUBSTR(cTest, nCnt, 1)
    IF ISDIGIT(cCharacter)
        cNumOnly = cNumOnly + cCharacter
    ENDIF
ENDFOR
? cTest
? cNumOnly
```

## Resultado

```
123-456-7 89 0
1234567890
```

El resultado parece correcto. Pero si cambia la cadena de prueba mientras comprueba los elementos, puede tener problemas. Escriba el siguiente comando en la ventana Comandos y ejecute de nuevo el programa:

```
cTest = "456-789 22"
```

El programa generará un mensaje de error. El bucle FOR ha intentado ejecutarse 14 veces, pero en la cadena sólo había 10 caracteres. Necesita una forma de ajustar las longitudes variables de las cadenas. Use [LEN\(\)](#) para devolver el número de caracteres de una cadena. Si sustituye este comando en el bucle FOR, verá que el programa funciona correctamente con ambas cadenas de prueba:

```
cNumOnly = ""
FOR nCnt = 1 TO LEN(cTest)
    cCharacter = SUBSTR(cTest, nCnt, 1)
    IF ISDIGIT(cCharacter)
        cNumOnly = cNumOnly + cCharacter
    ENDIF
ENDFOR
? cTest
? cNumOnly
```

## Agrupar los elementos

Para completar la solución de programación para este problema, quizá desee volver a leer sus datos de una tabla. Cuando tenga una tabla, explore los registros y aplique su código de programa a un campo de la tabla, en lugar de a una variable.

En primer lugar, podría crear una tabla temporal que contuviera diversas cadenas de prueba. Dicha tabla podría contener un único campo de caracteres llamado Testfield y cuatro o cinco registros:

### Contenido de Testfield

123-456-7 89 0	-9221 9220 94321 99-
456-789 22	000001 98-99-234

Cuando sustituya el nombre de la cadena de prueba por el nombre del campo, el programa será similar al siguiente:

```
FOR nCnt = 1 TO LEN(TestField)
    cCharacter = SUBSTR(TestField, nCnt, 1)
    IF ISDIGIT(cCharacter)
        cNumOnly = cNumOnly + cCharacter
    ENDIF
ENDFOR
? TestField
? cNumOnly
```

Puede ajustar manualmente el puntero de registro si [examina](#) la tabla y se desplaza por ella. Cuando el puntero de registro esté en cada uno de los registros, el programa funcionará de la forma deseada. O bien, ahora puede envolver el código de desplazamiento por la tabla en el resto de su programa:

```
SCAN
    cNumOnly = ""
    FOR nCnt = 1 TO LEN(TestField)
        cCharacter = SUBSTR(TestField, nCnt, 1)
        IF ISDIGIT(cCharacter)
```

```

        cNumOnly = cNumOnly + cCharacter
    ENDIF
ENDFOR
? TestField
? cNumOnly
?
ENDSCAN

```

## Resultado

```

123-456-7 89 0
1234567890

456-789 22
45678922

-9221 9220 94321 99-
922192209432199

000001 98-99-234
0000019899234

```

## Comprobar todo el programa

En lugar de imprimir la cadena al final del programa, quizá desee guardarla en la tabla. Para ello, utilice la siguiente línea de código:

```
REPLACE TestField WITH cNumOnly
```

El programa completo será el siguiente:

```

SCAN
    cNumOnly = ""
    FOR nCnt = 1 TO LEN(TestField)
        cCharacter = SUBSTR(TestField, nCnt, 1)
        IF ISDIGIT(cCharacter)
            cNumOnly = cNumOnly + cCharacter
        ENDIF
    ENDFOR
    REPLACE TestField WITH cNumOnly
ENDSCAN

```

Cuando tenga el programa completo, necesitará probarlo con los datos de ejemplo antes de probarlo con los datos reales.

## Aumentar la robustez del programa

Un programa robusto hace lo que usted quiere que haga, pero también se anticipa a posibles problemas y se encarga de ellos. Este programa hace lo que usted quiere, pero hace algunas suposiciones que deben ser verdaderas para que funcione:

- Hay una tabla abierta en el área de trabajo actual.
- La tabla tiene un campo de caracteres llamado `TestField`.

Si la tabla no está abierta en el área de trabajo actual o si la tabla no tiene un campo de caracteres con

el nombre esperado, el programa generará un mensaje de error y no realizará la tarea prevista.

### Programa para eliminar los caracteres no numéricos de todos los registros de un campo

Código	Comentarios
<code>lFieldOK = .F.</code>	Esta <a href="#">variable</a> determina si existen las condiciones necesarias para que el programa funcione. Inicialmente se establece la variable como falsa (.F.) para suponer que las condiciones necesarias no existen.
<pre> FOR nCnt = 1 TO FCOUNT( )   IF FIELD(nCnt) = ;     UPPER("TestField")     IF TYPE("TestField") = "C"       lFieldOK = .T.     ENDIF     EXIT   ENDIF ENDIF ENDFOR </pre>	Esta sección de código recorre todos los campos de la tabla actual hasta que encuentra un campo de caracteres llamado <code>TestField</code> . En cuanto se encuentra el campo correcto, <code>lFieldOK</code> se establece como verdadera (.T.) y <a href="#">EXIT</a> finaliza el bucle (no hay ninguna razón para continuar con la comprobación una vez identificado el campo correcto). Si ningún campo cumple el criterio, <code>lFieldOK</code> seguirá siendo falso (.F.).
<code>IF lFieldOK</code>	La sección de conversión del programa sólo se ejecuta si en la tabla activa actualmente hay un campo de caracteres llamado <code>TestField</code> .
<pre> SCAN   cNumOnly = ""   FOR nCnt = 1 TO LEN(TestField)     cCharacter = ;     SUBSTR(TestField, nCnt, 1)     IF ISDIGIT(cCharacter)       cNumOnly = cNumOnly + ;       cCharacter     ENDIF   ENDFOR </pre>	El código de conversión.
<pre>   REPLACE TestField WITH ;   cNumOnly ENDSCAN </pre>	
<code>ENDIF</code>	Fin de la condición <code>IF lFieldOK</code> .

La mayor limitación de este programa es que sólo puede utilizarlo para un campo. Si desea eliminar los caracteres no numéricos de un campo distinto de `TestField`, tendrá que recorrer el programa y cambiar cada aparición de `TestField` por el nombre del otro campo.

Convertir el programa a un procedimiento, según se explica en las próximas secciones, permite hacer que el código que ha escrito sea más genérico y más reutilizable, con lo que ahorrará trabajo más adelante.



## Usar procedimientos y funciones definidas por el usuario

Los procedimientos y funciones permiten mantener en un único lugar el código que utiliza con frecuencia y llamarlo a través de su aplicación siempre que lo necesite. Esto hace que su código sea más fácil de leer y mantener, ya que en un procedimiento el cambio se realiza una sola vez, no varias veces como ocurre en un programa.

En Visual FoxPro, los procedimientos son similares a éste:

```
PROCEDURE miproc
  * Esto es un comentario, pero podría ser código ejecutable
ENDPROC
```

Tradicionalmente, los procedimientos contienen código que usted escribe para realizar una operación y funciones que calculan y devuelven un valor. En Visual FoxPro, las funciones son similares a los procedimientos:

```
FUNCTION mifunción
  * Esto es un comentario, pero podría ser código ejecutable
ENDFUNC
```

Puede incluir procedimientos y funciones en un archivo de programa distinto o al final de un archivo de programa que contenga código normal de programa. En un archivo de programa no puede tener código ejecutable de programa a continuación de los procedimientos y las funciones.

Si incluye sus procedimientos y funciones en un archivo de programa distinto, podrá hacer accesibles estos procedimientos y funciones desde su programa si utiliza el comando [SET PROCEDURE TO](#). Por ejemplo, para un archivo llamado FUNPROC.PRG, utilice el siguiente comando en la ventana Comandos:

```
SET PROCEDURE TO funproc.prg
```

### Llamar a un procedimiento o a una función

Hay dos formas de llamar a un procedimiento o a una función en sus programas:

- Utilizar el comando [DO](#). Por ejemplo:

```
DO miproc
```

–O bien–

- Incluir detrás del nombre de la función un par de paréntesis. Por ejemplo:

```
mifunción( )
```

Cada uno de estos métodos puede ampliarse enviando o recibiendo valores desde el procedimiento o la función.

## Enviar valores a un procedimiento o a una función

Para enviar valores a procedimientos o funciones se incluyen [parámetros](#). Por ejemplo, el procedimiento siguiente acepta un solo parámetro:

```
PROCEDURE miproc( cString )
    * La línea siguiente muestra un mensaje
    MESSAGEBOX ("miproc" + cString)
ENDPROC
```

**Nota** Incluir los parámetros dentro de los paréntesis en la línea de definición de un procedimiento o una función, por ejemplo `PROCEDURE miproc(cString)`, indica que el parámetro tiene [alcance](#) local al procedimiento o la función. También puede permitir que una función o un procedimiento acepte parámetros de alcance local mediante [LPARAMETERS](#).

Los parámetros funcionan de manera idéntica en una función. Para enviar un valor como un parámetro de este procedimiento o a una función, podría utilizar una cadena o una [variable](#) que contuviera una cadena, como se muestra en la tabla siguiente.

## Transferencia de parámetros

Código	Comentarios
DO miproc WITH cTestString DO miproc WITH "cadena de prueba"	Llama a un procedimiento y transfiere una variable de caracteres o un literal de cadena.
mifunción("cadena de prueba") mifunción( cTestString )	Llama a una función y transfiere una copia de una cadena literal o una variable de caracteres.

**Nota** Si llama un procedimiento o función sin usar el comando DO, la configuración de [UDFPARMS](#) controla cómo se transfieren los parámetros. De forma predeterminada, UDFPARMS se establece como VALUE, por lo que se transferirán copias de los parámetros. Cuando utilice DO, se empleará el parámetro real (el parámetro se transfiere por referencia) y cualquier cambio realizado en el procedimiento o en la función se reflejarán en los datos originales, cualquiera que sea la configuración de UDFPARMS.

Puede enviar múltiples valores a un procedimiento o función si los separa mediante comas. Por ejemplo, el siguiente procedimiento espera tres parámetros: una fecha, una cadena de caracteres y un número.

```
PROCEDURE miproc( dDate, cString, nTimesToPrint )
    FOR nCnt = 1 to nTimesToPrint
        ? DTOC(dDate) + " " + cString + " " + STR(nCnt)
    ENDFOR
ENDPROC
```

Podría llamar a este procedimiento mediante la siguiente línea de código:

```
DO miproc WITH DATE(), "Hola", 10
```

## Recibir valores desde una función

El valor devuelto de forma predeterminada es verdadero (.T.), pero puede utilizar el comando [RETURN](#) para devolver cualquier valor. Por ejemplo, la siguiente función devuelve una fecha correspondiente a dos semanas después de la fecha que se ha pasado como parámetro.

```
FUNCTION plus2weeks
PARAMETERS dDate
    RETURN dDate + 14
ENDFUNC
```

La siguiente línea de código almacena el valor devuelto desde esta función en una variable:

```
dDeadLine = plus2weeks(DATE())
```

En la tabla siguiente se muestran las formas en que puede almacenar o mostrar valores devueltos por una función.

## Manipular valores devueltos

Código	Comentarios
<code>var = mifunción( )</code>	Almacena en una <a href="#">variable</a> el valor devuelto por la función.
<code>? mifunción( )</code>	Imprime en la ventana activa de salida el valor devuelto por la función.

## Comprobar parámetros en un procedimiento o en una función

Es conveniente comprobar que los parámetros enviados a su procedimiento o a su función son los que espera recibir. Puede utilizar las funciones [TYPE\(\)](#) y [PARAMETERS\(\)](#) para comprobar el tipo y el número de parámetros enviados a su procedimiento o a su función.

El ejemplo de la sección anterior necesita recibir un parámetro de tipo Fecha. Puede utilizar la función `TYPE( )` para asegurarse de que el valor que su función recibe es del tipo adecuado.

```
FUNCTION plus2weeks( dDate )
    IF TYPE("dDate") = "D"
        RETURN dDate + 14
    ELSE
        MESSAGEBOX( "requiere un parámetro de tipo Fecha" )
        RETURN {}      && Devuelve una fecha vacía
    ENDIF
ENDFUNC
```

Si un procedimiento espera menos parámetros de los que recibe, Visual FoxPro generará un mensaje de error. Por ejemplo, si especificó dos parámetros pero llamó al procedimiento con tres parámetros, obtendrá un mensaje de error. Pero si un procedimiento espera más parámetros de los que recibe, los parámetros adicionales simplemente se inicializarán como falso (.F.). Puesto que no hay ninguna forma de decir si el último parámetro se estableció como falso (.F.) o se omitió, el siguiente

procedimiento comprueba que se ha enviado el número correcto de parámetros:

```
PROCEDURE SaveValue( cStoreTo, cNewVal, lIsInTable )
  IF PARAMETERS( ) < 3
    MESSAGEBOX( "No se han pasado suficientes parámetros". )
    RETURN .F.
  ENDIF
  IF lIsInTable
    REPLACE (cStoreTo) WITH (cNewVal)
  ELSE
    &cStoreTo = cNewVal
  ENDIF
  RETURN .T.
ENDPROC
```

## Convertir el programa NUMONLY en una función

NUMONLY.PRG, el programa de ejemplo descrito en la sección [El proceso de la programación](#), puede hacerse más robusto y útil si crea una función para la parte del programa que elimina los caracteres no numéricos de una cadena.

### Procedimiento de ejemplo para devolver caracteres numéricos de una cadena

Código	Comentarios
FUNCTION NumbersOnly( cMixedVal )	Principio de la función, que acepta una cadena de caracteres.
<pre> cNumOnly = "" FOR nCnt = 1 TO LEN(cMixedVal)   cCharacter = ;   SUBSTR(cMixedVal, nCnt, 1)   IF ISDIGIT(cCharacter)     cNumOnly = ;     cNumOnly + cCharacter   ENDIF ENDFOR </pre>	Crea una cadena que sólo tiene los caracteres numéricos de la cadena original.
RETURN cNumOnly	Devuelve la cadena que sólo tiene caracteres numéricos.
ENDFUNC	Fin de la función.

Además de permitirle utilizar este código en múltiples situaciones, esta función mejora la legibilidad del programa:

```
SCAN
  REPLACE FieldName WITH NumbersOnly(FieldName)
ENDSCAN
```

O, más sencillo aún:

```
REPLACE ALL FieldName WITH NumbersOnly(FieldName)
```

## Cómo avanzar

La programación por procedimientos, junto con la programación orientada a objetos y las herramientas de diseño de Visual FoxPro, pueden ayudarle a programar una versátil aplicación de Visual FoxPro. El resto de este manual explica temas con los que se encontrará a medida que programe aplicaciones de Visual FoxPro.

Para obtener más información acerca de la programación orientada a objetos, consulte el capítulo 3, [Programación orientada a objetos](#). Para aprender a diseñar formularios con el Diseñador de formularios, consulte el capítulo 9, [Crear formularios](#).

## Capítulo 2: Programar una aplicación

Una aplicación de Visual FoxPro incluye normalmente una o más [bases de datos](#), un programa principal que configura el entorno del sistema para la aplicación y una interfaz de usuario compuesta por [formularios](#), [barras de herramientas](#) y [menús](#). Las [consultas](#) y los informes permiten que los usuarios extraigan información de sus datos.

Este capítulo trata los temas siguientes:

- [Diseñar la aplicación](#)
- [Crear bases de datos](#)
- [Crear clases](#)
- [Proporcionar acceso a la funcionalidad](#)
- [Proporcionar acceso a la información](#)
- [Pruebas y depuración](#)
- Para obtener información acerca de la programación de aplicaciones con el Generador de aplicaciones y el Marco de aplicaciones mejorado, vea [Programar aplicaciones con el Marco de aplicaciones](#).

### Diseñar la aplicación

Un diseño apropiado ahorra tiempo, esfuerzo, dinero y permite mantener la cordura al programar. Cuanto más implique a los usuarios en el proceso de diseño, mejor. No importa lo cuidadosamente que se diseñe; aun así, acabará refinando las especificaciones a medida que avance el proyecto y los usuarios le proporcionen información adicional.

Algunas de las decisiones de diseño que tome afectarán a la forma en que creará elementos de la aplicación. ¿Quién utilizará la aplicación? ¿Cuál es el centro de actividad del usuario? ¿Con qué cantidad de datos se supone que se trabajará? ¿Se utilizarán servidores de datos de apoyo o los datos serán exclusivamente locales para un único usuario o para múltiples usuarios a través de una red? Considere estos factores antes de avanzar demasiado en el proyecto.

#### Actividades frecuentes de usuario

Incluso si sus usuarios finales trabajan con clientes, pedidos y piezas, lo que determinará la forma en que la aplicación deberá tratar los datos es el modo en que los usuarios trabajan con esa información. Un formulario de entrada de pedidos, como el de Tastrade.app (en el directorio ...\\Samples\\Vfp98\\Tastrade de Visual Studio), puede ser necesario para algunas aplicaciones, pero no sería una buena herramienta para administrar inventarios o para hacer un seguimiento de las ventas, por ejemplo.

### **Tamaño de la base de datos**

Deberá pensar más en el rendimiento si trata con grandes conjuntos de datos. En el capítulo 15, [Optimizar aplicaciones](#), se explican los métodos para optimizar el rendimiento. También puede desear ajustar el modo en que los usuarios pueden desplazarse por los datos. Si tiene veinte o treinta registros en una tabla, está bien que permita que los usuarios desplacen el puntero de registro de una tabla un registro cada vez. Si tiene veinte o treinta mil registros, deberá proporcionar otros sistemas para obtener los datos deseados: agregar listas o cuadros de diálogos de búsqueda, [filtros](#), [consultas](#) personalizadas, etc. El capítulo 10, [Usar controles](#), explica la forma de utilizar una lista para seleccionar registros específicos de una tabla. En el capítulo 8, [Crear vistas](#), se explica la creación de consultas parametrizadas.

### **Usuario individual frente a múltiples usuarios**

Es conveniente crear la aplicación pensando que múltiples usuarios tendrán acceso simultáneo a la base de datos. Visual FoxPro facilita la programación para acceso compartido. En el capítulo 17, [Programar para acceso compartido](#), se describen técnicas para permitir que varios usuarios tengan acceso simultáneo a la base de datos.

### **Consideraciones internacionales**

Si sabe que su aplicación sólo se utilizará en el entorno de un único idioma, no debe preocuparse de la internacionalización. Por otra parte, si desea ampliar su mercado, o si sus usuarios deben trabajar con datos o configuraciones de entorno internacionales, deberá tener en cuenta estos factores al crear la aplicación. En el capítulo 18, [Programar aplicaciones internacionales](#), se explican los puntos que debe tener en cuenta cuando programe aplicaciones internacionales.

### **Datos locales frente a datos remotos**

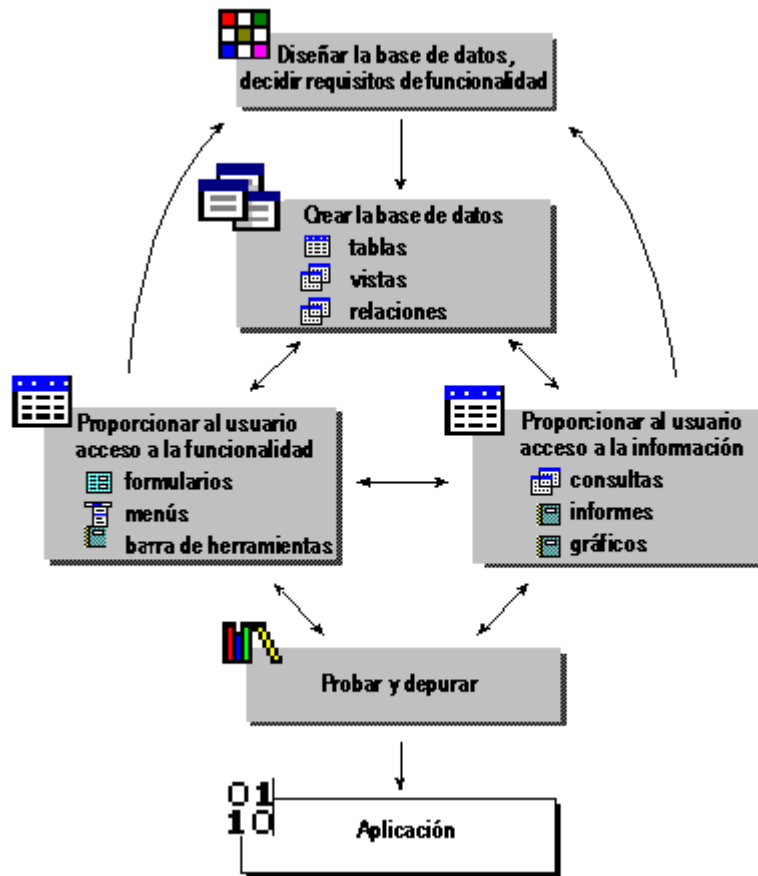
Si su aplicación trata con datos remotos, deberá administrarlos de forma diferente a como administraría los datos nativos de Visual FoxPro. En el capítulo 8, [Crear vistas](#), se explica la forma de crear vistas para datos locales o remotos. En la parte 6 del *Manual del programador*, [Crear soluciones cliente-servidor](#), se explica cómo diseñar aplicaciones que trabajen sin problemas con datos remotos.

## **Descripción general del proceso**

El proceso de crear una aplicación es muy repetitivo. Puesto que no hay dos aplicaciones exactamente iguales, lo que hará probablemente será definir prototipos y refinar algunos componentes varias veces antes de obtener un producto final. Las expectativas de los usuarios finales, o sus solicitudes, también pueden cambiar, lo que hará necesario redefinir aspectos de la aplicación. Además, nadie escribe código libre de errores, por lo que las pruebas y la depuración conducen a algún tipo de rediseño o

rescritura.

## El proceso de creación de una aplicación



Además de tener en cuenta la imagen general durante la fase de diseño, tendrán que decidir cuáles son las funciones necesarias, qué datos están implicados y cómo debe estar estructurada la base de datos. Tendrá que diseñar una interfaz para que el usuario tenga acceso a la funcionalidad de la aplicación. Puede crear informes y [consultas](#) para que los usuarios puedan extraer información útil de sus datos.

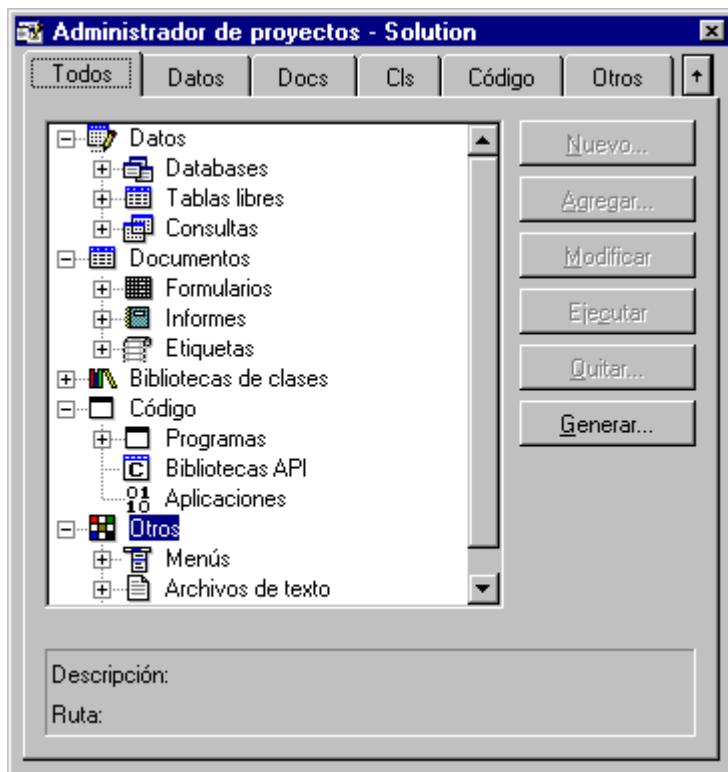
## Iniciar la programación

Después de haber pensado qué componentes son necesarios en la aplicación, es posible que quiera configurar un conjunto de directorios y un proyecto para organizar los archivos componentes que desea crear para la aplicación. Puede crear el conjunto de directorios mediante el Explorador de Windows y puede crear el proyecto en el [Administrador de proyectos](#) o usar el [Asistente para aplicaciones](#) para configurarlos a la vez. Este nuevo Asistente para aplicaciones abre el [Generador de aplicaciones](#) para que pueda personalizar aún más un proyecto y los componentes que inicie en el Asistente. Por compatibilidad con versiones anteriores, puede elegir el [Asistente para aplicaciones \(5.0\)](#) anterior.

## Usar el Administrador de proyectos

El Administrador de proyectos permite compilar la aplicación completa. En la fase de programación de la aplicación, el Administrador de proyectos facilita el diseño, la modificación y la ejecución de los componentes individuales de su aplicación.

## El Administrador de proyectos



Cuando utilice el Administrador de proyectos, podrá:

- Modificar y ejecutar partes de su aplicación (formularios, menús, programas) con tan sólo algunos clics.
- Arrastrar clases, tablas y campos desde el Administrador de proyectos hasta el [Diseñador de formularios](#) o el [Diseñador de clases](#).
- Arrastrar clases entre bibliotecas de clases.
- Ver y modificar fácilmente sus tablas y bases de datos.
- Agregar descripciones para los componentes de la aplicación.
- Arrastrar y colocar elementos entre proyectos.

Para obtener información detallada acerca del uso del Administrador de proyectos, consulte el capítulo 1, [Introducción](#), del *Manual del usuario*. Para obtener información acerca de la compilación de aplicaciones, consulte el capítulo 13, [Compilar una aplicación](#), en este manual.

## Crear bases de datos

Puesto que una aplicación de base de datos depende tanto de los datos subyacentes, la mejor forma de empezar a diseñar su aplicación es comenzar por los datos. Puede configurar su propia base de datos y determinar cuáles serán las relaciones entre las tablas, qué reglas comerciales va a exigir, etc. antes



de diseñar cualquier componente de interfaz o de manipulación de datos. La creación de una estructura sólida de la base de datos hará que el trabajo de programación sea mucho más sencillo.

En el capítulo 5, [Diseñar bases de datos](#), el capítulo 6, [Crear bases de datos](#) y el capítulo 7, [Trabajar con tablas](#), se explican los temas de diseño y uso de Visual FoxPro para diseñar tablas y bases de datos efectivas y eficaces.

## Crear clases

Puede crear una aplicación robusta, orientada a objetos y controlada por eventos utilizando únicamente las [clases](#) de base de Visual FoxPro. Es posible que no tenga que crear una clase, pero deseará hacerlo. Además de hacer que el código sea más manejable y sencillo de mantener, una biblioteca de clases sólida le permite crear rápidamente prototipos e incorporar funciones a una aplicación. Puede crear clases en un archivo de programa, en el [Diseñador de formularios](#) (mediante el comando **Guardar como clase** del menú **Archivo**) o en el [Diseñador de clases](#).

El capítulo 3, [Programación orientada a objetos](#), trata algunos de los beneficios de la creación de clases y detalla su creación con el Diseñador de clases o mediante programación.

## Proporcionar acceso a la funcionalidad

La satisfacción del usuario se verá influenciada en gran medida por la interfaz que le proporcione para la funcionalidad de su aplicación. Puede tener un modelo de clases muy limpio, código muy elegante y soluciones muy inteligentes para los problemas difíciles de su aplicación, pero esto casi siempre está oculto a los clientes. Lo que ellos ven es la interfaz que usted les proporciona. Afortunadamente, las herramientas de diseño de Visual FoxPro facilitan la creación de interfaces atractivas y ricas en características.

La interfaz de usuario consiste principalmente en [formularios](#), [barras de herramientas](#) y [menús](#). Puede asociar toda la funcionalidad de su aplicación con [controles](#) o comandos de menú en la interfaz. En el capítulo 9, [Crear formularios](#), se describe la creación de formularios y [conjuntos de formularios](#). La utilización de controles de Visual FoxPro en los formularios se trata en el capítulo 10, [Usar controles](#). Consulte el capítulo 11, [Diseñar menús y barras de herramientas](#), para dar los últimos retoques a la aplicación.

## Proporcionar acceso a la información

Probablemente muestre cierta información para los usuarios en [formularios](#), pero también deseará ofrecer a los usuarios la posibilidad de especificar exactamente la información que desean ver, así como la opción de imprimirla en informes o etiquetas. Las [consultas](#), especialmente las consultas que aceptan parámetros definidos por el usuario, permiten a los usuarios tener más control sobre sus datos. Los informes permiten a los usuarios imprimir imágenes totales, parciales o de resumen de sus datos. Los [controles ActiveX](#) y la [automatización](#) permiten que su aplicación comparta información y funciones con otras aplicaciones.

El [Diseñador de consultas](#) y el [Diseñador de informes](#) se describen en los capítulos 4 a 7 del *Manual del usuario*. En el capítulo 12 de este manual, [Agregar consultas e informes](#), se explica la integración de consultas e informes en una aplicación. El capítulo 16, [Agregar OLE](#) describe la integración de

OLE en una aplicación.

## Probar y depurar

La prueba y depuración es algo que la mayoría de los programadores hace en cada paso del proceso de desarrollo. Es conveniente probar y depurar a medida se va avanzando. Si crea un formulario, querrá asegurarse de que éste hace lo que se desea antes de continuar con otros elementos de su aplicación.

En el capítulo 14, [Probar y depurar aplicaciones](#), se explica el uso de las herramientas de depuración de Visual FoxPro para depurar sus aplicaciones y se ofrecen sugerencias para que el proceso resulte más sencillo.

## Capítulo 3: Programación orientada a objetos

Aunque Visual FoxPro admite la programación estándar por procedimientos, se ha ampliado la capacidad del lenguaje para proporcionar la potencia y la flexibilidad propias de la programación orientada a objetos.

El diseño orientado a objetos y la programación orientada a objetos representan un cambio de perspectiva con respecto a la programación estándar por procedimientos. En lugar de pensar en el flujo del programa desde la primera hasta la última línea de código, se debe pensar en la creación de objetos: componentes autocontenidos de una aplicación que tienen funcionalidad privada además de la funcionalidad que se puede exponer al usuario.

En este capítulo se tratan los temas siguientes:

- [Descripción de los objetos de Visual FoxPro](#)
- [Descripción de las clases de Visual FoxPro](#)
- [Adaptar la clase a la tarea](#)
- [Crear clases](#)
- [Agregar clases a formularios](#)
- [Definir clases mediante programación](#)

## Descripción de los objetos de Visual FoxPro

En Visual FoxPro, los [formularios](#) y los [controles](#) son objetos que puede incluir en sus aplicaciones. Puede manipular estos objetos a través de sus [propiedades](#), [eventos](#) y [métodos](#).

Las mejoras en el lenguaje orientado a objetos de Visual FoxPro proporcionan un mayor control sobre los objetos de las aplicaciones. Asimismo, facilitan la creación y el mantenimiento de bibliotecas de código reutilizable, proporcionando:

- Código más compacto.
- Incorporación más sencilla del código a las aplicaciones sin necesidad de elaborar esquemas de asignación de nombres.
- Menos complejidad al integrar código de distintos archivos en una aplicación.

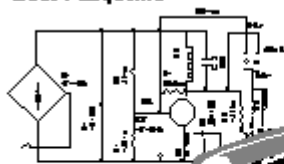
La programación orientada a objetos es en gran medida un modo de empaquetar código de manera que se pueda volver a utilizar y mantener más fácilmente. Los paquetes principales se llaman [clases](#).

## Clases y objetos: los bloques funcionales de las aplicaciones

Las clases y los objetos están estrechamente relacionados, pero no son lo mismo. Una clase contiene información sobre cuál debe ser la apariencia y el comportamiento de un objeto. Una clase es el plano o esquema de un objeto. Por ejemplo, el esquema eléctrico y de diseño de un teléfono sería algo similar a una clase. El objeto o una instancia de la clase sería el teléfono.

**La clase determina las características del objeto.**

**Clase / Esquema**



**Objeto / Teléfono**

## Los objetos tienen propiedades

Un objeto tiene ciertas [propiedades](#) o atributos. Por ejemplo, un teléfono tiene un color y un tamaño determinados. Cuando se instala un teléfono en la oficina, tiene una determinada posición sobre la mesa. El receptor puede estar colgado o descolgado.

Los objetos que se crean en Visual FoxPro también tienen propiedades que están determinadas por la clase en la que se basa el objeto. Estas propiedades pueden establecerse en [tiempo de diseño](#) o en [tiempo de ejecución](#).

Por ejemplo, en la tabla siguiente se indican algunas propiedades que puede tener una [casilla de verificación](#).

Propiedad	Descripción
Caption	Texto descriptivo que aparece junto a la casilla de verificación.
Enabled	Especifica si un usuario puede elegir o no la casilla de verificación.
ForeColor	Color del texto del título.
Left	Posición del extremo izquierdo de la casilla de verificación.
MousePointer	Apariencia del puntero del <i>mouse</i> (ratón) cuando está situado sobre la casilla de verificación.
Top	Posición de la parte superior de la casilla de verificación.

---



---

Visible	Especifica si la casilla de verificación es visible o no.
---------	---

---



---

## Los objetos tienen eventos y métodos asociados

Cada objeto reconoce y puede responder a determinadas acciones denominadas [eventos](#). Un evento es una actividad específica y predeterminada, iniciada por el usuario o por el sistema. Los eventos, en la mayor parte de los casos, se generan por interacción del usuario. Por ejemplo, con un teléfono, se desencadena un evento cuando un usuario descuelga el receptor. Los eventos también se desencadenan cuando el usuario presiona los botones para efectuar una llamada.

En Visual FoxPro, las acciones del usuario que desencadenan eventos incluyen clics, movimientos del *mouse* y pulsaciones de teclas. Inicializar un objeto y encontrar una línea de código que produce un error son eventos iniciados por el sistema.

Los [métodos](#) son procedimientos asociados a un objeto. Los métodos se diferencian de los procedimientos normales de Visual FoxPro en que están vinculados inseparablemente a un objeto y tienen nombres distintos que los procedimientos normales de Visual FoxPro.

Los eventos pueden tener métodos asociados. Por ejemplo, si escribe código de método para el evento Click, ese código se ejecutará cuando se produzca el evento Click. Los métodos también pueden existir independientemente de los eventos. Se debe llamar a estos métodos de forma explícita en el código.

El conjunto de eventos es limitado, aunque amplio. No es posible crear nuevos eventos. Sin embargo, el conjunto de métodos puede ampliarse indefinidamente.

La tabla siguiente muestra algunos de los eventos asociados a una [casilla de verificación](#):

Evento	Descripción
<a href="#">Click</a>	El usuario hace clic en la casilla de verificación.
<a href="#">GotFocus</a>	El usuario activa la casilla de verificación al hacer clic en ella o al llegar a ella a través de la tecla TAB.
<a href="#">LostFocus</a>	El usuario selecciona otro control.

La tabla siguiente muestra algunos métodos asociados a una casilla de verificación:

Método	Descripción
<a href="#">Refresh</a>	El valor de la casilla de verificación se actualiza para reflejar los cambios que se puedan haber producido en el origen de datos subyacente.
<a href="#">SetFocus</a>	El enfoque se establece en la casilla de verificación como si el usuario hubiera presionado la tecla TAB hasta activar la casilla de verificación.

Consulte el capítulo 4, [Descripción del modelo de eventos](#) si desea obtener una explicación del orden

en que se producen los eventos.

## Descripción de las clases de Visual FoxPro

Todas las [propiedades](#), [eventos](#) y [métodos](#) de un objeto se especifican en la definición de clase. Además, las clases tienen las siguientes características que las hacen especialmente útiles para crear código reutilizable y fácil de mantener:

- Encapsulamiento
- Subclases
- Herencia

## Ocultar la complejidad innecesaria

Cuando instale un teléfono en la oficina, lo más probable es que no le interese el funcionamiento interno del aparato para la recepción de llamadas, la realización o la finalización de conexiones con centralitas electrónicas o la conversión de las pulsaciones de tecla en señales electrónicas. Lo único que necesitará saber es que puede levantar el auricular, marcar los números apropiados y hablar con la persona con la que desea hablar. La complejidad de realizar esa conexión queda oculta. La ventaja de ignorar los detalles internos de un objeto para poder centrarse en los aspectos del objeto que necesita utilizar se denomina [abstracción](#).

**La complejidad interna puede estar oculta**



El [encapsulamiento](#), que empaqueta el código de métodos y propiedades en un mismo objeto, contribuye a la abstracción. Por ejemplo, las propiedades que determinan los elementos de un [cuadro de lista](#) y el código que se ejecuta al elegir un elemento de la lista pueden encapsularse en un único [control](#) que se agrega a un [formulario](#).

## Aprovechar la potencia de las clases existentes

Una [subclase](#) puede tener toda la funcionalidad de una clase existente, además de la funcionalidad y los controles adicionales que quiera darle. Si la clase es un teléfono básico, podrá tener subclases que tengan toda la funcionalidad del teléfono original y todas las características especializadas que desee darles.

**Las subclases le permiten reutilizar código.**



La creación de subclases es un modo de reducir la cantidad de código que hay que escribir. Puede comenzar definiendo un objeto que sea similar al deseado y personalizarlo.

## Simplificar el mantenimiento de código

Con la [herencia](#), si realiza un cambio en una clase, ese cambio se reflejará en todas las subclases que se basen en ella. Esta actualización automática ahorra tiempo y trabajo. Por ejemplo, si un fabricante de teléfonos quisiera cambiar los teléfonos de tipo marcación por aparatos de pulsación, se ahorraría mucho trabajo si pudiera hacer el cambio en el diagrama original y hacer que todos los teléfonos fabricados anteriormente con ese diagrama heredaran automáticamente la nueva característica, en lugar de tener que agregarla a todos los teléfonos existentes individualmente.

**La herencia facilita el mantenimiento del código.**

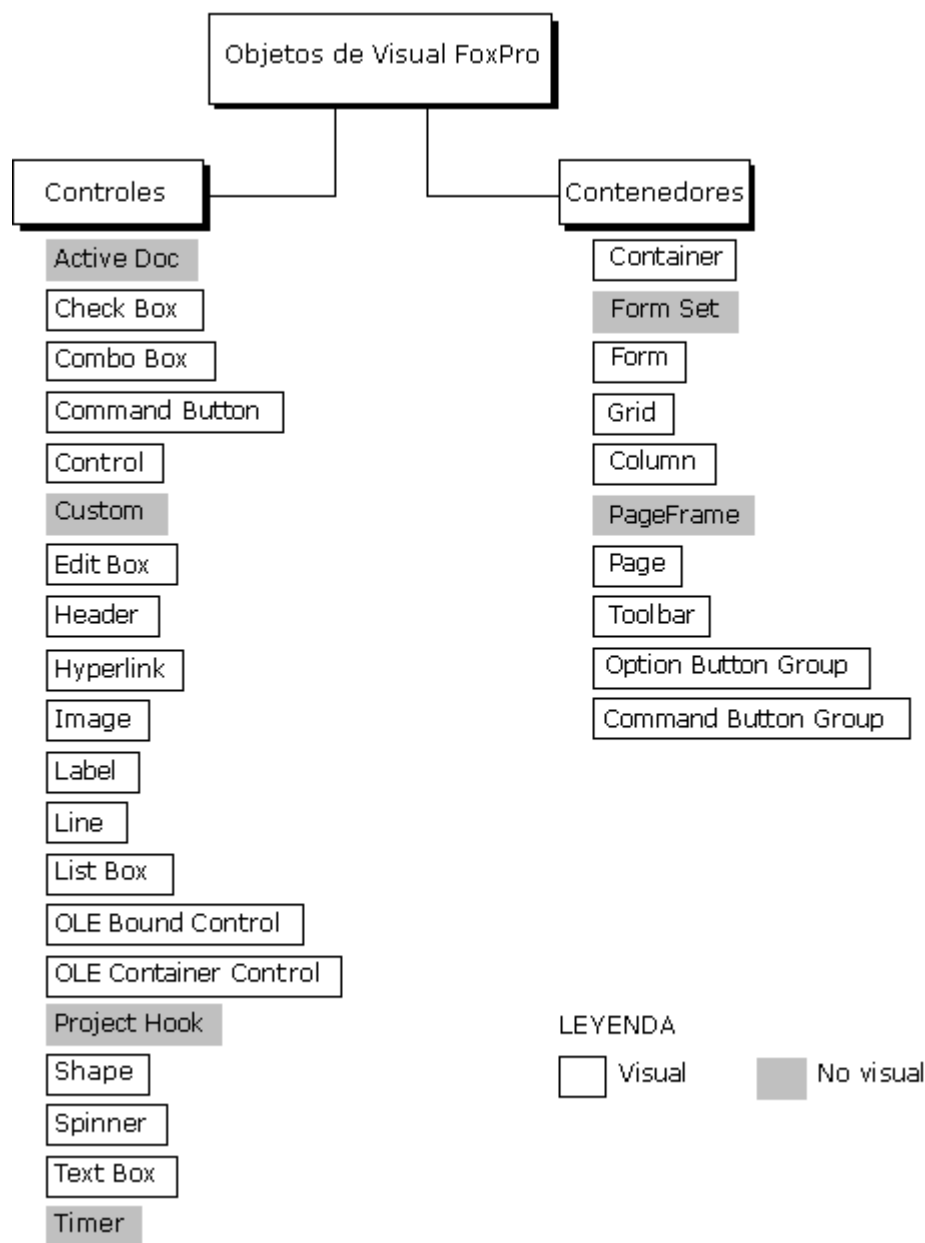


La herencia no funciona con el hardware, pero sí en el software. Si descubre un error en una clase, en lugar de tener que cambiar el código de todas las subclases podrá corregirlo una única vez en la clase y el cambio se propagará a todas las subclases pertenecientes a ella.

## Jerarquía de clases de Visual FoxPro

A la hora de crear clases definidas por el usuario, resulta útil comprender la jerarquía de clases de Visual FoxPro.

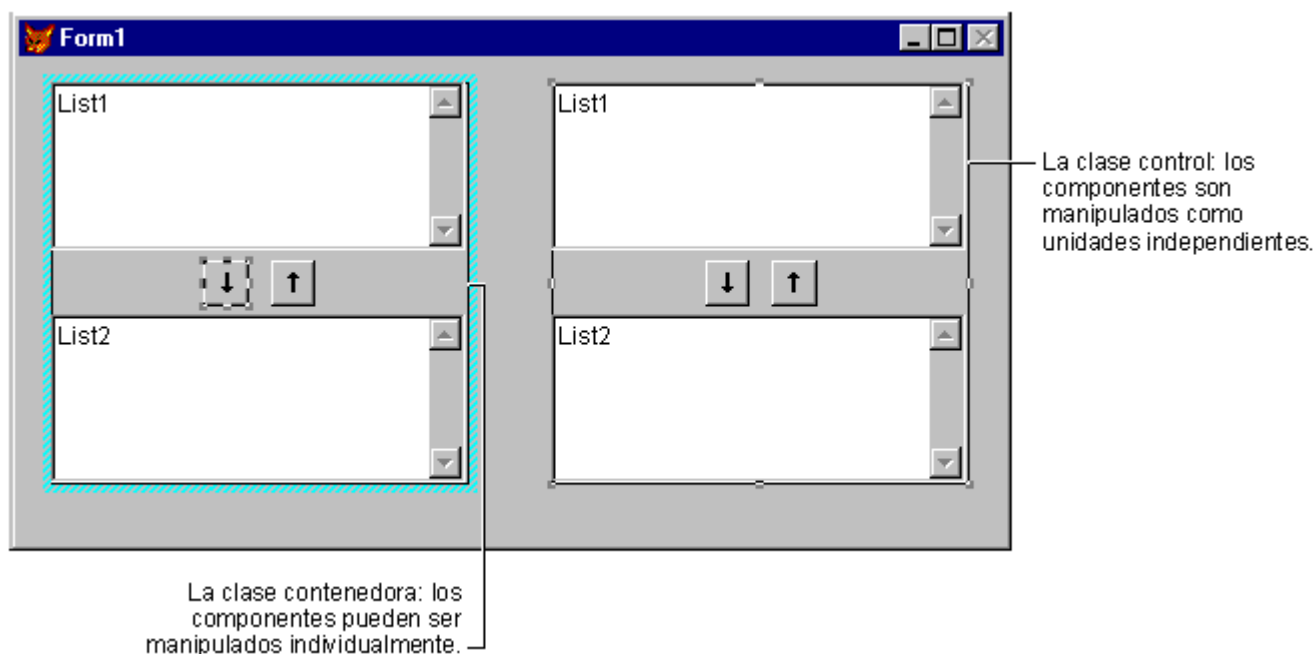
### Jerarquía de clases de Visual FoxPro



### Contenedores y no contenedores

Los dos tipos principales de clases de Visual FoxPro y por extensión, de objetos de Visual FoxPro, son las clases de contenedor y las clases de control.

### Clases de contenedor y clases de control



## Clases de contenedor

Los contenedores pueden incluir otros objetos y permiten el acceso a los objetos que contienen. Por ejemplo, si crea una clase de contenedor que consta de dos [cuadros de lista](#) y dos [botones de comando](#) y, a continuación, agrega a un formulario un objeto basado en esta clase, cada objeto individual podrá manipularse en [tiempo de ejecución](#) y en [tiempo de diseño](#). Puede cambiar fácilmente las posiciones de los cuadros de lista o los títulos de los botones de comando. También puede agregar objetos al [control](#) en tiempo de diseño; por ejemplo, puede agregar etiquetas para identificar los cuadros de lista.

La tabla siguiente muestra los posibles componentes de cada clase de contenedor:

Contenedor	Puede contener
Grupos de botones de comando	Botones de comando
Contenedor	Cualquier control
Control	Cualquier control
Personalizado	Cualquier control, marcos de página, contenedor, personalizado
<a href="#">Conjuntos de formularios</a>	Formularios, <a href="#">barras de herramientas</a>
<a href="#">Formularios</a>	Marcos de página, cualquier control, contenedores, personalizado
Columnas de cuadrícula	Encabezados de columnas, cualquier objeto excepto conjuntos de formularios, formularios, barras de herramientas, <a href="#">cronómetros</a> y otras columnas



<a href="#">Cuadrículas</a>	Columnas de cuadrícula
Grupos de botones de opción	Botones de opción
Marcos de página	Páginas
Páginas	Cualquier control, contenedores, personalizado
Proyecto	Archivos, servidores
Barras de herramientas	Cualquier control, marcos de página, contenedor

## Clases de control

Las clases de control están más encapsuladas que las clases de contenedor, pero por esa misma razón es posible que sean menos flexibles. Las clases de control no tienen un [método AddObject](#).

## Adaptar la clase a la tarea

Es conveniente poder usar clases en muchos contextos distintos. Un diseño inteligente le permitirá decidir con mayor efectividad qué clases desea diseñar y qué funcionalidad va a incluir en la clase.

## Decidir cuándo crear clases

Puede crear una clase para cada [control](#) y cada [formulario](#) que utilice, aunque éste no es el modo más efectivo de diseñar aplicaciones. Es muy probable que acabe con múltiples clases que tengan prácticamente la misma función y que deban mantenerse por separado.

## Encapsular funcionalidad genérica

Cree una clase de control para funcionalidad genérica. Por ejemplo, los [botones de comando](#) que permiten al usuario mover el puntero de registro en una tabla, un botón para cerrar un formulario y un botón de ayuda pueden guardarse como clases y agregarse a formularios en cualquier momento que desee que los formularios tengan esta funcionalidad.

Puede exponer las [propiedades](#) y los [métodos](#) en una clase de modo que el usuario pueda integrarlos en el [entorno de datos](#) concreto de un formulario o un conjunto de formularios.

## Proporcionar una apariencia y un uso coherentes

Puede crear clases de [conjunto de formularios](#), de [formulario](#) y de [control](#) con una apariencia característica, de modo que todos los componentes de la aplicación tengan la misma apariencia. Por ejemplo, podría agregar gráficos y patrones de color específicos a una clase de formulario y utilizarla como plantilla para todos los formularios que cree. Podría crear una clase de [cuadro de texto](#) con una apariencia característica, como un efecto de sombreado, y usar esta clase en la aplicación en cualquier momento que desee agregar un cuadro de texto.

## Decidir qué tipo de clase va a crear

Visual FoxPro permite crear distintos tipos de clases, cada uno con sus propias características. Especifique el tipo de clase que desea crear en el [cuadro de diálogo Nueva clase](#) o en la cláusula AS del comando [CREATE CLASS](#).

## Clases de base de Visual FoxPro

En el [Diseñador de clases](#) puede crear subclases para la mayoría de las clases de base de Visual FoxPro.

### Clases de base de Visual FoxPro

<a href="#">ActiveDoc</a>	<a href="#">Custom</a>	<a href="#">Label</a>	<a href="#">PageFrame</a>
<a href="#">CheckBox</a>	<a href="#">EditBox</a>	<a href="#">Line</a>	<a href="#">ProjectHook</a>
<a href="#">Column</a> *	<a href="#">Form</a>	<a href="#">ListBox</a>	<a href="#">Separator</a>
<a href="#">CommandButton</a>	<a href="#">FormSet</a>	<a href="#">OLEBoundControl</a>	<a href="#">Shape</a>
<a href="#">CommandGroup</a>	<a href="#">Grid</a>	<a href="#">OLEContainerControl</a>	<a href="#">Spinner</a>
<a href="#">ComboBox</a>	<a href="#">Header</a> *	<a href="#">OptionButton</a> *	<a href="#">TextBox</a>
<a href="#">Container</a>	<a href="#">Hyperlink Object</a>	<a href="#">OptionGroup</a>	<a href="#">Timer</a>
<a href="#">Control</a>	<a href="#">Image</a>	<a href="#">Page</a> *	<a href="#">ToolBar</a>

\* Estas clases son parte integral de un contenedor primario y no pueden usarse como subclases en el Diseñador de clases.

Todas las clases de base de Visual FoxPro reconocen el siguiente conjunto mínimo de eventos:

Evento	Descripción
<a href="#">Init</a>	Ocurre cuando se crea el objeto.
<a href="#">Destroy</a>	Ocurre cuando el objeto se libera de la memoria.
<a href="#">Error</a>	Ocurre siempre que tiene lugar un error en procedimientos de evento o de método de la clase.

Todas las clases de base de Visual FoxPro tienen el siguiente conjunto mínimo de propiedades:

Propiedad	Descripción
<a href="#">Class</a>	El tipo de clase de que se trata.
<a href="#">BaseClass</a>	La clase de base de la que se deriva, como Form, Commandbutton, Custom, etc.

---

<a href="#">ClassLibrary</a>	La biblioteca de clases en la que está almacenada.
<a href="#">ParentClass</a>	La clase de la que se deriva la clase actual. Si la clase se deriva directamente de una clase de base de Visual FoxPro, la propiedad ParentClass es la misma que la propiedad BaseClass.

---

## Extensión de las clases de base de Visual FoxPro

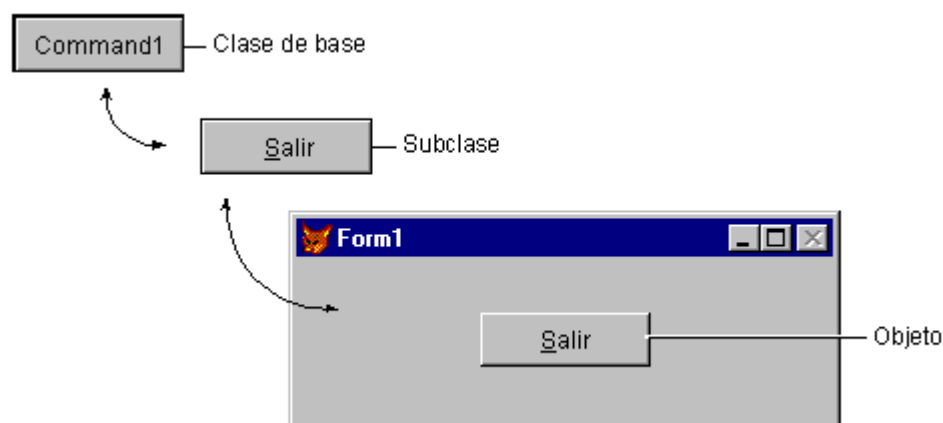
Puede convertir en subclases estas clases para establecer sus propias propiedades de control predeterminadas. Por ejemplo, si quiere que los nombres predeterminados de controles que agregue a formularios de sus aplicaciones reflejen automáticamente sus convenciones de nombres, puede crear clases basadas en las clases de base de Visual FoxPro para hacerlo. Puede crear clases de formulario con una apariencia o un comportamiento personalizado para que sirvan como plantillas para todos los formularios que cree.

También podría convertir en subclases las clases de base de Visual FoxPro para crear controles con funcionalidad encapsulada. Si quiere que un botón libere formularios cuando haga clic en él, puede crear una clase basada en la clase de botón de comando de Visual FoxPro, establecer como título "Salir" e incluir el siguiente comando en el evento Click:

```
THISFORM.Release
```

Puede agregar este nuevo botón a cualquier formulario de la aplicación.

### Botón de comando personalizado agregado a un formulario



## Crear controles con múltiples componentes

Las subclases no están limitadas a clases de base únicas. Puede agregar múltiples controles a una única definición de clase de contenedor. Muchas de las clases de la biblioteca de clases de ejemplo de Visual FoxPro están incluidas en esta categoría. Por ejemplo, la clase VCR de Buttons.vcx, ubicada en la carpeta ...\\Samples\\Vfp98\\Clases de Visual Studio, contiene cuatro [botones de comando](#) para desplazarse por los registros de una tabla.

## Creación de clases no visuales

Una clase basada en la clase personalizada de Visual FoxPro no tiene un elemento visual de tiempo de ejecución. Puede crear [métodos](#) y [propiedades](#) para la clase personalizada en el entorno del [Diseñador de clases](#). Por ejemplo, podría crear una clase personalizada llamada `StrMethods` e incluir en ella una serie de métodos para manipular cadenas de caracteres. Podría agregar esta clase a un formulario con un [cuadro de edición](#) y llamar a los métodos cuando lo necesitara. Si tuviera un método llamado `WordCount`, podría llamarlo cuando lo necesitara:

```
THISFORM.txtCount.Value = ;  
    THISFORM.StrMethods.WordCount ( THISFORM.edtText.Value )
```

Las clases no visuales (como el control personalizado y el control cronómetro) tienen una representación visual, únicamente en [tiempo de diseño](#), en el [Diseñador de formularios](#). Establezca la propiedad de imagen de la clase personalizada como el archivo .bmp que desea mostrar en el Diseñador de formularios cuando se agregue la clase personalizada a un formulario.

## Crear clases

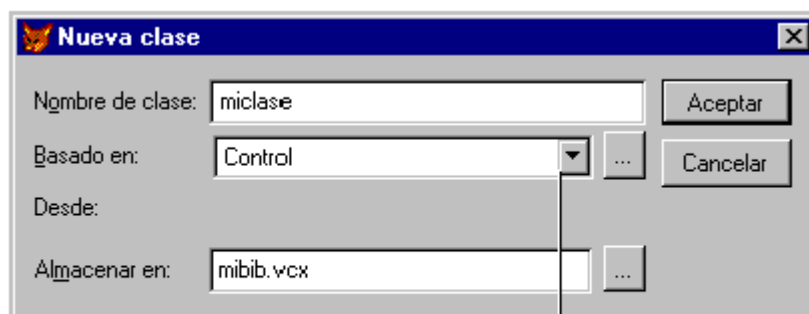
Puede crear nuevas clases en el [Diseñador de clases](#) y puede ver cómo verá el usuario cada objeto a medida que lo diseña.

### Para crear una clase nueva

- En el [Administrador de proyectos](#), seleccione la ficha **Clases** y elija **Nuevo**.  
–O bien–
- En el menú **Archivo**, elija **Nuevo**, seleccione **Clase** y elija **Nuevo archivo**.  
–O bien–
- Utilice el comando [CREATE CLASS](#).

El cuadro de diálogo **Nueva clase** le permite especificar el nombre de la nueva clase, la clase en la que se basa la nueva clase y la biblioteca en la que se almacenará.

### Crear una clase nueva



Elija la clase principal desde esta lista desplegable.

## Modificar una definición de clase

Cuando haya creado una clase, podrá modificarla. Los cambios realizados a una clase afectan a todas las subclases y a todos los objetos basados en esta clase. Puede agregar una mejora a una clase o reparar un error en la clase, y todas las subclases y los objetos basados en dicha clase heredarán el cambio.

Para modificar una clase en el [Administrador de proyectos](#)

1. Seleccione la clase que desea modificar.
2. Elija **Modificar**.

Se abrirá el **Diseñador de clases**.

También puede modificar una definición de clase visual mediante el comando [MODIFY CLASS](#).

**Importante** No cambie la propiedad Name de una clase si la usan otros componentes de la aplicación. De lo contrario, Visual FoxPro no podrá encontrar la clase cuando la necesite.

## Subclases de una definición de clase

Hay dos formas de crear una subclase de una clase definida por el usuario.

Para crear una subclase de una clase definida por el usuario

1. En el cuadro de diálogo [Nueva clase](#), haga clic en el botón de tres puntos situado a la derecha del cuadro **Basada en**.
2. En el cuadro de diálogo **Abrir**, elija la clase en la que desea basar la nueva clase.

–O bien–

- Utilice el comando [CREATE CLASS](#).

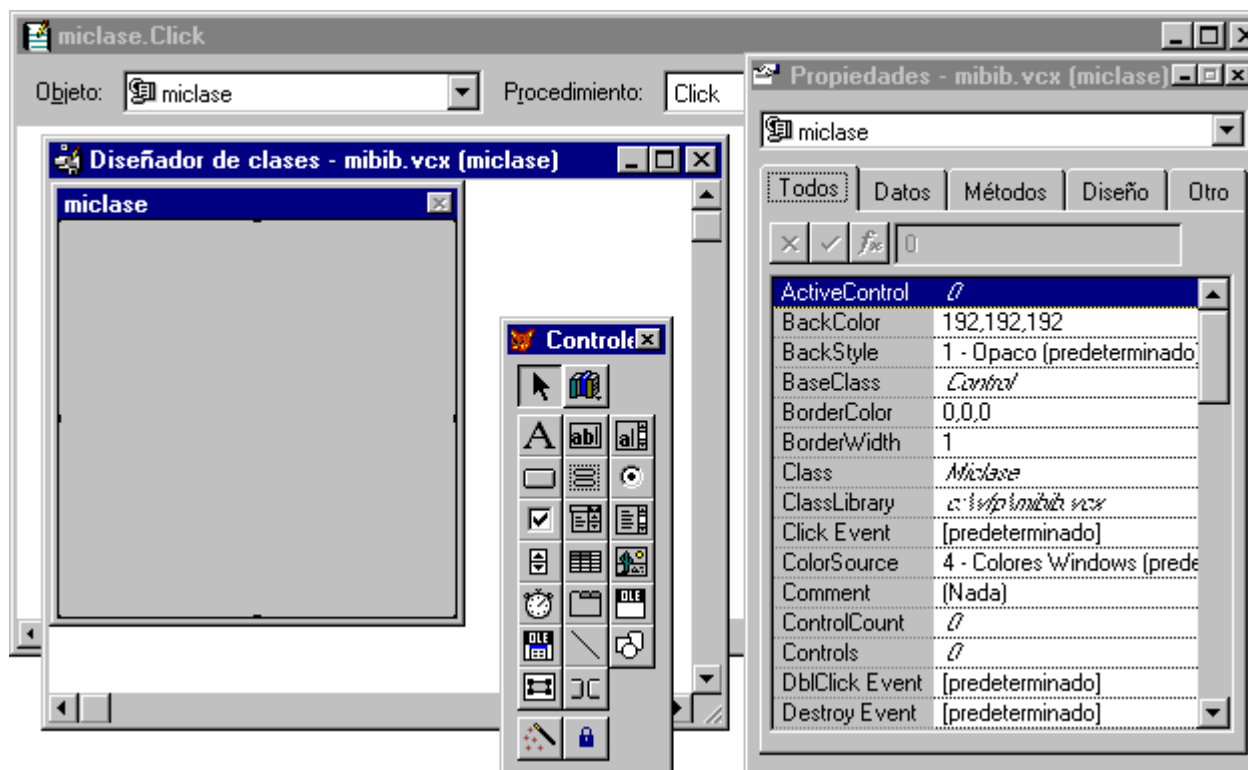
Por ejemplo, para basar una nueva clase, `x`, en `parentclass` de `Mylibrary.vcx`, use el código siguiente:

```
CREATE CLASS x OF y AS parentclass ;  
FROM mylibrary
```

## Utilizar el Diseñador de clases

Cuando especifica la clase en la que está basada la nueva clase y la biblioteca en la que se va a almacenar, se abre el Diseñador de clases.

**Diseñador de clases**



El [Diseñador de clases](#) proporciona la misma interfaz que el [Diseñador de formularios](#), que le permite ver y modificar las propiedades de la clase en la [ventana Propiedades](#). La ventana de edición de código le permite escribir código para que se ejecute cuando ocurran [eventos](#) o se llame a [métodos](#).

### Agregar objetos a una clase de control o a una clase de contenedor

Si basa la nueva clase en una clase de control o en una clase de contenedor, podrá agregarle controles del mismo modo que en el Diseñador de formularios: elija el botón del control en la [barra de herramientas Controles de formularios](#) y arrastre para ajustar el tamaño en el Diseñador de clases.

Independientemente del tipo de clase en el que base la nueva clase, puede establecer propiedades y escribir código de método. También podrá crear nuevas [propiedades](#) y [métodos](#) para la clase.

### Agregar propiedades y métodos a una clase

Puede agregar tantas [propiedades](#) y [métodos](#) nuevos a la nueva clase como desee. Las propiedades contienen valores, mientras que los métodos contienen código de procedimiento que se ejecutará cuando llame al método.

#### Crear propiedades y métodos nuevos

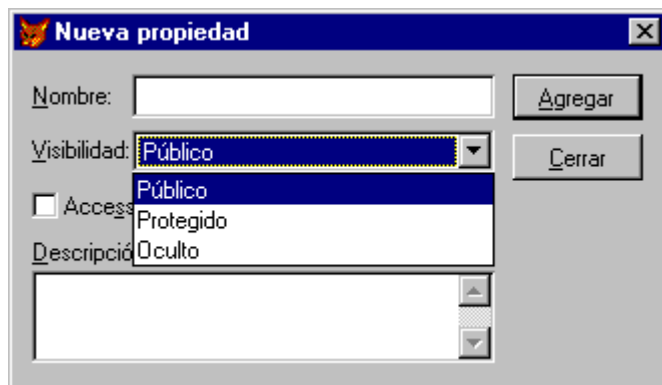
Cuando crea propiedades y métodos nuevos para clases, las propiedades y los métodos tienen como alcance la clase, no los componentes individuales de la misma.

#### Para agregar una propiedad nueva a una clase

1. En el menú **Clase**, elija **Nueva propiedad**.
2. En el cuadro de diálogo [Nueva propiedad](#), escriba el nombre de la propiedad.
3. Especifique la visibilidad: **Public**, **Protected** o **Hidden**.

Puede tener acceso a una propiedad Public desde cualquier lugar de la aplicación. Las propiedades Protected y Hidden se tratan en "[Proteger y ocultar miembros de clase](#)" más adelante en este mismo capítulo.

### Cuadro de diálogo Nueva propiedad



4. Elija **Agregar**.

También puede incluir una descripción de la propiedad que aparecerá en la parte inferior de la ventana [Propiedades](#) en el **Diseñador de clases** y en el **Diseñador de formularios** cuando se agregue el control a un formulario.

**Solución de problemas** Cuando agregue una propiedad a una clase que un usuario de la clase pueda establecer, el usuario puede introducir un valor incorrecto para la propiedad que cause errores en tiempo de ejecución. Tiene que documentar de forma explícita los valores válidos de la propiedad. Si la propiedad puede establecerse como 0, 1 ó 2, por ejemplo, indíquelo en el cuadro **Descripción** del [cuadro de diálogo Nueva propiedad](#). También es conveniente comprobar el valor de la propiedad en código que haga referencia a ella.

### Para crear una propiedad de matriz

- En el cuadro **Nombre** del cuadro de diálogo [Nueva propiedad](#), especifique el nombre, el tamaño y las dimensiones de la matriz.

Por ejemplo, para crear una propiedad de matriz llamada `mimatriz` con diez filas y dos columnas, escriba lo siguiente en el cuadro Nombre:

```
mimatriz[10,2]
```

La propiedad de matriz es de sólo lectura en [tiempo de diseño](#) y se muestra en cursiva en la ventana [Propiedades](#). Se puede administrar y redimensionar en [tiempo de ejecución](#). Para ver un ejemplo del

uso de una propiedad de matriz, consulte "Administrar varias instancias de un formulario" en el capítulo 9, [Crear formularios](#).

### Para agregar un método nuevo a una clase

1. En el menú **Clase**, elija **Nuevo método**.
2. En el cuadro de diálogo [Nuevo método](#), escriba el nombre del método.
3. Especifique la visibilidad: **Public**, **Protected** o **Hidden**.
4. Seleccione la casilla de verificación Access para crear un método de Access, seleccione la casilla de verificación para crear un método Assign o seleccione ambas casillas de verificación para crear métodos Access y Assign.

Los métodos Access y Assign le permiten ejecutar código cuando se consulta el valor de una propiedad o cuando se intenta cambiar su valor.

El código de un método Access se ejecuta cuando se consulta el valor de una propiedad, generalmente al utilizar la propiedad en una referencia de objeto, al almacenar el valor de una propiedad en una variable o al mostrar el valor de la propiedad con un signo de interrogación (?).

El código de un método Assign se ejecuta cuando intenta modificar el valor de una propiedad, generalmente mediante los comandos STORE o = para asignar un nuevo valor a la propiedad.

Para obtener más información acerca de los métodos Access y Assign, consulte [Métodos Access y Assign](#).

También puede incluir una descripción del método.

### Proteger y ocultar miembros de clases

Las [propiedades](#) y [métodos](#) de una definición de clase son **Public** de forma predeterminada: el código de otras clases u otros procedimientos puede establecer las propiedades o llamar a los métodos. A las propiedades y los métodos definidos como **Protected** sólo pueden tener acceso otros métodos de la definición de la clase o de subclases de la clase. A las propiedades y los métodos definidos como **Hidden** sólo pueden tener acceso otros miembros de la definición de la clase. Las subclases de la clase no pueden "ver" o hacer referencia a miembros ocultos.

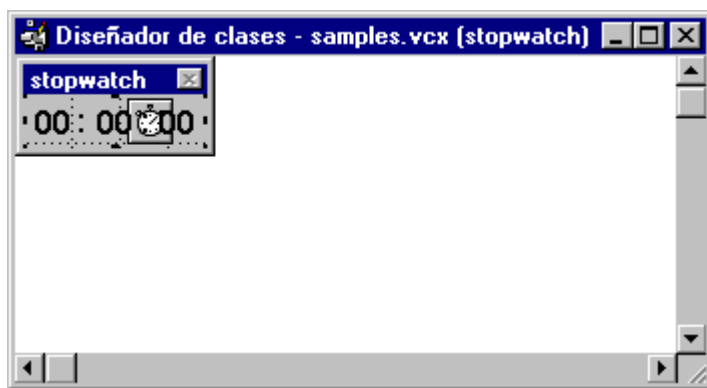
Para asegurar un correcto funcionamiento en algunas clases, deberá impedir que los usuarios cambien las propiedades o llamen al método desde fuera de la clase mediante programación.

El ejemplo siguiente ilustra el uso de propiedades y métodos protegidos de una clase

La clase Stopwatch incluida en Samples.vcx, en el directorio ...\\Samples\\Vfp98\\Classes de Visual Studio, incluye un [cronómetro](#) y cinco [etiquetas](#) que muestran el tiempo transcurrido:

#### La clase Stopwatch de Samples.vcx





La clase Stopwatch contiene etiquetas y un cronómetro.

### Valores de las propiedades de la clase Stopwatch

Control	Propiedad	Valor
lblSeconds	<a href="#">Caption</a>	00
lblColon1	Caption	:
lblMinutes	Caption	00
lblColon2	Caption	:
lblHours	Caption	00
tmrSWatch	<a href="#">Interval</a>	1000

Esta clase tiene también tres propiedades protegidas, nSec, nMin y nHour, así como un método protegido, UpdateDisplay. Los otros tres métodos personalizados de la clase, Start, Stop y Reset no están protegidos.

**Sugerencia** Elija **Información de clase** en el menú **Clase** para ver la visibilidad de todas las propiedades y métodos de una clase.

Las propiedades protegidas se utilizan en cálculos internos en el método UpdateDisplay y el evento Timer. El método UpdateDisplay establece los títulos de las etiquetas para que reflejen el tiempo transcurrido.

### Método UpdateDisplay

Código	Comentarios
<pre>CSecDisplay = ALLTRIM(STR(THIS.nSec)) cMinDisplay = ALLTRIM(STR(THIS.nMin)) cHourDisplay = ALLTRIM(STR(THIS.nHour))</pre>	Convierte las propiedades numéricas al tipo Character para mostrarlas en los títulos de etiqueta.
<pre>THIS.lblSeconds.Caption = ;     STR(THIS.nSec, 2, 0) + "</pre>	Establece los títulos de etiqueta,

```

    IIF(THIS.nSec < 10, ;
    "0", "") + cSecDisplay
THIS.lblMinutes.Caption = ;
    IIF(THIS.nMin < 10, ;
    "0", "") + cMinDisplay
THIS.lblHours.Caption = ;
    IIF(THIS.nHour < 10, ;
    "0", "") + cHourDisplay

```

conservando los 0 iniciales si el valor de la propiedad numérica es menor que 10.

La tabla siguiente muestra el código del evento `tmrSWatch.Timer`.

### Evento Timer

Código	Comentarios
<pre> THIS.Parent.nSec = THIS.Parent.nSec + 1 IF THIS.Parent.nSec = 60     THIS.Parent.nSec = 0     THIS.Parent.nMin = ;     THIS.Parent.nMin + 1 ENDIF </pre>	<p>Incrementa el valor de la propiedad <code>nSec</code> cada vez que se desencadena el evento de cronómetro cada segundo.</p> <p>Si <code>nSec</code> ha llegado a 60, lo restablece a 0 e incrementa la propiedad <code>nMin</code>.</p>
<pre> IF THIS.Parent.nMin = 60     THIS.Parent.nMin = 0     THIS.Parent.nHour = ;     THIS.Parent.nHour + 1 ENDIF THIS.Parent.UpdateDisplay </pre>	<p>Si <code>nMin</code> ha llegado a 60, lo restablece a 0 e incrementa la propiedad <code>nHour</code>.</p> <p>Llama al método <code>UpdateDisplay</code> cuando se establecen los nuevos valores de la propiedad.</p>

La clase `Stopwatch` tiene tres métodos que no están protegidos: `Start`, `Stop` y `Reset`. Un usuario debe poder llamar directamente a estos métodos para controlar el cronómetro.

El método `Start` contiene la línea de código siguiente:

```
THIS.tmrSWatch.Enabled = .T.
```

El método `Stop` contiene la línea de código siguiente:

```
THIS.tmrSWatch.Enabled = .F.
```

El método `Reset` establece las propiedades protegidas a 0 y llama al método protegido:

```

THIS.nSec = 0
THIS.nMin = 0
THIS.nHour = 0
THIS.UpdateDisplay

```

El usuario no puede establecer directamente estas propiedades o llamar a este método, pero el código del método `Reset` sí puede hacerlo.

### Especificar el valor predeterminado para una propiedad

Al crear una nueva propiedad, su valor predeterminado es falso (.F.). Para especificar un valor predeterminado distinto para una propiedad, utilice la [ventana Propiedades](#). En la ficha **Otras**, haga clic en la propiedad y establezca el valor deseado. Este será el valor inicial de la propiedad cuando se agregue la clase a un [formulario](#) o a un [conjunto de formularios](#).

También puede establecer cualquiera de las propiedades de clase de base en el [Diseñador de clases](#). Cuando un objeto basado en la clase se agregue al formulario, reflejará el valor de su propiedad en lugar del valor de la propiedad de la clase de base de Visual FoxPro.

**Sugerencia** Si desea convertir el valor predeterminado de una propiedad en una cadena vacía, seleccione el valor en el cuadro **Edición de propiedades** y presione la tecla RETROCESO.

## Especificar la apariencia en tiempo de diseño

Puede especificar el icono de barra de herramientas y el de contenedor para su clase en el cuadro de diálogo [Información de clase](#).

### Para establecer un icono de barra de herramientas para una clase

1. En el [Diseñador de clases](#), elija **Información de clase** en el menú **Clase**.
2. En el cuadro de diálogo [Información de clase](#), escriba el nombre y la ruta de acceso del archivo .BMP en el cuadro **Icono de la barra de herramientas**.

**Sugerencia** El archivo de mapa de bits (archivo .bmp) para el icono de la barra de herramientas debe tener 15 por 16 [píxeles](#). Si la imagen es mayor o menor, se ajustará a 15 por 16 píxeles y posiblemente no tendrá la apariencia deseada.

El icono de barra de herramientas especificado se muestra en la [barra de herramientas Controles de formularios](#) cuando se llena la barra de herramientas con las clases de la [biblioteca de clases](#).

También puede especificar que se muestre el icono para la clase en el [Administrador de proyectos](#) y el [Examinador de clases](#) si establece el icono contenedor.

### Para establecer un icono contenedor para una clase

1. En el [Diseñador de clases](#), elija **Información de clase** en el menú **Clase**.
2. En el cuadro **Icono de contenedor**, escriba el nombre y la ruta de acceso del archivo .bmp que se va a mostrar en el botón de la [barra de herramientas Controles de formularios](#).

## Usar archivos de bibliotecas de clases

Todas las clases diseñadas visualmente se almacenan en una biblioteca de clases con la extensión de archivo .vcx.

### Crear una biblioteca de clases

Una biblioteca de clases puede crearse de una de estas tres formas.

### Para crear una biblioteca de clases

- Cuando cree una clase, especifique un nuevo archivo de biblioteca de clases en el cuadro **Almacenar en** del cuadro de diálogo [Nueva clase](#).

–O bien–

- Utilice el comando [CREATE CLASS](#), especificando el nombre de la nueva biblioteca de clases.

Por ejemplo, la instrucción siguiente crea una nueva clase llamada `miclase` y una nueva biblioteca de clases llamada `nue_bib`:

```
CREATE CLASS miclase OF nue_bib AS CUSTOM
```

–O bien–

- Utilice el comando [CREATE CLASSLIB](#).

Por ejemplo, escriba el comando siguiente en la [ventana Comandos](#) para crear una biblioteca de clases llamada `nue_bib`:

```
CREATE CLASSLIB nue_bib
```

### Copiar y quitar clases de bibliotecas de clases

Cuando agregue una biblioteca de clases a un proyecto, podrá copiar clases de una biblioteca a otra con facilidad o, simplemente, quitar clases de las bibliotecas.

### Para copiar una clase de una biblioteca a otra

1. Asegúrese de que ambas bibliotecas están en un proyecto (no necesariamente en el mismo).
2. En el [Administrador de proyectos](#), seleccione la ficha **Clases**.
3. Haga clic en el signo más (+) situado a la izquierda de la biblioteca de clases en la que se encuentra ahora la clase.
4. Arrastre la clase desde la biblioteca original y colóquela en la nueva.

**Sugerencia** Es conveniente guardar en una biblioteca de clases una clase y todas las subclases basadas en ella. Si tiene una clase que contiene elementos de muchas bibliotecas de clases distintas, estas bibliotecas deberán estar abiertas, por lo que se tardará un poco más en cargar inicialmente la clase en [tiempo de ejecución](#) y en [tiempo de diseño](#).

### Para quitar una clase de una biblioteca

- Seleccione la clase en el [Administrador de proyectos](#) y elija **Quitar**.

–O bien–

- Utilice el comando [REMOVE CLASS](#).

Puede utilizar el comando [RENAME CLASS](#) para cambiar el nombre de una clase de una biblioteca de clases. Sin embargo, recuerde que cuando cambia el nombre de una clase, los formularios que contienen la clase y las subclases en otros archivos .vcx siguen haciendo referencia al nombre antiguo y no volverán a funcionar correctamente.

Visual FoxPro incluye un Examinador de clases para facilitar el uso y la administración de clases y bibliotecas de clases. Para obtener más información, vea la ventana [Examinador de clases](#).

## Agregar clases a formularios

Puede arrastrar una clase desde el [Administrador de proyectos](#) hasta el [Diseñador de formularios](#) o hasta el [Diseñador de clases](#). También puede registrar las clases de modo que puedan mostrarse directamente en la [barra de herramientas Controles de formularios](#) del Diseñador de formularios o el Diseñador de clases y agregarse a contenedores de la misma forma que los controles estándar.

### Para registrar una biblioteca de clases

1. En el menú **Herramientas**, elija **Opciones**.
2. En el cuadro de diálogo **Opciones**, elija la ficha [Controles](#).
3. Seleccione **Bibliotecas de clases visuales** y elija **Agregar**.
4. En el cuadro de diálogo **Abrir**, elija una biblioteca de clases para agregar el registro y, a continuación, elija **Abrir**.
5. Elija **Establecer como predeterminado** si desea que la biblioteca de clases esté disponible en la barra de herramientas Controles de formularios en sesiones futuras de Visual FoxPro.

También puede agregar la biblioteca de clases a la barra de herramientas Controles de formularios si elige **Agregar** en el submenú del botón **Ver clases**. Para que estas clases estén disponibles en la barra de herramientas Controles de formularios en sesiones futuras de Visual FoxPro, tendrán que establecer el valor predeterminado en el [cuadro de diálogo Opciones](#).

## Anular valores predeterminados de propiedades

Al agregar a un [formulario](#) objetos basados en una clase definida por el usuario, puede cambiar el valor de todas las [propiedades](#) de la clase que no estén protegidas, anulando los valores predeterminados. Si posteriormente cambia las propiedades de clase en el [Diseñador de clases](#), no se verá afectada la configuración del objeto del formulario. Si no ha cambiado el valor de una propiedad del formulario y cambia el de la clase, el cambio también surtirá efecto en el objeto.

Por ejemplo, un usuario puede agregar a un formulario un objeto basado en su clase y cambiar la propiedad BackColor de blanco a rojo. Si cambia a verde la propiedad BackColor de la clase, el objeto del formulario del usuario seguirá teniendo un valor rojo para BackColor. Por otra parte, si el usuario no cambia la propiedad BackColor del objeto y usted cambia a verde el color de fondo de la clase, la propiedad BackColor del objeto del formulario heredará el cambio y también será verde.

## Llamar al código de métodos de clase primaria

Un objeto o una clase que se basa en otra clase hereda automáticamente la funcionalidad de la clase original. Sin embargo, puede anular fácilmente el código de métodos heredado. Por ejemplo, puede escribir nuevo código para el [evento Click](#) de una clase después de haberla convertido en subclase o después de agregar al contenedor un objeto basado en la clase. En ambos casos, el nuevo código se ejecuta en [tiempo de ejecución](#); el código original no se ejecuta.

Sin embargo, es más frecuente que quiera agregar funcionalidad a la nueva clase u objeto conservando la funcionalidad original. De hecho, una de las decisiones clave que tiene que hacer en la programación orientada a objetos es qué funcionalidad va a incluir a nivel de clase, a nivel de subclase y a nivel de objeto. Puede optimizar el diseño de la clase con la función [DODEFAULT\(\)](#) o el [operador de resolución de alcance](#) (::) para agregar código a distintos niveles de la jerarquía del contenedor o de la clase.

### Agregar funcionalidad a subclases

Puede llamar al código de la clase primaria desde una subclase mediante la función [DODEFAULT\(\)](#).

Por ejemplo, `cmdOk` es una clase de botón de comando almacenada en `Buttons.vcx`, ubicada en el directorio `...\Samples\Vfp98\Classes de Visual Studio`. El código asociado al evento `Click` de `cmdOk` libera el formulario que contiene el botón. `cmdCancel` es una subclase de `cmdOk` de la misma biblioteca de clases. Para agregar funcionalidad a `cmdCancel` para descartar cambios, por ejemplo, puede agregar el código siguiente al evento `Click`:

```
IF USED( ) AND CURSORGETPROP("Buffering") != 1
    TABLEREVERT(.T.)
ENDIF
DODEFAULT( )
```

Como los cambios se escriben en una tabla almacenada en búfer de forma predeterminada cuando se cierra la tabla, no tiene que agregar código [TABLEUPDATE\(\)](#) a `cmdOk`. El código adicional de `cmdCancel` deshace los cambios realizados a la tabla antes de llamar al código de `cmdOk`, la clase primaria, para liberar el formulario.

## Jerarquías de clases y de contenedores

Las jerarquías de clases y de contenedores son dos entidades distintas. Visual FoxPro busca código de evento en la jerarquía de clases, mientras que se hace referencia a los objetos en la jerarquía de contenedores. La siguiente sección, "Referencias a objetos en la jerarquía de contenedores", trata la jerarquía de contenedores. Más adelante en este capítulo se explican las jerarquías de clases en la sección [Llamar a código de evento en la jerarquía de clases](#).

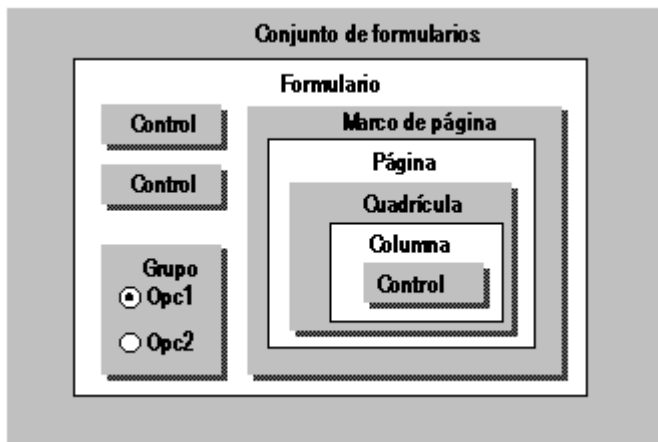
## Referencias a objetos en la jerarquía de contenedores

Para manipular un objeto, hay que identificarlo en relación a la jerarquía de contenedores. Por ejemplo, para manipular un [control](#) de un [formulario](#) perteneciente a un [conjunto de formularios](#), deberá hacer referencia al conjunto de formularios, al formulario y, por último, al control.

Hacer referencia a un objeto dentro de su jerarquía de contenedores se puede comparar con dar una dirección del objeto a Visual FoxPro. Cuando describe la ubicación de una casa a otra persona fuera de su marco inmediato de referencia, debe indicar el país, la provincia o la región, la ciudad, la calle o bien sólo el número de la calle donde se encuentra la vivienda, según lo lejos que se encuentre esa otra persona. De lo contrario, podría haber cierta confusión.

La ilustración siguiente muestra una posible situación de anidamiento del contenedor.

### Contenedores anidados



Para desactivar el control de la columna de cuadrícula, deberá proporcionar la dirección siguiente:

```
Formset.Form.PageFrame.Page. ;  
    Grid.Column.Control.Enabled = .F.
```

La [propiedad ActiveForm](#) del objeto aplicación (\_VFP) le permite manipular el formulario activo aunque no conozca su nombre. Por ejemplo, la siguiente línea de código cambia el color de fondo del formulario activo, independientemente del conjunto de formularios al que pertenezca:

```
_VFP.ActiveForm.BackColor = RGB(255,255,255)
```

De forma similar, la [propiedad ActiveControl](#) permite manipular el control activo del formulario activo. Por ejemplo, la expresión siguiente introducida en la [ventana Inspección](#) muestra el nombre del control activo de un formulario a medida que se eligen interactivamente los distintos controles:

```
_VFP.ActiveForm.ActiveControl.Name
```

### Referencias relativas

Cuando haga referencia a objetos desde la jerarquía de contenedores (por ejemplo, en el evento Click de un botón de comando de un formulario perteneciente a un conjunto de formularios), puede utilizar algunos métodos abreviados para identificar el objeto que desea manipular. La tabla siguiente indica las propiedades o las palabras clave que facilitan la referencia a un objeto desde la jerarquía del objeto:

Propiedad o palabra clave	Referencia
<a href="#">Parent</a>	El contenedor más inmediato del objeto
<a href="#">THIS</a>	El objeto
<a href="#">THISFORM</a>	El formulario que contiene el objeto
<a href="#">THISFORMSET</a>	El conjunto de formularios que contiene el objeto

**Nota** Sólo puede utilizar THIS, THISFORM y THISFORMSET en código de métodos y eventos.

La tabla siguiente proporciona ejemplos del uso de THISFORMSET, THISFORM, THIS y Parent para establecer propiedades de objetos:

Comando	Dónde incluir el comando
<code>THISFORMSET.frm1.cmd1.Caption = "Aceptar"</code>	En el código de evento o de método de cualquier control de cualquier formulario del conjunto de formularios.
<code>THISFORM.cmd1.Caption = "Aceptar"</code>	En el código de evento o de método de cualquier control del mismo formulario en el que está <code>cmd1</code> .
<code>THIS.Caption = "Aceptar"</code>	En el código de evento o de método del control cuyo título desee cambiar.
<code>THIS.Parent.BackColor = RGB(192,0,0)</code>	En el código de evento o de método de un control de un formulario. El comando cambia a rojo oscuro el color de fondo del formulario.

## Establecer propiedades

Las propiedades de un objeto pueden establecerse en [tiempo de ejecución](#) o en [tiempo de diseño](#).

### Para establecer una propiedad

- Utilice esta sintaxis:

*Contenedor.Objeto.Propiedad = Valor*



Por ejemplo, las instrucciones siguientes establecen varias propiedades de un [cuadro de texto](#) llamado txtDate en un formulario llamado frmPhoneLog:

```
frmPhoneLog.txtDate.Value = DATE( ) && Muestra la fecha actual
frmPhoneLog.txtDate.Enabled = .T. && El control está activado
frmPhoneLog.txtDate.ForeColor = RGB(0,0,0) && texto en negro
frmPhoneLog.txtDate.BackColor = RGB(192,192,192) && fondo en gris
```

Para la configuración de propiedades de los ejemplos anteriores, frmPhoneLog es el objeto contenedor de mayor nivel. Si frmPhoneLog estuviera incluido en un [conjunto de formularios](#), también debería incluir el conjunto de formularios en la ruta de acceso primaria:

```
frsContacts.frmPhoneLog.txtDate.Value = DATE( )
```

## Establecer múltiples propiedades

La estructura WITH ... ENDWITH simplifica el establecimiento de múltiples propiedades. Por ejemplo, para establecer múltiples propiedades de una columna en una [cuadrícula](#) de un [formulario](#) perteneciente a un [conjunto de formularios](#), podría utilizar la sintaxis siguiente:

```
WITH THISFORMSET.frmForm1.grdGrid1.grcColumn1
.Width = 5
.Resizable = .F.
.ForeColor = RGB(0,0,0)
.BackColor = RGB(255,255,255)
.SelectOnEntry = .T.
ENDWITH
```

## Llamar a métodos

Una vez creado un objeto, puede llamar a los [métodos](#) de ese objeto desde cualquier lugar de la aplicación.

### Para llamar a un método

- Utilice esta sintaxis:

*Primario.Objeto.Método*

Las instrucciones siguientes llaman a métodos para mostrar un [formulario](#) y establecer el enfoque en un [cuadro de texto](#):

```
frsFormSet.frmForm1.Show
frsFormSet.frmForm1.txtGetText1.SetFocus
```

Los métodos que devuelven valores y se utilizan en [expresiones](#) deben terminar en paréntesis de apertura y de cierre. Por ejemplo, la instrucción siguiente establece el [título](#) de un [formulario](#) como el valor devuelto por el método definido por el usuario GetNewCaption:

```
Form1.Caption = Form1.GetNewCaption( )
```

**Nota** Los [parámetros](#) transferidos a métodos deben incluirse entre paréntesis después del nombre del método; por ejemplo, `Form1.Show(nStyle)`. transfiere `nStyle` al código del método `Show` de `Form1`.

## Responder a eventos

El código incluido en un procedimiento de evento se ejecuta cuando se produce el [evento](#). Por ejemplo, el código incluido en el procedimiento de evento `Click` de un [botón de comando](#) se ejecutará cuando el usuario haga clic en el botón de comando.

Puede activar los eventos [Click](#), [DblClick](#), [MouseMove](#) y [DragDrop](#) con el evento [MOUSE](#) o usar el comando [ERROR](#) para generar eventos `Error` y el comando [KEYBOARD](#) para generar eventos `KeyPress`. No puede hacer que se produzca ningún otro evento mediante programación, pero sí puede llamar al procedimiento asociado con el evento. Por ejemplo, la instrucción siguiente hace que se ejecute el código del [evento Activate](#) de `frmPhoneLog`, pero no activa el formulario:

```
frmPhoneLog.Activate
```

Si desea activar el formulario, utilice el [método Show](#) del formulario. Al llamar al método `Show` se mostrará y activará el formulario, momento en el que también se ejecutará el código del evento `Activate`:

```
frmPhoneLog.Show
```

## Definir clases mediante programación

Las clases se pueden definir visualmente en el [Diseñador de clases](#) y el [Diseñador de formularios](#) o mediante programación en archivos `.PRG`. En esta sección se explica cómo escribir definiciones de clase. Para obtener información sobre comandos, funciones y operadores específicos, vea la Ayuda. Para obtener más información sobre formularios, consulte el capítulo 9, [Crear formularios](#)

En un archivo de programa es posible tener código de programa delante de las definiciones de clase, pero no después de ellas, del mismo modo que el código de programa no puede ir después de los procedimientos de un programa. El intérprete de comandos básico para la creación de clases tiene la sintaxis siguiente:

```
DEFINE CLASS NombreClase1 AS ClasePrimaria [OLEPUBLIC]
[[PROTECTED | HIDDEN NombrePropiedad1, NombrePropiedad2 ...]
[Object.]NombrePropiedad = eExpresión ...]
[ADD OBJECT [PROTECTED] NombreObjeto AS NombreClase2 [NOINIT]
[WITH cListaPropiedades]]...
[[PROTECTED | HIDDEN] FUNCTION | PROCEDURE Nombre[_ACCESS | _ASSIGN]
[NODEFAULT]
cInstrucciones
[ENDFUNC | ENDPROC]]...
ENDDEFINE
```

## Proteger y ocultar miembros de clase

Puede proteger u ocultar [propiedades](#) y [métodos](#) de una definición de clase con las palabras clave PROTECTED y HIDDEN del comando [DEFINE CLASS](#).

Por ejemplo, si crea una clase para almacenar información sobre empleados y no desea que los usuarios puedan modificar la fecha de contratación, puede proteger la propiedad FechaContr. Si los usuarios necesitan averiguar cuándo se contrató a un empleado determinado, podrá incluir un método para devolver la fecha de contratación.

```
DEFINE CLASS empleado AS CUSTOM
  PROTECTED FechaContr
    Nombre = ""
    Apellido = ""
    Dirección = ""
    FechaContr = { - - }

  PROCEDURE ObtFechaContr
    RETURN This.FechaContr
  ENDPROC
ENDDDEFINE
```

## Crear objetos a partir de clases

Cuando haya guardado una clase visual, puede crear un objeto basado en ella mediante la función [CREATEOBJECT\(\)](#). El ejemplo siguiente muestra la ejecución de un formulario guardado como una definición de clase en el archivo de biblioteca de clases Forms.vcx:

### Crear y mostrar un objeto Form cuya clase se diseñó en el Diseñador de formularios

Código	Comentarios
<a href="#">SET CLASSLIB</a> TO Forms ADDITIVE	Establece como biblioteca de clases el archivo .vcx en el que se guardó la definición del formulario. La palabra clave ADDITIVE impide que este comando cierre otras bibliotecas de clases que estuvieran abiertas.
frmTest = <a href="#">CREATEOBJECT</a> ("FormPrueba")	Este código supone que el nombre de la clase de formulario guardada en la biblioteca de clases es FormPrueba.
frmTest.Show	Muestra el formulario.

## Agregar objetos a una clase contenedor

Puede utilizar la cláusula ADD OBJECT en el comando [DEFINE CLASS](#) o en el [método AddObject](#) para agregar objetos a un contenedor.

Por ejemplo, la siguiente definición de clase se basa en un formulario. El comando ADD OBJECT agrega dos botones de comando al formulario:

```
DEFINE CLASS miForm AS FORM
    ADD OBJECT cmdOK AS COMMANDBUTTON
    ADD OBJECT PROTECTED cmdCancel AS COMMANDBUTTON
ENDDDEFINE
```

Utilice el método AddObject para agregar objetos a un contenedor después de crear el objeto contenedor. Por ejemplo, las líneas de código siguientes crean un objeto formulario y le agregan dos botones de comando:

```
frmMessage = CREATEOBJECT("FORM")
frmMessage.AddObject("txt1", "TEXTBOX")
frmMessage.AddObject("txt2", "TEXTBOX")
```

También puede utilizar el método AddObject en el código de método de una clase. Por ejemplo, la definición de clase siguiente utiliza AddObject en el código asociado al [evento Init](#) para agregar un [control](#) a una columna de cuadrícula.

```
DEFINE CLASS micuad AS GRID
    ColumnCount = 3
    PROCEDURE Init
        THIS.Column2.AddObject("cboCliente", "COMBOBOX")
        THIS.Column2.CurrentControl = "cboCliente"
    ENDPROC
ENDDDEFINE
```

## Agregar y crear clases en código de métodos

Puede agregar [objetos](#) a un contenedor mediante programación con el método AddObject. También puede crear objetos con la función [CREATEOBJECT\(\)](#) en los métodos Load, Init o en cualquier otro método de la clase.

Cuando agregue un objeto con el método AddObject, el objeto se convierte en un miembro del contenedor. La [propiedad Parent](#) del objeto agregado se refiere al contenedor. Cuando un objeto basado en el contenedor o en la clase del control se libera de la memoria, también se libera el objeto agregado.

Cuando crea un objeto con la función CREATEOBJECT( ), el objeto está en el alcance de una propiedad de la clase o [variable](#) del método que llama a esta función. La propiedad primaria del objeto no está definida.

## Asignar código de método y código de evento

Además de escribir código para los [métodos](#) y [eventos](#) de un [objeto](#), puede ampliar el conjunto de métodos en las subclases de clases de base de Visual FoxPro. Estas son las reglas para escribir código de evento y métodos:

- El conjunto de eventos para las clases de base de Visual FoxPro es limitado y no puede ampliarse.

- Todas las clases reconocen un conjunto limitado de eventos predeterminados, que incluye como mínimo los eventos [Init](#), [Destroy](#) y [Error](#).
- Al crear en una definición de clase un [método](#) con el mismo nombre que un [evento](#) reconocible por la [clase](#), el código del método se ejecutará cuando se produzca el evento.
- Puede agregar métodos a las clases mediante la creación de un [procedimiento](#) o una [función](#) en la definición de clase.
- Puede crear [métodos Access y Assign](#) para sus clases si crea un [procedimiento](#) o una [función](#) con el mismo nombre que una propiedad de clase y anexa `_ACCESS` o `_ASSIGN` al nombre de procedimiento o de función.

## Llamar al código de evento en la jerarquía de clases

Al crear una [clase](#), ésta hereda automáticamente todas las [propiedades](#), los [métodos](#) y los [eventos](#) de la clase primaria. Si se escribe código para un evento en la clase primaria, ese código se ejecutará cuando se produzca el evento con respecto a un objeto basado en la subclase. Sin embargo, podrá sobrescribir el código de la clase primaria escribiendo código para el evento en la subclase.

Para llamar explícitamente al código de evento en una clase primaria cuando la subclase tiene código escrito para el mismo evento, utilice la función [DODEFAULT\(\)](#).

Por ejemplo, podría tener una clase llamada `cmdBottom` basada en la clase de base del botón de comando que tuviera el código siguiente en el [evento Click](#):

```
GO BOTTOM  
THISFORM.Refresh
```

Al agregar un [objeto](#) basado en esta clase a un [formulario](#) llamado, por ejemplo, `cmdInferior1`, podría decidir que también desea mostrar un mensaje para informar al usuario de que el puntero de registro está en la parte inferior de la tabla. Podría agregar el código siguiente al evento Click del objeto para mostrar el mensaje:

```
WAIT WINDOW "En la parte inferior de la tabla" TIMEOUT 1
```

Sin embargo, al ejecutar el formulario se muestra el mensaje, pero el puntero de registro no se mueve porque nunca se ejecuta el código del evento Click de la clase primaria. Para asegurarse de que también se ejecuta el código del evento Click de la clase primaria, incluya las siguientes líneas de código en el procedimiento del evento Click del objeto:

```
DODEFAULT( )  
WAIT WINDOW "En la parte inferior de la tabla" TIMEOUT 1
```

**Nota** Puede utilizar la función [AClass\(\)](#) para determinar todas las clases de la jerarquía de clases de un objeto.

## Impedir la ejecución del código de clase de base

En algunos casos, deseará evitar que produzca el comportamiento predeterminado de la clase de base en un [evento](#) o [método](#). Para ello, incluya la palabra clave `NODEFAULT` en el código de método que escriba. Por ejemplo, el programa siguiente utiliza la palabra clave `NODEFAULT` en el [evento](#)

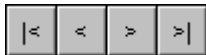
[KeyPress](#) de un [cuadro de texto](#) para impedir que se muestren en el cuadro los caracteres escritos:

```
frmKeyExample = CREATEOBJECT("prueba")
frmKeyExample.Show
READ EVENTS
DEFINE CLASS prueba AS FORM
    ADD OBJECT text01 AS TEXTBOX
    PROCEDURE text01.KeyPress
        PARAMETERS nKeyCode, nShiftAltCtrl
        NODEFAULT
        IF BETWEEN(nKeyCode, 65, 122) && entre 'A' y 'z'
            This.Value = ALLTRIM(This.Value) + "*"
            ACTIVATE SCREEN      && enviar el resultado a la ventana principal de Visua
            ?? CHR(nKeyCode)
        ENDIF
    ENDPROC
    PROCEDURE Destroy
        CLEAR EVENTS
    ENDPROC
ENDDEFINE
```

## Crear un conjunto de botones de desplazamiento por tablas

Una característica común de muchas aplicaciones es una serie de botones de desplazamiento que permiten a los usuarios moverse por una tabla. Suelen incluir botones para mover el puntero de registro al registro siguiente o anterior de la tabla, así como al registro superior o inferior de la tabla.

### Botones de desplazamiento por tablas



### Diseño de los botones de desplazamiento

Todos los botones tendrán algunas características y funciones comunes, por lo que es conveniente crear una clase de botones de desplazamiento. A continuación, los botones individuales pueden aprovechar fácilmente esta apariencia y funcionalidad comunes. Esta clase primaria es la clase `NavButton` que se definirá posteriormente en esta sección.

Una vez definida la clase primaria, las subclases siguientes definen la funcionalidad y apariencia específicas de cada uno de los cuatro botones de desplazamiento: `navTop`, `navPrior`, `navNext`, `navBottom`.

Por último se crea una [clase de contenedor](#) `vcr`, a la que se agregan todos los botones de desplazamiento. El contenedor puede agregarse a un [formulario](#) o una [barra de herramientas](#) para proporcionar funcionalidad de desplazamiento por tablas.

### Definición de la clase NAVBUTTON

Para crear `NavButton`, guarde las seis definiciones de clase siguientes (`Navbutton`, `navTop`, `navBottom`, `navPrior`, `navNext` y `vcr`) en un archivo de programa como `Navclass.prg`.

### Definición de la clase genérica botón de comando de desplazamiento

Código	Comentarios
<pre> DEFINE CLASS NavButton AS COMMANDBUTTON      Height = 25     Width = 25     TableAlias = "" </pre>	<p>Define la clase primaria de los botones de desplazamiento.</p> <p>Asigna dimensiones a la clase.</p> <p>Incluye una <a href="#">propiedad</a> personalizada, TableAlias, que contiene el nombre del <a href="#">alias</a> por el que desplazarse.</p>
<pre> PROCEDURE Click     IF NOT EMPTY(This.TableAlias)         SELECT (This.TableAlias)     ENDIF ENDPROC </pre>	<p>Si se ha establecido TableAlias, este procedimiento de clase primaria selecciona el alias antes de ejecutar el código real de desplazamiento en las subclases. De lo contrario, se supondrá que el usuario desea desplazarse por la tabla del área de trabajo seleccionada actualmente.</p>
<pre> PROCEDURE RefreshForm     _SCREEN.ActiveForm.Refresh ENDPROC </pre>	<p>Al emplear _SCREEN.ActiveForm.Refresh en lugar de THISFORM.Refresh puede agregar la clase a un <a href="#">formulario</a> o una <a href="#">barra de herramientas</a> y hacer que funcione con la misma precisión.</p>
<pre> ENDDEFINE </pre>	<p>Finaliza la definición de clase.</p>

Los botones de desplazamiento específicos se basan en la clase NavButton. El código siguiente define el botón Superior para el conjunto de botones de desplazamiento. Los tres botones de desplazamiento restantes se definen en la tabla siguiente. Las cuatro definiciones de clase son similares. Por ello, sólo se ofrecen comentarios extensos para la primera definición.

### Definición de la clase botón de desplazamiento Superior

Código	Comentarios
<pre> DEFINE CLASS navTop AS BotDespl     Caption = "&lt;" </pre>	<p>Define la clase botón de desplazamiento Superior y establece la <a href="#">propiedad Caption</a>.</p>
<pre> PROCEDURE Click </pre>	<p>Crea código de método que se ejecutará cuando se produzca el <a href="#">evento Click</a> para el <a href="#">control</a>.</p>
<pre>     DODEFAULT( ) </pre>	<p>Llama al código de evento Click de la clase primaria, Navbutton, de modo que se pueda seleccionar el <a href="#">alias</a> adecuado si se ha establecido la propiedad TableAlias.</p>
<pre>     GO TOP </pre>	<p>Incluye el código para establecer el puntero de registro en el primer registro de la tabla: GO TOP.</p>
<pre>     THIS.RefreshForm </pre>	

Llama al método RefreshForm de la clase primaria. No es necesario utilizar el [operador de resolución de alcance \(::\)](#) en este caso porque no hay ningún [método](#) en la subclase que tenga el mismo nombre que el método de la clase primaria. Por otra parte, tanto la clase primaria como la subclase tienen código de método para el evento Click.

---



---

ENDPROC

Termina el procedimiento Click.

---



---

ENDDEFINE

Termina la definición de clase.

---



---

Los restantes botones de desplazamiento tienen definiciones de clase similares.

### Definición de las demás clases de botones de desplazamiento

#### Código

#### Comentarios

---



---

```
DEFINE CLASS navNext AS Navbutton
    Caption = ">"
```

Define la clase de botón de desplazamiento Siguiente y establece la [propiedad Caption](#).

---



---



---



---

```
PROCEDURE Click
    DODEFAULT( )
    SKIP 1
    IF EOF( )
        GO BOTTOM
    ENDIF
    THIS.RefreshForm
ENDPROC
ENDDEFINE
```

Incluye el código para establecer el puntero de registro en el siguiente registro de la tabla.

Termina la definición de la clase.

---



---

```
DEFINE CLASS navPrior AS Navbutton
    Caption = "<"
```

Define la clase de botón de desplazamiento Anterior y establece la propiedad Caption.

---



---



---



---

```
PROCEDURE Click
    DODEFAULT( )
    SKIP -1
    IF BOF( )
        GO TOP
    ENDIF
    THIS.RefreshForm
ENDPROC
ENDDEFINE
```

Incluye el código para establecer el puntero de registro en el registro anterior de la tabla.

Termina la definición de clase.

---



---

```
DEFINE CLASS navBottom AS
Navbutton
    Caption = ">|"
```

Define la clase de botón de desplazamiento Inferior y establece la propiedad Caption.

---



---



---

```

PROCEDURE Click
    DODEFAULT( )
    GO BOTTOM
    THIS.RefreshForm
ENDPROC
ENDDEFINE

```

Incluye el código para establecer el puntero de registro en el último registro de la tabla.

Termina la definición de clase.

---

La siguiente definición de clase contiene los cuatro botones de desplazamiento para poder agregarlos como una unidad a un formulario. La clase también incluye un método para establecer la propiedad TableAlias de los botones.

### Definición de una clase de controles de desplazamiento por tabla

#### Código

---

```

DEFINE CLASS vcr AS CONTAINER
    Height = 25
    Width = 100
    Left = 3
    Top = 3

    ADD OBJECT cmdTop AS navTop ;
        WITH Left = 0
    ADD OBJECT cmdPrior AS navPrior ;
        WITH Left = 25
    ADD OBJECT cmdNext AS navNext ;
        WITH Left = 50
    ADD OBJECT cmdBot AS navBottom ;
        WITH Left = 75

```

#### Comentarios

Comienza la definición de clase. La [propiedad Height](#) se establece en el mismo alto que los botones de comando que contendrá.

---

```

PROCEDURE SetTable(cTableAlias)
    IF TYPE("cTableAlias") = 'C'
        THIS.cmdTop.TableAlias = ;
            cTableAlias
        THIS.cmdPrior.TableAlias = ;
            cTableAlias
        THIS.cmdNext.TableAlias = ;
            cTableAlias
        THIS.cmdBot.TableAlias = ;
            cTableAlias
    ENDIF
ENDPROC

```

Este método se utiliza para establecer la propiedad TableAlias de los botones. TableAlias se define en la clase primaria Navbutton.

También podría utilizar el [método SetAll](#) para establecer esta propiedad:

```

IF TYPE ("cTableAlias") = 'C'
    This.SetAll("TableAlias",
        "cTableAlias")
ENDIF

```

Sin embargo, esto produciría un error si se agregara a la clase un objeto que no tuviera la propiedad TableAlias.

---

```

ENDDEFINE

```

Termina la definición de clase.

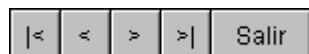
---

Una vez definida la clase, puede dividirla en subclases o agregarla a un [formulario](#).

## Crear una subclase basada en la nueva clase

También puede crear subclases basadas en `vcr` que tengan botones adicionales como Buscar, Modificar, Guardar y Salir. Por ejemplo, `vcr2` incluye un botón Salir:

## Botones de desplazamiento por tablas con un botón para cerrar el formulario



## Definición de una subclase de control de desplazamiento por tablas

Código	Comentarios
<pre> DEFINE CLASS vcr2 AS vcr ADD OBJECT cmdQuit AS COMMANDBUTTON WITH ;     Caption = "Salir",;     Height = 25, ;     Width = 50 Width = THIS.Width + THIS.cmdQuit.Width cmdQuit.Left = THIS.Width - ;     THIS.cmdQuit.Width </pre>	<p>Define una clase basada en <code>vcr</code> y le agrega un botón de comando.</p>
<pre> PROCEDURE cmdQuit.CLICK     RELEASE THISFORM ENDPROC </pre>	<p>Cuando el usuario haga clic en <code>cmdQuit</code>, este código liberará el <a href="#">formulario</a>.</p>
<pre> ENDDEFINE </pre>	<p>Termina la definición de clase.</p>

`vcr2` tiene todo lo de `vcr` más el nuevo botón de comando y no es necesario volver a escribir ninguna parte del código.

## Cambios en VCR reflejados en la subclase

A causa de la [herencia](#), los cambios realizados en la clase primaria se reflejan en todas las subclases que se basan en ella. Por ejemplo, puede informar al usuario de que se ha llegado al final de la tabla si cambia la instrucción `IF EOF( )` de `navNext.Click` por la siguiente:

```

IF EOF( )
    GO BOTTOM
    SET MESSAGE TO "Final de la tabla"
ELSE
    SET MESSAGE TO
ENDIF

```

Puede indicar al usuario que ha llegado al principio de la tabla si cambia la instrucción `IF BOF( )` de `navPrior.Click` por la siguiente:

```

IF BOF( )

```

```
GO TOP
SET MESSAGE TO "Principio de la tabla"
ELSE
SET MESSAGE TO
ENDIF
```

Si se realizan estos cambios en las clases `navNext` y `navPrior`, también se aplicarán automáticamente a los botones apropiados de `vcr` y `vcr2`.

### Agregar `vcr` a una clase de formulario

Una vez definido `vcr` como un [control](#), el control puede agregarse a la definición de un contenedor. Por ejemplo, el código siguiente agregado a `Navclass.prg` define un [formulario](#) al que se han agregado botones de desplazamiento:

```
DEFINE CLASS NavForm AS Form
ADD OBJECT oVCR AS vcr
ENDDDEFINE
```

### Ejecutar el formulario que contiene VCR

Una vez definida la subclase de formulario, podrá mostrarla fácilmente con los comandos apropiados.

#### Para mostrar el formulario

1. Cargue la definición de clase:

```
SET PROCEDURE TO navclass ADDITIVE
```

2. Cree un [objeto](#) basado en la clase `navForm`:

```
frmPrueba = CREATEOBJECT("navForm")
```

3. Invoque el método [Show](#) del formulario:

```
frmPrueba.Show
```

Si no llama al método `SetTable` de `oVCR` (el objeto VCR de `NavForm`), cuando el usuario haga clic en los botones de desplazamiento el puntero de registro se moverá por la tabla del área de trabajo seleccionada actualmente. Puede llamar al método `SetTable` para especificar en qué tabla se va a desplazar.

```
frmPrueba.oVCR.SetTable("customer")
```

**Nota** Cuando el usuario cierre el formulario, `frmPrueba` se establecerá a un [valor nulo](#) (.NULL.). Para liberar de la memoria la variable de objeto, utilice el comando [RELEASE](#). Las variables de objeto creadas en los archivos de programa se liberan de la memoria cuando se completa el programa.

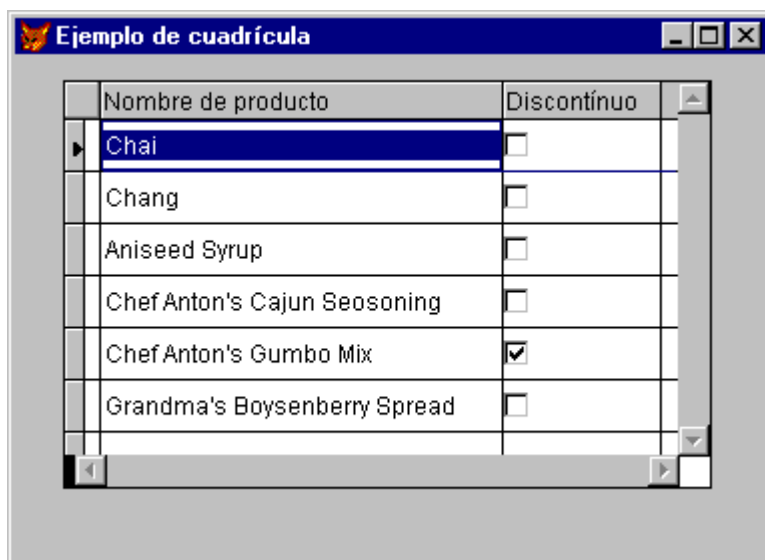
### Definir un control cuadrícula

Una [cuadrícula](#) contiene columnas que, a su vez, pueden contener encabezados y cualquier otro

[control](#). El control predeterminado contenido en una columna es un [cuadro de texto](#), por lo que la funcionalidad predeterminada de la cuadrícula se aproxima a una ventana [Examinar](#). Sin embargo, la arquitectura subyacente de la cuadrícula la abre hasta una extensión ilimitada.

El ejemplo siguiente crea un formulario que contiene un objeto Grid (Cuadrícula) con dos columnas. La segunda columna contiene una [casilla de verificación](#) para mostrar los valores en un campo lógico de una tabla.

### Control Grid (Cuadrícula) con una casilla de verificación en una columna



### Definición de una clase Grid con una casilla de verificación en una columna de cuadrícula

#### Código

```
DEFINE CLASS grdProducts AS Grid
    Left = 24
    Top = 10
    Width = 295
    Height = 210
    Visible = .T.
    RowHeight = 28
    ColumnCount = 2

    Column1.ControlSource = "prod_name"
    Column2.ControlSource = "discontinuo"

    Column2.Sparse = .F.
```

#### Comentarios

Comienza la definición de clase y establece las [propiedades](#) que determinan la apariencia de la cuadrícula.

Al establecer la propiedad ColumnCount en 2, se agregan dos columnas a la cuadrícula. Cada columna contiene un encabezado con el nombre Header1. Además, cada columna tiene un grupo de propiedades independiente que determina su apariencia y comportamiento.

Al establecer la propiedad ControlSource de una columna, la columna muestra los valores de ese campo para todos los registros de la tabla.

Discontinuo es un campo lógico.

Column2 contendrá la casilla de verificación. Establezca la [propiedad Sparse](#) de la columna

en .F. de modo que la casilla de verificación sea visible en todas las filas, no sólo en la celda seleccionada.

---

```

Procedure Init
    THIS.Column1.Width = 175
    THIS.Column2.Width = 68
    THIS.Column1.Header1.Caption = ;
        "Nombre de producto"
    THIS.Column2.Header1.Caption = ;
        "Suspendido"

    THIS.Column2.AddObject("chk1", ;
        "checkbox")
    THIS.Column2.CurrentControl = ;
        "chk1"
    THIS.Column2.chk1.Visible = .T.
    THIS.Column2.chk1.Caption = ""
ENDPROC

```

Establece el ancho de las columnas y los títulos de los encabezados.

El [método AddObject](#) permite agregar un objeto a un contenedor ; en este caso, una casilla de verificación llamada `chk1`. Establece la propiedad [CurrentControl](#) de la columna en la casilla de verificación, de modo que se muestre la casilla de verificación. Comprueba que la casilla de verificación es visible. Establece el título en una cadena vacía de modo que no se muestre el [título](#) predeterminado "chk1".

---

```

ENDDEFINE

```

Termina la definición de clase.

---

La siguiente definición de clase es el formulario que contiene la cuadrícula. Ambas definiciones de clase pueden incluirse en el mismo archivo de programa.

### Definición de una clase Form que contiene la clase Grid

#### Código

---

```

DEFINE CLASS GridForm AS FORM
    Width = 330
    Height = 250
    Caption = "Ejemplo de cuadrícula"
    ADD OBJECT grid1 AS grdProducts

```

#### Comentarios

Crea una clase de formulario y le agrega un [objeto](#) basado en la clase de cuadrícula.

---

```

PROCEDURE Destroy
    CLEAR EVENTS
ENDPROC

ENDDEFINE

```

El programa que crea un objeto basado en esta clase utilizará [READ EVENTS](#). Al incluir `CLEAR EVENTS` en el [evento Destroy](#) del formulario, el programa podrá terminar cuando el usuario cierre el formulario. Termina la definición de clase.

---

El programa siguiente abre la tabla donde están incluidos los campos que se van a mostrar en las columnas de cuadrícula, crea un objeto basado en la clase GridForm y ejecuta el comando [READ EVENTS](#).

```
CLOSE DATABASE
OPEN DATABASE (SYS(2004) + "samples\data\testdata.dbc")
USE products
frmTest= CREATEOBJECT("GridForm")
frmTest.Show
READ EVENTS
```

Este programa puede incluirse en el mismo archivo en el que están incluidas las definiciones de clase si aparece al principio del archivo. También puede emplear el comando [SET PROCEDURE TO](#) para especificar el programa que contiene las definiciones de clase e incluir este código en un programa distinto.

## Crear referencias a objetos

En lugar de realizar una copia de un [objeto](#), puede crear una referencia a dicho objeto. Una referencia ocupa menos memoria que un objeto adicional, puede transferirse fácilmente entre [procedimientos](#) y puede ayudar a escribir código genérico.

### Devolver una referencia a un objeto

En algunas ocasiones puede resultar conveniente manipular un [objeto](#) por medio de una o varias referencias al mismo. Por ejemplo, el programa siguiente define una clase, crea un objeto basado en la clase y devuelve una referencia al objeto:

```
*--NEWINV.PRG
*--Devuelve una referencia a un nuevo formulario de facturas.
frmInv = CREATEOBJECT("InvoiceForm")
RETURN frmInvoice

DEFINE CLASS InvoiceForm AS FORM
    ADD OBJECT txtCompany AS TEXTBOX
    * código para establecer propiedades, agregar otros objetos, etc.
ENDDEFINE
```

El programa siguiente establece una referencia al objeto creado en Newinv.prg. La [variable](#) de referencia puede manipularse exactamente del mismo modo que la variable de objeto:

```
frmInvoice = NewInv() && almacena la referencia al objeto en una variable
frmInvoice.SHOW
```

También puede crear una referencia a un objeto de un [formulario](#), como en el ejemplo siguiente.

```
txtCustName = frmInvoice.txtCompany
txtCustName.Value = "Usuario de Fox"
```

**Sugerencia** Cuando ha creado un objeto, puede usar el comando [DISPLAY OBJECTS](#) para mostrar la jerarquía de clases del objeto, los valores de las propiedades, los objetos contenidos y los métodos y eventos disponibles. Puede llenar una matriz con las propiedades (no los valores de las propiedades), eventos, métodos y objetos contenidos de un objeto con la función [AMEMBERS\(\)](#).

## Liberar objetos y referencias de la memoria

Si existe una referencia a un objeto, la liberación del [objeto](#) no borra el objeto de la memoria. Por ejemplo, el comando siguiente libera el objeto original, frmFactura:

```
RELEASE frmFactura
```

Sin embargo, puesto que sigue existiendo una referencia a un objeto perteneciente a frmFactura, el objeto no se liberará de la memoria hasta que se libere txtNombrePers con el comando siguiente:

```
RELEASE txtNombrePers
```

## Comprobar si existe un objeto

Puede utilizar las funciones [TYPE\(\)](#), [ISNULL\(\)](#) y [VARTYPE\(\)](#) para determinar si existe un objeto. Por ejemplo, las líneas de código siguientes comprueban si existe un objeto llamado oConexión:

```
IF TYPE("oConexión") = "O" AND NOT ISNULL(oConexión)
    * El objeto existe
ELSE
    * El objeto no existe
ENDIF
```

**Nota** El comando [ISNULL\(\)](#) es necesario porque .NULL. se almacena en la variable de objeto de formulario cuando un usuario cierra un formulario, pero el tipo de variable sigue siendo "O".

## Crear matrices de miembros

Puede definir miembros de clases como [matrices](#). En el ejemplo siguiente, elecc es una matriz de [controles](#).

```
DEFINE CLASS MoverListBox AS CONTAINER
    DIMENSION choices[3]
    ADD OBJECT lFromListBox AS LISTBOX
    ADD OBJECT lToListBox AS LISTBOX
    ADD OBJECT choices[1] AS COMMANDBUTTON
    ADD OBJECT choices[2] AS COMMANDBUTTON
    ADD OBJECT choices[3] AS CHECKBOX
    PROCEDURE choices.CLICK
        PARAMETER nIndex
        DO CASE
            CASE nIndex = 1
                * código
            CASE nIndex = 2
                * código
            CASE nIndex = 3
                * código
        ENDCASE
    ENDPROC
ENDDEFINE
```

Cuando un usuario hace clic en un control incluido en una matriz de controles, Visual FoxPro transfiere el número de índice del control al procedimiento de evento Click. En este procedimiento,

puede utilizar una instrucción [CASE](#) para ejecutar código distinto según el botón en el que se haya hecho clic.

## Crear matrices de objetos

También puede crear [matrices](#) de [objetos](#). Por ejemplo, `MiMatriz` contiene cinco botones de comando:

```
DIMENSION MiMatriz[5]
FOR x = 1 TO 5
    MiMatriz[x] = CREATEOBJECT("COMMANDBUTTON")
ENDFOR
```

Hay una serie de consideraciones que conviene tener en cuenta con respecto a las matrices de objetos:

- No se puede asignar un objeto a una matriz completa mediante un comando. Es necesario asignar individualmente el objeto a cada miembro de la matriz.
- No se puede asignar un valor a una [propiedad](#) de una matriz completa. El comando siguiente produciría un error:

```
MiMatriz.Enabled = .F.
```

- Al redimensionar una matriz de objetos para que sea más grande que la matriz original, los elementos nuevos se inicializarán como falso (.F.), como ocurre con todas las matrices de Visual FoxPro. Cuando redimensione una matriz de objetos para que sea más pequeña que la matriz original, se liberarán los objetos cuyo subíndice sea mayor que el mayor subíndice nuevo.

## Usar objetos para almacenar datos

En los lenguajes orientados a objetos, una [clase](#) ofrece un medio útil y cómodo para almacenar datos y [procedimientos](#) relacionados con una entidad. Por ejemplo, podría definir una clase de cliente para incluir en ella información sobre un cliente, así como un [método](#) para calcular la edad del cliente:

```
DEFINE CLASS cliente AS CUSTOM
    Apellidos = ""
    Nombre = ""
    FechaNacimiento = { - - }
    PROCEDURE Edad
        IF !EMPTY(THIS.FechaNacimiento)
            RETURN YEAR(THIS.FechaNacimiento) - YEAR(THIS.FechaNacimiento)
        ELSE
            RETURN 0
        ENDIF
    ENDPROC
ENDDEFINE
```

Sin embargo, los datos almacenados en objetos que se basan en la clase de cliente sólo se almacenan en memoria. Si estos datos estuvieran en una tabla, ésta se almacenaría en disco. Si tuviera que hacer un seguimiento de varios clientes, la tabla le daría acceso a todos los comandos y las funciones de administración de bases de datos de Visual FoxPro. De este modo, podría localizar información rápidamente, ordenarla, agruparla, realizar cálculos, crear informes y consultas basándose en la



información, etc.

Visual FoxPro ofrece un resultado incomparable en cuanto al almacenamiento y la manipulación de datos de [bases de datos](#) y [tablas](#). Sin embargo, en determinadas ocasiones deseará almacenar datos en [objetos](#). Generalmente, los datos sólo serán significativos mientras se esté ejecutando la aplicación y pertenecerán a una única entidad.

Por ejemplo, en una aplicación que incluye un sistema de seguridad, normalmente tendría una tabla de los usuarios que tienen acceso a la aplicación. La tabla incluiría la identificación, la contraseña y el nivel de acceso del usuario. Cuando un usuario haya iniciado una sesión no necesitará toda la información de la tabla. Lo único que necesitará es la información sobre el usuario actual y esta información se puede almacenar y manipular fácilmente en un objeto. Por ejemplo, la definición de clase siguiente inicia una sesión al crear un objeto basado en la clase:

```
DEFINE CLASS NuevoUsuario AS CUSTOM
    PROTECTED LogonTime, AccessLevel
    UserId = ""
    Password = ""
    LogonTime = { - - : : }
    AccessLevel = 0
    PROCEDURE Init
        DO FORM LOGON WITH ; && suponiendo que ha creado este formulario
            This.UserId, ;
            This.Password, ;
            This.AccessLevel
        This.LogonTime = DATETIME( )
    ENDPROC
* Crear métodos para devolver valores de propiedad protegidos.
    PROCEDURE GetLogonTime
        RETURN This.LogonTime
    ENDPROC
    PROCEDURE GetAccessLevel
        RETURN This.AccessLevel
    ENDPROC
ENDDEFINE
```

En el programa principal de la aplicación, podría crear un [objeto](#) basado en la clase NuevoUsuario:

```
oUser = CREATEOBJECT('NuevoUsuario')
oUser.Logon
```

En cualquier parte de la aplicación, cuando necesite información sobre el usuario actual, podrá obtenerla del objeto oUser. Por ejemplo:

```
IF oUser.GetAccessLevel( ) >= 4
    DO ADMIN.MPR
ENDIF
```

## Integrar objetos y datos

En la mayoría de las aplicaciones, puede sacar el máximo partido de la potencia de Visual FoxPro si integra objetos y datos. La mayoría de las clases de Visual FoxPro tienen [propiedades](#) y [métodos](#) que permiten integrar la potencia de un administrador de base de datos relacional y un sistema completamente orientado a objetos.

## Propiedades para integrar datos de clases y bases de datos de Visual FoxPro

Clase	Propiedades de datos
<a href="#">Cuadrícula</a>	<a href="#">RecordSource</a> , <a href="#">ChildOrder</a> , <a href="#">LinkMaster</a>
Todos los demás <a href="#">controles</a>	<a href="#">ControlSource</a>
<a href="#">Cuadro de lista</a> y <a href="#">cuadro combinado</a>	<a href="#">ControlSource</a> , <a href="#">RowSource</a>
<a href="#">Formulario</a> y <a href="#">conjunto de formularios</a>	<a href="#">DataSession</a>

Puesto que estas propiedades de datos pueden cambiarse en [tiempo de diseño](#) o en [tiempo de ejecución](#), puede crear controles genéricos con funcionalidad encapsulada que opere con datos diversos.

Para obtener más información sobre la integración de datos y objetos, consulte el capítulo 9, [Crear formularios](#) y el capítulo 10, [Usar controles](#).

## Capítulo 4: Descripción del modelo de eventos

Visual FoxPro ofrece un auténtico funcionamiento [no modal](#), por lo que es posible coordinar fácilmente múltiples [formularios](#) automáticamente y ejecutar simultáneamente múltiples instancias de un formulario. Además, Visual FoxPro se encarga del procesamiento de los eventos, por lo que puede ofrecer a sus usuarios un entorno interactivo mucho más rico.

En este capítulo se describe:

- [Eventos de Visual FoxPro](#)
- [Seguimiento de secuencias de eventos](#)
- [Asignar código a eventos](#)

### Eventos de Visual FoxPro

El sistema desencadena automáticamente un [código de evento](#) como respuesta a alguna acción del usuario. Por ejemplo, el sistema procesa automáticamente el código escrito para el evento Click cuando el usuario hace clic en un control. El código de un evento también puede desencadenarse mediante eventos del sistema, como es el caso del evento Timer en un control de cronómetro.

### Los eventos básicos

La tabla siguiente contiene una lista del principal conjunto de eventos de Visual FoxPro que se aplican a la mayoría de los controles.

### Conjunto básico de eventos

Evento	Cuándo se desencadena el evento
<a href="#">Init</a>	Al crear un objeto.
<a href="#">Destroy</a>	Al liberar de la memoria un objeto.
<a href="#">Click</a>	Cuando el usuario hace clic en el objeto con el botón principal del <i>mouse</i> .
<a href="#">DblClick</a>	Cuando el usuario hace doble clic en el objeto con el botón principal del <i>mouse</i> .
<a href="#">RightClick</a>	Cuando el usuario hace clic en el objeto con el botón secundario del <i>mouse</i> .
<a href="#">GotFocus</a>	Cuando el objeto recibe el enfoque, ya sea como resultado de una acción del usuario o al hacer clic, o porque se cambie el enfoque en el código mediante el <a href="#">método SetFocus</a> .
<a href="#">LostFocus</a>	Cuando el objeto pierde el enfoque, ya sea como resultado de una acción del usuario o al hacer clic, o porque se cambie el enfoque en el código mediante el <a href="#">método SetFocus</a> .
<a href="#">KeyPress</a>	Cuando el usuario presiona y suelta una tecla.
<a href="#">MouseDown</a>	Cuando el usuario presiona el botón del <i>mouse</i> mientras el puntero del <i>mouse</i> se encuentra sobre el objeto.
<a href="#">MouseMove</a>	Cuando el usuario mueve el <i>mouse</i> sobre el objeto.
<a href="#">MouseUp</a>	Cuando el usuario libera un botón del <i>mouse</i> mientras el puntero del <i>mouse</i> se encuentra sobre el objeto.

## Contenedores y eventos de objeto

A la hora de escribir código de eventos para los [controles](#) se deben tener en cuenta dos reglas básicas:

- Los contenedores no procesan los [eventos](#) asociados a los controles que contienen.
- Si no hay código de evento asociado a un control, Visual FoxPro comprobará si hay código asociado al evento en algún nivel superior de la jerarquía de clase para dicho control.

Cuando un usuario interactúa con un objeto de alguna forma, ya sea presionando la tecla tab, haciendo clic en él, moviendo el puntero del *mouse* sobre él, etc., tienen lugar los eventos de objeto. Cada objeto recibe sus eventos de forma independiente. Por ejemplo, aunque un botón de comando se encuentre en un formulario, el evento Click del formulario no se desencadenará cuando un usuario haga clic en el botón de comando; sólo se desencadenará el evento Click del botón de comando.

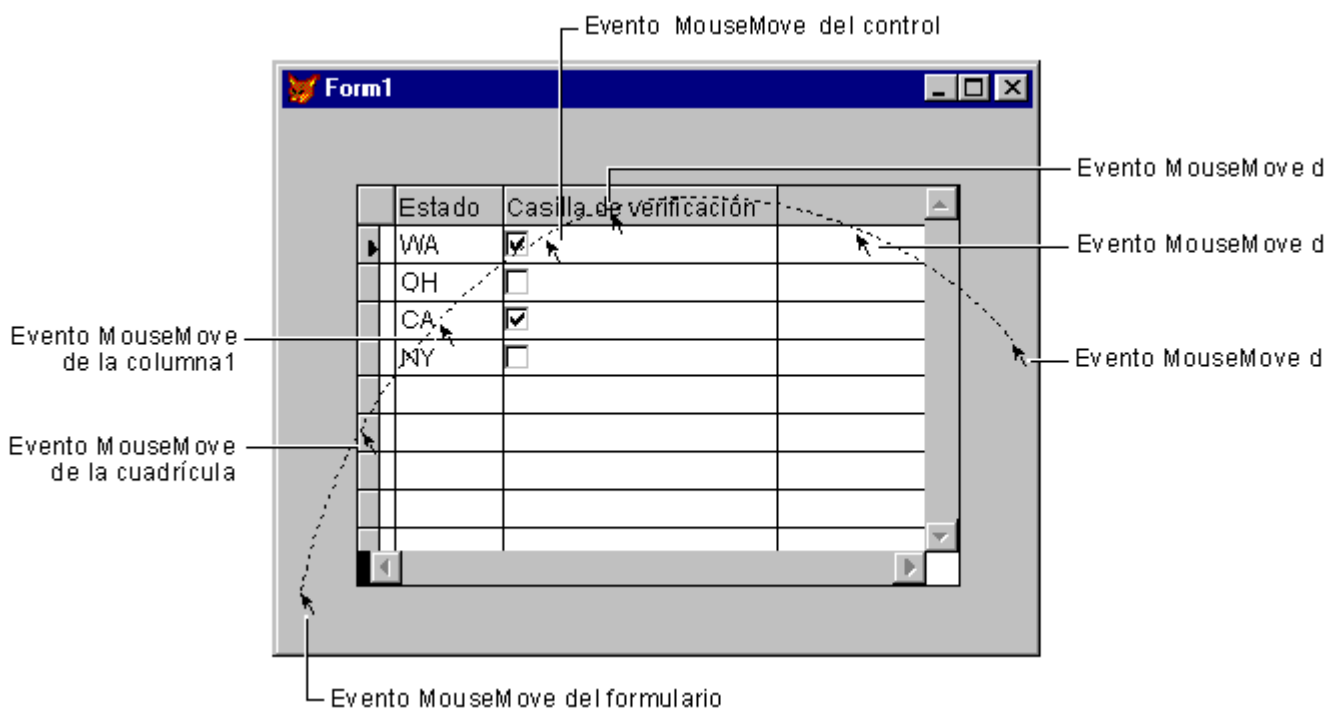
## El código de evento del contenedor es distinto del código de evento del control



Si no hay ningún código de evento Click asociado al botón de comando, cuando el usuario haga clic en el botón no ocurrirá nada, incluso cuando haya un código de evento Click asociado al formulario.

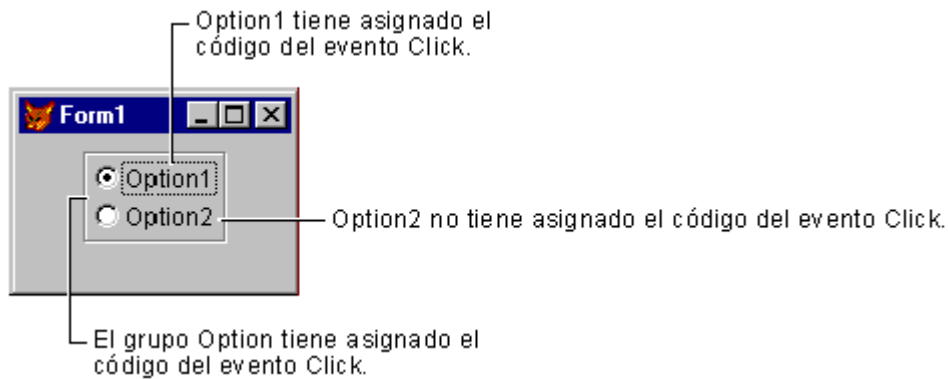
Esta regla también es aplicable a los controles [cuadrícula](#). La cuadrícula contiene columnas que a su vez contienen encabezados y controles. Cuando ocurren los eventos, sólo el objeto más interno implicado en el evento reconoce el evento. Los contenedores de mayor nivel no reconocen el evento. La ilustración siguiente muestra qué objetos procesan los eventos MouseMove que se generan cuando un usuario mueve el puntero del *mouse* por la cuadrícula.

### Eventos MouseMove para una cuadrícula



No obstante, hay una excepción a esta regla. Si ha escrito código de evento para un grupo de botones de opción o para un grupo de botones de comando pero no hay código para el evento en un determinado botón del grupo, el código de evento del grupo *se* ejecutará cuando se produzca el evento del botón.

Por ejemplo, podría tener un grupo de botones de opción con un código de evento Click asociado. Sólo uno de los dos botones de opción del grupo tienen asociado código de evento Click:

**El código de evento para los grupos de botones puede utilizarse como valor predeterminado**

Si un usuario hace clic en Opción1, se ejecutará el código de evento Click asociado a Opción1. El código de evento Click asociado al grupo de botones de opción no se ejecutará.

Puesto que no hay código de evento Click asociado a Opción2, si el usuario hace clic en Opción2 se ejecutará el código de evento Click del grupo de opciones.

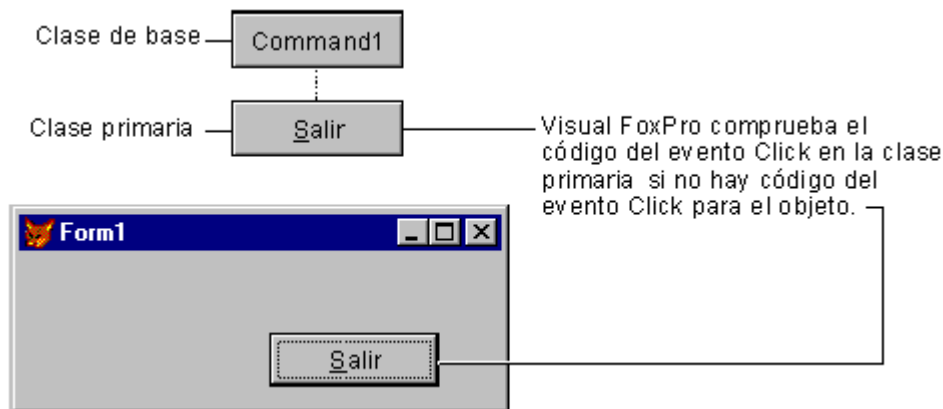
**Nota** Cuando se inicia una secuencia de eventos, como MouseDown y MouseUp, para un control, toda la secuencia de eventos pertenece al control.

Por ejemplo, si presiona el botón primario del *mouse* en un botón de comando y arrastra el puntero del *mouse* hacia fuera del botón de comando, los eventos MouseMove del botón de comando seguirán produciéndose, aunque el puntero del *mouse* se esté moviendo fuera del formulario. Si suelta el botón primario del *mouse* sobre el formulario en lugar de hacerlo sobre el botón de comando, el evento MouseUp que ocurrirá será el asociado al botón de comando, no al formulario.

**Clases y eventos de controles**

Si un [control](#) de un [formulario](#) está basado en una [clase definida por el usuario](#) (que, a su vez, está basada en otra clase definida por el usuario), cuando se produzca un evento, Visual FoxPro comprobará el [código de evento](#) del control inmediato. Si hay código en ese procedimiento de evento, Visual FoxPro lo ejecutará. Si no existe código en el procedimiento de evento, Visual FoxPro comprobará un nivel superior en la jerarquía de clases. Si en algún lugar de la jerarquía de clases Visual FoxPro encuentra código para el evento, se ejecutará dicho código. Cualquier código que haya más allá dentro de la jerarquía no se ejecutará.

**Si no hay código de evento asociado a un objeto, Visual FoxPro comprobará la clase primaria.**



No obstante, puede incluir código en un procedimiento de evento y llamar explícitamente al código en clases en las cuales se base el control; para ello se utiliza la función [DODEFAULT\(\)](#).

## Seguimiento de secuencias de eventos

El modelo de eventos de Visual FoxPro es amplio y le concede bastante control sobre los componentes de la aplicación en respuesta a una amplia variedad de acciones de usuario. Algunas de las secuencias de eventos son fijas como, por ejemplo, la creación o destrucción de un [formulario](#). Algunos [eventos](#) ocurren de forma independiente, pero la mayor parte ocurre con otros eventos basados en la interacción con el usuario.

### Establecer el seguimiento de eventos

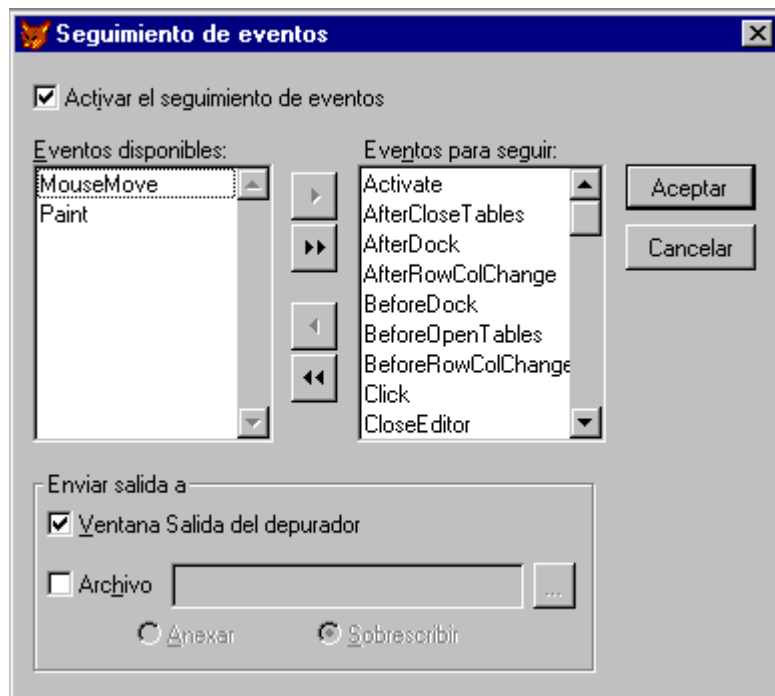
La mejor manera de ver las secuencias de eventos de Visual FoxPro es establecer el seguimiento de eventos en el depurador. El seguimiento de eventos le permite ver cuándo tiene lugar cada evento asociado a sus propios formularios y controles en relación a otros eventos, de forma que puede determinar el lugar más eficiente para incluir el código.

#### Para establecer el seguimiento de eventos

1. En el menú **Herramientas** de la ventana [Depurador](#), elija **Seguimiento de eventos**.
2. En el cuadro de diálogo **Seguimiento de eventos**, seleccione **Activar el seguimiento de eventos**.

Los eventos de la lista Eventos para seguir se escriben en la ventana Salida del depurador o en un archivo cuando tengan lugar.

#### El cuadro de diálogo Seguimiento de eventos



**Nota** En este ejemplo, los eventos `MouseMove` y `Paint` han sido eliminados de la lista `Eventos para seguir` porque ocurren con tanta frecuencia que hacen más difícil ver las secuencias de los otros eventos.

## Observar cómo ocurren los eventos

A veces una acción de un usuario desencadena un único evento, como mover el puntero del *mouse* sobre un control, por ejemplo. Sin embargo, con frecuencia una acción del usuario desencadena múltiples eventos.

En esta sección se describe el orden en que ocurren los eventos como respuesta a la interacción del usuario, utilizando el siguiente formulario como ejemplo.

## Formulario de ejemplo para ilustrar las secuencias de eventos



En esta situación de ejemplo, el usuario realiza las siguientes acciones en el formulario:

1. Ejecuta el formulario.

2. Escribe texto en Text1.
3. Selecciona el campo y lo copia al Portapapeles.
4. Va a Text2.
5. Pega el texto en Text2.
6. Cierra el formulario haciendo clic en Command2.

Estas acciones desencadenan uno o más eventos del sistema para cada objeto. En las tablas siguientes se describen los eventos desencadenados como respuesta a cada acción del usuario.

### Acción 1

El usuario ejecuta el formulario escribiendo el siguiente comando en la ventana [Comandos](#):

```
DO FORM form1 NAME frmObject
```

Visual FoxPro carga el formulario, inicializa cada objeto y después inicializa el formulario; el formulario se activa y el primer campo recibe el enfoque de entrada.

Objeto	Evento
DataEnvironment	<a href="#">BeforeOpenTables</a>
Form1	<a href="#">Load</a>
DataEnvironment	<a href="#">Init</a>
Text1	<a href="#">Init</a>
Text2	<a href="#">Init</a>
Command1	<a href="#">Init</a>
Command2	<a href="#">Init</a>
Form1	<a href="#">Init</a>
Form1	<a href="#">Activate</a>
Form1	<a href="#">GotFocus</a>
Text1	<a href="#">When</a>
Text1	<a href="#">GotFocus</a>

### Acción 2

El usuario escribe **Test** en Text1. Cada pulsación de teclas genera dos eventos. El [evento KeyPress](#)



recibe 2 parámetros: la tecla presionada y el estado de las teclas MAYÚS, ALT y CTRL.

Objeto	Evento
Text1	KeyPress(84, 1) “T”
Text1	InteractiveChange
Text1	KeyPress(101, 0) “e”
Text1	InteractiveChange
Text1	KeyPress(115,0) “s”
Text1	InteractiveChange
Text1	KeyPress(116,0) “t”
Text1	InteractiveChange

### Acción 3

El usuario hace doble clic en Text1 para seleccionar el texto y presiona CTRL+C para copiar el texto al [Portapapeles](#). Los eventos Mouse y un [evento Click](#) acompañan al [evento DblClick](#). Los eventos [MouseMove](#) y [MouseDown](#) reciben cuatro [parámetros](#): qué botón, el estado de MAYÚS y las ubicaciones X e Y. Las ubicaciones X e Y son relativas al formulario y reflejan el modo de escala (por ejemplo, [píxeles](#)) del formulario. Sólo se presenta un evento MouseMove para cada control. En realidad, este evento se activaría probablemente media docena de veces o más.

Objeto	Evento
Form1	MouseMove(0, 0, 100, 35)
Text1	MouseMove(0,0,44,22)
Text1	MouseDown(1, 0, 44, 22)
Text1	MouseUp(1, 0, 44, 22)
Text1	Click
Text1	MouseDown(1, 0, 44, 22)
Text1	MouseUp(1, 0, 44, 22)
Text1	DblClick

### Acción 4

El usuario presiona TAB para pasar a Text2.

Objeto	Evento
Text1	<a href="#">KeyPress(9, 0)</a>
Text1	<a href="#">Valid</a>
Text1	<a href="#">LostFocus</a>
Text2	<a href="#">When</a>
Text2	<a href="#">GotFocus</a>

**Acción 5**

El usuario pega en Text2 el texto copiado al presionar CTRL+V.

Objeto	Evento
Text2	InteractiveChange

**Acción 6**

El usuario hace clic en Command2, que cierra el formulario.

Objeto	Evento
Form1	<a href="#">MouseMove</a>
Command2	<a href="#">MouseMove</a>
Text2	<a href="#">Valid</a>
Command2	<a href="#">When</a>
Text2	<a href="#">LostFocus</a>
Command2	<a href="#">GotFocus</a>
Command2	<a href="#">MouseDown(1, 0, 143, 128)</a>
Command2	<a href="#">MouseUp(1, 0, 143, 128)</a>
Command2	<a href="#">Click</a>
Command2	<a href="#">Valid</a>
Command2	<a href="#">When</a>

Cuando se cierra el formulario y se libera el objeto se producen estos eventos adicionales, en orden inverso a los eventos de la Acción 1.

Objetos	Evento
Form1	<a href="#">Destroy</a>
Command2	<a href="#">Destroy</a>
Command1	<a href="#">Destroy</a>
Text2	<a href="#">Destroy</a>
Text1	<a href="#">Destroy</a>
Form1	<a href="#">Unload</a>
DataEnvironment	<a href="#">AfterCloseTables</a>
DataEnvironment	<a href="#">Destroy</a>

## La secuencia de eventos de Visual FoxPro

La tabla siguiente muestra la secuencia de activación general de los eventos de Visual FoxPro. Se supone que la [propiedad AutoOpenTables](#) del [entorno de datos](#) está establecida a verdadero (.T.). Otros eventos pueden tener lugar en base a interacciones de usuario y a respuesta del sistema.

Objeto	Evento
DataEnvironment	<a href="#">BeforeOpenTables</a>
FormSet	<a href="#">Load</a>
Form	<a href="#">Load</a>
Cursores DataEnvironment	<a href="#">Init</a>
DataEnvironment	<a href="#">Init</a>
Objects <sup>1</sup>	<a href="#">Init</a>
Form	<a href="#">Init</a>
FormSet	<a href="#">Init</a>
FormSet	<a href="#">Activate</a>
Form	<a href="#">Activate</a>
Object1 <sup>2</sup>	<a href="#">When</a>
Form	<a href="#">GotFocus</a>
Object1	<a href="#">GotFocus</a>
Object1	<a href="#">Message</a>

Object1	<a href="#">Valid</a> <sup>3</sup>
Object1	<a href="#">LostFocus</a>
Object2 <sup>3</sup>	<a href="#">When</a>
Object2	<a href="#">GotFocus</a>
Object2	<a href="#">Message</a>
Object2	<a href="#">Valid</a> <sup>4</sup>
Object2	<a href="#">LostFocus</a>
Form	<a href="#">QueryUnload</a>
Form	<a href="#">Destroy</a>
Object <sup>5</sup>	<a href="#">Destroy</a>
Form	<a href="#">Unload</a>
FormSet	<a href="#">Unload</a>
DataEnvironment	<a href="#">AfterCloseTables</a>
DataEnvironment	<a href="#">Destroy</a>
Cursores DataEnvironment	<a href="#">Destroy</a>

<sup>1</sup> Para cada objeto, desde el objeto más interno hasta el contenedor más externo

<sup>2</sup> Primer objeto según el orden de tabulación

<sup>3</sup> Siguiente objeto que va a recibir el enfoque

<sup>4</sup> Cuando el objeto pierde el enfoque

<sup>5</sup> Para cada objeto, desde el contenedor más externo hasta el objeto más interno

## Asignar código a eventos

A menos que asocie código a un evento, no pasará nada cuando se produzca dicho evento. Casi nunca escribirá código para los eventos asociados a cualquier objeto de Visual FoxPro, pero querrá incorporar funcionalidad como respuesta a ciertos eventos clave de sus aplicaciones. Para agregar código que se va a ejecutar cuando se produzca un evento, utilice la ventana [Propiedades del Diseñador de formularios](#).

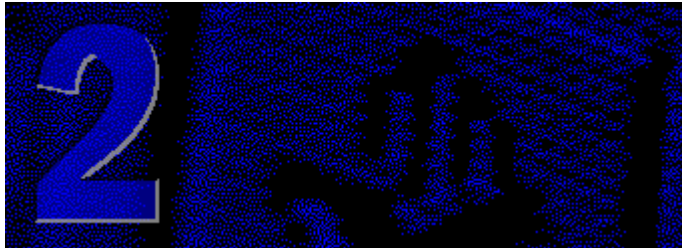
La secuencia de eventos afecta a dónde debe situar el código. Tenga en cuenta las siguientes sugerencias:

- El evento Init de todos los [controles](#) del [formulario](#) se ejecuta antes que el evento Init del formulario, por lo que puede incluir código en el evento Init del formulario para manipular cualquier control del formulario. Por ejemplo, si va a agregar controles para columnas de cuadrícula, podrá hacerlo en el evento Init del formulario.

- Si quiere que se procese parte del código siempre que el valor de [cuadro de lista](#), [cuadro combinado](#) o [casilla de verificación](#) cambia, asóciela al evento InteractiveChange. El evento Click puede no tener lugar o puede ser llamado incluso si el valor no ha cambiado.
- Cuando arrastra un [control](#), los otros eventos de *mouse* se paran. Por ejemplo, los eventos MouseUp y MouseMove no tienen lugar durante una operación de arrastrar y colocar.
- Los eventos Valid y When devuelven un valor, verdadero (.T.) de forma predeterminada. Si devuelve falso (.F.) ó 0 desde el evento When, el [control](#) no puede tener el enfoque. Si devuelve falso (.F.) ó 0 desde el evento Valid, el enfoque no puede abandonar el control.

Para obtener más información acerca del uso del Diseñador de formularios, consulte el capítulo 9, [Crear formularios](#). Para obtener información acerca de la codificación de clases y la forma de agregar código de eventos, consulte el capítulo 3, [Programación orientada a objetos](#).

# Manual del programador, Parte 2: Trabajar con datos



Para crear aplicaciones efectivas, debe comprender sus requisitos de datos y después diseñar sus bases de datos, índices y tablas de forma que resuelvan estas necesidades.

## Capítulo 5 [Diseñar una base de datos](#)

Aproveche la tecnología de bases de datos relacionales que ofrece Visual FoxPro creando bases de datos bien diseñadas.

## Capítulo 6 [Crear bases de datos](#)

Use bases de datos de Visual FoxPro para establecer relaciones entre tablas, hacer cumplir la integridad referencial y administrar datos locales y remotos.

## Capítulo 7 [Trabajar con tablas](#)

Asegúrese de que sus tablas tengan la estructura que necesita su aplicación. La elección de los tipos de datos y los índices es esencial para el éxito de su aplicación.

## Capítulo 8 [Crear vistas](#)

Use las vistas para tener acceso a registros de varias tablas y actualizarlos. Mediante vistas puede actualizar datos locales y remotos.

# Capítulo 5: Diseñar una base de datos

En Visual FoxPro se utilizan [bases de datos](#) para organizar y relacionar [tablas](#) y [vistas](#). La base de datos proporciona la arquitectura necesaria para almacenar los datos y cuenta además con otras ventajas. Al utilizar una base de datos, puede crear extensiones a nivel de tabla, tales como reglas a nivel de campo o de registro, valores predeterminados para los campos y [desencadenantes](#). También puede crear [procedimientos almacenados](#) y [relaciones persistentes](#) entre tablas. La base de datos puede utilizarse para tener acceso a conexiones con orígenes de datos remotos y también para crear [vistas](#) de tablas remotas y locales.

Este capítulo proporciona instrucciones para diseñar las tablas que se incluyen en una base de datos

de Visual FoxPro. Le muestra el diseño de base de datos de la base de datos de ejemplo Importadores Tasmanian y le proporciona diseños de base de datos de ejemplo adicionales. La base de datos de ejemplo Importadores Tasmanian, Tastrade.dbc, se encuentra en el directorio ...\\Samples\\Vfp98\\Tastrade\\Data de Visual Studio.

Para obtener información sobre la creación de bases de datos de Visual FoxPro después de diseñarlas, consulte el capítulo 6, [Crear bases de datos](#). Para obtener información sobre la creación de tablas de Visual FoxPro, consulte el capítulo 7, [Trabajar con tablas](#).

Este capítulo trata los siguientes temas:

- [Usar un proceso de diseño de base de datos](#)
- [Analizar los requisitos de datos](#)
- [Agrupar requisitos en tablas](#)
- [Determinar los campos que necesita](#)
- [Identificar relaciones](#)
- [Refinar el diseño](#)
- [Diagramas de base de datos de ejemplo](#)

## El proceso de diseño de una base de datos

Si usa un proceso de diseño de base de datos establecido, puede crear de forma rápida y efectiva una base de datos bien diseñada que le proporciona acceso conveniente a la información que desea. Con un diseño sólido tardará menos tiempo en construir la base de datos y obtendrá resultados más rápidos y precisos.

**Nota** Los términos "base de datos" y "tabla" no son sinónimos en Visual FoxPro. El término [base de datos](#) (archivo .dbc) se refiere a una base de datos relacional que almacena información sobre una o más [tablas](#) (archivos .dbf) o vistas.

La clave para entender el proceso de diseño de una base de datos radica en comprender la forma en que un sistema de administración de bases de datos relacionales, como Visual FoxPro, almacena los datos. Para ofrecer información de forma eficiente y precisa, Visual FoxPro debe tener almacenados los datos sobre distintos temas en tablas separadas. Por ejemplo, puede haber una tabla donde sólo se almacenen datos sobre empleados y otra tabla que sólo contenga datos de ventas.

Al organizar los datos de forma apropiada, proporciona flexibilidad a la base de datos y tiene la posibilidad de combinar y presentar información de muchas formas diferentes. Por ejemplo, podría imprimir informes que combinaran datos sobre empleados con datos sobre ventas.

**La separación de hechos en tablas agrega flexibilidad a su base de datos.**

Almacenar todos los datos de los empleados...

...y todos los datos de las ventas...

...en dos tablas diferentes.

Emp_id	Last_name	First_name	Title
5	Buchanan	Steven	Gerente de ventas
6	Suyama	Michael	Representante
7	King	Robert	Representante
8	Callahan	Laura	Coordinador ventas interno
9	Dodsworth	Anne	Representante
10	Hellstern	Albert	Director comercial
11	Smith	Tim	Operario de correo
12	Patterson		

Order_id	Cust_id	Emp_id	To_name	To_address
10000	FRANS	6	Franchi S.p.A.	Via Monte Bianco 34
10001	MEREP	8	Mère Paillarde	43 rue St. Laurent
10002	FOLKO	3	Folk och få HB	Åkergatan 24
10003	SIMOB	8	Simons bistro	Vinbæltet 34
10004	VAFFE	3	Vaffeljernet	Smagsloget 45
10005	WARTH	5	Wartian Herkku	Torikatu 38
10006	FRANS	8	Franchi S.p.A.	Via Monte Bianco 34
10007	MORGK	4	Morgenstern Gesundkost	Heerstr. 22

Al diseñar una base de datos, en primer lugar debe dividir la información que desea almacenar como temas distintos y después indicar a Visual FoxPro cómo se relacionan estos temas para que pueda recuperar la información correcta cuando sea necesario. Si mantiene la información en tablas separadas facilitará la organización y el mantenimiento de los datos y conseguirá aplicaciones de alto rendimiento.

A continuación se indican los pasos que hay que seguir en el proceso de diseño de una base de datos. Cada paso se trata con mayor detalle en las secciones restantes de este capítulo.

1. **Determinar el propósito de la base de datos** Este paso le ayudará a decidir los datos que desea que Visual FoxPro almacene.
2. **Determinar las tablas necesarias** Cuando ya conoce claramente el propósito de la base de datos, puede dividir la información en temas distintos, como "Empleados" o "Pedidos". Cada tema será una tabla de la base de datos.
3. **Determinar los campos necesarios** Tiene que decidir la información que desea incluir en cada tabla. Cada categoría de información de una tabla se denomina [campo](#) y se muestra en forma de columna al [examinar](#) la tabla. Por ejemplo, un campo de la tabla Empleado podría ser Apellidos y otro podría ser Fecha\_cont.
4. **Determinar las relaciones** Observe cada tabla y decida cómo se relacionan sus datos con los de las tablas restantes. Agregue campos a las tablas o cree tablas nuevas para clarificar las relaciones, si es necesario.
5. **Perfeccionar el diseño** Busque errores en el diseño. Cree las tablas y agregue algunos registros de datos de ejemplo. Vea si puede obtener los resultados que desea de sus tablas.



Haga los ajustes necesarios al diseño.

No se preocupe si se equivoca o si olvida algunos aspectos en el diseño inicial. Piense en él como en un borrador que podrá refinar posteriormente. Pruebe con datos de ejemplo y con prototipos de los [formularios](#) e informes. Con Visual FoxPro resulta sencillo modificar el diseño de la base de datos durante su creación. Sin embargo, es mucho más difícil modificar las tablas cuando ya están llenas de datos y se han generado formularios e informes. Por este motivo, debe asegurarse de tener un diseño sólido antes de llegar demasiado lejos en la programación de una aplicación.

## Analizar los requisitos de datos

El primer paso para diseñar una base de datos de Visual FoxPro es determinar el objetivo de la misma y cómo se va a utilizar. A partir de este punto, puede determinar sobre qué temas desea almacenar datos (las tablas) y qué datos necesita almacenar sobre cada tema (los campos de las tablas).

Consulte con las personas que vayan a utilizar la base de datos. Reflexione sobre las preguntas que desee que la base de datos responda. Haga un borrador de los informes que desee producir. Reúna los formularios que se utilicen actualmente para registrar los datos. Utilizará toda esta información en las etapas restantes del proceso de diseño.

### Ejemplo: hacer un seguimiento de ventas e inventario

Suponga que Importadores Tasmanian, una compañía de importación y exportación que vende productos alimenticios de todo el mundo, desea una base de datos que permita hacer un seguimiento de sus ventas y su inventario.

Comience por escribir una lista de preguntas que la base de datos tenga que responder. ¿Cuántas ventas de cierto producto logramos el mes pasado? ¿Dónde viven nuestros mejores clientes? ¿Quién es el proveedor de nuestro producto más vendido?

A continuación, reúna todos los formularios e informes que contengan la información que la base de datos deba producir. Actualmente, la compañía utiliza un informe impreso para hacer un seguimiento de los productos pedidos y un formulario para recibir nuevos pedidos. En la siguiente ilustración se muestran estos dos documentos.

**Los formularios y los informes presentan algunos requisitos de datos para su base de datos.**

<b>Productos pedidos</b>					
14-Jun-93					
Nom. categoria	Nom bre de producto	Existencias	Pedido	Nom bre del proveedor	Teléfono
Bebidas	16% del inventario total				
	Chai	39	0	Exotic Liquids	(71) 555-2222
	Chang	17	40	Exotic Liquids	(71) 555-2222
	Chartreuse Verte	49	0	Aux joyeux ecclésiastiques	(1) 800 83 00 48
	Côte de Blaye	17	0	Aux joyeux ecclésiastiques	(1) 800 83 00 48
	Gossum's Patriótica	20	0	Rafinesco Americano LTDA.	(11) 555 4 440
	Iphig Coffee	17	10	Lala Trading	55-8787
	Kahlúa88ri	57	0	Karlitz Oy	(93) 10954
	Langham Lumberjack Lager	52	0	Bigfoot Brewaris	(503) 555-9991
	Oatback Lager	15	10	Pardox, Ltd.	(83) 444-2343
	Rhinhorn Florentiner	125	0	Euro-par	(49) 992735
	Saguntia Ale	111	0	Labem mital grolmilita	(503) 555-9991
	Stakoya Stout	20	0	Bigfoot Brewaris	(503) 555-9991
		559	60		
Condimentos	16% del inventario total				
	Anisado Sabor	13	70	Exotic Liquids	(71) 555-2222

## Importadores Tasmanian

One Tasmanian Way, Twin Rivers, WA 6209  
Teléfono: (+61-8)-555-5555 Fax: (+61-8)-555-5559

---

**Facturar a:** Dame-a-muchallosite  
Lot Windok Ak  
Boxe 111111

**Entregar a:** Dame-a-muchallosite  
Lot Windok Ak  
Boxe 111111

**Fecha de pedido:** 14-jun-93

ID producto	Nombre de producto	Precio unitario	Cantidad	Precio acumulado
01	CHAI	01.25	39	4.88
02	CHANG	01.25	40	5.00
03	CHARTREUSE VERTE	01.25	49	5.12
04	COTE DE BLAYE	01.25	17	2.12

**Subtotal:**      17.24  
**Gastos de envío:**      0.00  
**Total:**      17.24

Importadores Tasmanian necesita también imprimir etiquetas de correo para los clientes, empleados y proveedores.

Una vez recopilada esta información, podrá continuar con el paso siguiente.

## Agrupar requisitos en tablas

Determinar las [tablas](#) de la base de datos puede ser la etapa más difícil del proceso de diseño, ya que los resultados que se esperan de la base de datos (los informes que desea imprimir, los formularios que desea utilizar, las preguntas que desea responder) no son necesariamente pistas sobre la estructura de las tablas que los producen. Esta información indica lo que se desea saber, pero no cómo clasificar la información en tablas.

Como ejemplo, observe el formulario de pedido anterior. En él se incluyen datos sobre el cliente (su dirección y su número de teléfono) junto con datos sobre el pedido. Este formulario contiene datos que se sabe que deben almacenarse en la base de datos, pero aparecerían problemas si se almacenasen los datos de los clientes en la misma tabla que los datos de los pedidos.

**El almacenamiento de información una vez reduce las posibilidades de error** Por ejemplo, si sólo usa una tabla para almacenar la información para un formulario de pedido, suponga que un cliente coloca tres pedidos diferentes. Podría agregar la dirección del cliente y el número de teléfono

a su base de datos tres veces, una para cada pedido. Pero esto multiplica la posibilidad de errores de entrada de datos.

**La tabla Customer almacena la información de dirección una sola vez.**

Customer\_orders

Order_id	Order_date	Company	Address
10927	01/27/95	La corne d'abondance	67, avenue de l'Europe
10972	02/15/95	La corne d'abondance	6, avenue De L' Europe
10973	02/15/95	La corne d'abondance	67, ave de l'Europe
10923	01/25/95	La maison d'Asie	1 rue Alsace-Lorraine
11051	03/21/95	La maison d'Asie	1 rue Alsace-Lorraine
10891	01/11/95	Lehmanns Marktstand	Magazinweg 7
10934	01/31/95	Lehmanns Marktstand	Magazinweg 7

¿Qué dirección es correcta?

Customer

Cust_id	Company	Address	City
LACDR	La corne d'abondance	67, avenue de l'Europe	Versailles
LAMAI	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse
LAUGB	Laughing Bacchus Wine Cellars	1900 Oak St	Vancouver
LAZYK	Lazy K Kountry Store	12 Orchestra Terrace	Walla Walla
LEHMS	Lehmanns Marktstand	Magazinweg 7	Frankfurt a.
LETSS	Let's Stop N Shop	87 Polk St, Suite 5	San Francis
LILA S	LILA Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Barquisimeta	

Para mayor precisión, almacene cada dato una sola vez.

Además, si el cliente cambia de domicilio, deberá aceptar información contradictoria o bien buscar y modificar cada uno de los registros de ventas del cliente en la tabla. Resulta mucho mejor crear una tabla Customer que almacene la dirección del cliente una sola vez en la base de datos. Así, si es necesario modificar los datos, sólo se hará una vez.

**Prevenir la eliminación de información valiosa** Suponga que un nuevo cliente hace un pedido y

que posteriormente lo cancela. Al eliminar el pedido de la tabla que contiene información de clientes y pedidos a la vez, se eliminaría también el nombre del cliente y su dirección. Sin embargo, es conveniente mantener a este nuevo cliente en la base de datos para poder enviarle el siguiente catálogo. Una vez más, es mejor conservar la información sobre el cliente en una tabla aparte. De esta forma se puede eliminar el pedido sin perder la información del cliente.

Estudie la información que desee obtener de la base de datos y divídala en temas fundamentales que desee mantener, como por ejemplo clientes, empleados, productos que usted vende, servicios ofrecidos, etc. Cada uno de estos temas es un candidato para una tabla independiente.

**Sugerencia** Una estrategia para dividir la información en tablas es observar primero los datos individuales y determinar de qué trata cada uno. Por ejemplo, en el formulario de pedidos de Importadores Tasmanian, la dirección del cliente no está asociada a la venta, sino al cliente. Este hecho sugiere la necesidad de una tabla independiente para los clientes. En el informe de productos pedidos, el número de teléfono del proveedor no está asociado al producto en existencias, sino al proveedor. Esto sugiere que es necesaria una tabla independiente para los proveedores.

### Ejemplo: diseñar tablas para la base de datos de Importadores Tasmanian

El formulario de pedidos y el informe de productos pedidos de Importadores Tasmanian contienen información sobre los temas siguientes:

- Empleados
- Clientes
- Proveedores
- Productos
- Pedidos

A partir de esta lista, puede esbozar un borrador de las tablas de la base de datos, así como apuntar algunos de los campos de cada una.

### Borrador de tablas y campos necesarios para la base de datos de Importadores Tasmanian

<b>BASE DE DATOS TASMANIAN</b>			
	Employees	Customers	Suppliers
	Name	Company Name	Company Name
	Address	Address	Address
		Contact	Contact
			Phone
	Products	Orders	
	Product Name	Order Date	
	Unit Price	Shipping Address	
	Units in Stock		
	Units on Order		

Aunque la base de datos terminada de Importadores Tasmanian contiene otras tablas, esta lista es un buen comienzo. Más adelante en este capítulo verá cómo agregar otras tablas para ajustar el diseño.

## Determinar los campos necesarios

Para determinar los campos de una tabla, decida qué necesita saber sobre las personas, cosas o eventos que se registran en ella. Puede considerar los campos como atributos de la tabla. Cada registro (o fila) de la tabla contiene el mismo conjunto de campos o atributos. Por ejemplo, un campo de dirección de una tabla de clientes contendrá las direcciones de los clientes. Cada registro de la tabla contendrá datos sobre un cliente y el campo de dirección contendrá la dirección de ese cliente determinado.

### Determinar los campos

A continuación se indican algunas sugerencias para determinar los campos:

**Relacione cada campo directamente con el tema de la tabla** Los campos que describan el tema de una tabla distinta pertenecerán a esa otra tabla. Posteriormente, al definir las relaciones entre las tablas, verá cómo combinar los datos de los campos de múltiples tablas. Por ahora, asegúrese de que cada campo de una tabla describe el tema de esa tabla. Si descubre que está repitiendo la misma información en varias tablas, será señal de que algunas tablas contienen campos innecesarios.

**No incluya datos derivados o calculados** En la mayoría de los casos no es conveniente almacenar en las tablas el resultado de cálculos. En lugar de ello, puede hacer que Visual FoxPro realice los cálculos cuando desee ver el resultado. Por ejemplo, el formulario de pedido mostrado anteriormente en este capítulo muestra el precio con descuento para cada línea de un pedido de la base de datos de Importadores Tasmanian. Sin embargo, no existe ningún campo de Subtotal Extended Price en ninguna tabla de Importadores Tasmanian, sino que la tabla Order\_Line\_Items incluye un campo de cantidad que almacena las unidades pedidas de cada producto individual, así como el precio unitario de cada producto pedido. Con esos datos, Visual FoxPro calcula el subtotal cada vez que se imprime un formulario de pedido y no es necesario almacenar en una tabla el subtotal propiamente dicho.

**Incluya toda la información necesaria** Es fácil pasar por alto información importante. Vuelva a la información que reunió en la primera etapa del proceso de diseño y observe los formularios e informes impresos para asegurarse de que toda la información que ha sido necesaria en el pasado está incluida en las tablas de Visual FoxPro o que puede derivarse de ellas. Piense en las preguntas que se formularán a Visual FoxPro. ¿Podrá Visual FoxPro encontrar todas las respuestas con la información que tiene en las tablas? ¿Ha identificado campos para almacenar datos únicos, como el Id. de cliente? ¿Qué tablas incluyen la información que desea combinar en un informe o en un formulario? Si desea más información sobre la forma de identificar campos clave y relacionar tablas, consulte las secciones [Uso de campos de clave principal](#) y [Determinación de las relaciones](#), más adelante en este mismo capítulo.

**Almacene la información en sus componentes lógicos más pequeños** Puede que sienta la tentación de utilizar un único campo para nombres completos o para nombres de productos, junto con su descripción. Si combina más de un tipo de información en un campo, después será difícil obtener datos individuales. Procure dividir la información en componentes lógicos; por ejemplo, cree campos distintos para el nombre y los apellidos, o para el nombre de producto, su categoría y su descripción.

Ejemplo: agregar campos a la tabla Products

Importadores Tasmanian vende especialidades alimentarias importadas de todo el mundo. Los empleados utilizan un informe Products On Order para hacer un seguimiento de los productos pedidos.


Informe para hacer un seguimiento del inventario de productos

Productos pedidos					
14-Jun-93					
Nom. categoría	Nombre del producto	Existencias	Pedido	Nombre del proveedor	Teléfono
Bebidas	100% del inventario total				
	Chai	39	0	Exotic Liquids	(71)555-2222
	Chang	17	40	Exotic Liquids	(71)555-2222
	Chartreuse Verte	49	0	Aux joyaux exotiques	(1)03.83.00.68
	Côte de Blaye	17	0	Aux joyaux exotiques	(1)03.83.00.68
	Guinness Extra Stout	20	0	Rafaelco, Americano, LTD.	(11)555-4440
	Ispah Coffee	17	10	Lola Trading	555-8787
	Lakka Lassi	37	0	Kashki Co.	(953)10954
	Laughing Lumberjack Lager	32	0	Bigfoot Beverages	(503)555-9931
	Oatmeal Lager	13	10	Pacific, Ltd.	(03)444-2343
	Robertson Elderberry	123	0	Pharmaceuticals	(049)992733
	Sagewatch Ale	111	0	Bigfoot Beverages	(503)555-9931
	Stout	20	0	Bigfoot Beverages	(503)555-9931
Condimentos	100% del inventario total				
	Aniseed Syrup	13	70	Exotic Liquids	(71)555-2222
	Chef Antoine's Cajun Seasoning	33	0	New Orleans Cajun Delights	(100)555-4822
	Chef Antoine's Gumbo Mix	0	0	New Orleans Cajun Delights	(100)555-4822
	Genoa Soy Sauce	39	0	Maryland	(04)431-7877

El informe indica que la tabla Products, que contiene datos sobre los productos vendidos, debe contener campos para el nombre del producto, las unidades en existencia y las unidades pedidas, entre otros. ¿Pero qué ocurre con los campos para el nombre y el número de teléfono del proveedor? Para producir el informe, Visual FoxPro necesitará saber a qué proveedor corresponde cada producto.

Borrador de la tabla Supplier con campos para el nombre y el número de teléfono

--	--



Products	Supplier
Product Name	Company Name
Unit Price	Address
Units in Stock	Contact
Units on Order	Phone

Puede resolver esto sin almacenar datos redundantes en sus tablas si crea una tabla Supplier con campos individuales para el nombre y el número de teléfono del suministrador. En el paso siguiente agregará un campo a la tabla Products para identificar la información de proveedor necesaria.

## Usar campos de clave principal

La eficacia de un sistema de administración de bases de datos relacionales como Visual FoxPro proviene de su capacidad de buscar y agrupar rápidamente información almacenada en tablas distintas. Para que Visual FoxPro funcione del modo más eficaz, cada tabla de la base de datos debe incluir un campo o un conjunto de campos que identifique de forma única cada registro individual almacenado en la tabla. A menudo, este campo será un número único de identificación, como un número de Id. de empleado o un número de serie. En la terminología de las bases de datos, esta información se denomina [clave principal](#) de la tabla. Visual FoxPro utiliza los campos de clave principal para asociar rápidamente datos de múltiples tablas y presentarlos agrupados.

Si ya dispone de un identificador único para una tabla, como por ejemplo una serie de números de producto que haya creado para identificar los productos en existencias, puede utilizarlo como clave principal de la tabla. Asegúrese de que los valores de este campo serán siempre distintos para cada registro, pues Visual FoxPro no permite valores duplicados en un campo de clave principal. Por ejemplo, no debe utilizar nombres de personas como clave principal, ya que no son únicos. Puede ocurrir fácilmente que haya en una misma tabla dos personas con el mismo nombre.

Al elegir campos de clave principal, tenga en cuenta lo siguiente:

- Visual FoxPro no permite valores duplicados ni [nulos](#) en los campos de clave principal. Por ello, no debe elegir una clave principal que pueda contener valores de este tipo.
- Puede utilizar el valor del campo de clave principal para buscar registros, por lo que no debe ser demasiado largo para recordarlo o escribirlo. Puede ser conveniente hacer que el valor tenga un número determinado de letras o dígitos, o que se encuentre en un cierto intervalo de valores.
- El tamaño de la clave principal afecta a la velocidad de las operaciones en la base de datos. Cuando cree campos de clave principal, utilice el menor tamaño que resulte adecuado a los valores que se almacenarán en ellos.

### Ejemplo: establecer la clave principal de la tabla Products

La clave principal de la tabla Products de Importadores Tasmanian contiene números de Id. de producto. Como cada número identifica a un producto distinto, no interesará que haya dos productos con el mismo número.

**La clave principal de la tabla Products es el campo Product\_id.**

Clave principal

Products			
Product_id	Prod_name	Eng_name	Unit_cost
1	Chai	Té Dharamsala	16.3
2	Chang	Cerveza tibetana Barley	17.2
3	Aniseed Syrup	Sirope de regaliz	9.1
4	Chef Anton's Cajun Seasoning	Especias Cajun del chef Anton	20.0
5	Chef Anton's Gumbo Mix	Mezcla Gumbo del chef Anton	19.4

No hay dos números de producto iguales...

...pero otros campos pueden contener valores repetidos

En algunos casos puede ser conveniente utilizar dos o más campos para formar la clave principal de una tabla. Por ejemplo, la tabla `Order_Line_Items` de la base de datos de Importadores Tasmanian utiliza dos campos como clave principal: `Order_id` y `Product_id`. En el paso siguiente se explicará la razón de esto.

## Determinar las relaciones

Ahora que ha dividido la información en [tablas](#) debe indicar a Visual FoxPro la manera de agruparla de nuevo de forma significativa. Por ejemplo, el [formulario](#) siguiente incluye información procedente de tablas distintas.

**El formulario Entrada de pedidos utiliza información de varias tablas.**

La información procede de la tabla Clientes...

...la tabla Productos...

...de la tabla Elementos del pedido...

**Entrada de pedidos**

**Cliente** Alfred's Futterkiste

**Enviar a** Alfred's Futterkiste

**Dirección** Obere Str. 57

**Ciudad** Berlin **Código postal** 12209

**Región** **País** Alemania

**Nº de pedido**

**Fecha de pedido**

**Información de entrega** Speedy Express

**Fecha de vencimiento**

Producto	Cantidad	Precio/Unidad	Extensión
Chef Anton's Gumbo Mix	8.000	14.9000	
Manjimup Dried Apples	12.000	37.1000	
Pâté chino's	20.000	16.8000	

**Notas** Haga clic con el botón derecho en la cuadrícula para el menú

**Último pedido**

**Crédito disponible:** \$6.300

**Ayuda**

**Subtotal de artículos** 10

**% de descuento**

**Pagado** ☒ **Gastos de envío**

**Total de factura**



Visual FoxPro es un sistema de administración de bases de datos relacionales. Esto significa que se almacenan datos relacionados en tablas separadas y que luego se definen entre las tablas relaciones que Visual FoxPro utilizará para buscar información asociada almacenada en la base de datos.

Por ejemplo, suponga que desea telefonar a un empleado para consultarle sobre una venta que ha realizado. Los números de teléfono de los empleados están registrados en la tabla Employee, y las ventas se registran en la tabla Orders. Al indicar a Visual FoxPro la venta en la que estamos interesados, éste puede buscar el número de teléfono utilizando la relación existente entre las dos tablas. El mecanismo funciona gracias a que Employee\_id, la [clave principal](#) de la tabla Employee, es también un campo de la tabla Orders. En la terminología de las bases de datos, el campo Employee\_id de la tabla Orders es una [clave externa](#), ya que se refiere a la clave principal de una tabla distinta, o externa.

### Campo Employee\_id como clave principal de la tabla Employee y como clave externa de la tabla Orders

Clave principal

Emp_id	Last_name	First_name	Title
5	Buchanan	Steven	Gerente de ventas
6	Suyama	Michael	Representante

El campo Employee ID aparece en ambas tablas.

Clave externa

Order_id	Cust_id	Emp_id	To_name	To_address
10005	WARTH	5	Wartian Herkkku	Torikatu 38
10006	FRANS	8	Franchi S.p.A.	Via Monte Bianco 34
10007	MORGK	4	Moraenstern Gesundkost	Heerstr. 22

Así, para establecer una relación entre dos tablas (tabla A y tabla B) se agrega la clave principal de una de ellas a la otra, de forma que aparezca en ambas. Pero, ¿cómo se decide de qué tabla se toma la clave principal? Para configurar correctamente las relaciones, primero es necesario determinar su naturaleza. Hay tres tipos de relaciones entre tablas:

- [Relaciones de uno a varios](#)
- [Relaciones de varios a varios](#)
- [Relaciones de uno a uno](#)

En el resto de esta sección se presenta un ejemplo de cada tipo de relación y se explica cómo diseñar

En el resto de esta sección se presenta un ejemplo de cada tipo de relación y se explica cómo diseñar las tablas de forma que Visual FoxPro pueda asociar sus datos correctamente. El propósito de cada ejemplo es explicar la forma de determinar las relaciones entre las tablas y cómo decidir los campos de las tablas que deben utilizarse para las relaciones. No se pretende describir el uso de la interfaz de Visual FoxPro para relacionar tablas.

**Ejemplo: crear una relación de uno a varios**

La relación de [uno a varios](#) es la más común en una base de datos relacional. En una relación de este tipo, un registro de la tabla A puede tener más de un registro coincidente en la tabla B, pero cada registro de la Tabla B tendrá como máximo un registro coincidente en la tabla A.

Por ejemplo, las tablas Supplier y Products de la base de datos de Importadores Tasmanian tienen una relación de uno a varios.

**Las tablas Category y Products representan una relación de uno a varios.**

Category_id	Nombre	Description	Picture_file	Picture
1	Bebidas	Memo	Memo	Gen
2	Condimentos	Memo	Memo	Gen
3	Repostería	Memo	Memo	Gen
4	Lácteos	Memo	Memo	Gen

Una categoría puede incluir varios productos.

Product_id	Supplier_id	Category_id	English_name
43	20	1	Café de Malasia
67	16	1	Cerveza Laughing Lumberjack
70	7	1	Cerveza Outback
75	12	1	Cerveza Rhönbräu
76	23	1	Licor Cloudberry
3	1	2	Sirope de regaliz

Para establecer la relación, agregue el campo o los campos que forman la [clave principal](#) del lado "uno" de la relación a la tabla del lado "varios". En el lado "uno" se utiliza una clave de índice principal o [candidato](#) y en el lado "varios" puede usarse una clave de [índice normal](#). En este caso, el campo Supplier\_id de la tabla Supplier se agregaría a la tabla Products, ya que *un* proveedor puede suministrar *varios* productos. Visual FoxPro utilizará el número de Id. de proveedor para localizar el proveedor correspondiente a cada producto.

Para obtener informacion sobre la creacion de claves de indice, consulte el capitulo 7, [Trabajar con tablas](#).

**Ejemplo: crear una relación de varios a varios**

En una relación de [varios a varios](#), un registro de la tabla A puede tener más de un registro coincidente en la tabla B y a un registro de la tabla B también puede corresponderle más de un registro de la tabla A. Este tipo de relación requiere cambios en el diseño de la base de datos para su correcta especificación en Visual FoxPro.

Para detectar relaciones de varios a varios entre las tablas es importante observar los dos sentidos de cada relación. Por ejemplo, considere la relación entre pedidos y productos de la base de datos de Importadores Tasmanian. Un pedido puede incluir más de un producto, de forma que para cada registro de la tabla Orders puede haber varios registros en la tabla Products. Pero eso no es todo: cada producto puede aparecer en varios pedidos, por lo que para cada registro de la tabla Products puede haber varios registros en la tabla Orders.

**Las tablas Orders y Products representan una relación de varios a varios.**

Entrada de pedidos

Cliente

Alfred's Futterkiste

Enviar a

Alfred's Futterkiste

Dirección

Obere Str. 57

Ciudad

Berlin

Código postal

12209

Región

País

Alemania

Nº de pedido

Fecha de pedido

21/09

Información de entrega

Speedy Express

Fecha de vencimiento

18/09

Producto	Cantidad	Precio/Unidad	Extensión
Chef Anton's Gumbo Mix	8.000	14.9000	119.200
Manjimup Dried Apples	12.000	37.1000	445.200
Pâté chinois			

Notas Haga clic

Entrada de pedidos

Cliente

Centro comercial Moctezuma

Enviar a

Centro comercial Moctezuma

Dirección

Sierras de Granada 9993

Ciudad

México D.F.

Código postal

05022

Región

País

México

Nº de

Fecha de

Información de

Speedy Expre

Fecha de ven

Producto	Cantidad	Precio/Unidad
Manjimup Dried Apples	1.000	14.9000
Filn Mix	8.000	37.1000

Un pedido puede tener varios productos.

Cada producto puede

aparecer en varios pedidos.

Perth Pasties	12.000	16.8000
---------------	--------	---------

**Notas** Haga clic con el botón derecho en la cuadrícula para el menú

Último pedido

Crédito disponible: \$1.000

Pagado ☒ Gastos

Ayuda

Subtotal de

10 % de de

Total de

Los temas de las dos tablas (pedidos y productos) tienen una relación de [varios a varios](#). Esto supone un problema en el diseño de la base de datos. Para entender este problema, imagine lo que ocurriría si intentase establecer la relación entre las dos tablas agregando el campo Product\_id a la tabla Orders. Para tener más de un producto por pedido, necesitaría más de un registro en la tabla Orders por cada pedido real y tendría que repetir una y otra vez la información para cada registro relacionado con el mismo pedido, lo que supone un diseño poco eficaz que puede llevar a una falta de exactitud en los datos. El mismo problema aparece si se incluye el campo Order\_id en la tabla Products: sería necesario más de un registro en la tabla Products para cada producto real. ¿Cómo puede resolverse este problema?

La respuesta consiste en crear una tercera tabla que divida la relación de varios a varios en dos relaciones de [uno a varios](#). Esta tercera tabla se denomina [tabla de unión](#), ya que actúa como conexión entre las dos tablas y en ella se incluirá la [clave principal](#) de cada una de las dos tablas anteriores.

**La tabla Order\_Line\_Items crea un vínculo de uno a varios entre Orders y Products.**

Clave principal de la tabla Pedidos

Clave principal de la tabla Productos

Line_no	Order_id	Product_id	Unit_price	Quantity
1	10000	17	27.0000	4.000
1	10001	25	9.8000	42.000
2	10001	40	12.8000	36.000
3	10001	59	38.5000	24.000
4	10001	64	23.0000	12.000
1	10002	31	8.0000	15.000
2	10002	39	12.6000	19.000
3	10002	71	15.0000	15.000
1	10003	18	13.7000	12.000

Información contenida en ambas tablas Pedidos y Productos.

Una [tabla de conexión](#) podría guardar únicamente las dos [claves principales](#) de las tablas que vincula o, como en la tabla Order\_Line\_Items, la tabla de conexión podría contener información adicional.

Cada [registro](#) de la tabla Order\_Line\_Items representa una línea de un pedido. La [clave principal](#) de la tabla Order\_Line\_Items consta de dos [campos](#): las [claves externas](#) de las tablas Orders y Products. El campo Order\_id por sí solo no sirve como clave principal de la tabla, ya que un mismo pedido puede tener varias líneas de productos. El Id. de pedido se repite para cada línea de un pedido, por lo que el campo no contendrá valores únicos. El campo Product\_id tampoco funcionará, ya que un mismo producto puede aparecer en varios pedidos diferentes. Sin embargo, los dos campos juntos en la [tabla de unión](#) producirán siempre un valor único para cada registro. La tabla de conexión no requiere su

propia [clave principal](#).

En la base de datos de Importadores Tasmanian, las tablas Orders y Products no se relacionan directamente, sino que lo hacen a través de la tabla Order\_Line\_Items. La relación de varios a varios entre los pedidos y los productos se representa en la base de datos mediante dos relaciones de uno a varios:

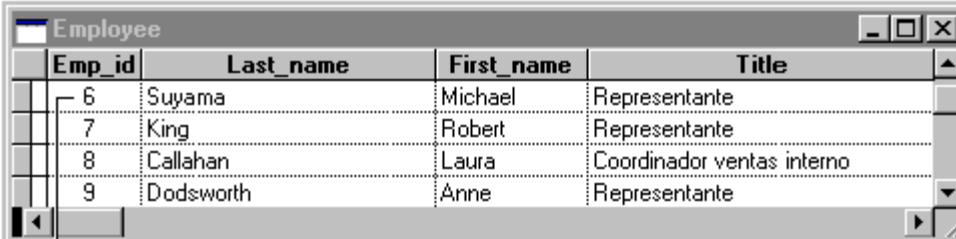
- Las tablas Orders y Order\_Line\_Items tienen una relación de uno a varios. Cada pedido puede tener más de una línea de producto, pero cada línea sólo estará conectada a un pedido.
- Las tablas Products y Order\_Line\_Items tienen una relación de uno a varios. Cada producto puede tener más de una línea asociada, pero cada línea sólo se refiere a un producto.

### Ejemplo: crear una relación de uno a uno

En una relación de [uno a uno](#), cada registro de la tabla A no puede tener más de un registro coincidente en la tabla B y a cada registro de la tabla B no puede corresponderle más de un registro coincidente en la tabla A. Este tipo de relación es poco habitual y puede indicar la necesidad de una modificación en el diseño de la base de datos.

Las relaciones de uno a uno entre dos tablas son poco usuales ya que, en muchos casos, la información de ambas tablas puede combinarse de forma sencilla en una tabla única. Por ejemplo, suponga que se crea una tabla, llamada Jugadores de ping-pong, para hacer un seguimiento de un torneo benéfico de ping-pong en Importadores Tasmanian. Como todos los jugadores son empleados, esta tabla tendrá una relación de uno a uno con la tabla Employee de la base de datos de Importadores Tasmanian.

**Las tablas Employee y Jugadores\_de\_ping\_pong representan una relación de uno a uno.**



Emp_id	Last_name	First_name	Title
6	Suyama	Michael	Representante
7	King	Robert	Representante
8	Callahan	Laura	Coordinador ventas interno
9	Dodsworth	Anne	Representante

— Cada jugador de ping-pong player tiene un registro coincidente en la tabla Empleados.

Emp_id	Player_nickname	Preferred_date	Skill_level	Pledge
6	Ace	07/07/96	1	2.0000
7	King John	09/07/96	2	2.0000
8	Calmeister	07/07/96	1	2.0000
9	Slammin' Nan	07/07/96	1	2.0000

Este conjunto de valores es un subconjunto del campo Employee\_id field en la tabla Empleados.

Sería posible agregar todos los campos de la tabla Jugadores de ping-pong a la tabla Employee. Sin embargo, la tabla Jugadores de ping-pong se utiliza para un acontecimiento puntual y la información dejará de ser necesaria cuando éste termine. Además, no todos los empleados juegan al Ping-pong, de modo que si los campos se incorporasen a la tabla Employee, estarían vacíos en muchos registros. Por estas razones tiene sentido crear una tabla distinta.

Cuando detecte la necesidad de una relación de [uno a uno](#) en una base de datos, considere primero si es conveniente agrupar la información en una sola tabla. Por ejemplo, en la tabla Employee, un empleado puede tener su propio director, que también es un empleado. Puede agregar un campo para el número de identificación del director. Para reunir la información más tarde, puede usar una autocombinación en la [consulta](#) o [vista](#). No necesita una tabla independiente para resolver la relación uno a uno. Si no quiere hacerlo por alguna razón, a continuación se muestra cómo configurar la relación uno a uno entre las dos tablas:

- Si las dos tablas tratan del mismo tema, probablemente podrá definir la relación utilizando el mismo campo de [clave principal](#) en ambas.
- Si las dos tablas tratan de temas distintos y tienen claves principales diferentes, elija una de ellas e incluya su campo de clave principal en la otra como [clave externa](#).

## Perfeccionar el diseño

Cuando ya se tienen las tablas, los campos y las relaciones necesarios, es el momento de estudiar el diseño y detectar los posibles fallos que puedan quedar.

Puede encontrar varios fallos mientras está diseñando su base de datos. Estos problemas comunes pueden hacer que sus datos sean más difíciles de usar que de mantener:

- ¿Tiene una [tabla](#) con un gran número de [campos](#) que no relaciona con el mismo asunto? Por ejemplo, una tabla puede contener campos que pertenecen a sus clientes así como información de ventas. Intente asegurarse de que cada tabla contiene datos sobre un solo tema.
- ¿Tiene [campos](#) que se dejan intencionadamente en blanco en muchos [registros](#) porque no son aplicables a dichos registros? Esto generalmente significa que los campos pertenecen a otra [tabla](#).
- ¿Tiene un gran número de [tablas](#), muchas de las cuales contienen los mismos [campos](#)? Por ejemplo, tiene tablas distintas para las ventas de enero y las de febrero, o para clientes locales y clientes remotos, en las que almacena el mismo tipo de información. Intente consolidar toda la información perteneciente a un solo asunto en una sola tabla. Es posible que tenga que agregar un campo adicional, por ejemplo, para identificar la fecha de venta.

Cree sus tablas, especifique relaciones entre las tablas e introduzca algunos registros de datos en cada tabla. Vea si puede usar la base de datos para obtener las respuestas que desea. Cree borradores de sus formularios e informes y vea si muestra los datos que espera. Busque duplicados de datos innecesarios y elimínelos.

Al probar la base de datos inicial, probablemente descubrirá posibles mejoras. A continuación se indican algunos puntos que debe comprobar:

- ¿Ha olvidado algún [campo](#)? ¿Existe información necesaria que no se ha incluido? Si es así, ¿corresponde a las [tablas](#) existentes? Si la información trata de un tema distinto, puede ser necesario crear otra tabla.
- ¿Ha elegido una buena [clave principal](#) para cada [tabla](#)? Si la va a utilizar para buscar [registros](#) específicos, ¿es fácil de recordar y escribir? Asegúrese de que no se dará el caso de tener que introducir en un campo de clave principal un valor que duplique al de otro registro.
- ¿Tiene que introducir información repetida una y otra vez en una de las [tablas](#)? Si es así, probablemente deba dividirla en dos tablas con una relación de [uno a varios](#).
- ¿Tiene [tablas](#) con muchos [campos](#), un número de [registros](#) limitado y numerosos campos vacíos en algunos registros? En caso afirmativo, considere la posibilidad de volver a diseñar la tabla de forma que tenga menos campos y más registros.

A medida que identifique los cambios que desea realizar, puede modificar las tablas y los campos para reflejar las mejoras en el diseño. Si desea información sobre cómo modificar las tablas, consulte el capítulo 7, [Trabajar con tablas](#).

### Ejemplo: perfeccionar la tabla Products

Cada producto en existencias de Importadores Tasmanian pertenece a una categoría general, como Bebidas, Condimentos o Pescados. La [tabla](#) Products podría contener un [campo](#) para mostrar la categoría de cada producto.

#### Tabla Products con un campo Category\_name

Product_id	Product_name	Units_in_stock	Category_name
1	Chai	39.000	Productos Agrícolas
2	Chang	17.000	Bebidas
3	Aniseed Syrup	13.000	Condimentos
4	Chef Anton's Cajun Seasoning	53.000	Bebidas
5	Chef Anton's Gumbo Mix	0.000	Condimentos
6	Grandma's Boysenberry Spread	120.000	Condimentos
7	Uncle Bob's Organic Dried Pears	15.000	Productos Agrícolas
8	Northwoods Cranberry Sauce	6.000	Condimentos

Suponga que al examinar y perfeccionar la [base de datos](#) Importadores Tasmanian decide almacenar una descripción de la categoría junto con su nombre. Si agregase un campo Category Description a la tabla Products, tendría que repetir la descripción de una categoría para cada producto que perteneciese a ella, lo que no es una buena solución.

Un método más adecuado es hacer de la categoría un nuevo tema del que tenga que ocuparse la base

Un método más adecuado es hacer de la categoría un dato de clave que ocupase la parte de datos, con su propia tabla y su propia [clave principal](#). De esta forma puede agregar la clave principal de la tabla Category a la tabla Products como [clave externa](#).

La tabla Category proporciona un lugar donde almacenar de forma eficaz la información sobre categorías.

Category_id	Nombre	Description	Picture_file	Picture
2	Condimentos	Memo	Memo	Gen
3	Repostería	Memo	Memo	Gen
4	Lácteos	Memo	Memo	Gen
5	Granos/Cereales	Memo	Memo	Gen
6	Carnes	Memo	Memo	Gen
7				
8				

Product_id	Product_name	English_name	Category_id
1	Chai	Té Dharamsala	7
2	Chang	Cervez tibetana Barley	1
3	Aniseed Syrup	Sirope de regaliz	2
4	Chef Anton's Cajun Seasoning	Especias Cajun del chef Anton	1
5	Chef Anton's Gumbo Mix	Mezcla Gumbo del chef Anton	2
6	Grandma's Boysenberry Spread	Mermelada de grosellas de la abuela	2
7	Uncle Bob's Organic Dried Pears	Peras secas orgánicas del tío Bob	7
8	Northwoods Cranberry Sauce	Salsa de arándanos Northwoods	2
9	Mishi Kobe Niku	Buey Mishi Kobe	6

Las tablas Category y Products tienen una relación de [uno a varios](#): una categoría puede tener más de un producto, pero cada producto sólo pertenece a una categoría.

## Ejemplos de diagramas de base de datos

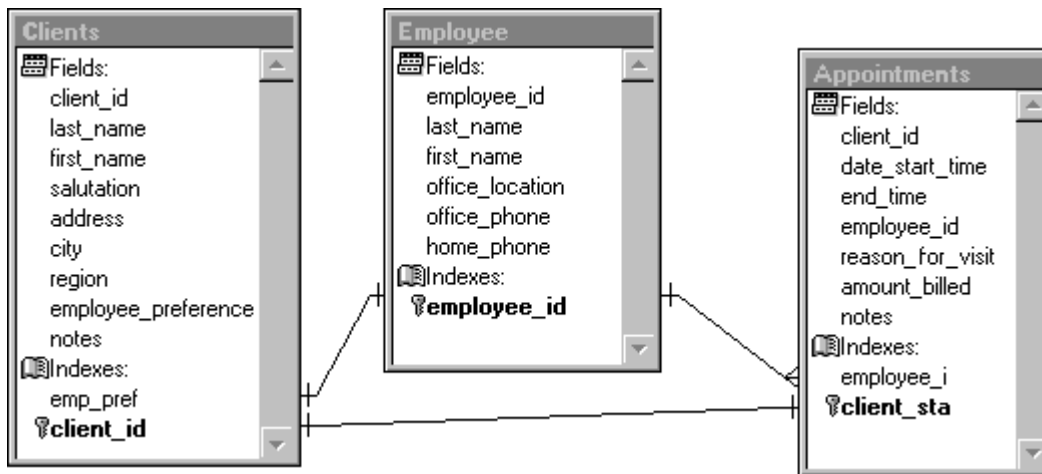
Los diagramas de bases de datos de esta sección pueden sugerirle ideas para el diseño de su propia base de datos. Estas bases no se incluyen con Visual FoxPro; sólo se ofrecen aquí como ejemplo de los tipos de [bases de datos](#) y [tablas](#) que puede crear.

### Base de datos de citas

Esta estructura de base de datos almacena las citas de una oficina profesional, y puede modificarse fácilmente para que la usen médicos, odontólogos, abogados o contables. La tabla Citas cuenta con una [clave principal](#) de múltiples campos para identificar cada cita de forma única. Esta clave principal, el índice 'client\_sta', se crea indexando por una [expresión](#) que combina los campos cliente\_id y date\_start.

### Ejemplo de una base de datos de citas

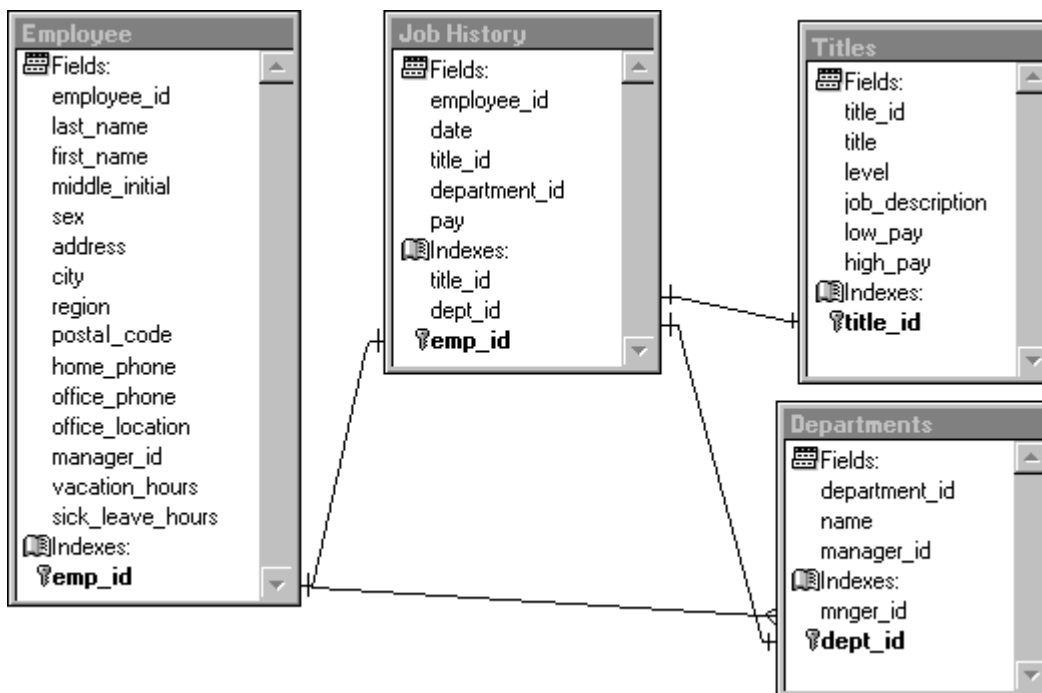




## Base de datos de personal

Esta estructura de base de datos almacena información sobre recursos humanos. La tabla Historial de trabajo almacena información sobre cada contrato o ascenso, de forma que puede contener varios registros por cada empleado.

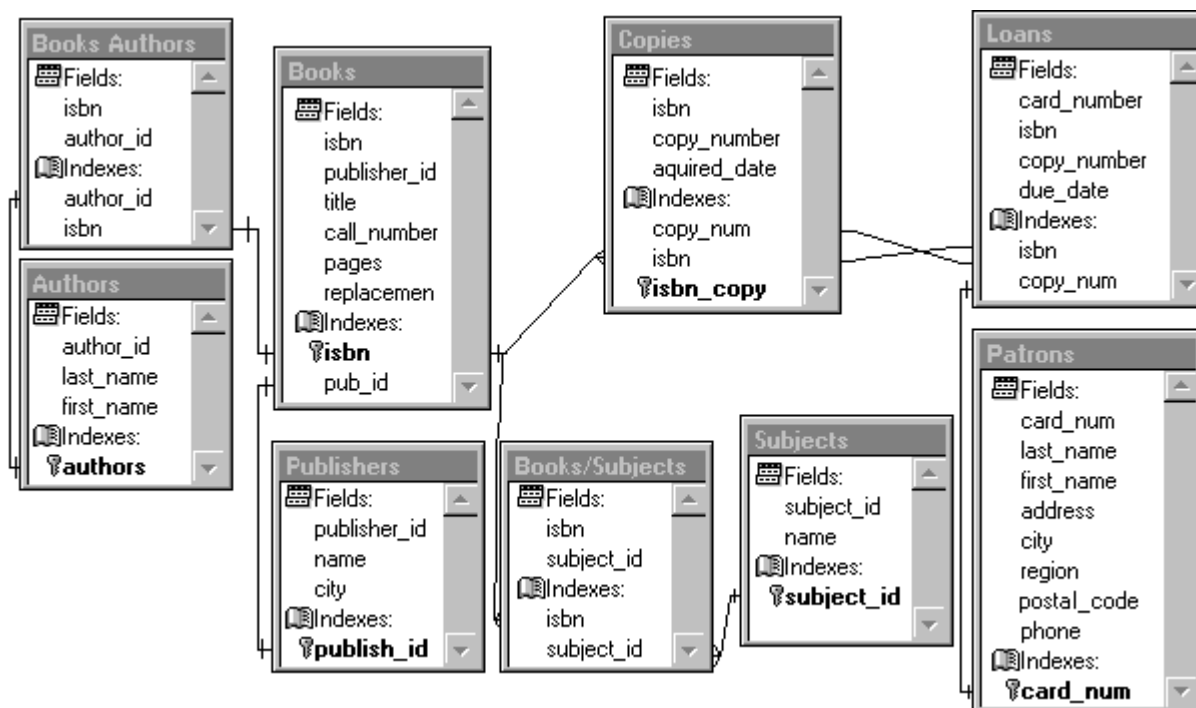
### Ejemplo de una base de datos de personal



## Base de datos de biblioteca

Esta base de datos almacena información sobre los libros de una biblioteca y los préstamos a los lectores. Observe la relación de [varios a varios](#) entre las tablas Libros y Autores y entre las tablas Libros y Temas.

### Ejemplo de una base de datos de biblioteca



## Capítulo 6: Crear bases de datos

Tras haber diseñado una [base de datos](#), puede generarla a través de la interfaz o mediante el lenguaje. Es posible que quiera agregar [tablas](#) existentes a la base de datos y, a continuación, modificarlas para aprovechar las características de [diccionario de datos](#) de Visual FoxPro. Si está trabajando en un proyecto en el [Administrador de proyectos](#), puede agregar las tablas a medida que las crea.

Para obtener más información sobre la forma de crear una base de datos para un entorno de múltiples usuarios, consulte el capítulo 17, [Programar para acceso compartido](#).

En este capítulo se tratan los temas siguientes:

- [Crear una base de datos](#)
- [Ver y modificar la arquitectura de una base de datos](#)
- [Administrar una base de datos](#)
- [Hacer referencia a múltiples bases de datos](#)
- [Controlar errores de bases de datos](#)

## Crear una base de datos

Al crear una [base de datos](#), usted reúne [tablas](#) en un conjunto y aprovecha las características de [diccionario de datos](#).

Un diccionario de datos proporciona mayor flexibilidad al diseñar y modificar la base de datos y le libera de tener que escribir código para crear [validación](#) a nivel de campos y a nivel de filas o para asegurar la unicidad de valores en campos de clave principal. El diccionario de datos de Visual FoxPro le permite crear o especificar:

- Claves [principales](#) y candidatas.
- [Relaciones persistentes](#) entre tablas de bases de datos.
- [Nombres largos](#) para tablas y campos.
- [Títulos](#) de campos que aparecen como encabezados en ventanas [Examinar](#) y en columnas de cuadrícula.
- Valores predeterminados en campos.
- La [clase de control](#) predeterminada usada en [formularios](#).
- [Máscaras de entrada](#) y formatos de presentación para campos.
- Reglas a [nivel de campo](#) y reglas a [nivel de registro](#).
- [Desencadenantes](#).
- [Procedimientos almacenados](#).
- [Conexiones](#) a orígenes de datos remotos.
- Vistas [locales](#) y [remotas](#).
- Comentarios para cada [campo](#), [tabla](#) y [base de datos](#).

Algunas características del diccionario de datos, como nombres de campo largos, claves principales y candidatas, valores predeterminados, reglas a nivel de campo y a nivel de registro y desencadenantes se almacenan en el archivo .dbc, pero se crean como parte del proceso de generación de una tabla o una vista. Para obtener más información, consulte el capítulo 7, [Trabajar con tablas](#), y el capítulo 8, [Crear vistas](#).

## Reunir tablas en una base de datos

Para reunir tablas en una base de datos tiene que crear un contenedor de base de datos para guardar todos los [objetos](#) como [vistas](#), [conexiones](#) y [procedimientos almacenados](#) asociados a las tablas que forman la base de datos.

### Para crear una nueva base de datos

- En el [Administrador de proyectos](#), seleccione la ficha **Datos**, luego **Bases de datos** y finalmente **Nuevo**.

–O bien–

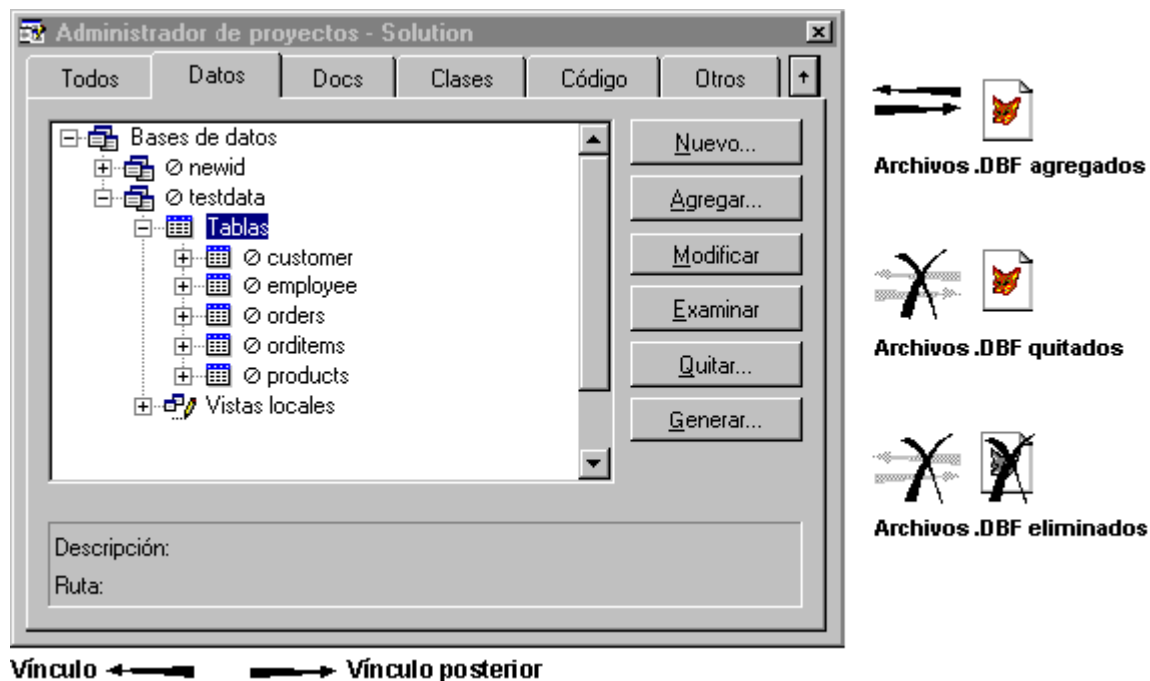
- Utilice el comando [CREATE DATABASE](#).

Por ejemplo, el código siguiente crea y abre de forma exclusiva una nueva base de datos, llamada Ejemplo:

```
CREATE DATABASE Ejemplo
```

Al crear una nueva base de datos, ésta estará vacía, pues no contendrá tablas asociadas ni ningún otro objeto. Al agregar una tabla se crean vínculos entre el archivo de tabla y el contenedor de la base de datos. La información de vínculo sobre una tabla almacenada en la base de datos es un vínculo posterior. La información de vínculo almacenada en la tabla sobre el contenedor de base de datos es el vínculo anterior.

**Los vínculos especifican las asociaciones entre un contenedor de bases de datos y las tablas.**



Para trabajar con una base de datos y sus objetos mediante programación, puede utilizar los [comandos](#) y [funciones](#) siguientes.

### Comandos y funciones para manipular bases de datos y sus objetos

<a href="#">ADATABASES()</a>	<a href="#">CREATE VIEW</a>	<a href="#">MODIFY CONNECTION</a>
<a href="#">ADBOBJECTS()</a>	<a href="#">DBC()</a>	<a href="#">MODIFY DATABASE</a>
<a href="#">ADD TABLE</a>	<a href="#">DBGETPROP()</a>	<a href="#">MODIFY PROCEDURE</a>
<a href="#">ALTER TABLE</a>	<a href="#">DBSETPROP()</a>	<a href="#">MODIFY STRUCTURE</a>
<a href="#">APPEND PROCEDURES</a>	<a href="#">DELETE CONNECTION</a>	<a href="#">MODIFY VIEW</a>

<a href="#">APPEND PROCEDURES</a>	<a href="#">DELETE CONNECTION</a>	<a href="#">MODIFY VIEW</a>
<a href="#">CLOSE DATABASE</a>	<a href="#">DELETE DATABASE</a>	<a href="#">OPEN DATABASE</a>
<a href="#">COPY PROCEDURES</a>	<a href="#">DELETE VIEW</a>	<a href="#">PACK DATABASE</a>
<a href="#">CREATE CONNECTION</a>	<a href="#">DISPLAY DATABASE</a>	<a href="#">RENAME TABLE</a>
<a href="#">CREATE DATABASE</a>	<a href="#">DROP TABLE</a>	<a href="#">REMOVE TABLE</a>
<a href="#">CREATE SQL VIEW</a>	<a href="#">INDBC()</a>	<a href="#">SET DATABASE</a>
<a href="#">CREATE TABLE</a>	<a href="#">LIST DATABASE</a>	<a href="#">VALIDATE DATABASE</a>

## Agregar tablas a una base de datos

Cada tabla de Visual FoxPro puede existir en uno de dos estados: como [tabla libre](#), que es un archivo .DBF no asociado a ninguna base de datos, o como [tabla de base de datos](#), que es un archivo .dbf asociado a una base de datos. Las tablas asociadas a una base de datos pueden tener [propiedades](#) que no tienen las tablas libres, como las [reglas a nivel de campo](#) y a [nivel de registro](#), los [desencadenantes](#) y las [relaciones persistentes](#).

Las tablas se asocian a una base de datos al crearlas desde dentro de una base de datos abierta o al agregar tablas existentes a una base de datos. Si desea información sobre la forma de crear tablas nuevas, consulte el capítulo 7, [Trabajar con tablas](#).

### Para agregar una tabla libre a una base de datos

- En el [Administrador de proyectos](#), seleccione **Tablas** en la ficha **Todos** o en la ficha **Datos** y, a continuación, elija **Agregar**.

-O bien-

- Utilice el comando [ADD TABLE](#).

Por ejemplo, el código siguiente abre la base de datos `testdata` y le agrega la tabla `orditems`:

```
OPEN DATABASE testdata
ADD TABLE orditems
```

Para que una tabla libre existente pase a formar parte de una base de datos, debe agregarla explícitamente. La modificación de la estructura de una tabla libre no hace que Visual FoxPro la agregue a una base de datos, incluso cuando la base de datos se encuentre abierta al utilizar el comando [MODIFY STRUCTURE](#).

### Usar tablas libres

Una tabla determinada sólo se puede asociar a una base de datos. Sin embargo, es posible utilizar los datos de un archivo .dbf existente sin necesidad de incorporarlo a una base de datos.

**Para tener acceso a una tabla de otra base de datos**

#### Para tener acceso a una tabla de otra base de datos

- Cree una [vista](#) en la base de datos actual que haga referencia a la tabla en cuestión.

-O bien-

- Tenga acceso a la tabla con el comando [USE](#) y el símbolo “!”.

El símbolo “!” permite hacer referencia a una tabla de una base de datos distinta de la actual. Por ejemplo, si desea examinar la tabla `orditems` de la base de datos `testdata`, puede escribir:

```
USE testdata!orditems  
BROWSE
```

En el ejemplo anterior, la base de datos `testdata` se abre automáticamente al utilizar el comando `USE`, pero Visual FoxPro no la establece como base de datos actual. Las bases de datos abiertas automáticamente como en el ejemplo anterior se cierran también automáticamente al cerrar la tabla, a menos que se abra explícitamente la base de datos antes de cerrar la tabla.

Si desea información sobre el uso de una vista para tener acceso a información externa a la base de datos actual, consulte el capítulo 8, [Crear vistas](#).

#### Quitar una tabla de una base de datos

Al agregar una tabla a una base de datos, Visual FoxPro modifica el registro de encabezado del archivo para documentar la ruta de acceso y el nombre de archivo de la base de datos a la que ahora pertenece la tabla. Esta información de ruta y nombre de archivo se denomina [vínculo anterior](#), ya que vincula la tabla a la base de datos a la que pertenece. El proceso de quitar una tabla de una base de datos no solamente suprime la tabla y la información de [diccionario de datos](#) asociada del archivo de la base de datos, sino que también actualiza la información de vínculo anterior para reflejar el nuevo estado de la tabla como [libre](#).

Para quitar una tabla de una base de datos puede utilizar la interfaz o bien el comando [REMOVE TABLE](#). Al quitar la tabla de la base de datos, puede elegir también eliminar físicamente del disco el archivo de la tabla.

#### Para quitar una tabla de una base de datos

- En el [Administrador de proyectos](#), seleccione el nombre de la tabla y luego elija **Quitar**.

-O bien-

- En el [Diseñador de bases de datos](#), seleccione la tabla y elija **Quitar** en el menú **Base de datos**.

-O bien-

- Utilice el comando [REMOVE TABLE](#).

Por ejemplo, el código siguiente abre la base de datos `testdata` y quita la tabla `orditems`.

Por ejemplo, el código siguiente abre la base de datos `testdata` y quita la tabla `orditems`:

```
OPEN DATABASE testdata
REMOVE TABLE orditems
```

Al quitar una tabla de una base de datos, el archivo de tabla no se elimina automáticamente. Si desea quitar la tabla de la base de datos y además eliminar del disco su archivo `.dbf`, utilice la cláusula `DELETE` del comando `REMOVE TABLE` o el comando [DROP TABLE](#). Por ejemplo, el código siguiente abre la base de datos `testdata` y elimina la tabla `orditems` del disco:

```
OPEN DATABASE testdata
REMOVE TABLE orditems DELETE
```

El código siguiente también abre la base de datos `testdata` y elimina la tabla `orditems` sin mover una copia a la Papelera de reciclaje de Windows:

```
OPEN DATABASE testdata
DROP TABLE orditems NORECYCLE
```

## Actualizar vínculos de tablas y bases de datos

Si mueve archivos de base de datos (`.dbf`, `.dct` y `.dcx`) o una tabla asociada con la base de datos, las rutas relativas cambian y pueden romper los vínculos anteriores y posteriores que Visual FoxPro usa para asociar archivos de bases de datos y de tablas:

- El [vínculo anterior](#) vincula la tabla con la base de datos propietaria de la tabla. Está formado por la ruta relativa y el nombre de archivo para el archivo `.dbc` asociado a la tabla, y está almacenado en el encabezado del archivo de tabla de Visual FoxPro (`.dbf`).
- El [vínculo posterior](#) dice a la base de datos qué tablas le pertenecen. Los vínculos posteriores están almacenados en el archivo de base de datos (`.dbc`) y están formados por la ruta relativa y el nombre de archivo para cada archivo de tabla asociado.

Puede restablecer vínculos y actualizar la información de ruta relativa de forma que refleje la nueva ubicación del archivo.

## Para actualizar vínculos después de mover una tabla o una base de datos

- Use la cláusula `RECOVER` del comando [VALIDATE DATABASE](#).

Por ejemplo, el código siguiente abre la base de datos `testdata` y muestra cuadros de diálogo que le permiten buscar tablas que no están en las ubicaciones contenidas en la base de datos:

```
OPEN DATABASE testdata
VALIDATE DATABASE RECOVER
```

**Sugerencia** Si quiere usar una tabla sin perder tiempo restableciendo los vínculos para todas las tablas de la base de datos, puede abrir la tabla con el comando [USE](#). Visual FoxPro muestra el [cuadro de diálogo Abrir](#) que le permitirá buscar la base de datos propietaria o eliminar los vínculos.

Para obtener información sobre la eliminación del vínculo anterior de una tabla cuya base de datos

propietaria se ha eliminado accidentalmente del disco, vea [FREE TABLE](#).

## Crear relaciones persistentes

Puede crear [relaciones persistentes](#) entre las [tablas](#) de una [base de datos](#). Las relaciones persistentes son relaciones entre tablas de una base de datos que se almacenan en el archivo de la base de datos y tienen las características siguientes:

- Se utilizan automáticamente como condiciones de combinación predeterminadas en los [Diseñadores de consultas y vistas](#).
- Se representan en el [Diseñador de bases de datos](#) como líneas que relacionan los [índices](#) de las tablas.
- Aparecen en el [Diseñador de entorno de datos](#) como relaciones predeterminadas para los formularios e informes.
- Se utilizan para almacenar información de [integridad referencial](#).

A diferencia de las relaciones [temporales](#) creadas con el comando [SET RELATION](#), las relaciones [persistentes](#) no necesitan restablecerse cada vez que se utilizan las tablas. Sin embargo, como las relaciones persistentes no controlan la relación entre los punteros de registros de las tablas, al programar aplicaciones de Visual FoxPro se utilizan relaciones de ambos tipos.

En Visual FoxPro se utilizan los [índices](#) para establecer relaciones persistentes entre las tablas de una base de datos. La relación se define entre los índices y no entre los campos, lo que permite relacionar las tablas basándose en una expresión de índice simple o compleja.

### Para crear una relación persistente entre tablas

- En el [Diseñador de bases de datos](#), elija el nombre del índice que desee relacionar y arrástrelo hasta el nombre del índice de la tabla relacionada.

–O bien–

- Utilice la cláusula FOREIGN KEY en los comandos [CREATE TABLE](#) o [ALTER TABLE](#).

Por ejemplo, el comando siguiente agrega una relación persistente de [uno a varios](#) entre las tablas customer y orders, basándose en la clave [principal](#) cust\_id de la tabla customer y en una nueva [clave externa](#), cust\_id, de la tabla orders:

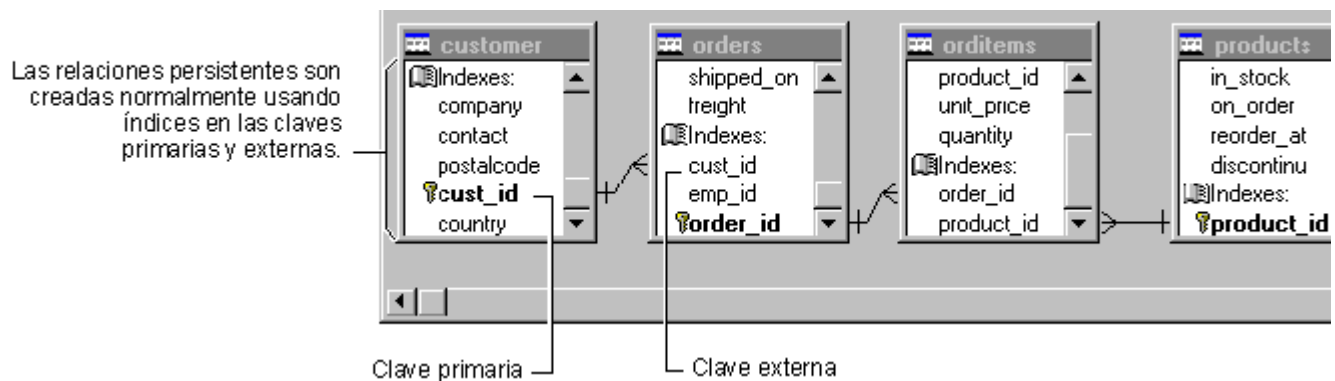
```
ALTER TABLE orders;  
  ADD FOREIGN KEY cust_id TAG ;  
    cust_id REFERENCES customer
```

Si examina el esquema de la base de datos en el Diseñador de bases de datos, verá una línea que une orders y customer, lo que representa la nueva relación persistente.

### Los índices proporcionan la base para relaciones persistentes







El tipo de etiqueta o clave de índice determinará el tipo de [relación persistente](#) que puede crear. Es necesaria una etiqueta de índice [principal](#) o [candidato](#) para el lado 'uno' de una relación de [uno a varios](#); para el lado 'varios' debe utilizar una etiqueta o clave de índice [normal](#). Si desea más información sobre los tipos de índices y cómo crearlos, consulte el capítulo 7, [Trabajar con tablas](#).

### Para eliminar una relación persistente entre tablas

1. En el [Diseñador de bases de datos](#), haga clic en la línea de relación entre las dos tablas.

El ancho de la línea aumentará para indicar que ha seleccionado la relación.

2. Presione la tecla **supr.**

—O bien—

Use la cláusula **DROP FOREIGN KEY** con el comando [ALTER TABLE](#).

Por ejemplo, el comando siguiente elimina una relación persistente entre las tablas `customer` y `orders` basada en la clave [principal](#) `cust_id` de la tabla `customer` y en una clave [externa](#), `cust_id`, de la tabla `orders`:

```
ALTER TABLE orders DROP FOREIGN KEY TAG cust_id SAVE
```

### Generar integridad referencial

Establecer [integridad referencial](#) implica la creación de un conjunto de reglas para preservar las relaciones definidas entre las [tablas](#) al introducir o eliminar [registros](#).

Si exige la integridad referencial, Visual FoxPro impedirá las acciones siguientes:

- Agregar registros a una tabla relacionada cuando no haya ningún registro asociado en la [tabla principal](#).
- Cambiar valores de una tabla principal cuando tales cambios supongan dejar registros huérfanos en una tabla relacionada.
- Eliminar registros de una tabla principal cuando tengan registros relacionados coincidentes.

Si lo desea, puede escribir sus propios [desencadenantes](#) y [procedimientos almacenados](#) para exigir la integridad referencial. Sin embargo, el [Generador de integridad referencial \(IR\)](#) de Visual FoxPro

integridad referencial. Sin embargo, el [Generador de integridad referencial \(IR\)](#) de Visual FoxPro permite determinar los tipos de reglas que desea exigir, las tablas a las que desea exigir las reglas y los eventos del sistema que harán que Visual FoxPro las compruebe.

El Generador de IR trata múltiples niveles de eliminaciones y actualizaciones en cascada y es recomendable como herramienta para asegurar la integridad referencial.

### Para abrir el Generador de IR

1. Abra el [Diseñador de bases de datos](#).
2. En el menú **Base de datos**, elija **Modificar integridad referencial**.

Al utilizar el Generador de IR para crear reglas que se van a aplicar a la base de datos, Visual FoxPro guarda el código generado para exigir las reglas de integridad referencial como [desencadenantes](#) que hacen referencia a [procedimientos almacenados](#). Para ver este código puede abrir el editor de texto de procedimientos almacenados en la base de datos. Si desea información sobre la forma de crear desencadenantes por programa, consulte Uso de desencadenantes en el capítulo 7, [Trabajar con tablas](#).

**Precaución** Cuando haga cambios en el diseño de una [base de datos](#), como modificaciones en sus [tablas](#) o en los [índices](#) utilizados en una [relación persistente](#), debe volver a ejecutar el Generador de IR antes de utilizar de nuevo la base de datos. De esta forma se revisará el código de [procedimiento almacenado](#) y los [desencadenantes](#) utilizados para exigir la integridad referencial, de forma que reflejen el nuevo diseño. Si no vuelve a ejecutar el Generador de RI, puede que obtenga resultados inesperados, ya que no se habrán actualizado los procedimientos almacenados y los desencadenantes para ajustarlos a las modificaciones.

### Crear procedimientos almacenados

Puede crear procedimientos almacenados para las tablas de una base de datos. Un [procedimiento almacenado](#) está formado por código de Visual FoxPro incluido en el archivo .dbc. Los procedimientos almacenados son procedimientos de código que operan específicamente sobre los datos de la base de datos. Su uso puede mejorar el rendimiento, ya que se cargan en memoria en el momento de abrir la base de datos.

### Para crear, modificar o quitar un procedimiento almacenado

- En el [Administrador de proyectos](#), seleccione una base de datos, seleccione **Procedimientos almacenados** y, a continuación, elija **Nuevo**, **Modificar** o **Quitar**.

—O bien—

- En el [Diseñador de bases de datos](#), elija **Modificar procedimientos almacenados** en el menú **Base de datos**.

—O bien—

- En la ventana **Comandos**, utilice el comando **MODIFY PROCEDURE**.

- En la ventana **Comandos**, utilice el comando [MODIFY PROCEDURE](#).

Cada una de estas opciones abre el editor de texto de Visual FoxPro, que permite crear o modificar en el procedimientos almacenados de la base de datos activa.

Los procedimientos almacenados se utilizan principalmente para crear [funciones definidas por el usuario](#) a las que se hace referencia en una regla de validación a nivel de [campo](#) o [registro](#). Al guardar una función definida por el usuario como procedimiento almacenado en la base de datos, el código de la función se guarda en el archivo .dbc y se desplaza automáticamente con la base de datos cuando ésta cambia de lugar. El uso de procedimientos almacenados permite también una mayor portabilidad de la aplicación, ya que no es necesario administrar los archivos de funciones definidas por el usuario independientemente del archivo de base de datos.

## Mostrar y establecer de propiedades de base de datos

Todas las bases de datos de Visual FoxPro contienen las propiedades Comment y Version. Para verlas puede utilizar las funciones [DBGETPROP\(\)](#) y [DBSETPROP\(\)](#).

Por ejemplo, el código siguiente muestra el número de versión de la base de datos testdata:

```
? DBGETPROP('testdata', 'database', 'version')
```

El valor devuelto representa el número de versión de .dbc de Visual FoxPro y es de sólo lectura. Con la misma función puede ver el comentario, si existe, para la base de datos:

```
? DBGETPROP('testdata', 'database', 'comment')
```

A diferencia de la propiedad Version, la propiedad Comment se puede establecer. Use la función [DBSETPROP\(\)](#) para introducir una descripción u otro texto que quiera almacenar con la base de datos.

### Para establecer la propiedad de comentario de la base de datos activa

- En el [Diseñador de bases de datos](#), elija **Propiedades** en el menú **Base de datos** y escriba un comentario en el cuadro **Comentario**.

–O bien–

- Utilice la opción de comentario de la función [DBSETPROP\(\)](#).

Por ejemplo, el código siguiente modifica el comentario de la base de datos testdata:

```
? DBSETPROP('testdata', 'database', 'comment', ;  
    'TestData está incluida a Visual FoxPro')
```

También puede utilizar las funciones [DBGETPROP\(\)](#) y [DBSETPROP\(\)](#) para ver y establecer propiedades de otros objetos de base de datos como [conexiones](#) y [vistas](#).

## Ver y modificar la arquitectura de las bases de datos

Al crear una base de datos, Visual FoxPro crea y abre de forma exclusiva un archivo .dbc (Contenedor de base de datos). El archivo .dbc almacena toda la información sobre la base de datos, incluidos los nombres de los archivos y los [objetos](#) asociados a ella. El archivo .dbc no contiene físicamente ningún objeto de alto nivel, como tablas o campos; Visual FoxPro almacena en él punteros de rutas de archivo que apuntan a las tablas.

Para examinar la arquitectura de una base de datos, puede [examinar](#) el archivo de base de datos, ver el [esquema](#), validar la base de datos, comprobar los vínculos anteriores e incluso extender el archivo .dbc.

## Mostrar el esquema de la base de datos

El [esquema](#) de la base de datos es una representación visual de la estructura de las tablas y las relaciones persistentes establecidas en ella. La ventana [Diseñador de bases de datos](#) muestra el esquema de la base de datos abierta.

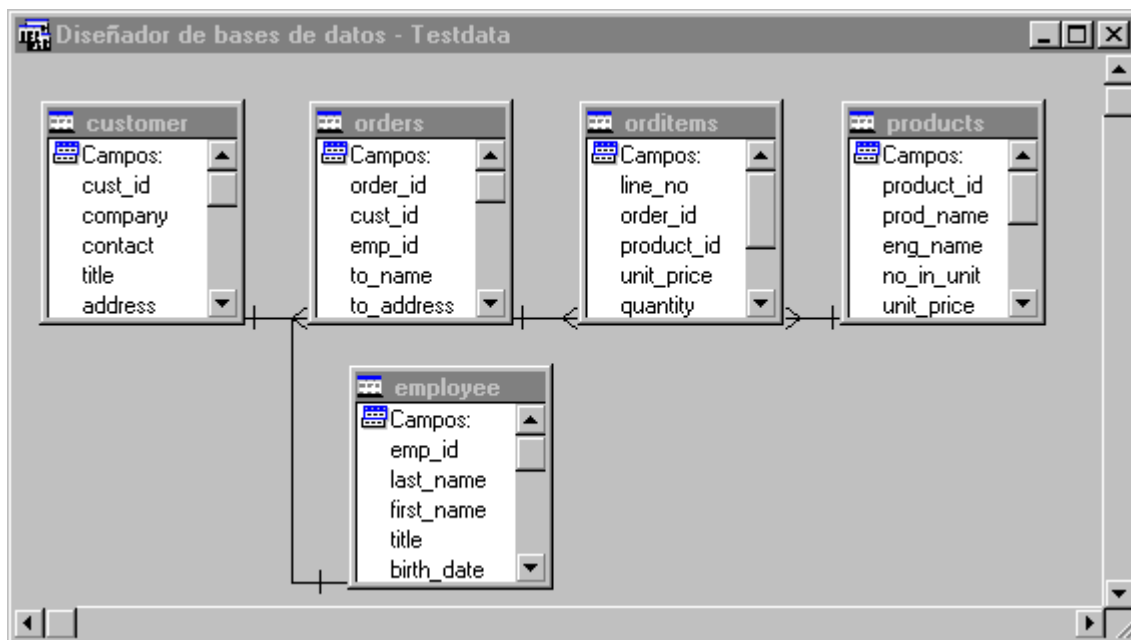
### Para ver el esquema de la base de datos

- Utilice el comando [MODIFY DATABASE](#).

Por ejemplo, el código siguiente abre la base de datos `testdata` y muestra su esquema en el Diseñador de bases de datos:

```
MODIFY DATABASE testdata
```

**Un esquema de base de datos es una representación de los objetos de la base de datos.**



En el Diseñador de bases de datos, puede utilizar la barra de herramientas Base de datos para crear una nueva tabla, agregar a la base de datos una tabla existente, quitar una tabla de la base de datos o

una nueva tabla, agregar a la base de datos una tabla existente, quitar una tabla de la base de datos o modificar la estructura de una tabla. También puede modificar [procedimientos almacenados](#).

## Examinar archivos de base de datos

El archivo de base de datos contiene un registro por cada [tabla](#), [vista](#), [índice](#), [etiqueta](#) de índice, [relación persistente](#) y [conexión](#) asociados a la base, y también por cada campo de tabla o de vista con propiedades extendidas. También incluye un registro único que contiene todos los [procedimientos almacenados](#) de la base de datos.

Para obtener más información acerca de la estructura del archivo .dbc, vea [Estructura de archivos de tabla](#).

Aunque el [Diseñador de bases de datos](#) proporciona una representación conceptual del [esquema](#) de la base de datos, puede que en ocasiones sea necesario examinar el contenido del propio archivo de base de datos. Para examinar una base de datos cerrada puede utilizar el comando [USE](#) con el archivo .dbc. En el ejemplo siguiente se abre una ventana [Examinar](#) que muestra el contenido de la base de datos sales en formato de tabla.

```
CLOSE DATABASE sales
USE sales.dbc EXCLUSIVE
BROWSE
```

**Precaución** No utilice el comando [BROWSE](#) para modificar el archivo de base de datos si no conoce la estructura de los archivos .dbc. Si comete un error al intentar modificar el archivo .dbc, puede invalidar la base de datos y provocar una pérdida de datos.

## Extender los archivos de base de datos

Todos los archivos .dbc contienen un campo Memo, llamado User, que puede utilizar para almacenar su propia información sobre cada registro de la base de datos. También puede extender un archivo .dbc agregándole campos para ajustarlo a sus necesidades como programador. Para modificar la estructura de un archivo .dbc es necesario tener acceso al mismo de forma exclusiva.

### Para agregar un campo a un archivo .dbc

1. Abra el archivo .dbc para uso exclusivo con el comando [USE](#).
2. Utilice el comando [MODIFY STRUCTURE](#).

Por ejemplo, el código siguiente abre el [Diseñador de tablas](#) para permitirle agregar un campo a la estructura de Testdata.dbc:

```
USE TESTDATA.DBC EXCLUSIVE  
MODIFY STRUCTURE
```

Al agregar un nuevo campo o un archivo de base de datos, utilice como primer carácter de su nombre “U” para indicar que se trata de un campo definido por el usuario. Esta convención evita que el campo entre en conflicto con futuras extensiones del archivo .dbc.

**Precaución** No modifique ninguno de los campos definidos por Visual FoxPro en un archivo .dbf. Las modificaciones de un archivo .dbc pueden afectar a la integridad de la base de datos.

## Validar una base de datos

La validación de una base de datos asegura que las filas de la base de datos almacenan representaciones precisas de los metadatos de la base de datos. Para comprobar la integridad de la base de datos activa, puede utilizar el comando `VALIDATE DATABASE`.

### Para validar una base de datos

- Utilice el comando [VALIDATE DATABASE](#).

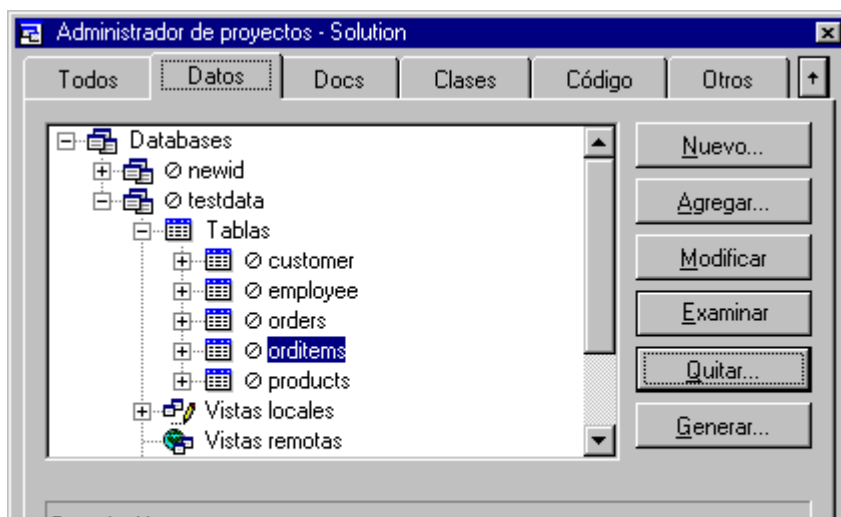
Por ejemplo, el código siguiente utiliza y valida el archivo .dbc de la base de datos `testdata`:

```
OPEN DATABASE testdata EXCLUSIVE  
VALIDATE DATABASE
```

## Administrar una base de datos

Después de crear una base de datos, es posible que quiera agregarla a un [proyecto](#) si no forma parte de uno. Si su base de datos ya forma parte de un proyecto, puede quitarla del mismo. Además, si ya no necesita la base de datos, puede eliminarla del disco.

### Una base de datos en el Administrador de proyectos



Descripción:
Ruta:

## Agregar una base de datos a un proyecto

Al crear una base de datos con el comando [CREATE DATABASE](#), la base de datos no pasa automáticamente a formar parte de un proyecto, aún cuando el [Administrador de proyectos](#) esté abierto. Puede agregar la base de datos a un proyecto para facilitar la organización, la presentación y la manipulación de los objetos de base de datos mediante la interfaz y también para simplificar el proceso de generación de una aplicación. Sólo se puede agregar una base de datos a un proyecto mediante el Administrador de proyectos.

### Para agregar una base de datos a un proyecto

- En el [Administrador de proyectos](#), seleccione **Bases de datos** y, a continuación, elija **Agregar**.

## Quitar una base de datos de un proyecto

Sólo se puede quitar una base de datos de un proyecto mediante el Administrador de proyectos.

### Para quitar una base de datos de un proyecto

- En el [Administrador de proyectos](#), seleccione la base de datos y elija **Quitar**; a continuación, elija **Quitar** de nuevo.

## Eliminar una base de datos

Para eliminar del disco una base de datos puede utilizar el Administrador de proyectos o el comando [DELETE DATABASE](#).

### Para eliminar una base de datos

- En el [Administrador de proyectos](#), seleccione la base de datos, elija **Quitar** y, a continuación, elija **Eliminar**.

–O bien–

- Utilice el comando [DELETE DATABASE](#).

Por ejemplo, el código siguiente elimina la base de datos ejemplo:

```
DELETE DATABASE sample
```

Utilice siempre alguno de los métodos anteriores para eliminar una base de datos del disco. El uso del [Administrador de proyectos](#) o del comando [DELETE DATABASE](#) permite a Visual FoxPro suprimir los [vínculos anteriores](#) que unen la base de datos a sus tablas. Otras utilidades de manipulación de

los [vínculos anteriores](#) que unen la base de datos a sus tablas. Otras unidades de manipulación de archivos, como el Administrador de archivos de Windows, no suprimen los vínculos anteriores.

**Nota** El comando DELETE DATABASE no elimina del disco las tablas asociadas a la base de datos, sino que las convierte en tablas libres. Si desea eliminar del disco una base de datos y también todas sus tablas asociadas, utilice la cláusula DELETETABLES con el comando DELETE DATABASE.

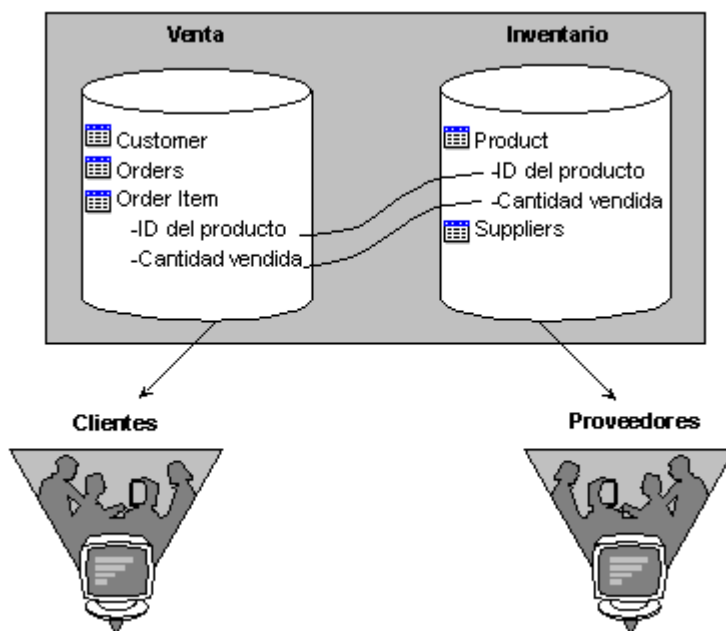
## Hacer referencia a múltiples bases de datos

En un sistema es posible disponer de múltiples bases de datos de Visual FoxPro para satisfacer las necesidades de un entorno multiusuario. El uso de múltiples bases de datos ofrece las siguientes ventajas:

- Controlan el acceso de usuarios a un subconjunto de tablas del sistema global.
- Organizan los datos para satisfacer de forma eficaz las necesidades de información de varios grupos que usan el sistema.
- Permitiendo el uso exclusivo de un subconjunto de tablas para crear vistas [locales](#) y [remotas](#) en [tiempo de ejecución](#).

Por ejemplo, es posible que tenga una base de datos de ventas que guarda información de ventas usada principalmente por la fuerza de ventas que trabaja con clientes y otra base de datos que guarda información de inventario usada principalmente por los compradores que trabajan con los suministradores. A veces es necesario que la información de estos grupos se solape. Estas bases de datos se pueden abrir simultáneamente y se puede tener acceso a las mismas como se desee, pero contendrán tipos de información completamente diferentes.

### Múltiples bases de datos pueden agregar flexibilidad a su sistema



Puede usar múltiples bases de datos abriendo más de una base de datos simultáneamente o estableciendo referencias en una base de datos cerrada. Cuando están abiertas varias bases de datos, puede establecer la base de datos actual y seleccionar tablas de ella.



puede establecer la base de datos actual y seleccionar tablas de ella.

## Abrir más de una base de datos

Cuando hay abierta una base de datos, las tablas y las relaciones entre ellas están controladas por la información almacenada en la base de datos abierta. Puede tener abierta más de una base de datos a la vez. Por ejemplo, puede utilizar múltiples bases de datos abiertas al ejecutar varias aplicaciones, cada una basada en una base de datos distinta. También puede ser conveniente abrir varias bases de datos para utilizar información (como por ejemplo [controles personalizados](#)) almacenada en una base de datos distinta de la que utiliza la aplicación.

### Para abrir más de una base de datos

- En el [Administrador de proyectos](#), seleccione una base de datos y elija **Modificar** o **Abrir**.  
–O bien–
- Utilice el comando [OPEN DATABASE](#).

Al abrir una nueva base de datos no se cierran las que se hayan abierto previamente. Las bases abiertas continuarán en este estado y la última en abrirse pasará a ser la base de datos activa.

### Establecer la base de datos activa

Al abrir varias bases de datos, Visual FoxPro establece como activa la abierta en último lugar. De forma predeterminada, todas las tablas u objetos que cree o agregue formarán parte de la base de datos activa. Los comandos y funciones que manipulan bases de datos abiertas, como ADD TABLE y DBC( ), se aplican también a la base de datos activa.

Puede elegir una base de datos distinta y establecerla como activa mediante la interfaz o mediante el comando SET DATABASE.

### Para establecer la base de datos activa

- En la barra de herramientas estándar, seleccione una base de datos en el cuadro **Bases de datos**.  
–O bien–
- Utilice el comando [SET DATABASE](#).

Por ejemplo, el código siguiente abre tres bases de datos, establece la primera como activa y luego utiliza la función DBC( ) para mostrar el nombre de la base de datos activa:

```
OPEN DATABASE testdata
OPEN DATABASE tastrade
OPEN DATABASE sample
SET DATABASE TO testdata
? DBC( )
```

**Sugerencia** Visual FoxPro puede abrir una o más bases de datos automáticamente al ejecutar una [consulta](#) o un [formulario](#) que requiera que esas bases de datos se encuentren abiertas. Para mantener el control, establezca la base de datos activa explícitamente antes de utilizar comandos que operen sobre la base de datos activa.

## Seleccionar tablas de la base de datos activa

Para elegir en una lista con las tablas de la base de datos activa, puede utilizar el comando USE.

### Para elegir una tabla de la base de datos activa

- Escriba el comando [USE](#) con el símbolo “?”.

Aparecerá el cuadro de diálogo **Usar** en el que puede seleccionar la tabla que desea abrir.

Por ejemplo, el código siguiente abre la base de datos `ventas` y solicita que se seleccione una de las tablas que contiene.

```
OPEN DATABASE SALES
USE ?
```

Si desea seleccionar una tabla no asociada a la base de datos abierta, podrá elegir "Otras" en el cuadro de diálogo Usar.

## Cerrar una base de datos

Puede cerrar una base de datos abierta mediante el Administrador de proyectos o mediante el comando CLOSE DATABASE.

### Para cerrar una base de datos

- En el [Administrador de proyectos](#), seleccione la base de datos y elija **Cerrar**.

–O bien–

- Utilice el comando [CLOSE](#) DATABASE.

Por ejemplo, el código siguiente cierra la base de datos `testdata`:

```
SET DATABASE TO testdata
CLOSE DATABASE
```

Ambas opciones cierran automáticamente la base de datos. También puede cerrar bases de datos y todos los demás objetos abiertos con la cláusula `ALL` del comando [CLOSE](#).

El uso del comando [CLOSE](#) DATABASE desde la ventana [Comandos](#) no cerrará las bases de datos que se hayan abierto de las formas siguientes:

- Con el Administrador de proyectos al expandir el esquema para ver el contenido de una base de datos.
- Con un [formulario](#) que se ejecute en su propia [sesión de datos](#).

En estas circunstancias, la base de datos permanecerá abierta hasta que el Administrador de proyectos la cierre o hasta que se cierre el formulario que la utiliza.

## Resolución del alcance

Visual FoxPro utiliza la base de datos activa como alcance principal para los objetos con nombre, como las tablas. Cuando una base de datos está abierta, Visual FoxPro busca primero en ella los objetos que se soliciten, como pueden ser [tablas](#), [vistas](#), [conexiones](#), etcétera. Si el objeto no se encuentra en la base de datos, Visual FoxPro lo buscará en la ruta de búsqueda predeterminada.

Por ejemplo, si la tabla `customer` está asociada a la base de datos `sales`, Visual FoxPro encontrará siempre la tabla `customer` de la base de datos al utilizar los comandos siguientes:

```
OPEN DATABASE SALES
ADD TABLE F:\SOURCE\CUSTOMER.DBF
USE CUSTOMER
```

Si utiliza el comando siguiente, Visual FoxPro buscará la tabla `products` comenzando por la base de datos activa:

```
USE PRODUCTS
```

Si `products` no se encuentra en la base de datos activa, Visual FoxPro intentará buscar fuera de ella en la ruta de búsqueda predeterminada.

**Nota** Puede especificar la ruta de acceso completa de una tabla si desea tener acceso a la misma desde dentro o desde fuera de una base de datos; por ejemplo, si prevé un cambio en la ubicación de la tabla. Sin embargo, el rendimiento es mejor cuando sólo se hace referencia al nombre de la tabla, ya que en Visual FoxPro es más rápido el acceso a los nombres de tablas de base de datos que a los nombres especificados con la ruta de acceso completa.

## Controlar errores de bases de datos

Los errores de base de datos, también llamados “errores de motor”, ocurren cuando se dan errores en tiempo de ejecución en código de eventos a nivel de registros. Por ejemplo, un error de base de datos ocurre cuando el usuario intenta almacenar un valor nulo en un campo que no admite valores nulos.

Cuando ocurre un error de base de datos, el motor de base de datos subyacente que detecta el error

envía normalmente un mensaje de error. Sin embargo, la naturaleza exacta del mensaje de error depende de la base de datos a la que se tenga acceso, por ejemplo, los mensajes de error producidos por un servidor de base de datos remota (como Microsoft SQL Server) probablemente serán distintos de los que se producen si ocurre un error en una tabla local de Visual FoxPro.

Además, los errores a nivel de motor son a veces muy genéricos, porque el motor de base de datos no tiene información sobre el contexto en el que se actualiza un registro. Como consecuencia, los mensajes de error producidos por un motor de base de datos suelen ser menos útiles para el usuario de una aplicación de Visual FoxPro.

Para controlar errores de base de datos de forma más específica para la aplicación, puede crear desencadenantes con el comando [CREATE TRIGGER](#). El desencadenante se llama cuando se intenta la actualización de un registro (eliminar, insertar o actualizar). El código de desencadenante actualizado puede buscar entonces condiciones de error específicas de la aplicación e informar de ellas.

Si controla errores de base de datos mediante desencadenantes, debería activar el almacenamiento local. De esta forma, cuando se actualiza un registro, se llama al desencadenante pero el registro no se envía automáticamente a la base de datos subyacente. Así evita la posibilidad de producir dos mensajes de error: uno del desencadenante y otro del motor de base de datos subyacente.

#### **Para crear mensajes de error personalizados mediante desencadenantes**

1. Dentro de una función definida por el usuario o un procedimiento almacenado, escriba su propio texto de mensaje.
2. Active el almacenamiento local con la función [CURSORSETPROP\(\)](#) para mostrar su texto personalizado. Si el almacenamiento local está desactivado, el usuario verá tanto su texto personalizado como el mensaje de error del motor.

## **Capítulo 7: Trabajar con tablas**

La creación de una [base de datos](#) implica la creación de [tablas](#). Al diseñar la base de datos especificó los [campos](#) de tabla y las relaciones necesarias para su aplicación. Ahora, al crear las tablas, deberá indicar con más detalle los tipos de datos, los títulos y los posibles valores predeterminados de cada campo, los desencadenantes de cada tabla y los índices que se utilizan para establecer relaciones entre las tablas. En este capítulo se describe el proceso de crear, ajustar y relacionar tablas e índices al programar una aplicación. Se tratará principalmente el uso del lenguaje para trabajar con tablas y registros, pero también se explicará el uso de la interfaz para realizar tareas habituales.

En este capítulo se tratan los temas siguientes:

- [Crear tablas](#)
- [Trabajar con registros](#)
- [Indexar tablas](#)
- [Usar múltiples tablas](#)

### **Crear tablas**

Puede crear una tabla en una base de datos o una tabla libre que no esté asociada a ninguna base de datos. A medida que crea la tabla puede crear nombres largos de tabla y de campo, y aprovechar las posibilidades del [diccionario de datos](#) para tablas de base de datos, nombres largos de campo, valores de campo predeterminados, reglas a nivel de [campo](#) y a [nivel de registro](#), y [desencadenantes](#).

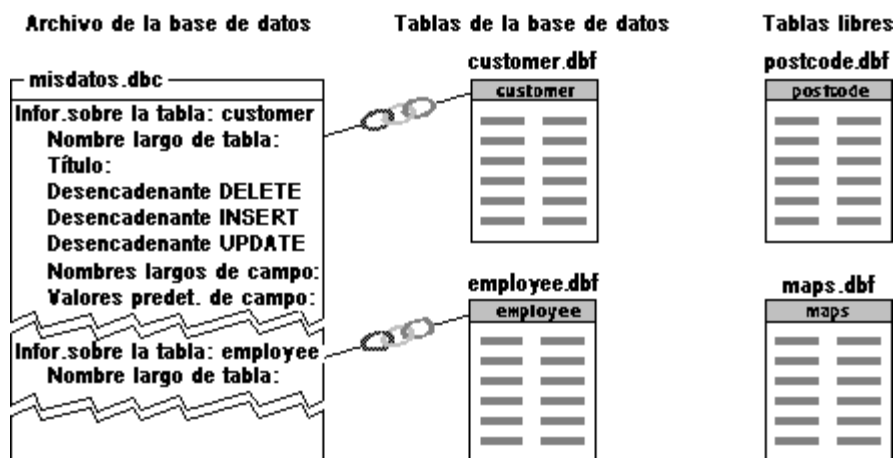
## Diseñar tablas de base de datos y tablas libres

Las tablas de Visual FoxPro, o archivos .dbf, pueden existir en dos estados: como tabla de base de datos (una tabla asociada a una base de datos) o como una tabla libre no asociada a ninguna base de datos. Las tablas asociadas a una base de datos cuentan con ciertas ventajas sobre las tablas libres. Cuando una tabla forma parte de una base de datos, es posible crear:

- Nombres largos para la tabla y para cada uno de sus campos.
- [Títulos](#) y comentarios para cada campo de la tabla.
- [Valores predeterminados](#), [máscaras de entrada](#) y formato para los campos de la tabla.
- [Clases de control](#) predeterminada para campos de tablas.
- Reglas a [nivel de campo](#) y a [nivel de registro](#).
- [Índices](#) de clave primaria y relaciones de tablas para compatibilidad con reglas de [integridad referencial](#).
- Un [desencadenante](#) para cada evento INSERT, UPDATE o DELETE.

Algunas características sólo son aplicables a las tablas de base de datos. Si desea información sobre la forma de asociar tablas a una base de datos, consulte el capítulo 6, [Crear bases de datos](#).

## Las tablas de base de datos tienen propiedades con las que no cuentan las tablas libres



Puede diseñar y crear una tabla de forma interactiva mediante el [Diseñador de tablas](#), accesible a través del [Administrador de proyectos](#) o del menú Archivo, o mediante el lenguaje de programación. En esta sección se describe principalmente la creación de una tabla mediante programación. Si desea información sobre el uso del Diseñador de tablas para crear tablas de forma interactiva, consulte el capítulo 2, [Crear tablas e índices](#), del *Manual del usuario*.

Para crear y modificar una tabla mediante programación se utilizan los comandos siguientes:

## Comandos para crear y modificar tablas

[ALTER TABLE](#)[CLOSE TABLES](#)[CREATE TABLE](#)[DELETE FILE](#)[REMOVE TABLE](#)[RENAME TABLE](#)[DROP TABLE](#)

## Crear una tabla de base de datos

Para crear una nueva tabla en una base de datos puede utilizar el sistema de menús, el Administrador de proyectos o puede hacerlo mediante el lenguaje. Al crear la tabla puede especificar nombres largos de tabla y de campo, valores de campo predeterminados, reglas a nivel de campo y a nivel de registro, y desencadenantes.

### Para crear una nueva tabla de base de datos

- En el [Administrador de proyectos](#) seleccione una base de datos, elija **Tablas** y, a continuación, **Nuevo** para abrir el [Diseñador de tablas](#).

–O bien–

- Utilice el comando [CREATE TABLE](#) con una base de datos abierta.

Por ejemplo, el código siguiente crea la tabla `pegtbl` con una columna, llamada `nombre`:

```
OPEN DATABASE Sales
CREATE TABLE pegtbl (nombre c(50))
```

La nueva tabla se asocia automáticamente a la base de datos abierta en el momento de su creación. Esta asociación está definida por un [vínculo anterior](#) almacenado en el registro de encabezado de la tabla.

## Crear una tabla libre

Para crear una nueva tabla libreLas tablas libres no están asociadas a ninguna base de datos. Puede ser conveniente crear una tabla libre, por ejemplo, para almacenar información de consulta que comparten múltiples bases de datos

computar múltiples bases de datos.

- En el [Administrador de proyectos](#), seleccione **Tablas libres** y, a continuación, **Nuevo** para abrir el **Diseñador de tablas**.

–O bien–

- Use la palabra clave FREE con el comando [CREATE TABLE](#).

Por ejemplo, el código siguiente crea la tabla libre peqtbl con una columna, llamada nombre:

```
CLOSE DATABASES
CREATE TABLE peqtbl FREE (nombre c(50))
```

Si no hay ninguna base de datos abierta cuando crea la tabla, no tiene que usar la palabra clave FREE.

## Asignar un nombre a una tabla

Al utilizar el comando CREATE TABLE se especifica el nombre del archivo .DBF que Visual FoxPro creará para almacenar la nueva tabla. El nombre del archivo será el nombre predeterminado de la tabla, tanto en el caso de tablas de base de datos como en el de tablas libres. Los nombres de las tablas pueden contener letras, dígitos o signos de subrayado y deben empezar por una letra o un signo de subrayado.

Si la tabla es de base de datos, también podrá especificar un nombre largo de tabla. Los nombres largos de tabla pueden tener hasta 128 caracteres y pueden usarse en lugar de los nombres cortos para identificar la tabla en la base de datos. Cuando se han definido nombres largos de tabla, Visual FoxPro los mostrará siempre que la tabla aparezca en la interfaz, como por ejemplo en el Diseñador de bases de datos, el Diseñador de consultas y el Diseñador de vistas, así como en la barra de título de las ventanas Examinar.

## Para asignar un nombre largo a una tabla de base de datos

- En el [Diseñador de tablas](#), escriba un nombre largo en el cuadro de texto **Nombre** de la ficha **Tabla**.

–O bien–

- Utilice la cláusula NAME del comando [CREATE TABLE](#).

Por ejemplo, el código siguiente crea la tabla provintl y le asigna el nombre largo vendors\_international, más legible:

```
CREATE TABLE provintl NAME vendors_international (company C(40))
```

También puede utilizar el [Diseñador de tablas](#) para cambiar el nombre de las tablas o agregar nombres largos a tablas creadas sin ellos. Por ejemplo, cuando agregue una tabla libre a una base de datos, puede utilizar el Diseñador de tablas para agregarle un nombre largo de tabla. Los nombres largos pueden contener letras, dígitos o signos de subrayado y deben empezar por una letra o un

largos pueden contener letras, dígitos o signos de subrayado, y deben empezar por una letra o un signo de subrayado. No se permite el uso de espacios en blanco en los nombres largos de tabla.

## Cambiar el nombre de una tabla

Puede cambiar el nombre de tablas de base de datos a través de la interfaz porque va a cambiar el nombre largo. Si quita la tabla de la base de datos, el nombre de archivo de la tabla conserva el nombre original. Las tablas libres no tienen un nombre largo y sólo se puede cambiar su nombre a través del lenguaje.

### Para cambiar el nombre de una tabla de una base de datos

1. En el [Diseñador de bases de datos](#), seleccione la tabla cuyo nombre desea cambiar.
2. En el menú **Base de datos**, elija **Modificar**.
3. En el **Diseñador de tablas** escriba un nuevo nombre para la tabla en el cuadro **Nombre de tabla** de la ficha **Tabla**.

### Para cambiar el nombre de una tabla

- Use el comando [RENAME](#).

**Precaución** Si usa el comando RENAME en tablas asociadas a la base de datos, el comando no actualiza el [vínculo anterior](#) a la base de datos y puede producir errores de acceso a la tabla.

## Eliminar una tabla de base de datos

Si una tabla está asociada a una base de datos, para eliminarla quítela de la base de datos. Sin embargo, eliminar una tabla es distinto de quitarla de una base de datos. Si desea quitar la tabla de la base de datos, pero no desea eliminarla físicamente del disco, consulte "Quitar una tabla de una base de datos" en el capítulo 6, [Crear bases de datos](#).

### Para eliminar una tabla de base de datos del disco

- En el [Administrador de proyectos](#), seleccione el nombre de la tabla, elija **Quitar** y, a continuación, elija **Eliminar**.  
–O bien–
- En el [Diseñador de bases de datos](#), seleccione la tabla, elija **Quitar** del menú **Base de datos** y, a continuación, elija **Eliminar**.  
–O bien–
- Para eliminar la tabla junto a todos los índices principales, valores predeterminados y reglas de validación asociadas a la tabla, use el comando [DROP TABLE](#).

–O bien–



–O bien–

- Para eliminar únicamente el archivo de tabla (.dbf), use el comando [ERASE](#).

**Precaución** Si usa el comando ERASE en tablas asociadas a la base de datos, el comando no actualiza el [vínculo anterior](#) a la base de datos y puede producir errores de acceso a tabla.

Por ejemplo, el código siguiente abre la base de datos `testdata` y elimina la tabla `orditems` y sus índices, valores predeterminados y reglas de validación:

```
OPEN DATABASE testdata
DROP TABLE orditems
```

Al eliminar una tabla con la cláusula DELETE del comando [REMOVE TABLE](#) también desaparecen el archivo memo .fpt y el archivo de índice estructural .cdx.

## Eliminar una tabla libre

Si una tabla no está asociada a ninguna base de datos, podrá eliminar su archivo mediante el Administrador de proyectos o con el comando DELETE FILE.

### Para eliminar una tabla libre

- En el [Administrador de proyectos](#), seleccione la tabla libre, elija **Quitar** y, a continuación, elija **Eliminar**.

–O bien–

- Utilice el comando [DELETE FILE](#).

Por ejemplo, si la tabla actual es `ejemplo`, el código siguiente cerrará la tabla y eliminará su archivo del disco:

```
USE
DELETE FILE ejemplo.dbf
```

El archivo que desea eliminar no puede estar abierto en el momento de utilizar DELETE FILE. Si elimina una tabla que tiene otros archivos asociados, como un archivo memo .fpt o archivos de índice (.cdx o .idx), asegúrese de eliminar también este archivo. Por ejemplo, si el archivo `ejemplo.dbf` tuviese además un archivo memo asociado, podría eliminar ambos archivos con los comandos siguientes:

```
USE
DELETE FILE ejemplo.dbf
DELETE FILE ejemplo.fpt
```

## Duplicar una tabla

Puede hacer una copia idéntica de una tabla, sus [procedimientos almacenados](#), expresiones [desencadenantes](#), valores de campos predeterminados y su contenido a través de programación. No hay ninguna opción de menú que realice la misma función. Este procedimiento no copia el contenido

de la tabla.

### Para duplicar una tabla

1. Abra la tabla original.
2. Use el comando [COPY STRUCTURE](#) para hacer una copia de la tabla original.
3. Abra la tabla vacía creada con el comando COPY STRUCTURE.
4. Use el comando [APPEND FROM](#) para copiar los datos desde la tabla original.

### Copiar y modificar la estructura de una tabla

Para modificar la estructura de una tabla puede abrir el [Diseñador de tablas](#) o bien puede realizar los cambios mediante programación con el comando [ALTER TABLE](#). De forma alternativa, puede crear una tabla nueva en base a la estructura de una tabla existente y, a continuación, modificar la estructura de la nueva tabla.

### Para copiar y modificar la estructura de una tabla

1. Abra la tabla original.
2. Use el comando [COPY STRUCTURE EXTENDED](#) para producir una tabla nueva que contenga la información estructural de la tabla antigua.
3. Modifique la nueva tabla que contiene la información estructural para variar la estructura de cualquier tabla nueva creada a partir de esa información.
4. Cree una tabla nueva con el comando [CREATE FROM](#).

La nueva tabla está vacía.

5. Use [APPEND FROM](#) o uno de los comandos de copia de datos para rellenar la tabla si es necesario.

### Guardar una tabla como HTML

Puede utilizar la opción **Guardar como HTML** del menú **Archivo** cuando examine el contenido de una tabla para guardarla como un archivo HTML (Lenguaje de marcado de hipertexto).

### Para guardar una tabla como HTML

1. Abra la tabla.
2. Examine la tabla; para ello, ejecute el comando [BROWSE](#) en la ventana Comandos o elija **Examinar** en el menú **Ver**.

3. Elija **Guardar como HTML** en el menú **Archivo**.
4. Escriba el nombre del archivo HTML que desea crear y elija **Guardar**.

## Crear campos

Al crear campos de tabla debe especificar un nombre de campo, un tipo de datos y un ancho de campo. También puede indicar si el campo debe permitir [valores nulos](#) y especificar el valor predeterminado del campo. Al configurar las propiedades de presentación, puede especificar el tipo de control de formulario que se crea cuando el campo se agrega a un [formulario](#), el formato del contenido de los [campos](#) o el [título](#) del contenido del campo.

**Nota** Las tablas de Visual FoxPro puede contener hasta 255 campos. Si uno o más campos pueden contener valores nulos, el número máximo de campos que la tabla puede contener se reduce en una unidad, de 255 a 254.

## Asignar nombres a los campos

Los nombres de los campos se especifican al crear la nueva tabla. Estos nombres pueden tener hasta 10 caracteres en las tablas libres y hasta 128 en las tablas de base de datos. Si quita una tabla de una base de datos, los nombres largos de campo se truncarán a 10 caracteres.

### Para asignar nombre a un campo de tabla

- En el [Diseñador de tablas](#), escriba un nombre de campo en el cuadro de texto **Nombre**.  
–O bien–
- Utilice el comando [CREATE TABLE](#) o el comando [ALTER TABLE](#).

Por ejemplo, para crear y abrir la tabla `customer` con tres campos, `cust_id`, `company` y `contact`, podría utilizar el comando siguiente:

```
CREATE TABLE customer (cust_id C(6), company C(40), contact C(30))
```

En el ejemplo anterior, `C(6)` indica que el campo tiene el tipo de datos Character y un ancho de 6. La elección del tipo de datos de los campos de tablas se trata más adelante en esta sección.

Con el comando `ALTER TABLE` agrega los campos, `company`, y `contact`, a una tabla `customer` existente:

```
ALTER TABLE customer ;  
    ADD COLUMN (company C(40), contact C(30))
```

## Usar nombres cortos de campo

Al crear una tabla en una base de datos, Visual FoxPro almacena el nombre largo de los campos de la tabla en un registro del archivo `.dbc`. Los 10 primeros caracteres de cada nombre largo se almacenan también en el archivo `.dbf` como nombre de campo.

Si los 10 primeros caracteres del nombre largo de campo no son únicos en la tabla, Visual FoxPro generará un nombre formado por los  $n$  primeros caracteres del nombre largo y un número secuencial a continuación, de forma que el nombre del campo tenga 10 caracteres. Por ejemplo, los nombres largos de campo siguientes se convierten en los nombres de 10 caracteres que se indican:

Nombre largo	Nombre corto
cliente_contacto_nombre	cliente_co
cliente_contacto_dirección	cliente_c2
customer_contact_city	cliente_c3
...	...
cliente_contacto_fax	cliente_11

Mientras una tabla esté asociada a una base de datos, deberá usar los nombres largos para referirse a sus campos. No puede utilizar los nombres de 10 caracteres para hacer referencia a los campos de una tabla de base de datos. Si quita una tabla de su base de datos, los nombres largos de los campos se perderán y deberá utilizar los de 10 caracteres (almacenados en el archivo .dbf).

Puede utilizar nombres largos de campo en los archivos de [índice](#). Sin embargo, si crea un índice con nombres largos de campo y luego quita la tabla correspondiente de la base de datos, el índice no funcionará. En este caso, puede eliminar el índice y volver a crearlo con nombres cortos de campo. Si desea información sobre la forma de eliminar un índice, consulte ["Eliminar un índice"](#), más adelante en este mismo capítulo.

Las reglas para crear nombres largos de campo son las mismas que para cualquier identificador de Visual FoxPro, salvo que pueden contener hasta 128 caracteres.

Si desea más información sobre la asignación de nombres a los identificadores de Visual FoxPro, vea [Crear nombres de Visual FoxPro](#).

## Elegir los tipos de datos

Al crear cada campo de una tabla, deberá elegir el tipo de los datos que ese campo va a almacenar. Al elegir el tipo de datos, decide lo siguiente:

- La clase de valores que se van a permitir en el campo. Por ejemplo, no puede almacenar texto en un campo [Numeric](#).
- El espacio de almacenamiento que Visual FoxPro debe reservar para los valores almacenados en el campo. Por ejemplo, todos los valores con [tipo de datos Currency](#) ocuparán 8 bytes.
- Los tipos de operaciones que pueden realizarse con los valores almacenados en el campo. Por ejemplo, Visual FoxPro puede hallar la suma de valores numéricos o de moneda, pero no la de valores de tipo [character](#) o [general](#).
- Si Visual FoxPro puede o no indexar u ordenar los valores del campo. No es posible ordenar ni crear índices para los campos de tipo [memo](#) o [general](#).

**Sugerencia** Para los números de teléfono, números de pieza y otros números que no vaya a utilizar en cálculos matemáticos, debe elegir el tipo de datos Character, en lugar de Numeric.

### Para elegir el tipo de datos de un campo

- En el [Diseñador de tablas](#), elija un tipo de datos en la lista **Tipo**.

–O bien–

- Utilice el comando [CREATE TABLE](#).

Por ejemplo, para crear y abrir la tabla `products` con tres campos, `prod_id`, `prod_name` y `unit_price`, podría utilizar el comando siguiente:

```
CREATE TABLE products (prod_id C(6), prod_name C(40), unit_price Y)
```

En el ejemplo anterior, la ‘Y’ que aparece después de `unit_price` especifica el tipo de datos Moneda.

Para obtener más información acerca de tipos de datos específicos, vea [Tipos de datos y tipos de campos](#).

### Agregar rápidamente un índice normal

Al agregar un campo, puede definir rápidamente un [índice normal](#) en el campo si especifica ascendente o descendente en la columna Índice del [Diseñador de tablas](#). El índice que cree se agrega automáticamente a la ficha Índices y usa el campo como expresión. Para modificar el índice, puede cambiar a la ficha Índices para cambiar el nombre de índice, el tipo o agregar un [filtro](#).

### Usar valores nulos

Al crear una tabla nueva, puede especificar si uno o más campos de la misma deben aceptar valores nulos. Al usar un valor nulo, se indica que la información que normalmente se almacenaría en un campo o registro no está disponible en ese momento. Por ejemplo, las ventajas sanitarias o el estado impositivo de un empleado pueden no estar determinados en el momento de rellenar un registro. En lugar de almacenar un cero o dejar los campos en blanco, lo que podría interpretarse de modo erróneo, puede almacenar en ellos un valor nulo hasta que la información correspondiente esté disponible.

### Para controlar la introducción de valores nulos campo a campo

- En el [Diseñador de tablas](#), seleccione o desactive la columna **Null** del campo correspondiente.

Cuando la columna **Null** está seleccionada, se pueden introducir valores nulos en el campo.

–O bien–

- Utilice las cláusulas `NULL` y `NOT NULL` del comando [CREATE TABLE](#).

Por ejemplo, el comando siguiente crea y abre una tabla que no admite valores nulos en los campos `cust_id` y `company` pero que sí los permite en el campo `contact`:

```
CREATE TABLE customer (cust_id C(6) NOT NULL, ;
                        company C(40) NOT NULL, contact C(30) NULL)
```

También puede controlar si se admiten o no valores nulos en los campos de la tabla con el comando [SET NULL ON](#).

### Para permitir valores nulos en todos los campos de la tabla

- En el [Diseñador de tablas](#), seleccione la columna **Null** para todos los campos de la tabla.

–O bien–

- Utilice el comando [SET NULL ON](#) antes del comando [CREATE TABLE](#).

Al especificar el comando `SET NULL ON`, Visual FoxPro marca automáticamente la columna `NULL` para cada campo de la tabla a medida que usted agrega campos en el Diseñador de tablas. Si ejecuta el comando `SET NULL` antes de `CREATE TABLE`, no será necesario especificar las cláusulas `NULL` o `NOT NULL`. Por ejemplo, el código siguiente crea una tabla que permite valores nulos en todos sus campos:

```
SET NULL ON
CREATE TABLE test (field1 C(6), field2 C(40), field3 Y)
```

La presencia de valores nulos afecta al comportamiento de las tablas y de los [índices](#). Por ejemplo, si utiliza `APPEND FROM` para copiar registros de una tabla que contiene valores nulos a otra que no los admite, los campos anexados que contengan valores nulos se considerarán en blanco, vacíos o con valor cero en la tabla actual.

Si desea más información sobre la forma en que los valores nulos interactúan con los comandos, consulte [Controlar valores nulos](#).

### Agregar comentarios a los campos

Una vez creada una tabla en una base de datos abierta, puede agregar una descripción de cada campo de la tabla para facilitar la comprensión y la actualización de la tabla. Visual FoxPro muestra el texto de comentario de los campos en el [Administrador de proyectos](#) cuando se seleccionan en la lista de campos de la tabla.

### Para agregar un comentario a un campo de una tabla de base de datos

- En el [Diseñador de tablas](#), escriba el texto del comentario en el área de edición de **Comentario de campo**.

–O bien–

Utilice la función `DBSETPROP()`.

- Utilice la función [DBSETPROP\(\)](#).

Por ejemplo, puede ser conveniente aclarar el contenido del campo `unit_price` de la tabla `orditems` con el comentario “Precio actual por unidad” como texto asociado al campo:

```
?DBSETPROP('orditems.price', 'field', 'comment', ;
           'Precio de venta al por menor actual')
```

Si desea más información sobre el uso de `DBSETPROP()` para establecer propiedades de los campos de una tabla de base de datos, consulte el capítulo 6, [Crear bases de datos](#).

## Crear valores predeterminados de campo

Si desea que Visual FoxPro rellene automáticamente un campo al agregar un nuevo registro, puede crear un [valor predeterminado](#) para ese campo. El valor predeterminado se aplica al introducir datos mediante un [formulario](#), en una ventana [Examinar](#), en una [vista](#) o mediante programación, y permanece en el campo hasta que se escriba un nuevo valor.

Los valores predeterminados se crean con el Diseñador de tablas o a través del lenguaje. Es posible especificar valores predeterminados para todos los tipos de datos, excepto [General](#).

### Para asignar un valor predeterminado a un campo de una tabla de base de datos

- En el [Diseñador de tablas](#), introduzca el valor en el cuadro de texto **Valor predeterminado** de la sección **Propiedades de campo**.

–O bien–

- Utilice la cláusula `DEFAULT` del comando [CREATE TABLE](#).

Por ejemplo, puede estimar conveniente que su aplicación limite la mercancía que puede pedir un nuevo cliente hasta que haya tenido tiempo de realizar una comprobación de su crédito y determinar la cantidad que desea autorizarle. En el ejemplo siguiente se crea un campo `maxordamt` con un valor predeterminado de 1000:

```
CREATE TABLE customer (cust_id C(6), company C(40), contact C(30), ;
                        maxordamt Y(4) DEFAULT 1000)
```

Si la tabla `customer` ya incluye la columna `maxordamt`, podrá agregar un valor predeterminado para la columna con el comando siguiente:

```
ALTER TABLE customer ALTER COLUMN maxordamt SET DEFAULT 1000
```

### Usar valores predeterminados para acelerar la escritura de datos

Puede utilizar [valores predeterminados](#) para acelerar la escritura de datos por parte de los usuarios de la aplicación, permitiéndoles así saltar un campo si no desean introducir en él un valor distinto. Por ejemplo, si en su negocio trata principalmente con clientes nacionales, puede hacer que el campo `país` de la tabla `cliente` de una base de datos se rellene automáticamente con el nombre de su país. Al introducir el registro de un cliente extranjero, puede sobrescribir el nombre de su país con el del

Al introducir el registro de un cliente extranjero, puede sobrescribir el nombre de su país con el del país correspondiente.

**Sugerencia** Si alguna de las reglas de la aplicación requiere que un campo contenga siempre un valor, el uso de un valor predeterminado ayudará a asegurar que se cumpla una regla concreta a [nivel de campo](#) o de [registro](#).

Si quita o elimina una tabla de una base de datos, todos los valores predeterminados de la tabla se eliminarán de la base de datos. Los [procedimientos almacenados](#) a los que haga referencia el valor predeterminado quitado o eliminado permanecerán aún después de la eliminación.

Cuando no especifique un valor predeterminado, se insertará un valor en blanco (tal y como se define para cada tipo de datos), salvo en el caso en que esté activado [SET NULL](#). De esta forma se mantiene la compatibilidad con el código de FoxPro de versiones anteriores que pueda conservar.

Puede indicar .NULL. como valor predeterminado si desea que el campo utilice valores nulos. Está activado o no SET NULL, si utiliza .NULL. como valor predeterminado, Visual FoxPro insertará .NULL. para todos los comandos excepto APPEND BLANK.

### Valores predeterminados permitidos

Puede especificar valores predeterminados escalares (por ejemplo, 'un número'), o expresiones que se evalúen como una cantidad escalar. También puede especificar cualquier expresión de Xbase válida que devuelva un valor coherente con el tipo de datos del campo.

Visual FoxPro evalúa las [expresiones](#) para el tipo de datos cuando se cierra la estructura de la tabla. Si el tipo de datos no coincide con el tipo asociado al campo, Visual FoxPro generará un error. Si la expresión es una [función definida por el usuario](#) (FDU) o contiene una FDU, no se evaluará.

Al crear el valor predeterminado por medio del lenguaje, los comandos CREATE TABLE y ALTER TABLE generarán un error cuando los tipos de datos no coincidan. Si la expresión es una FDU o contiene una FDU, no se evaluará al ejecutar CREATE y no se generará ningún error.

### Cuándo se aplican los valores predeterminados

Los [valores predeterminados](#) se evalúan (si es necesario) y se colocan en los campos correspondientes cuando se ejecutan los comandos APPEND, APPEND BLANK o INSERT.

Al asignar valores con los comandos APPEND FROM o INSERT - SQL, Visual FoxPro asignará valores predeterminados a todos los campos que no los tengan asignados explícitamente. Los comandos APPEND FROM e INSERT - SQL también respetan los valores predeterminados. Sin embargo, cuando se utiliza alguno de estos comandos, los valores predeterminados no sobrescriben a los valores existentes en los campos. Si los campos anexados o insertados contienen valores, los valores existentes se conservarán al anexar o insertar el registro, y no se utilizará el valor predeterminado.

### Usar valores predeterminados para rellenar automáticamente campos NOT NULL

Los [valores predeterminados](#) son especialmente útiles para rellenar automáticamente los campos que



no permiten [valores nulos](#). Al agregar un nuevo registro, los valores predeterminados se aplican en primer lugar y después se comprueba cada campo según su orden de definición para determinar si falta información. Así se asegura que los campos designados como NOT NULL tengan ocasión de rellenarse con valores predeterminados antes de aplicar la restricción NOT NULL.

### Especificar una máscara de entrada

Al especificar una [máscara de entrada](#), define la puntuación, el espacio y otros atributos de formato de valores que se introducen en el campo. Los valores se almacenan así de una forma uniforme que puede reducir los errores de entrada de datos, haciendo que se procesen de forma más eficaz. Por ejemplo, al agregar una máscara a un campo numérico que almacena números de teléfono ayuda al usuario a rellenar rápidamente el campo porque la puntuación y los espacios ya los proporciona la máscara.

### Para proporcionar una máscara de entrada

- En el [Diseñador de tablas](#), escriba la máscara en el cuadro **Máscara de entrada** en el área **Mostrar**.

–O bien–

- Use la función [DBSETPROP\(\)](#) para establecer la propiedad InputMask.

Por ejemplo, el código siguiente especifica una máscara de entrada para una fecha:

```
DBSetProp("orders.postalcode","field","InputMask","99999-9999")
```

### Controlar la presentación de un campo

Las propiedades adicionales para campos le permiten controlar cómo aparecen un campo y sus valores en [formularios](#), [ventanas Examinar](#) e informes. Puede especificar un formato de presentación, un [título](#) de campo predeterminado y una [clase](#) y una [biblioteca de clases](#) predeterminadas.

### Definir un formato

Un formato proporciona una máscara de salida que determina la manera en que se presenta el valor de un campo en un formulario, una ventana Examinar o un informe. Por ejemplo,

### Para proporcionar un formato

- En el [Diseñador de tablas](#), escriba la máscara en el cuadro **Formato** en el área **Mostrar**.

–O bien–

- Use la función [DBSETPROP\(\)](#) para establecer la propiedad Format.

Por ejemplo, el código siguiente especifica un formato de presentación para una tarjeta postal:

```
DBSetProp("orders.postalcode","field","Format","@R 99999-9999")
```

## Crear títulos para campos

Puede crear un [título](#) para cada campo de una tabla de base de datos. Visual FoxPro muestra el texto del título de un campo como encabezado de columna en una ventana Examinar y como nombre de encabezado predeterminado en una cuadrícula de formulario.

### Para agregar un título a un campo de una tabla de base de datos

- En el [Diseñador de tablas](#), escriba el texto para el título en el cuadro **Título** de la sección **Mostrar**.

–O bien–

- Use la función [DBSETPROP\(\)](#).

Por ejemplo, es posible que quiera crear un título para el campo fax de la tabla supplier escribiendo “Supplier\_Fax” como título del campo:

```
?DBSETPROP('supplier.fax', 'field', 'caption', 'Supplier_Fax')
```

Para obtener más información sobre el uso de DBSETPROP( ) para establecer propiedades de campos de tablas de base de datos, consulte el capítulo 6, [Crear bases de datos](#).

## Establecer una clase predeterminada

Para ahorrar tiempo más tarde al crear [formularios](#) puede establecer una [clase](#) predeterminada para un campo. Una vez establecida, cada vez que agrega un campo a un formulario, el [control](#) del formulario usa la clase que especifique como predeterminada. Por ejemplo, los campos de tipo Character aparecen automáticamente como controles cuadro de texto cuando los agrega al formulario. Si en lugar de esto quiere agregar automáticamente un control cuadro combinado cuando use el campo en un formulario, puede establecer la clase como predeterminada para este campo. También puede usar [bibliotecas de clases](#) que haya creado.

### Para establecer una clase predeterminada

- En el [Diseñador de tablas](#), elija una clase y una biblioteca en los cuadros **Mostrar clase** y **Mostrar biblioteca**.

Si cambia a menudo la biblioteca y la clase para sus campos, puede asignar los tipos de datos de los campos a una biblioteca y a una clase en el [cuadro de diálogo Opciones](#). Para obtener más información sobre la asignación de tipos de datos de campo a clases, consulte el capítulo 3, [Configurar Visual FoxPro](#), de la *Guía de instalación e Índice principal*. Para obtener más información sobre la creación de clases, consulte el capítulo 3, [Programación orientada a objetos](#), en este manual.

## Exigir reglas comerciales

Puede exigir reglas comerciales para la introducción de datos si crea a nivel de campo y a nivel de

registro las llamadas [reglas de validación](#), con las que se pueden controlar los datos introducidos en los campos y registros de las tablas de base de datos. Las reglas a nivel de campo y a nivel de registro comparan los valores introducidos con expresiones definidas previamente. Si el valor introducido no satisface los requisitos de la regla, se rechazará el valor. Las reglas de validación sólo pueden darse para las tablas de base de datos.

Las reglas a nivel de campo y a nivel de registro permiten controlar el tipo de información introducido en una tabla, tanto si el acceso a los datos se realiza mediante una [ventana Examinar](#), un [formulario](#) o mediante programación. Las reglas permiten exigir su cumplimiento en un campo con menos código que si se escribiera la expresión de la regla en una cláusula VALID de un formulario o en un fragmento de código del programa. Además, las reglas establecidas para una base de datos se aplicarán a todos los usuarios de la tabla, independientemente de los requisitos de la aplicación.

También puede crear índices [candidatos](#) o [principales](#) que eviten entradas duplicadas en un campo, y desencadenantes que exijan la [integridad referencial](#) o que realicen otras acciones cuando se modifica la información de la base de datos.

### Cuándo se establecen las restricciones

Las restricciones de la base de datos se eligen basándose en el nivel al que se desea exigir una regla comercial o de integridad referencial, y también la acción que causa que la restricción se active. En la tabla siguiente se enumeran las restricciones de validación de datos en el orden en que las exige el motor de Visual FoxPro, el nivel al que se aplican y cuándo activa el motor la validación.

Mecanismo de exigencia	Nivel	Activación
Validación NULL	Campo o columna	Al salir del campo o columna en una ventana Examinar o cuando el valor del campo cambia a causa de INSERT o REPLACE.
Reglas a nivel de campo	Campo o columna	Al salir del campo o columna en una ventana Examinar o cuando el valor del campo cambia a causa de INSERT o REPLACE.
Reglas a nivel de registro	Registro	Al producirse la actualización del registro.
Índice candidato o principal	Registro	Al producirse la actualización del registro.
Cláusula VALID	Formulario	Al salir del registro.
Desencadenantes	Tabla	Al cambiar los valores de la tabla a causa de un evento INSERT, UPDATE o DELETE.

Las restricciones se activan en el orden en que aparecen en la tabla. El primer incumplimiento de una

validación detiene el comando correspondiente.

Los índices candidatos y principales se tratan más adelante en este capítulo, dentro de la sección ["Controlar los valores duplicados"](#).

### Limitar los valores de un campo

Cuando desee controlar el tipo de información que un usuario puede introducir en un campo y sea posible validarlo independientemente de las restantes entradas del registro, puede utilizar una regla de validación a nivel de campo. Por ejemplo, puede utilizar una regla de validación a nivel de campo para asegurarse de que el usuario no escriba un número negativo en un campo que sólo debe contener valores positivos. También puede utilizar una regla a nivel de campo para comparar los valores introducidos en un campo con los de otra tabla.

No debe crear reglas específicas de la aplicación a nivel de campo o a nivel de registro. Utilícelas para exigir reglas de integridad de datos y reglas comerciales que sean siempre aplicables a la información de la base de datos, independientemente de quién tenga acceso a la misma. Por ejemplo, podría crear una regla que comparase el valor introducido en el campo `código postal` de una tabla con una tabla de búsqueda con los códigos abreviados de su país y que rechazase los valores que no fueran una abreviatura válida de código postal.

### Para crear una regla a nivel de campo

- En el [Diseñador de tablas](#), introduzca la expresión de la regla en el cuadro de texto **Regla** de la sección **Validación de campos**.

–O bien–

- Utilice la cláusula CHECK del comando [CREATE TABLE](#).

–O bien–

- Utilice la cláusula SET CHECK del comando [ALTER TABLE](#).

Por ejemplo, el código siguiente agrega a la tabla `orditems` una regla de validación a nivel de campo que requiere que el número introducido en el campo `quantity` sea mayor o igual que 1:

```
ALTER TABLE orditems
  ALTER COLUMN quantity SET CHECK quantity >= 1
```

Cuando el usuario intente introducir un valor menor que 1, Visual FoxPro mostrará un mensaje de error y rechazará el valor.

Puede personalizar el mensaje que aparece cuando se infringe la regla si agrega texto de validación al campo. El texto especificado se mostrará en el cuadro de diálogo en lugar del mensaje de error predeterminado.

### Para agregar un mensaje de error personalizado a una regla a nivel de campo

- En el [Diseñador de tablas](#), escriba el mensaje de error que desea en el cuadro **Mensaje del área**

- En el [DISEÑADOR DE TABLAS](#), escriba el mensaje de error que desee en el cuadro **Mensaje** del área **Validación de campos**.

–O bien–

- Utilice la cláusula opcional **ERROR** con la cláusula **CHECK** de los comandos [CREATE TABLE](#) o [ALTER TABLE](#).

Por ejemplo, el código siguiente agrega una regla de validación a nivel de campo para la tabla `orditems` que requiere que los números introducidos en la columna `quantity` sean mayores o iguales que 1 y además define un mensaje de error personalizado:

```
ALTER TABLE orditems ;
  ALTER COLUMN quantity SET CHECK quantity >= 1 ;
  ERROR "Las cantidades deben ser mayores o iguales que 1"
```

Cuando el usuario intente escribir un valor menor que 1, Visual FoxPro mostrará un cuadro de diálogo de error con el mensaje de error personalizado que usted ha definido y rechazará el valor. También puede utilizar la cláusula **SET CHECK** del comando **ALTER TABLE** con la cláusula opcional **ERROR** para crear un mensaje de error personalizado.

### Cuándo se comprueban las reglas a nivel de campo

Las reglas a nivel de campo se comprueban cuando se modifica el valor del campo. Al contrario que los [desencadenantes](#), las reglas a nivel de campo se activan incluso cuando los datos se encuentran en búfer. Al trabajar con los datos en una ventana Examinar, en un formulario u otra ventana de interfaz de usuario, Visual FoxPro comprueba las reglas a nivel de campo al salir del campo correspondiente. Si el valor del campo no ha cambiado, la regla no se comprobará. Esto significa que puede desplazarse por los campos sin que el sistema valide su contenido.

### Comprobación de las reglas a nivel de campo

Método de introducción de datos	Ventana o comando	Comprobación de la regla a nivel de campo
Interfaz de usuario	<a href="#">Ventana Examinar</a> <a href="#">Formulario</a> Otra ventana	Al salir del campo, si el valor ha variado. (Si el valor no cambia, la regla no se comprueba).
Comandos que no especifican campos	<a href="#">APPEND</a> <a href="#">APPEND GENERAL</a> <a href="#">APPEND MEMO</a> <a href="#">BROWSE</a> <a href="#">CHANGE</a> <a href="#">DELETE</a> <a href="#">EDIT</a> <a href="#">GATHER</a>	Al cambiar el valor del campo, en el orden de definición de los campos.
	<a href="#">APPEND BLANK</a> <a href="#">INSERT</a> <a href="#">INSERT . SOI</a>	Al anexar o insertar el registro.

<a href="#">INSERT - SQL</a>		
Comandos que especifican campos	<a href="#">UPDATE</a> <a href="#">UPDATE - SQL</a> <a href="#">REPLACE</a>	En el orden en que aparecen los campos en el comando.

### Validar valores a nivel de registro

Las reglas de validación a nivel de registro permiten controlar el tipo de información que el usuario puede introducir en los registros. Normalmente, las reglas de validación a nivel de registro comparan los valores de dos o más campos del mismo registro para asegurarse de que cumplen las reglas comerciales establecidas para la base de datos. Por ejemplo, puede utilizar una regla de validación a nivel de registro para asegurarse de que el valor de un campo sea siempre superior al de otro campo del mismo registro.

### Para crear una regla de validación a nivel de registro y un mensaje de error personalizado

- En la ficha **Tabla** del [Diseñador de tablas](#), escriba la regla y el mensaje de error que desee en los cuadros **Regla** y **Mensaje**.

–O bien–

- Utilice la cláusula CHECK de los comandos [CREATE TABLE](#) o [ALTER TABLE](#).

Por ejemplo, puede querer asegurarse de que los empleados tengan 18 o más años en el momento de su contratación. En el código siguiente se agrega a la tabla `employee` una regla de validación a nivel de registro y un texto de error para requerir que la fecha de contratación introducida en `hire_date` sea mayor o igual que la fecha de nacimiento más 18 años:

```
ALTER TABLE employee SET CHECK ;
hire_date >= birth_date + (18 * 365.25) ;
ERROR "Los empleados deben tener 18 años en la fecha de contratación"
```

Si el usuario introduce un registro de empleado con una fecha no válida, Visual FoxPro mostrará un cuadro de diálogo de error con el mensaje de error personalizado definido y no actualizará el registro.

También puede utilizar la cláusula SET CHECK del comando [ALTER TABLE](#) para crear una regla de validación a nivel de registro. Debe asegurarse de que las reglas especificadas para los campos no entren en conflicto semántico con las definidas para la tabla. Visual FoxPro no comprueba la coherencia entre las expresiones a nivel de campo y a nivel de registro.

### Cuándo se comprueban las reglas a nivel de registro

Las reglas a nivel de registro, al igual que las reglas a nivel de campo, se activan cuando el valor del

registro cambia. Independientemente de la forma de trabajar con los datos, ya sea en una ventana Examinar, en un formulario, en otra ventana de interfaz de usuario o mediante comandos que alteren los datos, Visual FoxPro comprobará las reglas a nivel de registro a medida que usted desplace el puntero de registro fuera del registro. Si no ha cambiado ningún valor del registro, la regla a nivel de registro no se comprobará al desplazar el puntero, por lo que es posible desplazarse por los registros sin que el sistema valide los datos que contienen.

Si modifica un registro, pero no desplaza el puntero y cierra la ventana Examinar, la regla se comprobará, aparecerá un aviso si se producen errores y luego se cerrará la ventana Examinar.

**Precaución** No incluya en las reglas de validación comandos o funciones que intenten desplazar el puntero de registro del área de trabajo actual (es decir, del [área de trabajo](#) cuyas reglas se están comprobando). El uso de comandos o funciones tales como SEEK, LOCATE, SKIP, APPEND, APPEND BLANK, INSERT o AVERAGE, COUNT, BROWSE y REPLACE FOR en las reglas de validación puede causar su desencadenamiento recursivo, produciendo una condición de error.

Al contrario que los [desencadenantes](#), las reglas a nivel de registro se activan incluso cuando los datos se encuentran en búfer. Para el caso de que durante la ejecución de una aplicación se active una regla a nivel de registro, es necesario incluir código de control de errores. Normalmente, este código no permitirá que la aplicación abandone el formulario (o que cambie el entorno activo, hablando de forma más genérica), hasta que el usuario corrija el error detectado o cancele la actualización.

### Quitar de una base de datos una tabla con reglas asociadas

Si quita o elimina una tabla de una base de datos, se eliminarán todas las reglas a nivel de campo y a nivel de registro de la base de datos. Esto se debe a que las reglas se almacenan en el archivo .dbc, y al quitar una tabla de la base de datos se rompe el vínculo entre el archivo .dbf y su archivo .dbc. Sin embargo, los [procedimientos almacenados](#) a los que haga referencia la regla quitada o eliminada no se eliminan automáticamente, ya que aún pueden utilizarlos otras reglas de otras tablas que permanezcan en la base de datos.

### Usar desencadenantes

Un desencadenante es una expresión dependiente de una tabla que se invoca cuando se modifica alguno de los registros de la tabla con alguno de los comandos de modificación de datos especificados. Los desencadenantes pueden utilizarse para realizar cualquier operación que requiera una aplicación de base de datos cuando se modifique la información. Por ejemplo, puede utilizar desencadenantes para llevar a cabo las acciones siguientes:

- Registrar las modificaciones de la base de datos.
- Exigir la [integridad referencial](#).
- Crear un pedido automáticamente para un producto cuyas existencias están a punto de agotarse.

Los desencadenantes se crean y almacenan como [propiedades](#) de una tabla específica. Si quita una tabla de una base de datos, los desencadenantes asociados a la misma se eliminarán. Los desencadenantes se activan una vez realizadas todas las comprobaciones restantes, como las reglas de validación, la exigencia de clave principal y la de valor nulo. Al contrario de lo que ocurre con las reglas de validación a nivel de campo y a nivel de registro, los desencadenantes no se activan para los datos almacenados en búfer.

## Crear desencadenantes

Para crear desencadenantes puede utilizar el Diseñador de tablas o el comando CREATE TRIGGER. Puede crear un desencadenante para cada tabla y para cada uno de los tres eventos INSERT, UPDATE y DELETE. En un momento dado, una tabla puede tener un máximo de tres desencadenantes. Los desencadenantes deben devolver un valor verdadero (.T.) o falso (.F.).

### Para crear un desencadenante

- En el cuadro de diálogo **Tabla** del [Diseñador de tablas](#), escriba la expresión del desencadenante o el nombre del procedimiento almacenado que la contiene en los cuadros **Insertar desencadenante**, **Actualizar desencadenante**, o **Eliminar desencadenante**.

–O bien–

- Utilice el comando [CREATE TRIGGER](#).

Por ejemplo, puede que cada vez que Importadores Tasmanian venda un producto desee comparar las unidades en existencias `Units_in_stock` con el nivel del nuevo pedido `Reorder_level` y recibir una notificación si es necesario volver a pedir ese producto. Podría crear un desencadenante de actualización para la tabla `products` para conseguirlo. El desencadenante elegido es de actualización (y no de eliminación o inserción), ya que debe activarse cada vez que se venda un producto, y el campo `Units_in_stock` se actualizará cada vez que se venda un producto para reflejar los que queden en existencias.

Para crear el desencadenante puede especificar `updProductsTrigger( )` como desencadenante de actualización de la tabla `products`. Puede agregar un campo a `products`, llamado `reorder_amount`, para almacenar el importe que desee pedir cada vez que realice un nuevo pedido del producto en cuestión y crear una tabla `reorder` con los campos: `product_id` y `reorder_amount`. A continuación puede agregar este código al procedimiento almacenado:

```
PROCEDURE updProductsTrigger
  IF (units_in_stock+units_on_order) <= reorder_level
    INSERT INTO Reorder VALUES(Products.product_id, ;
      Products.reorder_amount)
  ENDIF
ENDPROC
```

Puede crear desencadenantes similares para los eventos de inserción o eliminación mediante las cláusulas FOR INSERT o FOR DELETE, respectivamente, en lugar de la cláusula FOR UPDATE. Si intenta crear un desencadenante que ya existe para un evento y una tabla determinados y está activado [SET SAFETY](#), Visual FoxPro le preguntará si desea sobrescribir el desencadenante existente.

## Quitar o eliminar desencadenantes

Puede quitar un desencadenante de una tabla de base de datos mediante la interfaz o el comando DELETE TRIGGER.

### Para eliminar un desencadenante



- En el cuadro de diálogo **Tabla** del [Diseñador de tablas](#), seleccione la expresión de desencadenante en el cuadro **Insertar desencadenante**, **Actualizar desencadenante** o **Eliminar desencadenante** y elimínela.

–O bien–

- Utilice el comando [DELETE TRIGGER](#).

En el ejemplo siguiente se elimina el desencadenante de actualización de la tabla `customer`:

```
DELETE TRIGGER ON customer FOR UPDATE
```

Si quita o elimina una tabla de una base de datos, todos los desencadenantes dependientes de ella se eliminarán de la base de datos. Sin embargo, los [procedimientos almacenados](#) a los que haga referencia el desencadenante quitado o eliminado no se eliminarán.

## Modificar desencadenantes

Puede modificar los desencadenantes mediante el Diseñador de tablas o a través del lenguaje.

### Para modificar un desencadenante

- En el cuadro de diálogo **Tabla** del [Diseñador de tablas](#), introduzca la nueva expresión del desencadenante en el cuadro **Insertar desencadenante**, **Actualizar desencadenante** o **Eliminar desencadenante**.

–O bien–

- Ejecute el comando [SET SAFETY OFF](#), seguido del comando [CREATE TRIGGER](#).

Si modifica un desencadenante utilizando primero el comando [SET SAFETY OFF](#) y volviendo a crear después el desencadenante, la expresión anterior se eliminará y se sustituirá automáticamente con la expresión recién creada.

## Usar desencadenantes para generar integridad referencial

Visual FoxPro cuenta con un [Generador de integridad referencial](#) para generar desencadenantes y procedimientos almacenados que exijan la integridad referencial (IR) de la base de datos. Si desea más información sobre el uso del Generador de integridad referencial, consulte el capítulo 6, [Crear bases de datos](#).

## Modificar la estructura de la tabla

Una vez creada una tabla, siempre puede modificar su estructura y sus propiedades. Puede ser conveniente agregar, modificar o eliminar los nombres, anchos y tipos de datos de los campos, así como sus valores predeterminados o sus reglas, y también agregar comentarios o títulos.

Para modificar la estructura de una tabla puede abrir el Diseñador de tablas o bien puede realizar los

Para modificar la estructura de una tabla puede usar el Diseñador de tablas o bien puede realizar los cambios mediante programación con el comando ALTER TABLE. Asegúrese de que tiene acceso exclusivo a la tabla antes de modificar su estructura.

### Para modificar la estructura de una tabla con el Diseñador de tablas

- En el [Administrador de proyectos](#), seleccione el nombre de la tabla y, a continuación, elija **Modificar**.  
–O bien–
- En el [Diseñador de bases de datos](#), seleccione la tabla en el esquema y elija **Modificar** en el menú **Base de datos**.  
–O bien–
- Utilice el comando [MODIFY STRUCTURE](#).

Por ejemplo, puede modificar la estructura de la tabla de base de datos `employee` con los comandos siguientes:

```
OPEN DATABASE testdata
USE employee EXCLUSIVE
MODIFY STRUCTURE
```

Todas las opciones anteriores abren el Diseñador de tablas.

### Para modificar la estructura de una tabla por programa

- Utilice el comando [ALTER TABLE](#).

El comando ALTER TABLE ofrece diversas cláusulas que permiten agregar o eliminar campos, crear o eliminar claves principales o únicas, o etiquetas de claves externas, así como cambiar el nombre de los campos existentes. Algunas cláusulas sólo son aplicables a las tablas asociadas a una base de datos. En esta sección se incluyen algunos ejemplos específicos.

### Agregar campos

Para agregar un nuevo campo a una tabla puede utilizar el Diseñador de tablas o hacerlo a través del lenguaje.

#### Para agregar un campo a una tabla

- En el [Diseñador de tablas](#), elija **Insertar**.  
–O bien–
- Utilice la cláusula ADD COLUMN del comando [ALTER TABLE](#).

Por ejemplo, el comando siguiente agrega un campo llamado `fax` a la tabla `customer` y permite que

el nuevo campo tenga valores nulos:

```
ALTER TABLE customer ADD COLUMN fax c(20) NULL
```

## Eliminar campos

Puede eliminar un campo existente de una tabla mediante el Diseñador de tablas o a través del lenguaje.

### Para eliminar un campo de una tabla

- En el [Diseñador de tablas](#), seleccione el campo y elija **Eliminar**.  
–O bien–
- Utilice la cláusula DROP COLUMN del comando [ALTER TABLE](#).

Por ejemplo, el comando siguiente elimina el campo fax de la tabla customer:

```
ALTER TABLE customer DROP COLUMN fax
```

Al quitar un campo de una tabla también se quita su valor predeterminado, sus definiciones de reglas y su título. Si hay referencias al campo en la clave de índice o en expresiones de desencadenante, estas expresiones dejarán de ser válidas cuando se elimine el campo. Las expresiones no válidas de clave de índice o de desencadenante no generarán errores hasta el [tiempo de ejecución](#).

## Cambiar el nombre de los campos

Hay dos formas de cambiar el nombre de los campos existentes en una tabla.

### Para cambiar el nombre de un campo de una tabla

- En el [Diseñador de tablas](#), escriba un nuevo nombre en el cuadro de texto **Nombre** del campo correspondiente.  
–O bien–
- Utilice la cláusula RENAME COLUMN del comando [ALTER TABLE](#).

Por ejemplo, para cambiar el nombre de la columna company de la tabla customer, podría utilizar el comando siguiente:

```
ALTER TABLE customer RENAME COLUMN company TO company_long_new_name
```

En el ejemplo anterior, el nuevo nombre del campo aprovecha la posibilidad de crear nombres largos de campo de la que disfrutaban las tablas de base de datos.

## Establecer o modificar las reglas a nivel de campo o de tabla

Puede establecer nuevas expresiones y texto de reglas a nivel de campo o de tabla, así como modificar las reglas y el texto especificados con los comandos CREATE TABLE o ALTER TABLE.

### Para cambiar una regla existente

- En el [Diseñador de tablas](#), seleccione la ficha **Tabla** y escriba la nueva expresión de regla o texto de regla en los cuadros **Regla** y **Mensaje** de la sección **Validación de registros**.  
–O bien–
- Utilice el comando [ALTER TABLE](#).

Puede utilizar la función [DBGETPROP\(\)](#) para ver la expresión o el texto actual de una regla; estos valores son de sólo lectura para las tablas y sólo se pueden modificar mediante el comando ALTER TABLE.

### Establecer o modificar los valores predeterminados

Una vez generada una tabla, puede establecer o cambiar los valores predeterminados de sus campos.

#### Para cambiar un valor predeterminado existente

- En el [Diseñador de tablas](#), escriba el nuevo valor en el cuadro **Valor predeterminado** de la ficha **Campos**.  
–O bien–
- Utilice el comando [ALTER TABLE](#).

Puede utilizar la función [DBGETPROP\(\)](#) para ver el valor predeterminado actual de un campo; estos valores son de sólo lectura para las tablas y sólo se pueden cambiar mediante el comando ALTER TABLE.

## Trabajar con registros

Una vez diseñada y creada la estructura de una tabla, puede almacenar datos en ella agregándole registros. Después podrá modificar y eliminar los registros existentes. Cada una de estas tareas puede realizarse a través de la interfaz o mediante comandos. En esta sección se tratará principalmente el trabajo con los registros por programa. Si desea más información sobre el trabajo con registros a través de la interfaz, consulte el capítulo 2, [Crear tablas e índices](#), del *Manual del usuario*.

### Agregar registros

Cuando cree una tabla de Visual FoxPro, la tabla estará abierta, pero vacía. Si intenta almacenar datos en una tabla sin crear primero un registro en ella, no sucederá nada. El primer paso para agregar registros a una tabla nueva es agregar filas en las que almacenar los nuevos datos.

#### Para agregar registros a una tabla

o o o

- Utilice el comando [INSERT - SQL](#).

El comando [INSERT - SQL](#) puede utilizarse para insertar valores indicados en el mismo o tomados de una [matriz](#) o [variable](#) de memoria. Por ejemplo, para insertar un nuevo registro en la tabla `customer` de la base de datos `TasTrade`, puede utilizar el comando siguiente:

```
INSERT INTO customer (cust_id, company, contact) ;
VALUES ("SMI007", "Calzados Arbor", "Daniel Rendich")
```

El comando `INSERT - SQL` es útil en el caso de datos remotos, ya que sigue una sintaxis `SQL` compatible con [ANSI](#).

Para agregar un registro en blanco a una tabla y almacenar datos en sus campos puede utilizar el comando `APPEND BLANK` seguido del comando `REPLACE`. El comando `APPEND BLANK` anexa un nuevo registro en blanco a la tabla. El comando `REPLACE` sustituye el valor actual de un campo (incluso si está vacío) con un nuevo valor.

El comando [REPLACE](#) requiere:

- Una [tabla](#) abierta.
- Un [registro](#) existente.
- El nombre del [campo](#) en el que desea almacenar el valor.
- Un valor para cada campo válido según el [tipo de datos](#) correspondiente.

En el ejemplo siguiente se utiliza el comando [APPEND BLANK](#) para crear un registro en el que almacenar datos con el comando `REPLACE`:

```
APPEND BLANK                                && registro disponible ahora
REPLACE lastname WITH "SMITH"               && almacenar valor character en el campo
```

Puede utilizar el comando [UPDATE - SQL](#) en lugar de `REPLACE` para actualizar los registros de una tabla.

## Agregar registros de otra tabla

Otra forma de almacenar datos en los registros es copiarlos de otras tablas o archivos. Por ejemplo, puede [anexar](#) registros de otra tabla o archivo.

### Para anexar registros de otro archivo

- Utilice el comando [APPEND FROM](#).

–O bien–

- Utilice el comando [IMPORT](#).

Los registros pueden aceptar los datos directamente, como en el ejemplo anterior, en el que el

comando INSERT especificaba el texto que se va a insertar en campos específicos de la tabla `customer` y también tomarlos de [constantes](#), [variables](#), [matrices](#), [objetos](#) u otros orígenes de datos. Si desea más información sobre otras formas de importar datos, consulte el capítulo 9, [Importar y exportar datos](#), del *Manual del usuario*.

## Agregar registros en modo Examinar

Si desea agregar un nuevo registro a una tabla mientras se encuentra en modo Examinar, puede elegir Anexar registro en el [menú Tabla](#). Por el contrario, si desea evitar que los usuarios puedan anexar registros en el modo Examinar, puede utilizar la cláusula NOAPPEND del comando [BROWSE](#).

## Introducir datos en una tabla

Puede introducir datos en una tabla de forma interactiva, a través de una [ventana Examinar](#) o mediante programación, con los comandos [REPLACE](#) o [UPDATE - SQL](#). Al utilizar REPLACE o UPDATE - SQL en una aplicación multiusuario, puede activar el almacenamiento en búfer de registros o tablas, lo que le permitirá modificar los datos sin bloquear el registro hasta que desee hacer efectivos los cambios. Si desea más información sobre el búfer de registros y tablas, consulte el capítulo 17, [Programar para acceso compartido](#).

## Modificar los registros de una tabla

Para mostrar y modificar los registros existentes en una tabla puede utilizar la interfaz o hacerlo mediante programación.

### Para mostrar los registros y modificarlos

- Utilice el comando [EDIT](#).
- O bien–
- Utilice el comando [CHANGE](#).

Por ejemplo, el código siguiente muestra la tabla `customer` en una ventana Examinar en modo de edición:

```
USE customer
EDIT
```

Si desea utilizar un [formulario](#) para modificar un registro, cree un [cuadro de texto](#) en el formulario y establezca su [propiedad DataSource](#) como el nombre de la tabla que desee modificar. Si desea más información sobre los formularios, consulte el capítulo 9, [Crear formularios](#).

También puede utilizar los comandos [CHANGE](#) y [EDIT](#) para modificar campos específicos de una tabla.

## Agregar gráficos a una tabla

Puede almacenar gráficos en una tabla de Visual FoxPro si crea un campo de tipo [general](#) e importa o

Puede almacenar gráficos en una tabla de Visual FoxPro si crea un campo de tipo [general](#) e importa o pega en él objetos OLE tales como mapas de bits o gráficos. El comando [APPEND GENERAL](#) sitúa un objeto OLE en un campo General. En el ejemplo siguiente se almacena un archivo gráfico de Microsoft Excel situado en el directorio predeterminado de Visual FoxPro en un campo General llamado Gráfico:

```
APPEND GENERAL Chart FROM "CHART1.CLX" CLASS EXCELCHART
```

Si desea más información sobre el manejo de objetos OLE en las tablas de Visual FoxPro, consulte el capítulo 16, [Agregar OLE](#).

## Introducir valores nulos en los campos

Para introducir un [valor nulo](#) en un campo puede utilizar el lenguaje, con el elemento .NULL, o hacerlo a través de la interfaz con una combinación de teclas si el campo admite valores nulos.

### Para almacenar un valor nulo en un campo

- En una ventana [Examinar](#) o en un control de formulario, presione CTRL+0 (cero).  
—O bien—
- Utilice el elemento .NULL.

Por ejemplo, el código siguiente sustituye el valor existente en el campo `automóvil` por un valor nulo:

```
REPLACE automóvil WITH .NULL
```

**Nota** Use el comando [SET NULLDISPLAY](#) para especificar el texto mostrado para los valores nulos.

## Eliminar registros

Para eliminar registros debe marcarlos primero para eliminación y luego suprimir los registros marcados. Hasta su supresión, los registros marcados permanecen en el disco y puede quitarles la marca para restaurarlos. En esta sección se explica cómo marcar, desmarcar y suprimir los registros de una tabla.

### Marcar registros para su eliminación

Para marcar registros para su eliminación puede utilizar la interfaz o bien el comando [DELETE - SQL](#).

### Para marcar un registro para su eliminación

- En una ventana [Examinar](#) haga clic en el marcador de eliminación para marcar el registro.  
—O bien—

- En el [menú Tabla](#), elija **Eliminar registros**.

–O bien–

- Utilice el comando [DELETE - SQL](#).

También puede utilizar el comando DELETE - SQL para especificar un intervalo de registros, así como una condición basada en una expresión lógica que los registros deben satisfacer para que se marquen para su eliminación. Por ejemplo, el código siguiente marca para eliminación todos los registros de productos que contienen 'T' en el campo `Discontinuo`:

```
USE products
DELETE FROM products WHERE discontinuo = .T.
BROWSE
```

Los registros marcados para su eliminación no se suprimen físicamente de la tabla hasta que no se especifique el comando [PACK](#). Al ver la tabla en la ventana Examinar, cada registro eliminado tendrá un indicador de eliminación, pero aún será visible en la tabla, siempre que [SET DELETED](#) esté desactivado. Si SET DELETED está activado, los registros marcados para su eliminación no aparecerán en la ventana Examinar.

La configuración del comando SET DELETED también afecta al acceso a los registros marcados para su eliminación por parte de los comandos que operan sobre registros.

## Desmarcar los registros marcados para su eliminación

Para desmarcar los registros marcados para su eliminación puede utilizar el comando RECALL. El comando RECALL sólo podrá recuperar los registros cuando aún no se haya especificado un comando PACK o ZAP, los cuales eliminan físicamente los registros de la tabla.

### Para quitar la marca de un registro marcado para su eliminación

- En una ventana [Examinar](#), haga clic en el marcador de eliminación para quitar el indicador del registro.

–O bien–

- En el menú [Tabla](#), elija **Desmarcar registros**.

–O bien–

- Utilice el comando [RECALL](#).

Con el comando RECALL puede especificar un intervalo de registros, así como una condición basada en una expresión lógica que los registros deben satisfacer para que se les quite la marca de eliminación. Por ejemplo, el código siguiente quita la marca de eliminación de todos los registros de productos que contengan 'T' en el campo `discontinuo`:



```
USE products
RECALL FOR discontinu = .T.
BROWSE
```

Cuando vea la tabla en la ventana Examinar, los registros especificados ya no tendrán la marca de eliminación.

## Suprimir registros marcados para eliminación

Una vez marcado un grupo de registros para su eliminación, puede suprimirlos de forma definitiva del disco a través de la interfaz o mediante el lenguaje.

### Para suprimir del disco los registros marcados para eliminación

- En una ventana [Examinar](#), elija **Quitar registros eliminados** en el menú **Tabla**.  
–O bien–
- Utilice el comando [PACK](#).

El comando PACK tiene dos cláusulas: MEMO y DBF. Si especifica PACK sin MEMO ni DBF, se suprimirán tanto los registros del archivo de tabla como los del archivo memo asociado. Por ejemplo, el código siguiente suprime los registros marcados para eliminación:

```
USE customer EXCLUSIVE
PACK
```

Para eliminar solamente los registros del archivo de tabla y dejar intactos los del archivo memo, utilice PACK DBF.

## Conservar espacio

La información de los [campos memo](#) de una tabla se almacena en un archivo memo asociado con el mismo nombre de la tabla y la extensión .FPT. Si desea eliminar el espacio no utilizado en el archivo memo, pero no quiere suprimir los registros de la tabla marcados para eliminación, ejecute el comando [PACK](#) con la cláusula MEMO. Asegúrese de tener acceso exclusivo a la tabla.

## Suprimir todos los registros de una tabla

Si desea suprimir todos los registros de una tabla, dejando sólo su estructura, puede utilizar el comando [ZAP](#). El uso de ZAP equivale a especificar DELETE ALL y a continuación PACK, pero ZAP es mucho más rápido. Asegúrese de tener acceso exclusivo a la tabla.

**Precaución** Los registros de la tabla actual suprimidos con ZAP no pueden recuperarse.

## Indexar tablas

Para desplazarse por los registros de una tabla, verlos o manipularlos en un orden determinado, debe

utilizar un índice. Visual FoxPro utiliza los índices como mecanismos de ordenación para ofrecer flexibilidad y eficacia al programar una aplicación. La flexibilidad supone crear y utilizar múltiples claves de índice distintas para la misma tabla, lo que permite trabajar con los registros ordenados de formas diferentes, según las necesidades de la aplicación. La potencia implica crear relaciones personalizadas entre las tablas, basadas en sus índices, lo que le permite tener acceso a los registros que desea.

Un índice de Visual FoxPro es un archivo que contiene punteros ordenados lógicamente según los valores de una clave de índice. El archivo de índice es independiente del archivo .DBF de la tabla, y no cambia el orden físico de los registros contenidos en la misma. Al crear un índice se crea un archivo que mantiene punteros a los registros del archivo .DBF. Cuando desee trabajar con los registros de la tabla en un orden determinado, elija un índice para controlar el orden en que se ve la tabla y se tiene acceso a ella.

## Crear un índice

Al crear una tabla, Visual FoxPro crea el archivo .DBF correspondiente y, si la tabla contiene campos de tipo [Memo](#) o [General](#), el archivo .FPT asociado. En este momento no se genera ningún archivo de índice. Los registros introducidos en la nueva tabla se almacenarán en el orden de introducción, y al examinar la tabla aparecerán en ese orden.

Normalmente, será conveniente ver y tener acceso a los registros en un orden específico. Por ejemplo, puede ser adecuado ver los registros de la tabla de clientes ordenados alfabéticamente por nombres de empresa. Cuando se desea controlar el orden en que se muestran los registros y se tiene acceso a los mismos, es necesario crear un archivo de índice para la tabla creando la primera ordenación de registros, o clave de índice, de la tabla. A continuación puede establecer como orden de la tabla la nueva clave de índice y tener acceso a los registros siguiendo ese orden.

### Para crear una clave de índice de una tabla

- En el [Diseñador de tablas](#), elija la ficha Índices e introduzca la información de una clave de índice. Elija **Normal** como tipo de índice.

—O bien—

- Utilice el comando [INDEX](#).

Por ejemplo, en el código siguiente se utiliza la tabla `customer` y se crea una clave de índice sobre el campo `city`. La palabra clave TAG y la palabra 'city' que la sigue especifican un nombre, o 'etiqueta', para la nueva clave de índice asociada al campo de ciudad.

```
USE customer
INDEX ON city TAG city
```

En el ejemplo anterior, la etiqueta de la clave de índice tiene el mismo nombre que el campo de índice. Estos nombres no tienen que coincidir necesariamente (y podría haber elegido uno distinto para la etiqueta).

Al crear un índice con el comando INDEX, Visual FoxPro lo utiliza automáticamente para establecer el orden de los registros de la tabla. Por ejemplo, si introduce algunos datos en la tabla utilizada en el

el orden de los registros de la tabla. Por ejemplo, si introduce algunos datos en la tabla utilizada en el ejemplo anterior y a continuación la examina, los registros aparecerán ordenados por ciudad.

## Crear un archivo de índice

Al crear la primera clave de índice para la tabla del ejemplo anterior, Visual FoxPro creó automáticamente un nuevo archivo, Customer.cdx, para almacenar la nueva clave de índice. Las claves de índice se almacenan en archivos con la extensión .cdx. El archivo de índice .cdx, llamado [índice compacto estructural compuesto](#), es el tipo de archivo de índice más importante y más común en Visual FoxPro. El archivo .cdx estructural:

- Se abre automáticamente al abrir la tabla.
- Puede contener múltiples ordenaciones de registros, o claves de índice, en el mismo archivo.
- Se modifica automáticamente al agregar, cambiar o eliminar registros de la tabla.

Si una tabla de Visual FoxPro tiene un archivo de índice asociado, normalmente se tratará de un archivo .cdx estructural. El término “estructural” hace referencia al hecho de que Visual FoxPro trata al archivo como una parte intrínseca de la tabla y lo abre automáticamente al abrir aquélla. Ya sea a través del Diseñador de tablas o con la forma más simple del comando INDEX, como se mostraba en el ejemplo anterior, Visual FoxPro crea el archivo .cdx con el mismo nombre que la tabla actual y almacena la información de índice de la nueva clave, o etiqueta, dentro de él. Los archivos .cdx estructurales se utilizan para claves de índice de uso frecuente, como las que ordenan los registros para tareas diarias de presentación o introducción de datos, vínculos con SET RELATION, optimización [Rushmore™](#) al ver los registros o para informes que se imprimen con frecuencia.

Visual FoxPro ofrece otros dos tipos de archivos de índice: el archivo .cdx no estructural y el archivo .idx de una sola clave. Al ser el índice .cdx, o índice estructural compuesto, el más importante, la mayoría de los ejemplos de esta sección utilizarán claves de índice de archivos .cdx para ordenar los registros de las tablas. Los dos tipos de índice restantes se usan con menos frecuencia y se tratan al final de la sección.

## Ver la información de índice

Para ver el número registros que se indexan durante el proceso de indexación, puede establecer [TALK](#) a ON. El intervalo de registros mostrado durante la indexación puede especificarse con [SET ODOMETER](#). Si desea más información sobre los archivos de índice abiertos, utilice el comando [DISPLAY STATUS](#). Este comando muestra los nombres de todos los archivos de índice abiertos, su tipo (estructural, .cdx o .idx), sus expresiones de índice y el nombre del archivo de índice principal o etiqueta principal.

El número de archivos de índice (.idx o .cdx) que pueden abrirse sólo está limitado por la memoria y los recursos del sistema.

## Controlar los valores duplicados

Visual FoxPro cuenta con cuatro tipos de índices: principal, candidato, único y normal. Estos tipos determinan si se admiten o no valores duplicados en los campos y registros de la tabla.

## Impedir valores duplicados

Un [índice principal](#) nunca permite valores duplicados en la expresión o en los campos especificados. Los índices principales se utilizan sobre todo en la tabla principal o “referenciada” para establecer la integridad referencial en una relación persistente. Sólo es posible crear un índice principal para cada tabla. Visual FoxPro devolverá un error si especifica un índice principal sobre un campo que contenga datos duplicados.

Un [índice candidato](#) nunca permite valores duplicados en la [expresión](#) o en los [campos](#) especificados. El nombre “candidato” hace referencia al estado del índice: puesto que estos índices no admiten valores duplicados, se convierten en “candidatos” para ser elegidos como índice principal de la tabla.

Puede crear múltiples índices candidatos para una tabla. Los índices candidatos pueden utilizarse para hacer referencia o para ser referenciados en una [relación persistente](#) con el fin de establecer la [integridad referencial](#).

Visual FoxPro generará un error si especifica un índice candidato sobre un campo que contenga datos duplicados.

### Establecer un índice principal o candidato

Los índices principales y candidatos pueden crearse con los comandos CREATE TABLE y ALTER TABLE. Puede utilizar ambos tipos de índices para definir el lado “uno” de una relación persistente de [uno a varios](#) o de [uno a uno](#).

### Para crear un índice principal o candidato

- En el [Diseñador de tablas](#), elija la ficha **Índices** y cree un índice, seleccionando como tipo **Principal** o **Candidato**.

–O bien–

- Utilice el comando [ALTER TABLE](#).

Por ejemplo, cualquiera de los dos comandos siguientes hace de `cust_id` la clave principal de la tabla `customer`:

```
ALTER TABLE customer ADD PRIMARY KEY cust_id TAG cust_id
ALTER TABLE customer ALTER COLUMN cust_id c(5) PRIMARY KEY
```

Los índices principal y candidatos se almacenan en el archivo `.cdx` estructural de la tabla correspondiente y también en la base de datos con las propiedades “Primary” o “Candidate”. No se puede almacenar estos tipos de índices en otros archivos `.cdx`, ni tampoco utilizar para ellos archivos `.idx`. El motivo principal es que el archivo que contiene estos índices siempre debe estar abierto cuando esté abierta la tabla a la que están asociados.

Las [claves principales](#) forman parte de una tabla que pertenece a una base de datos. Si libera una tabla de una base de datos, se elimina la clave principal.

Si utiliza una [función definida por el usuario](#) en una expresión de índice asociada a una base de datos,

Visual FoxPro tratará la expresión de la misma forma que las expresiones de [reglas](#) y [desencadenantes](#) que contienen FDU.

### Permitir valores duplicados

En Visual FoxPro, los [índices únicos](#) no evitan el uso de valores duplicados, aunque sólo almacenan la primera aparición del valor o valores en el archivo de índice. En este sentido, la palabra “único” se refiere a las entradas del archivo de índice, que sólo contiene valores únicos, pues no almacena cada clave más de una vez e ignora la segunda y posteriores apariciones de los valores no únicos. Las tablas indexadas con índices únicos pueden contener valores duplicados. Los índices únicos se admiten principalmente por compatibilidad con las versiones anteriores.

Un [índice normal](#) es simplemente un índice que no es único, principal ni candidato. Los índices normales se utilizan para ordenar y buscar registros, pero no se exige que los datos de dichos registros sean únicos. También puede usar un índice normal como lado “varios” de una relación persistente de [uno a varios](#).

### Para crear un índice normal

- En el [Diseñador de tablas](#), elija la ficha **Índices** y cree un índice seleccionando **Normal** como tipo.  
  
–O bien–
- Utilice el comando [INDEX](#).

Por ejemplo, los comandos siguientes hacen de `city` una clave normal para la tabla `customer`:

```
USE customer  
INDEX ON city TAG city
```

### Crear múltiples índices

A medida que trabaje con los registros de una tabla, descubrirá la necesidad de tener acceso a los registros utilizando varias secuencias diferentes. Por ejemplo, puede ser conveniente ordenar la tabla `customer` por el campo `contact` para encontrar rápidamente un nombre que se busca, o por `postal_code`, para generar etiquetas de correo ordenadas para un envío más eficiente.

Puede crear y almacenar varias ordenaciones distintas para una tabla creando múltiples claves de índice para la misma. Esto permite ordenar los registros de una forma distinta en cada momento, según las operaciones que vaya a realizar.

### Para crear claves de índice adicionales para una tabla

- En el [Diseñador de tablas](#) elija la ficha **Índices** e introduzca la información de las claves adicionales.  
  
–O bien–

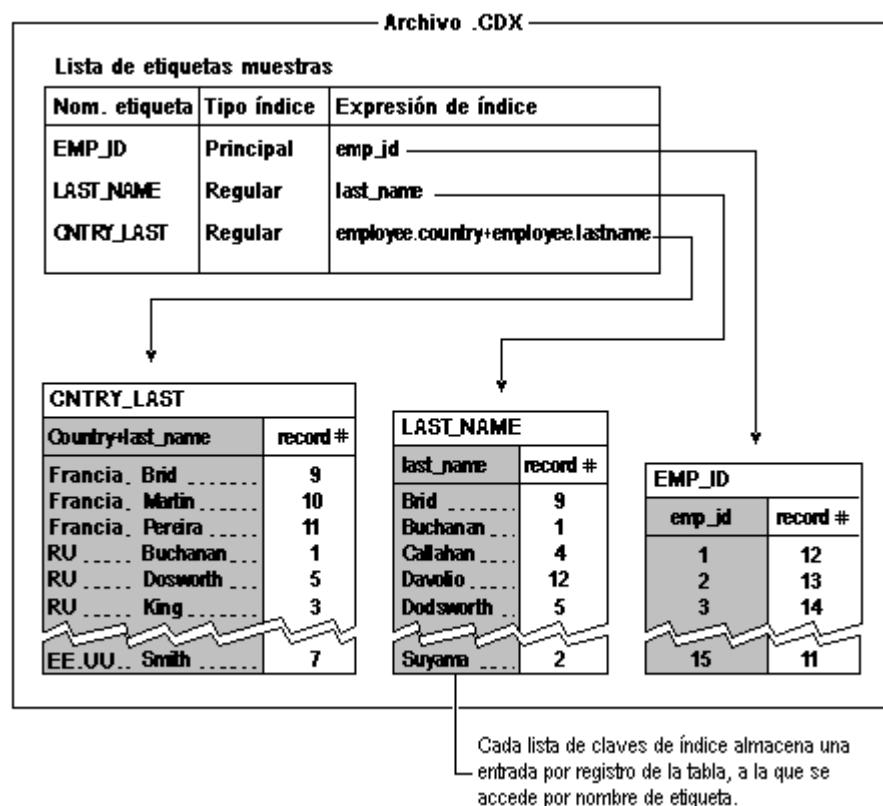
- Utilice el comando [INDEX](#).

Por ejemplo, el código siguiente crea dos nuevas claves de índice en la tabla `employee`: una sobre el campo `last_name` y otra sobre el campo `country`:

```
USE employee
INDEX ON last_name TAG last_name
INDEX ON country TAG country
```

Al crear una etiqueta de índice sin especificar el nombre de un archivo de índice, la etiqueta se agregará automáticamente al archivo `.cdx` estructural de la tabla. En el diagrama siguiente se muestra un archivo de índice `.cdx` con tres etiquetas.

### Índice `.cdx` con múltiples etiquetas que representan múltiples ordenaciones de registros



Dos de las etiquetas del diagrama, `emp_id` y `last_name`, representan índices basados en un solo campo. El índice `cntry_last` ordena los registros según una expresión sencilla de dos campos. Si desea más información sobre la forma de generar índices basados en múltiples campos, consulte "[Indexar por prestaciones](#)", más adelante en este mismo capítulo.

### Controlar el orden de acceso a los registros

Una vez creadas las claves de índice para la tabla `customer` sobre los campos `company`, `city`, y `country` puede tener acceso a la tabla y mostrarla en secuencias distintas; para ello, debe elegir la clave de índice que desee. El comando [SET ORDER](#) permite elegir un índice determinado como clave de ordenación de la tabla.

Por ejemplo, el código siguiente abre una [ventana Examinar](#) con los registros de la tabla `customer` ordenados por países:

```
SET ORDER TO country  
BROWSE
```

## Establecer el orden de los registros en tiempo de ejecución

[SET ORDER](#) permite designar el archivo o etiqueta de índice que controla el orden de los registros. Una tabla puede tener abiertos simultáneamente varios archivos de índice. Sin embargo, para determinar el orden en que se muestran los registros o se tiene acceso a los mismos, hay que establecer como índice de control un archivo de una sola clave (.idx, archivo de control) o una etiqueta de un archivo de índice compuesto (.cdx, etiqueta de control). Algunos comandos, como SEEK, utilizan la etiqueta de índice de control para buscar registros. No tiene que usar SET ORDER para ejecutar [consultas](#).

## Establecer el orden de los registros de forma interactiva en un formulario

Puede utilizar [SET ORDER](#) en [tiempo de ejecución](#) para cambiar el orden de los registros en un [formulario](#). Por ejemplo, puede ser conveniente que los usuarios de su aplicación puedan cambiar el orden de los registros de una [cuadrícula](#) haciendo clic en el encabezado de la columna que deseen ordenar.

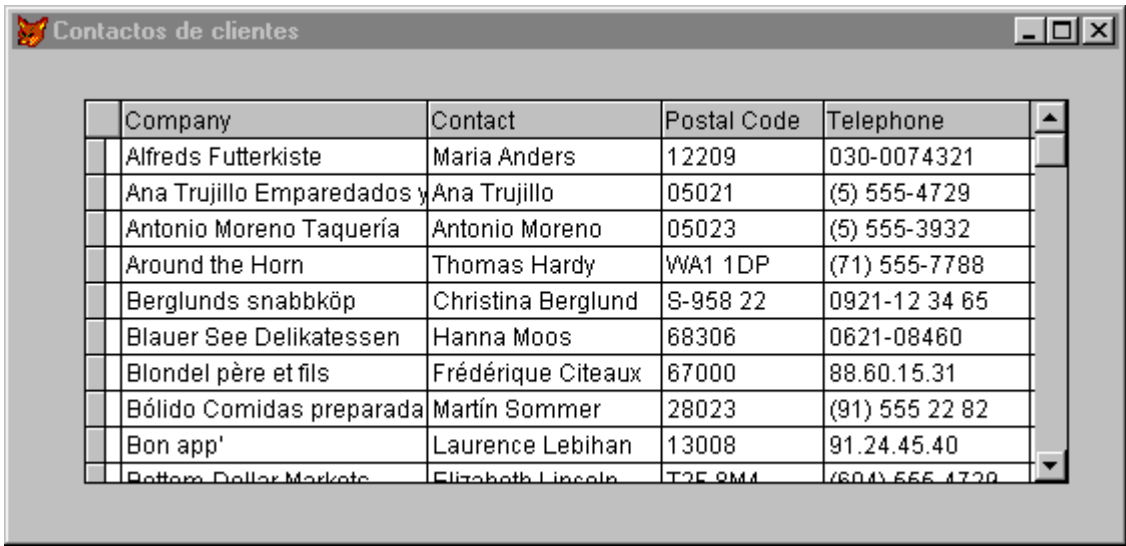
## Para ordenar los registros de una cuadrícula por una columna

1. Cree un formulario con un control [Grid](#).

2. Establezca en la propiedad [ColumnCount](#) de la cuadrícula el número de campos que desee mostrar en ella.
3. En el evento [Click](#) del encabezado de cada columna de la cuadrícula, inserte código para:
  - Establecer como orden de los registros una clave de índice basada en la columna.
  - Actualizar el formulario.

Por ejemplo, si crea un formulario basado en la tabla Customer de la base de datos Testdata con una cuadrícula que contenga las cuatro columnas, company, contact, postal\_code y phone, la cuadrícula aparecerá en primer lugar ordenada alfabéticamente, ya que los registros de esa tabla se introdujeron por ese orden.

**Tabla Customer en una cuadrícula, ordenada alfabéticamente por nombre de empresa**



	Company	Contact	Postal Code	Telephone
	Alfreds Futterkiste	Maria Anders	12209	030-0074321
	Ana Trujillo Emparedados y	Ana Trujillo	05021	(5) 555-4729
	Antonio Moreno Taquería	Antonio Moreno	05023	(5) 555-3932
	Around the Horn	Thomas Hardy	WA1 1DP	(71) 555-7788
	Berglunds snabbköp	Christina Berglund	S-958 22	0921-12 34 65
	Blauer See Delikatessen	Hanna Moos	68306	0621-08460
	Blondel père et fils	Frédérique Citeaux	67000	88.60.15.31
	Bólido Comidas preparada	Martín Sommer	28023	(91) 555 22 82
	Bon app'	Laurence Lebihan	13008	91.24.45.40
	Bottom Dollar Markets	Elizabeth Lincoln	T2F 9M4	(604) 555 4720

A continuación puede permitir que el usuario ordene la cuadrícula por contact o postal\_code si inserta el código siguiente en el evento Click de cada encabezado de columna:

**Ejemplo de código de evento para ordenar los registros de una cuadrícula al hacer clic en el encabezado de las columnas**

Código	Comentario
SET ORDER TO company GO TOP THISFORM.Refresh	En el código de evento Click del encabezado de Company se reordena la cuadrícula por la clave de índice company y se actualiza el formulario para mostrar los registros ordenados por empresas.
SET ORDER TO contact GO TOP THISFORM.Refresh	En el código de evento Click del encabezado de Contact se reordena la cuadrícula por la clave de índice contact y se actualiza el formulario para mostrar los registros ordenados por nombres de contacto.



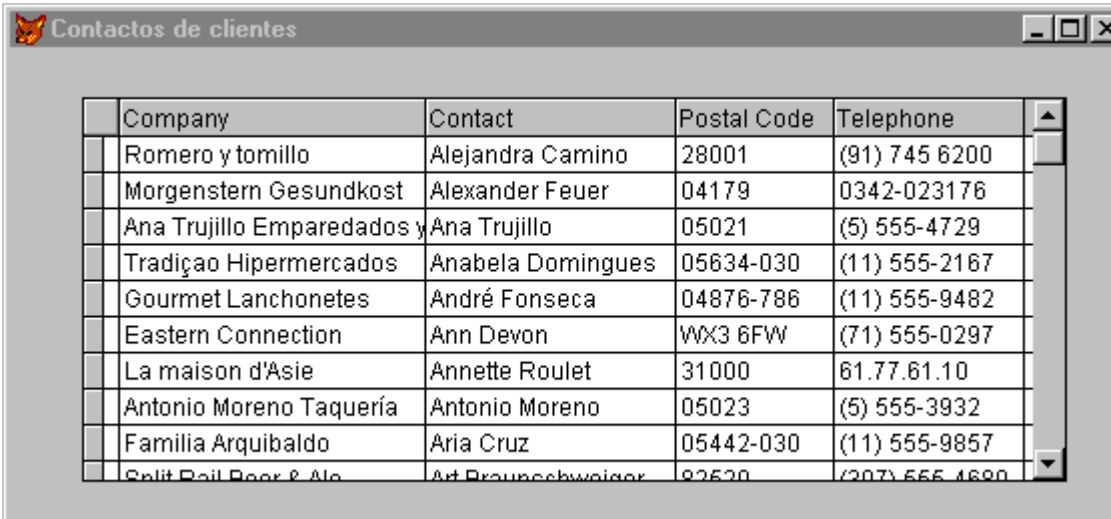
```
SET ORDER TO
  postalcode
GO TOP
THISFORM.Refresh
```

En el código de evento Click del encabezado de Postal\_Code se reordena la cuadrícula por la clave de índice postalcode y se actualiza el formulario para mostrar los registros ordenados por códigos postales.

Como la ordenación por números de teléfono no es adecuada para esta aplicación, se deja en blanco el código del evento Click del encabezado de Phone.

En este ejemplo, al mostrar un [formulario](#) la [cuadrícula](#) aparece ordenada alfabéticamente por empresas. Cuando el usuario hace clic en el encabezado de la columna Contact, Visual FoxPro muestra los registros en la cuadrícula ordenados alfabéticamente por el nombre del contacto.

### Tabla Customer en una cuadrícula, reordenada alfabéticamente por nombre de contacto



	Company	Contact	Postal Code	Telephone
	Romero y tomillo	Alejandra Camino	28001	(91) 745 6200
	Morgenstern Gesundkost	Alexander Feuer	04179	0342-023176
	Ana Trujillo Emparedados y	Ana Trujillo	05021	(5) 555-4729
	Tradiçao Hipermercados	Anabela Domingues	05634-030	(11) 555-2167
	Gourmet Lanchonetes	André Fonseca	04876-786	(11) 555-9482
	Eastern Connection	Ann Devon	WX3 6FW	(71) 555-0297
	La maison d'Asie	Annette Roulet	31000	61.77.61.10
	Antonio Moreno Taquería	Antonio Moreno	05023	(5) 555-3932
	Familia Arquibaldo	Aria Cruz	05442-030	(11) 555-9857
	Split Rail Beer & Ale	Art Braunschweiger	02620	(207) 555-1690

Si el usuario hace clic en el encabezado de la columna Postal\_code la cuadrícula se reordenará y aparecerá ordenada por códigos postales.

### Tabla Customer en una cuadrícula, reordenada por códigos postales



	Company	Contact	Postal Code	Telephone
	Hungry Owl All-Night Groce	Patricia McKenna		2967 542
	Wolski Zajazd	Zbyszek Piestrzeniewi	01-012	(26) 642-7012
	QUICK-Stop	Horst Kloss	01307	0372-035188
	Que Delícia	Bernardo Batista	02389-673	(21) 555-4252
	Ricardo Adocicados	Janete Limeira	02389-890	(21) 555-3412
	Morgenstern Gesundkost	Alexander Feuer	04179	0342-023176
	Gourmet Lanchonetes	André Fonseca	04876-786	(11) 555-9482
	Ana Trujillo Emparedados y	Ana Trujillo	05021	(5) 555-4729
	Centro comercial Moctezum	Francisco Chang	05022	(5) 555-3392
	Antonio Moreno Taquería	Antonio Moreno	05023	(5) 555-3932



Como en esta aplicación de ejemplo no hay necesidad de ordenar los contactos por sus números de teléfono, no se inserta el código con [SET ORDER](#) en el evento Click del encabezado de la columna `phone`. Cuando el usuario hace clic en el encabezado de `Phone`, la cuadrícula no cambia.

## Usar otros tipos de índices

Además del índice más habitual, el compacto estructural compuesto `.cdx`, Visual FoxPro admite otros dos tipos de archivos de índice: el `.cdx` no estructural y el índice autónomo `.idx`. Los índices `.cdx` no estructurales se utilizan como etiquetas de clave múltiple de uso menos frecuente. Los índices autónomos, o `.idx`, se usan como índices de clave única de uso poco frecuente o temporal, y se admiten principalmente por la compatibilidad con las versiones anteriores.

La tabla siguiente es un resumen de los tres tipos de índices, su nombre, el número de claves que pueden contener y las limitaciones en caracteres de cada uno de ellos.

### Tipos de índices de Visual FoxPro

Tipo de índice	Descripción	Número de claves	Límites
<code>.cdx</code> estructural	Utiliza el mismo nombre base que el archivo de tabla; se abre automáticamente con la tabla	Expresiones de múltiples claves, llamadas etiquetas	Límite de 240 caracteres en las expresiones evaluadas
<code>.cdx</code> no estructural	Debe abrirse explícitamente; utiliza un nombre distinto del nombre base de la tabla	Expresiones de múltiples claves, llamadas etiquetas	Límite de 240 caracteres en las expresiones evaluadas
<code>.idx</code> autónomo	Debe abrirse explícitamente; el usuario define el nombre del archivo <code>.idx</code>	Expresión de una sola clave	Límite de 100 caracteres en las expresiones evaluadas

### Usar índices `.cdx` no estructurales

Un índice `.cdx` no estructural resulta útil cuando desea crear múltiples etiquetas de índice con un fin determinado, pero no quiere cargar sus aplicaciones manteniendo estos índices de una forma continua. Por ejemplo, su aplicación puede tener un conjunto especial de informes que analicen los datos basándose en campos normalmente no indexados. En este caso el programa puede crear un índice `.cdx` no estructural con las etiquetas necesarias, ejecutar los informes especiales y a continuación eliminar el índice `.cdx` no estructural.

### Para crear una etiqueta de índice `.cdx` no estructural

- Utilice el comando [INDEX](#) con las cláusulas TAG y OF.

Al especificar la cláusula OF con el comando INDEX, se indica a Visual FoxPro que debe almacenar la etiqueta en un archivo distinto del índice .cdx estructural de la tabla. Por ejemplo, el comando siguiente crea etiquetas llamadas `title` y `hire_date` en la tabla `employee` y las almacena en un índice .cdx no estructural llamado `QRTLYRPT.CDX`:

```
USE employee
INDEX ON title TO TAG title OF QRTLYRPT
INDEX ON hire_date TO TAG hiredate OF QRTLYRPT
```

### Usar índices autónomos

Los índices autónomos, basados en una expresión de una sola clave, se almacenan como archivos .idx. Al contrario que los índices .cdx, que pueden almacenar expresiones de múltiples claves, los índices .idx se limitan a una sola.

Normalmente, los índices autónomos se utilizan de forma temporal, creándolos o reindexándolos inmediatamente antes de su uso. Por ejemplo, puede tener un índice que sólo utilice para un informe resumen trimestral o anual. En lugar de incluir este índice de uso poco frecuente en el .cdx estructural, donde se actualizará cada vez que se utilice la tabla, puede crear un índice .idx autónomo. Una tabla determinada puede tener tantos archivos .idx como se desee.

### Para crear un índice .idx autónomo

- Utilice la cláusula COMPACT del comando [INDEX](#).

–O bien–

- Utilice el comando [COPY TAG](#).

Al utilizar el comando INDEX con la cláusula COMPACT se crea un nuevo índice autónomo en un archivo pequeño y de acceso rápido. Puede omitir la cláusula COMPACT si desea crear un archivo .idx autónomo no compacto y mantener así la compatibilidad con los formatos de índice de FoxBASE+® y FoxPro® versión 1.0.

El código siguiente crea un archivo .idx autónomo sobre `order` para la tabla `orders`, establece como orden el nuevo índice y luego abre una [ventana Examinar](#), que muestra los pedidos ordenados por `order_date`:

```
USE ORDERS
INDEX ON order_date TO orddate COMPACT
SET ORDER TO orddate
BROWSE
```

Puede utilizar el comando COPY TAG para generar un archivo de índice autónomo a partir de una etiqueta de índice de un archivo .cdx existente. Esto puede ser útil, por ejemplo, si comprueba que uno de los índices que actualmente se incluyen en el .cdx estructural sólo se utiliza para informes

trimestrales o anuales. En el código siguiente se crea un índice autónomo a partir de una etiqueta `birth_date` de la tabla `employee`:

```
COPY TAG birth_date to birthdt COMPACT
```

Una vez creado un índice autónomo a partir de una etiqueta de un archivo `.cdx`, normalmente deberá eliminar la etiqueta, ahora innecesaria, del archivo `.cdx`. En la sección siguiente se describe la forma de eliminar índices.

## Eliminar un índice

Cuando no vaya a utilizar un índice, puede eliminar su etiqueta en el archivo `.cdx` o bien puede eliminar el propio archivo `.idx` en el caso de los índices autónomos. La eliminación de las etiquetas de índice no utilizadas mejora el rendimiento, ya que suprime la necesidad de que Visual FoxPro actualice etiquetas no utilizadas para reflejar los cambios en los datos de la tabla.

## Eliminar una etiqueta del archivo `.idx` estructural

Puede eliminar una etiqueta del archivo `.idx` estructural mediante el Diseñador de tablas o bien mediante el lenguaje.

### Para eliminar una etiqueta de índice del `.cdx` estructural

- En el [Diseñador de tablas](#), utilice la ficha **Índices** para seleccionar el índice y eliminarlo.  
–O bien–
- Utilice el comando [DELETE TAG](#).  
–O bien–
- Utilice las cláusulas `DROP PRIMARY KEY` o `DROP UNIQUE TAG` del comando [ALTER TABLE](#).

Por ejemplo, si la tabla `employee` contiene una etiqueta llamada `title`, puede eliminarla con el código siguiente:

```
USE employee  
DELETE TAG title
```

En el caso de que la etiqueta que desea eliminar sea la clave principal de la tabla `employee`, puede utilizar el comando `ALTER TABLE`:

```
USE employee  
ALTER TABLE DROP PRIMARY KEY
```

## Eliminar una etiqueta de un archivo `.cdx` no estructural

Los índices `.cdx` no estructurales y sus etiquetas no son visibles en el [Diseñador de tablas](#). Para eliminar las etiquetas de un archivo `.cdx` no estructural debe utilizar el lenguaje

eliminar las etiquetas de un archivo .cdx no estructural debe utilizarse el lenguaje.

### Para eliminar un índice de un archivo .cdx no estructural

- Utilice la cláusula OF del comando [DELETE TAG](#).

Al utilizar la cláusula OF con el comando DELETE TAG, se indica a Visual FoxPro que debe eliminar una etiqueta de un archivo .cdx distinto del .cdx estructural. Por ejemplo, si tiene un archivo .cdx no estructural llamado QRTLYRPT.CDX con una etiqueta llamada title, puede eliminar la etiqueta title con el comando siguiente:

```
DELETE TAG title OF qtrlyrpt
```

Para eliminar todas las etiquetas de un archivo .cdx estructural o no estructural, utilice la cláusula ALL del comando DELETE TAG.

### Eliminar un archivo de índice autónomo .idx

Como los archivos de índice autónomos solamente contienen una expresión de clave única, para eliminar la expresión basta con eliminar el archivo .IDX del disco.

### Para eliminar un archivo de índice autónomo .idx

- Utilice el comando [DELETE FILE](#).

Por ejemplo, el código siguiente elimina el archivo de índice autónomo .idx Orddate.idx:

```
DELETE FILE orddate.idx
```

También puede utilizar el sistema operativo para eliminar los archivos .idx autónomos innecesarios.

## Indexar sobre expresiones

Para hacer más eficaces sus aplicaciones puede crear índices basados en expresiones. Estas expresiones pueden ser simples o complejas, dependiendo de lo que se quiera lograr.

### Indexar sobre expresiones simples

Las expresiones de índice simples se basan en un solo campo o en la concatenación de dos o más campos de caracteres para formar una clave de múltiples campos. Por ejemplo, para la tabla Customer de la base de datos TasTrade podría crear un índice basado en la expresión:

```
country + region + cust_id
```

Al [examinar](#) la tabla Customer ordenada según esta etiqueta de índice, los clientes aparecerán ordenados por países, dentro de cada país por regiones y dentro de cada región por Id. de cliente.

### Usar una expresión para evitar valores duplicados en una combinación de campos

Si desea evitar la duplicación de valores en múltiples campos, puede crear un índice [principal](#) o

[candidato](#) basado en una expresión que combine múltiples campos.

Por ejemplo, si tiene una tabla que almacena el prefijo de zona y el número de teléfono en dos columnas:

Prefijo de zona	Número de teléfono
206	444-nnnn
206	555-nnnn
313	444-nnnn

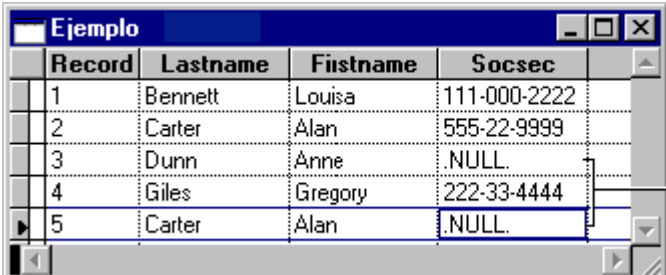
En ambas columnas existen valores duplicados en otras filas. Sin embargo, no hay ningún número de teléfono duplicado, ya que la combinación de los dos campos siempre es única. Por ello, si se especifican las dos columnas en la expresión de un índice principal o candidato, las filas del ejemplo no se considerarán duplicadas. Si intenta introducir un valor con el mismo prefijo de zona y el mismo número de teléfono que una de las filas existentes, Visual FoxPro rechazará la entrada por estar duplicada.

### Usar valores nulos en expresiones de índice

Puede crear índices sobre campos que contengan [valores nulos](#). Las expresiones de índice cuyo resultado es .NULL. se insertan en el archivo .CDX o .IDX delante de las entradas no nulas. Todos los valores nulos quedan por tanto al principio del índice.

El ejemplo siguiente ilustra un efecto de indexar valores nulos. Es el estado de la tabla antes de aplicar el índice:

### Hay valores nulos en el campo SocSec de dos registros



Record	Lastname	Firstname	Socsec
1	Bennett	Louisa	111-000-2222
2	Carter	Alan	555-22-9999
3	Dunn	Anne	.NULL.
4	Giles	Gregory	222-33-4444
5	Carter	Alan	.NULL.

El valor .NULL. en dos registros representa el hecho de que los números de seguridad social de Anne Dunn y Alan Carter son desconocidos o no están disponibles. Puede crear un índice con el número SocSec, como en el ejemplo siguiente:

```
INDEX ON SocSec + LastName + FirstName TAG MyIndex
```

Al ver la tabla ordenada por este índice, la secuencia es:

**Después de indexar sobre SocSec, los registros con valores nulos en SocSec aparecen en primer lugar**

Figura

Record	Lastname	Firstname	Socsec
3	Dunn	Anne	.NULL.
5	Carter	Alan	.NULL.
1	Bennett	Louisa	111-000-2222
4	Giles	Gregory	222-33-4444
2	Carter	Alan	555-22-9999

Cuando la expresión de índice contiene valores nulos, los registros cuyo valor de SocSec es .NULL. se sitúan los primeros (ordenados por LastName), y a continuación se colocan los registros cuyos valores de SocSec son no nulos. Observe que hay dos entradas para Alan Carter. Como el registro 5 contiene un valor nulo, se indexa antes que el registro 2.

### Indexar sobre expresiones complejas

También puede crear índices basados en expresiones más complejas. Las expresiones de clave de índice de Visual FoxPro pueden contener [funciones](#) de Visual FoxPro, [constantes](#) o [funciones definidas por el usuario](#).

La expresión que cree debe ofrecer como resultado no más de 100 caracteres en el caso de índices autónomos (.idx) o de 240 caracteres para las etiquetas de índice .cdx. Puede utilizar campos de [tipos de datos](#) diferentes en una sola etiqueta, convirtiendo los componentes individuales de la expresión a datos de tipo Character.

Para aprovechar la optimización [Rushmore™](#), la expresión de índice debe cumplir exactamente los criterios.

### Usar funciones de Visual FoxPro en una etiqueta de índice

Puede utilizar funciones de Visual FoxPro en las etiquetas de índice. Por ejemplo, puede utilizar la función [STR\(\)](#) para convertir un valor numérico en una cadena de caracteres. Si desea crear una etiqueta de índice para la tabla customer que combine el campo cust\_id con el campo maxordamt, puede convertir maxordamt del tipo Currency con ancho 8 a un campo de 8 caracteres con dos lugares decimales mediante el código siguiente:

```
INDEX ON cust_id + STR(maxordamt, 8, 2) TAG custmaxord
```

Si quiere reducir el tamaño de los índices para campos que contienen valores enteros, puede convertir los valores enteros en una representación Character (Binary) mediante la función [BINTOC\(\)](#). También puede convertir los valores binarios mediante la función [CTOBIN\(\)](#).

Si desea crear un índice para ordenar una tabla en orden cronológico, puede utilizar la función [DTOS\(\)](#) para convertir un campo de fecha en una cadena de caracteres. Para tener acceso a la tabla `employee` por `hire_date` y `emp_id`, puede crear la siguiente expresión de índice:

```
INDEX ON DTOS(hire_date) + emp_id TAG id_hired
```

### Incluir procedimientos almacenados o funciones definidas por el usuario en una etiqueta de índice

Para hacer más eficaz el índice puede hacer referencia en su expresión a un [procedimiento almacenado](#) o a una [función definida por el usuario](#). Por ejemplo, puede utilizar un procedimiento almacenado o una FDU para extraer el nombre de la calle de un campo único que incluya además del nombre el número de la calle. Si el número de la calle es siempre numérico, el procedimiento almacenado o la FDU puede devolver la parte de caracteres del campo y rellenar el resto con espacios hasta completar una clave de índice de longitud constante. A continuación puede utilizar esta clave de índice para tener acceso a los registros de la tabla ordenados por el nombre de calle.

Puede ser preferible utilizar un procedimiento almacenado en vez de una FDU en las etiquetas de índice. Como las FDU se almacenan en archivos independientes de la base de datos, puede mover o eliminar el archivo de la FDU, lo que hará que la etiqueta que hace referencia a la misma deje de ser válida. El código de los procedimientos almacenados, por el contrario, se conserva en el archivo .DBC y Visual FoxPro puede encontrarlo siempre.

Otra ventaja de los procedimientos almacenados en las etiquetas de índice es que la referencia a un procedimiento almacenado garantiza que el índice se basará en el código especificado. Si utiliza una FDU en la expresión de índice, se usará en la indexación toda FDU que se encuentre al alcance y que tenga el mismo nombre que la referenciada en el índice.

**Nota** Tenga cuidado al referenciar un procedimiento almacenado o una FDU en una expresión de índice, ya que el tiempo necesario para crear el índice aumentará.

### Usar datos de un campo de otra tabla en una etiqueta de índice

Puede crear una etiqueta de índice que haga referencia a una tabla abierta en otra [área de trabajo](#). Lo mejor es utilizar un índice autónomo (.idx) para las etiquetas que hagan referencia a más de una tabla, ya que si se incluye una etiqueta de este tipo en un archivo .cdx estructural, Visual FoxPro no permitiría abrir la tabla hasta que esté abierta la tabla referenciada en la etiqueta de índice.

### Acceso a los registros en orden descendente

Para ver los registros en orden descendente puede crear un índice descendente o bien leer un índice existente en orden descendente.

### Para crear un índice descendente

- En la ficha **Índices** del [Diseñador de tablas](#), elija el botón de flecha situado a la izquierda del cuadro **Nombre** de forma que la flecha apunte hacia abajo.

—O bien—



- Utilice la cláusula DESCENDING con el comando [INDEX ON](#) para crear un índice descendente.

Para crear archivos de índice estructural compuesto, puede utilizar los dos métodos. Para crear otros tipos de archivos de índice, sólo puede utilizar el segundo. Por ejemplo, puede crear un nuevo índice descendente que ordene la tabla `product` de mayor a menor por `unit_price` y examinar la tabla con el nuevo orden, mediante el código siguiente:

```
USE products
INDEX ON unit_price TAG unit_price DESCENDING
BROWSE
```

### Para leer un índice existente en orden descendente

- Utilice la cláusula DESCENDING del comando [SET ORDER](#) para leer el índice en orden descendente.

Leer los índices en orden descendente permite aprovechar los índices existentes, en lugar de crear otros nuevos. Por ejemplo, puede que ya haya creado un índice que ordene la tabla `product` por `unit_price` con el código siguiente:

```
USE products
INDEX ON unit_price TAG unit_price
```

De forma predeterminada, el orden es ascendente. Ahora puede examinar la tabla en orden descendente con el siguiente código:

```
USE products
SET ORDER TO unit_price DESCENDING
BROWSE
```

En los ejemplos anteriores se trata el acceso a la información en orden descendente. Tanto [SET ORDER](#) como [INDEX](#) cuentan con una cláusula ASCENDING. Si combina estos dos comandos puede obtener una gran flexibilidad en sus aplicaciones. Por ejemplo, si utiliza la cláusula ASCENDING o DESCENDING para crear un índice en el orden de uso más frecuente, puede utilizar la cláusula opuesta con el comando SET ORDER para ver o tener acceso a la información en el orden inverso cuando resulte más adecuado.

## Filtrar datos

Puede limitar el acceso a los registros que se desee con un índice filtrado. Al crear un índice filtrado, solamente estarán disponibles para su presentación y acceso los registros que satisfagan la expresión de filtro.

### Para filtrar los datos con un índice filtrado

- En el [Diseñador de tablas](#), escriba una expresión de filtro en el cuadro de texto **Filtro** del índice correspondiente.

~ . .

–O bien–

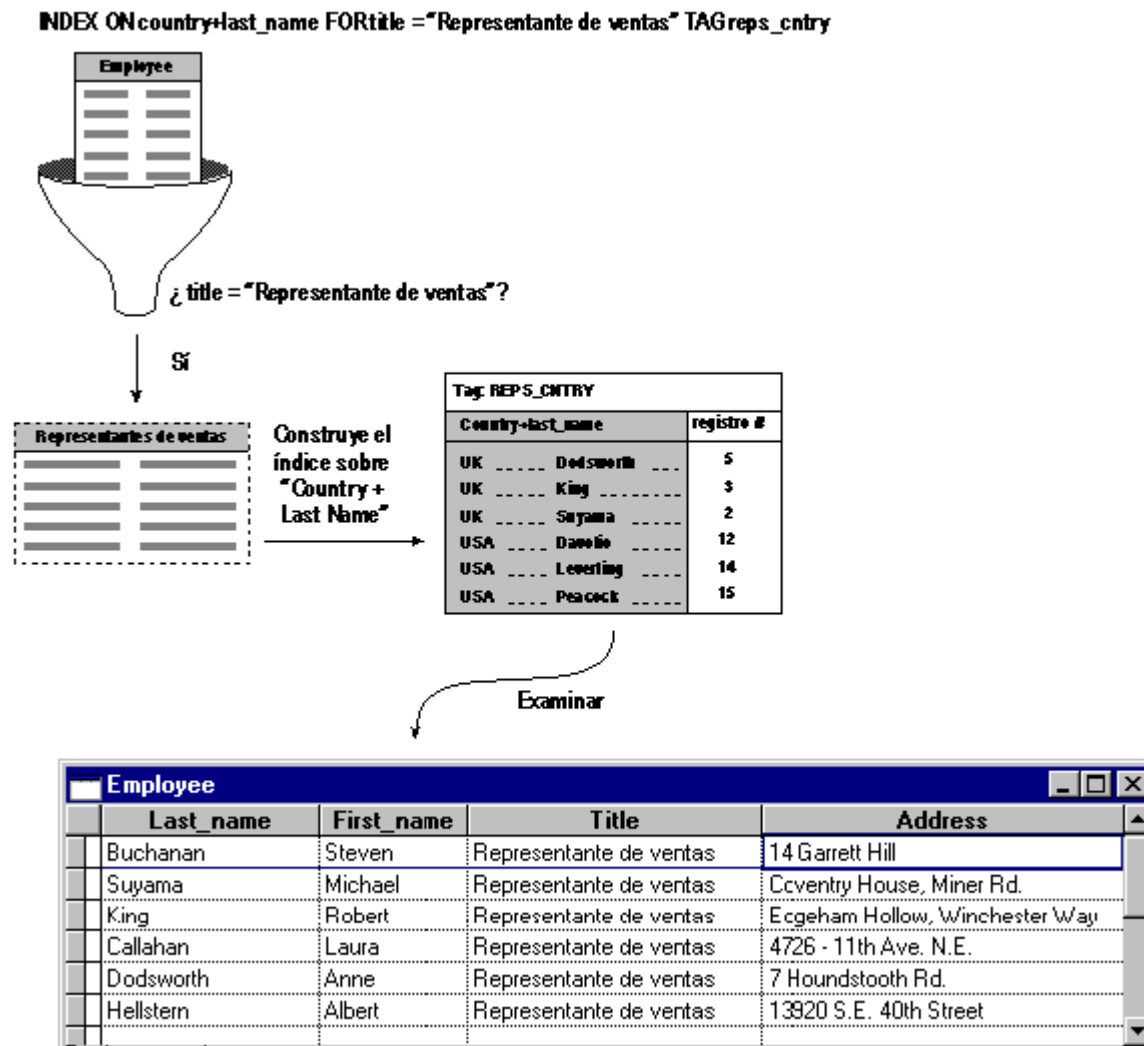
- Utilice la cláusula opcional FOR con el comando [INDEX](#).

Si incluye la cláusula opcional FOR con el comando INDEX, el archivo de índice actuará como un filtro para la tabla. En el archivo de índice sólo se crearán claves para los registros que satisfagan la expresión de filtro. Por ejemplo, si prepara una circular destinada a los representantes de ventas de su empresa y desea ordenar el envío por países, puede crear un índice que filtre la tabla `employee` de forma que sólo aparezcan los registros de los representantes de ventas, ordenados por países y por apellidos. El código siguiente crea tal índice filtrado y muestra los datos resultantes en una ventana Examinar:

```
USE employee
INDEX ON country+last_name FOR title = "Representante de ventas" ;
TAG reps_cntry
BROWSE
```

Al ver la ventana Examinar sólo aparecerán los representantes de ventas, no los registros de los demás empleados.

**Un índice filtrado se aplica solamente a los registros que satisfacen la expresión de filtro.**





## Filtrar datos de forma temporal

Puede utilizar el comando [SET FILTER](#) para filtrar los datos de forma temporal, sin crear un índice filtrado especial. Este comando es especialmente útil cuando se desea especificar una condición temporal que los registros de una tabla deben satisfacer para ser accesibles. Por ejemplo, puede utilizar el comando siguiente para filtrar la tabla `customer` de forma que sólo muestre los clientes de Alemania:

```
USE customer
SET FILTER TO country = "Germany"
BROWSE
```

El comando `SET FILTER` admite como condición de filtro cualquier expresión lógica válida en Visual FoxPro. Una vez especificado el comando `SET FILTER`, sólo estarán disponibles en la tabla los registros que cumplan la condición de filtro. Todos los comandos de acceso a la tabla respetan la condición de `SET FILTER`. Puede establecer un filtro distinto para cada tabla abierta.

## Usar los índices de forma eficaz

Puede mejorar el rendimiento de las tablas indexadas si mantiene los índices actualizados y utiliza en ellos expresiones optimizables.

## Volver a generar un archivo de índice activo

Los archivos de índice quedan obsoletos cuando se abre una tabla sin abrir sus archivos de índice correspondientes y se modifican los campos clave. Los archivos de índice también pueden quedar invalidados a consecuencia de una caída del sistema o al tener acceso a una tabla y actualizarla desde otro programa. Cuando los archivos de índice quedan obsoletos, puede actualizarlos con el comando `REINDEX`.

## Para volver a generar un archivo de índice activo

- En el [menú Tabla](#), elija **Volver a generar índices**.

–O bien–

- Utilice el comando [REINDEX](#).

Por ejemplo, el código siguiente actualiza el archivo de índice de la tabla Customer:

```
USE customer  
REINDEX
```

REINDEX actualiza todos los archivos de índice abiertos en el [área de trabajo](#) seleccionada. Visual FoxPro reconoce cada tipo de archivo de índice (.cdx compuestos, .cdx estructurales e .idx de índice único) y efectúa la reindexación según corresponda. Al hacerlo, actualiza todas las etiquetas de los archivos .cdx y también los archivos .cdx estructurales, que se abren automáticamente con la tabla.

También puede actualizar los archivos de índice obsoletos con el comando REINDEX.

### Reindexar en tiempo de ejecución

La reindexación lleva tiempo, especialmente cuando se trata de tablas de gran tamaño. Sólo debe reindexar cuando sea necesario. Para mejorar el rendimiento, puede reindexar durante la inicialización o la finalización del programa, en lugar de hacerlo en la ejecución principal.

### Usar índices para optimizar consultas

Puede utilizar índices para acelerar las [consultas](#) y otras operaciones. Si desea información sobre la forma de crear expresiones de índice optimizables Rushmore, consulte el capítulo 15, [Optimizar aplicaciones](#)

## Usar múltiples tablas

Para usar varias tablas es necesario emplear áreas de trabajo. Un [área de trabajo](#) es una región numerada que identifica una tabla abierta. Puede abrir y manipular tablas de Visual FoxPro en 32.767 áreas de trabajo. Las áreas de trabajo se suelen identificar en las aplicaciones por el alias de la tabla abierta en ellas. Un alias de tabla es un nombre que se refiere a una tabla abierta en un área de trabajo.

### Usar sesiones de datos

Además de las áreas de trabajo visibles en la ventana Sesión de datos, Visual FoxPro proporciona automáticamente un entorno independiente para cada instancia de un [formulario](#) o [conjunto de formularios](#) a través de las [sesiones de datos](#). Una sesión de datos es una representación del entorno de trabajo dinámico actual utilizado por un formulario, un conjunto de formularios o un informe. Cada sesión de datos contiene su propio conjunto de áreas de trabajo. Estas áreas de trabajo contienen las tablas abiertas en las áreas de trabajo, sus índices y sus relaciones. Para obtener información sobre el uso de sesiones de datos, consulte el capítulo 17, [Programar para acceso compartido](#).

### Ver las áreas de trabajo

Para ver la lista de tablas abiertas en una sesión de Visual FoxPro, puede abrir la ventana Vista.

### Para abrir la ventana Vista

- En el menú **Ventana**, elija **Sesión de datos**.

–O bien–

- Utilice el comando [SET](#).

Al escribir SET en la ventana Comandos, Visual FoxPro abre la ventana [Sesión de datos](#) y muestra los alias de las áreas de trabajo de las tablas abiertas en la sesión actual.

### Ventana Sesión de datos con tabla Employees abierta



### Abrir una tabla en un área de trabajo

Para abrir una tabla en un área de trabajo puede utilizar la ventana Sesión de datos o el comando USE.

### Para abrir una tabla en un área de trabajo

- En la ventana [Sesión de datos](#) elija **Abrir**.

–O bien–

- Escriba [USE](#) en la ventana **Comandos**.

Para abrir una tabla en la menor área de trabajo disponible, utilice la cláusula IN del comando USE con el área de trabajo 0. Por ejemplo, si hay tablas abiertas en las áreas de trabajo de 1 a 10, el comando siguiente abrirá la tabla customer en el área de trabajo 11.

```
USE customer IN 0
```

```
USE customer IN 0
```

También puede elegir **Abrir** en el menú **Archivo** para abrir una tabla en un área de trabajo.

## Cerrar una tabla en un área de trabajo

Para cerrar una tabla en un área de trabajo puede utilizar la ventana Sesión de datos o bien hacerlo mediante el lenguaje.

### Para cerrar una tabla en un área de trabajo

- En la ventana [Sesión de datos](#), seleccione el alias de la tabla y luego elija **Cerrar**.  
–O bien–
- Escriba [USE](#) sin indicar un nombre de tabla.  
–O bien–
- Utilice la cláusula IN del comando USE para referirse al área de trabajo de la tabla que desee cerrar.

Cuando especifique el comando USE sin ningún nombre de tabla y haya un archivo de tabla abierto en el área de trabajo seleccionada actualmente, la tabla se cerrará. Por ejemplo, el código siguiente abre la tabla `customer`, muestra una ventana Examinar y luego cierra la tabla:

```
USE customer  
BROWSE  
USE
```

También se cierra la tabla automáticamente cuando se abre otra en la misma área de trabajo o cuando se especifica el comando USE con la cláusula IN y se hace referencia al área de trabajo actual. El código siguiente abre, muestra y cierra la tabla `customer` utilizando USE IN y el alias de tabla `customer`:

```
USE customer  
BROWSE  
USE IN customer
```

No puede tener abierta más de una tabla a la vez en la misma área de trabajo.

## Hacer referencia a un área de trabajo

Antes de abrir una tabla, puede referirse a la siguiente área de trabajo disponible con el número de área de trabajo como se indica a continuación:

```
SELECT 0
```

## Usar alias de tablas

Un alias de tabla es el nombre que utiliza Visual FoxPro para referirse a una tabla abierta en un área

de trabajo. Visual FoxPro utiliza automáticamente el nombre de archivo como alias predeterminado al abrir una tabla. Por ejemplo, si abre con los comandos siguientes el archivo CUSTOMER.DBF en el área de trabajo 0, se asignará automáticamente a la tabla el alias predeterminado `customer`:

```
SELECT 0
USE customer
```

A continuación puede utilizar el alias `customer` para identificar la tabla en un comando o función. También puede especificar otro alias.

### Crear un alias definido por el usuario

Puede asignar el alias que desee a una tabla en el momento de abrirla.

#### Para abrir una tabla con un alias definido por el usuario

- Escriba [USE](#) con un nombre de alias de tabla.

Por ejemplo, para abrir el archivo CUSTOMER.DBF en el área de trabajo 0 y asignarle el alias `people`, utilice los comandos siguientes:

```
SELECT 0
USE customer ALIAS people
```

A continuación podrá utilizar el alias `people` para referirse a la tabla abierta. Un alias puede tener hasta 254 letras, dígitos o signos de subrayado, y debe comenzar por una letra o un signo de subrayado. Visual FoxPro crea automáticamente un alias alternativo cuando el especificado contiene algún carácter no admitido.

### Usar un alias asignado por Visual FoxPro

Visual FoxPro asigna alias automáticamente a las tablas en determinadas situaciones:

- Al abrir una misma tabla simultáneamente en varias áreas de trabajo incluyendo la cláusula `AGAIN` en el comando [USE](#) y sin especificar alias distintos al abrir la tabla en cada área.
- Cuando se produce un conflicto con los alias.

Los alias predeterminados asignados a las primeras 10 áreas de trabajo son las letras de área de trabajo de A a J; los asignados a las áreas de 11 a 32767 van de W11 a W32767. Puede utilizar estos alias asignados por Visual FoxPro exactamente de la misma forma que cualquier otro alias predeterminado o definido por el usuario para referirse a una tabla abierta en un área de trabajo.

### Seleccionar un área de trabajo con un alias

Para pasar de un área de trabajo a otra puede utilizar el comando `SELECT`. Por ejemplo, si se ha abierto Customer.dbf en un área de trabajo y tiene asignado su alias predeterminado `CUSTOMER`, puede pasar a esta área de trabajo con el comando `SELECT` siguiente:

```
SELECT customer
```

-- -- -- -- --

## Hacer referencia a tablas abiertas en otras áreas de trabajo

También puede hacer referencia a campos de otras áreas de trabajo si precede el nombre del campo con el nombre de alias correspondiente y un punto, o con el alias y el operador `->`. Por ejemplo, si se encuentra en un área de trabajo y desea tener acceso al campo `contact` de la tabla `Customer`, abierta en un área distinta, puede utilizar la siguiente expresión para hacer referencia al campo:

```
customer.contact
```

Si la tabla a la que desea hacer referencia se ha abierto con un alias, puede utilizar el nombre del alias. Por ejemplo, si la tabla `Customer` se ha abierto con el alias `people`, puede referirse al campo `lastname` con la expresión siguiente:

```
people.lastname
```

El uso del nombre o el alias de una tabla la identifica específicamente, independientemente del área de trabajo en la que esté abierta.

## Establecer relaciones temporales entre tablas

Al establecer una [relación temporal](#) entre tablas, se hace que el puntero de registro de una tabla (la tabla secundaria) siga automáticamente los movimientos del de la otra (la tabla primaria), lo que permite seleccionar un registro en el lado ‘uno’, o primario, de una relación y tener acceso automáticamente a los registros relacionados del lado ‘varios’, o secundario.

Por ejemplo, puede estimar conveniente relacionar las tablas `customer` y `orders` de forma que al situar el puntero de registro en un cliente determinado de la tabla `customer`, el puntero de la tabla `orders` se desplace hasta el registro con el mismo número de cliente.

Puede utilizar las áreas de trabajo y los alias de las tablas al establecer relaciones entre dos tablas abiertas con el comando `SET RELATION`. Si utiliza un [formulario](#) para trabajar con las tablas, puede almacenar estas relaciones como parte del [entorno de datos](#) del formulario.

## Tablas relacionadas temporalmente

Puede utilizar la ventana Sesión de datos o el lenguaje para crear relaciones temporales entre las tablas.

### Para relacionar tablas temporalmente

- En la ventana [Sesión de datos](#), seleccione las tablas y utilice el botón **Relaciones** para crear las relaciones.
- –O bien–
- Utilice el comando [SET RELATION](#).

Con el comando `SET RELATION` puede establecer una relación entre una tabla abierta en el área de trabajo seleccionada actualmente y otra tabla abierta en otra área. Normalmente se relacionan tablas



que tienen un campo común y la expresión utilizada para establecer la relación es habitualmente la expresión del índice que controla la tabla secundaria.

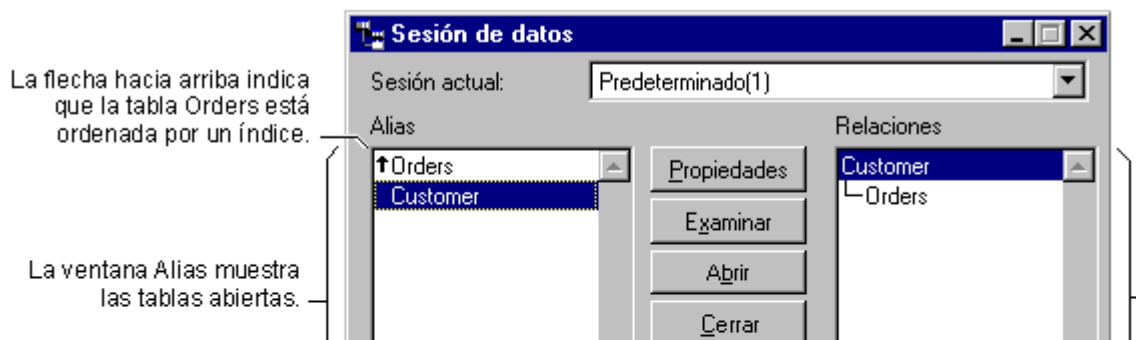
Por ejemplo, puede que un cliente tenga varios pedidos asociados a su registro de cliente. Si crea una relación entre el campo común a ambas tablas, podrá ver fácilmente todos los pedidos de un cliente determinado. En el programa siguiente se utiliza un campo, `cust_id`, común a las dos tablas, para crear una relación entre el campo `cust_id` de la tabla `customer` y la etiqueta de índice `cust_id` de la tabla `orders`.

### Uso de SET RELATION para establecer una relación entre dos tablas

Código	Comentarios
USE customer IN 1	Abre la tabla <code>customer</code> (tabla primaria) en el área de trabajo 1
USE orders IN 2	Abre la tabla <code>orders</code> (tabla secundaria) en el área de trabajo 2
SELECT orders	Selecciona el área de trabajo secundaria.
SET ORDER TO TAG cust_id	Especifica el orden de la tabla secundaria mediante la etiqueta de índice <code>cust_id</code> .
SELECT customer	Selecciona el área de trabajo independiente.
SET RELATION TO cust_id INTO orders	Crea la relación entre la tabla primaria y el índice que controla la tabla secundaria.
SELECT orders BROWSE NOWAIT SELECT customer BROWSE NOWAIT	Abre dos ventanas Examinar. Observe que al mover el puntero de registro en la tabla primaria los datos de la tabla secundaria cambian.

La ventana Sesión de datos muestra las dos tablas abiertas, `Orders` y `Customer`, y la relación establecida con el comando `SET RELATION`.

### La ventana Sesión de datos muestra los alias y las relaciones temporales de las tablas abiertas





El índice creado para la tabla secundaria, `orders`, ordena los registros de la tabla `orders` por grupos, según el cliente que ha originado el pedido. Al establecer una relación entre la tabla primaria y el índice de la tabla secundaria, Visual FoxPro selecciona solamente los registros de la tabla secundaria cuya clave de índice coincide con la del registro primario seleccionado.

En el ejemplo anterior se ha establecido una relación simple entre dos tablas. También puede utilizar el comando [SET RELATION](#) para establecer relaciones múltiples entre una tabla primaria y varias tablas secundarias.

### Guardar relaciones entre tablas en un entorno de datos

Si va a crear un [formulario](#) que utiliza más de una tabla, puede utilizar el [entorno de datos](#) para crear relaciones entre tablas y almacenarlas en el formulario. Las relaciones establecidas en el entorno de datos se abren automáticamente al ejecutar el formulario. Si desea información sobre cómo crear un entorno de datos, consulte el capítulo 9, [Crear formularios](#).

### Relacionar registros de una sola tabla

También puede crear una relación entre registros de una sola tabla. Esta relación, denominada con referencia a sí misma, puede ser útil en aquellas situaciones en las que se tiene toda la información necesaria almacenada en una sola tabla. Por ejemplo, puede ser conveniente desplazarse por los supervisores de la tabla `Employee` haciendo que los empleados a cargo de cada supervisor cambien automáticamente al mover el puntero de registro de un supervisor a otro.

### Para relacionar temporalmente registros de una sola tabla

- En la ventana [Sesión de datos](#), seleccione la tabla y utilice el botón **Relaciones** para establecer relaciones.
- -O bien-
- Utilice el comando [SET RELATION](#).

Para crear una relación con referencia a sí misma, debe abrir la misma tabla dos veces, una en un [área de trabajo](#) y una segunda vez, con el comando [USE AGAIN](#), en otra área distinta. A continuación utilizará un índice para relacionar los registros. Por ejemplo, el código siguiente establece y examina una relación con referencia a sí misma creando una etiqueta de índice llamada `mgr_id` que ordena la tabla `Employee` por el campo `reports_to`:

```
SELECT 0
USE employee ALIAS managers
```

```
-----  
SELECT 0  
USE employee AGAIN ALIAS employees  
INDEX ON reports_to TAG mgr_id  
SET ORDER TO mgr_id  
SELECT managers  
SET RELATION TO emp_id INTO employees  
BROWSE  
SELECT employees  
BROWSE
```

Al desplazar el puntero de registro de la ventana Examinar `managers`, la ventana Examinar `employees` se actualiza de forma que sólo muestra los empleados que dependen del supervisor seleccionado.

## Establecer relaciones persistentes con índices

Los índices permiten establecer [relaciones persistentes](#) entre las tablas de una base de datos. Las relaciones persistentes son relaciones entre tablas de base de datos almacenadas en el archivo de la base de datos y se utilizan automáticamente como condiciones predeterminadas de combinación en el [Diseñador de consultas](#) y en el [Diseñador de vistas](#). Las relaciones persistentes aparecen también en el [Diseñador de bases de datos](#) en forma de líneas que unen los índices de las tablas y son las relaciones predeterminadas al utilizar las tablas en el entorno de datos.

A diferencia de las [relaciones temporales](#) establecidas con el comando [SET RELATION](#), las [relaciones persistentes](#) no tienen que establecerse de nuevo cada vez que se utilizan las tablas. Sin embargo, las relaciones persistentes no controlan la relación entre los punteros de registros, por lo que deberá utilizar ambos tipos al programar aplicaciones con Visual FoxPro. Si desea más información sobre el establecimiento de relaciones persistentes, consulte el capítulo 6, [Crear bases de datos](#).

## Capítulo 8: Crear vistas

Si desea un conjunto de datos personalizado y actualizable para su aplicación, puede utilizar vistas. Las vistas combinan las cualidades de las [consultas](#) y las [tablas](#): al igual que una consulta, puede

utilizar una vista para extraer un conjunto de datos de una o más tablas relacionadas; y como en una tabla, puede utilizar una vista para actualizar la información de la misma y almacenar definitivamente en disco sus resultados.

En este capítulo se describe la creación y actualización de vistas mediante programación, así como el establecimiento de sus [propiedades](#) para optimizar su rendimiento. Para obtener más información sobre bases de datos, consulte el capítulo 6, [Crear bases de datos](#). Si desea más información sobre tablas o índices, consulte el capítulo 7, [Trabajar con tablas](#). Para obtener más información sobre el Diseñador de vistas, consulte el capítulo 5, [Actualizar datos con vistas](#), del *Manual del programador*.

Este capítulo incluye:

- [Crear una vista](#)
- [Usar las vistas](#)
- [Actualizar datos en una vista](#)
- [Combinar vistas](#)
- [Trabajar con datos fuera de línea](#)
- [Optimizar el rendimiento de una vista](#)

## Crear una vista

Como las vistas y las [consultas](#) tienen mucho en común, se siguen los mismos pasos para crear una vista y una consulta. Elija las tablas y los campos que desee incluir en la vista, especifique las condiciones de combinación utilizadas para relacionar las tablas y especifique [filtros](#) para seleccionar registros específicos. A diferencia de las consultas, en las vistas también puede seleccionar cómo se envían las modificaciones realizadas a los datos de una vista a las tablas originales, o [tablas de base](#), a partir de las cuales se construye la vista.

Cuando usted crea una vista, Visual FoxPro almacena una [definición de vista](#) en la base de datos actual. Esta definición contiene los nombres de las tablas y los campos de la vista, así como los valores de sus [propiedades](#). Cuando use la vista, la definición de la misma se usará para generar una instrucción SQL que define el conjunto de datos de la vista.

Para obtener información acerca de las propiedades de la vista, vea [Establecer las propiedades de la vista y de las conexiones](#) más adelante en este capítulo y vea [DBGETPROP\(\)](#) o [CURSORGETPROP\(\)](#).

.

Puede crear dos tipos de vistas: [local](#) y [remota](#). Las vistas remotas utilizan sintaxis remota de SQL para seleccionar información de tablas de un origen de datos ODBC remoto. Las vistas locales usan la sintaxis SQL de Visual FoxPro para seleccionar información de tablas o vistas. Puede agregar una o más vistas remotas a una vista local, lo que le permitirá tener acceso a información de orígenes de datos de Visual FoxPro y ODBC remoto en la misma vista. Para obtener más información sobre el acceso a datos remotos y locales en una única vista, consulte [Combinar datos locales y remotos en una vista](#), más adelante en este mismo capítulo.

### Crear una vista local

Puede crear una vista local con el Diseñador de vistas o mediante el comando CREATE SQL VIEW.

### Para crear una vista local

- En el [Administrador de proyectos](#), seleccione una base de datos, elija **Vistas locales** y, a continuación, elija **Nuevo** para abrir el [Diseñador de vistas](#).  
  
–O bien–
- Use el comando [CREATE SQL VIEW](#) cuando haya una base de datos abierta para mostrar el Generador de vistas.  
  
–O bien–
- Use el comando CREATE SQL VIEW con la cláusula AS.

Por ejemplo, el código siguiente crea una vista que contiene todos los campos de la tabla `products`:

```
CREATE SQL VIEW product_view AS SELECT * ;  
FROM testdata!products
```

El nombre de la nueva vista aparecerá en el Administrador de proyectos. Si abre el [Diseñador de bases de datos](#), la vista se mostrará de la misma manera que una tabla en el [esquema](#), con el nombre de la vista en lugar del nombre de la tabla.

En el ejemplo anterior, el nombre de la tabla está precedido, o [cualificado](#), por el nombre de la base de datos de la tabla y el símbolo "!". Si cualifica el nombre de la tabla al crear una vista, Visual FoxPro buscará la tabla tanto en la lista de bases de datos abiertas, incluyendo las bases de datos actuales y las no actuales, como en la ruta de acceso predeterminada de la tabla.

Si no cualifica una tabla con un nombre de base de datos en una [definición de vista](#), la base de datos deberá estar abierta antes de poder utilizar la vista.

**Sugerencia** Cuando cree o use una vista en el [Administrador de proyectos](#), éste abrirá la base de datos automáticamente. Por consiguiente, si usa una vista fuera del proyecto, deberá abrir la base de datos o asegurarse de que la base de datos está en el alcance antes de poder utilizar la vista.

### Crear vistas con instrucciones SQL SELECT almacenadas

Puede usar [sustitución de macros](#) para almacenar la instrucción SQL SELECT en una [variable](#) que puede llamar con la cláusula AS del comando CREATE SQL VIEW. Por ejemplo, el código siguiente almacena una instrucción SQL SELECT en la variable `emp_cust_sql`, que se utiliza para crear una vista nueva.

```
emp_cust_sql = "SELECT employee.emp_id, ;  
customer.cust_id, customer.emp_id, ;  
customer.contact, customer.company ;  
FROM employee, customer ;  
WHERE employee.emp_id = customer.emp_id"
```

```
CREATE SQL VIEW emp_cust_view AS &emp_cust_sql
```

## Modificar vistas

Puede modificar vistas existentes en el Diseñador de vistas mediante el Administrador de proyectos o a través del lenguaje. Si quiere modificar la cadena SQL de la vista mediante programación debe crear una vista nueva. Entonces, puede guardar la nueva [definición de vista](#) y sobrescribir el nombre de la vista existente. Para modificar las propiedades de una vista, consulte [Establecer propiedades de las vistas y las conexiones](#) más adelante en este mismo capítulo.

**Sugerencia** En el Diseñador de vistas, puede abrir una vista existente y, a continuación, copiar la cadena SQL de sólo lectura y pegarla en el código como método abreviado para la creación de una vista mediante programación.

### Para modificar una vista

- En el [Administrador de proyectos](#), seleccione el nombre de la vista y elija **Modificar** para abrir el [Diseñador de vistas](#).

–O bien–

- Abra una base de datos y utilice el comando [MODIFY VIEW](#) con el nombre de la vista.

En el Diseñador de vistas, puede utilizar el menú Consulta o la [barra de herramientas Diseñador de vistas](#) para agregar una nueva tabla a la vista. El código siguiente muestra `product_view` en el Diseñador de vistas:

```
OPEN DATABASE testdata  
MODIFY VIEW product_view
```

## Cambiar el nombre de una vista

Puede cambiar el nombre de una vista desde el Administrador de proyectos o mediante el comando `RENAME VIEW`.

### Para cambiar el nombre de una vista

- En el [Administrador de proyectos](#), seleccione una base de datos, después el nombre de la vista y, a continuación, elija **Cambiar el nombre del archivo** en el menú **Proyecto**.

–O bien–

- Use el comando [RENAME VIEW](#).

Por ejemplo, el código siguiente cambia el nombre de `product_view` a `products_all_view`:

```
RENAME VIEW product_view TO products_all_view
```

La base de datos que contiene la vista debe estar abierta para que usted pueda cambiar el nombre de la vista.

## Eliminar una vista

Puede eliminar la [definición de una vista](#) de una base de datos en el Administrador de proyectos o con el comando DELETE VIEW. Antes de eliminar la vista, asegúrese de que la base de datos que la contiene está abierta y establecida como base de datos actual.

### Para eliminar una vista

- En el [Administrador de proyectos](#), seleccione una base de datos, después el nombre de la vista y, a continuación, elija **Quitar**.  
–O bien–
- Use el comando [DELETE VIEW](#) o el comando [DROP VIEW](#).

Por ejemplo, el código siguiente elimina product\_view y customer\_view de la base de datos:

```
DELETE VIEW product_view  
DROP VIEW customer_view
```

**Nota** Estos comandos producen el mismo resultado; DROP VIEW es la sintaxis estándar ANSI SQL para eliminar una vista SQL.

## Crear una vista de múltiples tablas

Para tener acceso a información relacionada almacenada en tablas distintas, puede crear una vista y agregar dos o más tablas, o puede modificar una vista existente agregando tablas. Para agregar las tablas, puede usar el Diseñador de vistas o el comando CREATE SQL VIEW. Después de agregar las tablas, puede expandir su control de los resultados de la vista con la condición de combinación definida entre las tablas.

### Para crear una vista de múltiples tablas

- En el [Administrador de proyectos](#), cree una vista y agregue las tablas que quiera en el [Diseñador de vistas](#).  
–O bien–
- Abra una base de datos y use el comando [CREATE SQL VIEW](#); agregue nombres de tabla a la cláusula FROM y condiciones de combinación.

Cuando se agregan las tablas al comando CREATE SQL VIEW se produce un producto cartesiano. Tiene que especificar una [condición de combinación](#) en la cláusula FROM o en la cláusula WHERE de la instrucción para emparejar registros relacionados entre las tablas. Si existen relaciones persistentes entre las tablas, se usan automáticamente como condiciones de

existen [relaciones persistentes](#) entre las tablas, se usan automáticamente como condiciones de combinación.

## Definir y modificar condiciones de combinación

Normalmente, para definir una condición de combinación se usan las relaciones establecidas en los campos clave [principal](#) y clave [externa](#) entre las tablas. Por ejemplo, puede que quiera buscar información sobre pedidos, incluyendo información sobre el cliente que hizo el pedido. Puede crear una vista con las tablas Customer y Orders. Especifique una condición de combinación para comparar valores de los campos que tienen en común y, generalmente, devolver los que son iguales. En el ejemplo, Customer y Orders tienen un campo Customer ID.

### Para definir condiciones de combinación en una vista

- En el [Administrador de proyectos](#), cree o modifique una vista y, a continuación, agregue las tablas que quiera en el [Diseñador de vistas](#).

–O bien–

- Abra una base de datos y use el comando [CREATE SQL VIEW](#); agregue nombres de tabla y condiciones de combinación a la cláusula FROM.

### Condiciones de combinación especificadas en el Diseñador de vistas y mostradas en la instrucción SELECT - SQL

The screenshot shows the 'Diseñador de vistas - vista\_cust\_orders' window. It displays three tables: Customer, Orders, and Orditems. The 'Combinación' tab is selected, showing the following joins:

Tipo	Nombre de campo	No	Criterios	Valor	Lógico
↔ Inner Joi	Customer.cust_id	=		Orders.cust_id	
↔ Inner Joi	Orders.order_id	=		Orditems.order_id	

Below the window, the SQL query for the view is shown:

```
SELECT *;
FROM testdata!customer INNER JOIN testdata!orders;
INNER JOIN testdata!orditems ;
ON Orders.order_id = Orditems.order_id ;
ON Customer.cust_id = Orders.cust_id
```



El código siguiente crea la nueva vista como se describe en el ejemplo anterior, usando la cláusula FROM para especificar las condiciones de combinación para la vista:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_view AS ;
    SELECT * FROM testdata!customer ;
        INNER JOIN testdata!orders ;
            ON customer.cust_id = orders.cust_id
```

La condición de combinación tiene varias características: el tipo de combinación, los campos que hay que combinar y el operador para comparar los campos. En este caso, en el que tenemos una [combinación interna](#), sólo se incluyen en el resultado las filas de la tabla `customer` que coinciden con uno o más registros de la tabla `orders`.

Para cambiar el resultado de la vista de forma que satisfaga sus necesidades específicas, puede especificar:

- [Campos](#) de la combinación
- Operadores de comparación entre los campos
- Una secuencia de combinaciones, si tiene dos [tablas](#) en la vista
- El tipo de [combinación](#)

Especificar combinaciones en campos que no sean clave [principal](#) ni clave [externa](#) puede ser útil en casos específicos, pero no se suele hacer en la mayor parte de las vistas.

Si cambia el operador de combinación, puede controlar qué registros se comparan y se devuelven de forma similar a un [filtro](#). Por ejemplo, si usa un campo de fecha en la combinación, puede usar el operador de combinación para incluir sólo los registros anteriores a una fecha específica.

Para obtener más información sobre la secuencia de combinaciones, consulte [Definir múltiples condiciones de combinación](#) más adelante en este mismo capítulo.

Si elige un tipo de [combinación](#) diferente puede expandir el resultado de su [consulta](#) para incluir los registros que cumplan la [condición de combinación](#) y los que no la cumplan. Si tiene más de dos tablas en la vista, puede cambiar el resultado si cambia el orden de combinaciones en la cláusula FROM.

Puede modificar los tipos de combinación de la vista con el Diseñador de vistas o a través del lenguaje.

### Para modificar un tipo de combinación

- Seleccione la ficha **Combinación**.  
–O bien–
- Haga doble clic en la línea de combinación.  
–O bien–

- Abra una base de datos y use el comando [CREATE SQL VIEW](#); agregue nombres de tablas y condiciones de combinación a la cláusula FROM.

### Incluir registros no coincidentes en el resultado

Si desea incluir filas no coincidentes en el resultado, puede usar una [combinación externa](#). Por ejemplo, puede querer una lista de todos los clientes y saber si han hecho un pedido o no. Además, para los clientes que han hecho pedidos, es posible que quiera incluir los números de pedido en la vista. Cuando usa una combinación externa, los [campos](#) vacíos de las filas no coincidentes devuelven [valores nulos](#).

También puede usar el lenguaje para crear esta vista con el código siguiente:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_view AS ;
    SELECT * FROM testdata!customer ;
        LEFT OUTER JOIN testdata!orders ;
            ON customer.cust_id = orders.cust_id
```

Para controlar qué registros no coincidentes están incluidos en la vista, puede elegir los siguientes tipos de combinación.

Para	Utilice
Devolver sólo registros desde ambas tablas que satisfagan la condición de combinación establecida entre los dos campos de la condición de combinación.	Condición interna
Devolver todos los registros de la tabla situada a la izquierda de la palabra clave JOIN y únicamente los registros coincidentes de la tabla situada a la derecha de la palabra clave.	Combinación externa izquierda
Devolver todos los registros de la tabla situada a la derecha de la palabra clave JOIN y únicamente los registros coincidentes de la tabla situada a la izquierda de la palabra clave.	Combinación externa derecha
Devolver registros coincidentes y no coincidentes de ambas tablas	Combinación externa completa

### Definir múltiples condiciones de combinación

Si crea vistas o [consultas](#) con más de dos tablas, puede cambiar el resultado por el orden en que estén especificadas las condiciones de combinación. Por ejemplo, es posible que quiera buscar información sobre los pedidos, incluyendo información sobre el empleado que hizo la venta y el cliente que hizo el pedido. Puede crear una vista con las tablas `customer`, `orders` y `employee` y especificando condiciones de combinación interna en los campos que tienen en común: `customer` y `orders` tienen un campo `customer ID`; `orders` y `employee` tienen un campo `employee ID`.

Esta vista tiene la siguiente instrucción SQL subyacente:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_emp_view AS ;
    SELECT * FROM testdata!customer ;
        INNER JOIN testdata!orders ;
            ON customer.cust_id = orders.cust_id ;
        INNER JOIN testdata!employee ;
            ON orders.emp_id = employee.emp_id
```

### Usar combinaciones en la cláusula WHERE

Puede especificar sus [condiciones de combinación](#) en la cláusula WHERE; sin embargo, no puede especificar un tipo de combinación de la misma forma que puede hacerlo en combinaciones en la cláusula FROM. Para vistas remotas, la cláusula de combinación aparece siempre en la cláusula WHERE.

El código siguiente crea la misma vista que el ejemplo anterior, con la cláusula WHERE para especificar las condiciones de combinación para la vista:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_emp_view AS ;
    SELECT * FROM testdata!customer, ;
        testdata!orders, testdata!employee ;
    WHERE customer.cust_id = orders.cust_id ;
    AND orders.emp_id = employee.emp_id
```

### Acceso a datos remotos

Cuando desee utilizar datos situados en un servidor remoto, deberá crear una [vista remota](#). Para crear una vista remota, primero debe ser capaz de conectarse a un [origen de datos](#).

#### Conectarse a un origen de datos remoto

Un origen de datos remoto es generalmente un servidor remoto para el que ha instalado un controlador ODBC y establecido un nombre de origen de datos ODBC. Para tener un [origen de datos](#) válido, debe asegurarse de que [ODBC](#) está instalado. Desde Visual FoxPro, puede definir un origen de datos y conexiones.

Para obtener información sobre cómo establecer un origen de datos ODBC, consulte el capítulo 1, [Instalar Visual FoxPro](#), de la *Guía de instalación e Índice principal*.

#### Definir una conexión

En Visual FoxPro, puede crear y almacenar en una [base de datos](#) una definición de [conexión con nombre](#), a la que se puede referir desde ese momento por su nombre cuando cree una [vista remota](#). También puede establecer las [propiedades](#) de la [conexión con nombre](#) para optimizar la comunicación entre Visual FoxPro y el origen de [datos remoto](#). Cuando active una vista remota, la [conexión](#) de la vista se convertirá en el canal hacia el origen de datos remoto.

#### Para crear una conexión con nombre

- En el [Administrador de proyectos](#), seleccione **Conexiones** y, a continuación, elija **Nuevo** para abrir el [Diseñador de conexiones](#).

–O bien–

- Abra una base de datos y utilice el comando [CREATE CONNECTION](#) para abrir el **Diseñador de conexiones**.

–O bien–

- Use el comando [CREATE CONNECTION](#) con un nombre de conexión.

Por ejemplo, para crear una conexión en la base de datos `testdata` que almacene la información necesaria para conectarse al origen de datos ODBC `sqlremote`, puede escribir el código siguiente:

```
OPEN DATABASE testdata
CREATE CONNECTION remote_01 DATASOURCE sqlremote userid password
```

Visual FoxPro muestra `remote_01` como el nombre de la conexión en el Administrador de proyectos.

La creación de una conexión con nombre en su base de datos no utiliza ningún recurso remoto ni de red, ya que Visual FoxPro no activa la conexión hasta que usted utiliza la vista. Hasta que active la conexión, la conexión con nombre sólo existe como una definición de conexión almacenada como una fila en el archivo `.dbc` de la base de datos. Cuando utilice una vista remota, Visual FoxPro usará la conexión con nombre mencionada en la vista para crear una conexión activa con el origen de datos remoto y enviará la solicitud de datos al origen remoto utilizando la conexión activa como canal.

Puede crear opcionalmente una vista que especifique únicamente el nombre del [origen de datos](#), en lugar del nombre de la [conexión](#). Cuando utilice la vista, Visual FoxPro usará la información de [ODBC](#) acerca del origen de datos para crear y activar una conexión con el origen de datos. Cuando cierre la vista se cerrará la conexión.

### Prioridad de nombres para conexiones y orígenes de datos

Cuando utilice el comando [CREATE SQL VIEW](#) con la cláusula `CONNECTION`, especifique un nombre que represente una [conexión](#) o bien un [origen de datos](#). Visual FoxPro buscará primero en la base de datos actual una conexión con el nombre que especificó. Si no existe ninguna conexión con ese nombre en la base de datos, Visual FoxPro buscará entonces un origen de datos ODBC establecido con el nombre especificado. Si su base de datos actual contiene una conexión con nombre cuyo nombre coincide con el de un origen de datos ODBC de su sistema, Visual FoxPro la encontrará y utilizará la conexión con nombre.

### Mostrar instrucciones de inicio de sesión ODBC

Cuando use una vista cuya información de registro de [conexión](#) no esté totalmente especificada, Visual FoxPro puede mostrar un cuadro de diálogo específico del [origen de datos](#) que le pida la información que falte.

Puede controlar si Visual FoxPro le pedirá la información que dejó sin especificar en el momento de la conexión.

### Para controlar la presentación de instrucciones de inicio de sesión ODBC

- En el [Administrador de proyectos](#), seleccione el nombre de la conexión y elija **Modificar** para abrir el [Diseñador de conexiones](#).
- En el área **Mostrar instrucciones de inicio de sesión ODBC**, elija una opción.  
–O bien–
- Use la propiedad DispLogin de las funciones [DBSETPROP\(\)](#) o [SQLSETPROP\(\)](#).

### Usar una conexión existente

Puede utilizar una [conexión con nombre](#) ya existente para crear una [vista remota](#). Puede ver una lista de las conexiones disponibles en la base de datos y utilizar el Administrador de proyectos o el comando DISPLAY CONNECTIONS.

### Para determinar las conexiones existentes

- En el [Administrador de proyectos](#), seleccione una base de datos y seleccione **Conexiones**.  
–O bien–
- Use el comando [DISPLAY CONNECTIONS](#).

Por ejemplo, el código siguiente muestra las conexiones de la base de datos testdata:

```
OPEN DATABASE testdata  
DISPLAY CONNECTIONS
```

### Crear una vista remota

Cuando tiene un [origen de datos](#) o una [conexión con nombre](#) válidos, puede crear una [vista remota](#) con el Administrador de proyectos o el lenguaje de programación. Una vista remota es similar a una [vista local](#), pero usted agrega un nombre de conexión o de origen de datos al definir la vista. La instrucción SQL de la vista remota utiliza el dialecto nativo del servidor.

### Para crear una vista remota

- En el [Administrador de proyectos](#), seleccione una base de datos, seleccione **Vistas remotas** y elija **Nuevo** para abrir el [Diseñador de vistas](#).  
–O bien–
- Use el comando [CREATE SQL VIEW](#) con la cláusula REMOTE o la cláusula CONNECTION

## CONNECTION.

Si usa la cláusula **CONNECTION** con el comando **CREATE SQL VIEW**, no necesitará incluir la palabra clave **REMOTE**. Visual FoxPro identifica la vista como remota por la presencia de la palabra clave **CONNECTION**. Por ejemplo, si tiene la tabla `products` de la base de datos `Testdata` en un servidor remoto, el código siguiente creará una vista remota de la tabla `products`:

```
OPEN DATABASE testdata
CREATE SQL VIEW product_remote_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM products
```

Puede utilizar el nombre de un origen de datos en lugar del nombre de una conexión cuando cree una vista remota. También puede elegir entre omitir el nombre de la conexión o del origen de datos remoto cuando utilice el comando **CREATE SQL VIEW** con la cláusula **REMOTE**. Visual FoxPro mostrará entonces el cuadro de diálogo Selección de conexión u [Origen de datos](#), en el que podrá elegir un origen de datos o una conexión válidos.

Después de crear una vista, puede abrir el [Diseñador de bases de datos](#); la vista se mostrará de la misma manera que una tabla en el [esquema](#) con el nombre de la vista y un icono en lugar del nombre de la tabla y un icono.

Si combina dos o más tablas en el Diseñador de vistas remotas, el Diseñador usa combinaciones internas (o [equicombinaciones](#)) y coloca la condición de combinación en la cláusula **WHERE**. Si quiere usar una [combinación externa](#), el Diseñador de vistas remotas sólo proporciona combinaciones externas izquierdas, la [sintaxis](#) con la que es compatible ODBC. Si necesita combinaciones externas derechas o completas o sólo quiere usar una sintaxis nativa para una combinación externa izquierda, cree la vista mediante programación.

## Usar vistas

Después de haber creado una vista, puede utilizarla para mostrar y actualizar datos. También puede modificar las [propiedades](#) de una vista para aumentar el rendimiento de la misma. Se trata la vista como una tabla:

- Abra la vista con el comando [USE](#) y el nombre de la vista.
- Cierre la vista con el comando **USE**.
- Muestre los registros de la vista en una [ventana Examinar](#).
- Muestre los alias de vista abierta en la [ventana Sesión de datos](#).
- Use la vista como un [origen de datos](#) como en un formulario o un control Grid.

Puede utilizar una vista con el Administrador de proyectos o con el lenguaje de programación.

### Para usar una vista

- En el [Administrador de proyectos](#), seleccione una base de datos, elija el nombre de la vista y elija **Examinar** para mostrar la vista en una ventana **Examinar**.

—O bien—

- Tenga acceso a la vista mediante programación con el comando [USE](#).

El código siguiente muestra `product_view` en una ventana Examinar:

```
OPEN DATABASE testdata
USE product_view
BROWSE
```

Cuando utilice una vista, ésta se abrirá como un [cursor](#) en su propia [área de trabajo](#). Si la vista está basada en tablas locales, Visual FoxPro también abrirá las [tablas base](#) en áreas de trabajo distintas. Las tablas base de una vista son las tablas a las que tiene acceso la instrucción `SELECT - SQL` que usted incluye en el comando [CREATE SQL VIEW](#) al crear una vista. En el ejemplo anterior, al usar `product_view` también se abre automáticamente la tabla `products`.

### La ventana Sesión de datos muestra la vista y su tabla base



Cuando una vista está basada en tablas remotas, estas tablas no se abren en áreas de trabajo. En la ventana Sesión de datos sólo aparece el nombre de la vista remota.

### Limitar el alcance de una vista

Cuando tiene acceso a un [origen de datos](#) remoto, tendrá acceso a cantidades potencialmente masivas de datos. Puede limitar el alcance de los datos seleccionados en la vista únicamente a los registros que necesite en cada momento. Esto reduce el tráfico en la red y aumenta el rendimiento de su vista. Por ejemplo, si desea ver información acerca de los clientes de un determinado país y sus pedidos, mejorará el rendimiento si descarga en la vista sólo los registros correspondientes a ese país, en lugar de los de todos los clientes.

Un método que puede utilizar para limitar el alcance de su vista es agregar una cláusula `WHERE` a la instrucción `SQL` de su vista. Si deseara buscar los registros de los clientes de Suecia, podría crear esta cláusula `SQL WHERE` para su vista:

cláusula SQL **WHERE** para su vista:

```
SELECT * FROM customer ;  
WHERE customer.country = 'Suecia'
```

Este código limitará el alcance de su vista al descargar únicamente los registros correspondientes a los clientes de Suecia, pero también puede que sea necesaria la creación de una vista distinta para cada país, ya que el valor real de `customer.country` para un país está codificado en la instrucción **SELECT** de su vista.

### Crear una vista parametrizada

Puede limitar el alcance de una vista sin crear una vista distinta para cada subconjunto de registros si crea vistas parametrizadas. Una vista parametrizada crea una cláusula **WHERE** en la instrucción SQL **SELECT** de la vista que limita los registros descargados únicamente a aquellos que cumplen las condiciones de la cláusula **WHERE** que se creó con los valores proporcionados para el parámetro. Este valor se puede proporcionar en tiempo de ejecución o se puede transferir a la vista mediante programación.

En el caso del ejemplo anterior, puede crear una vista que le permita descargar los registros para cualquier país escribiendo simplemente el nombre del país al abrir la vista.

### Para crear una vista parametrizada

- En el [Diseñador de vistas](#), elija **Parámetros de vista** en el menú **Consulta**.

–O bien–

- Use el comando [CREATE SQL VIEW](#) con un símbolo "?" y un [parámetro](#).

El parámetro que usted proporciona se evalúa como una [expresión](#) de Visual FoxPro y el valor se envía como parte de la instrucción SQL de la vista. Si la evaluación falla, Visual FoxPro pedirá el valor del parámetro. Por ejemplo, si tiene la tabla `customer` de la base de datos `Testdata` en un servidor remoto, el código siguiente creará una vista remota parametrizada que limita la vista a aquellos clientes cuyo país coincida con el valor proporcionado para el parámetro `?cCountry`:

```
OPEN DATABASE testdata  
CREATE SQL VIEW customer_remote_view ;  
CONNECTION remote_01 ;  
AS SELECT * FROM customer ;  
WHERE customer.country = ?cCountry
```

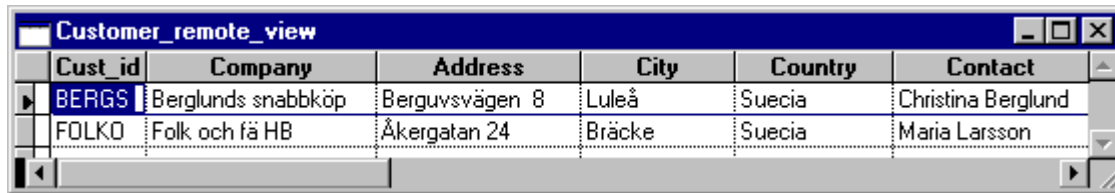
Puede proporcionar un valor para `?cCountry` mediante programación cuando utilice la vista. Por ejemplo, podría escribir el código siguiente:

```
cCountry = 'Suecia'  
USE Testdata!customer_remote_view IN 0  
BROWSE
```

Visual FoxPro mostrará los registros de los clientes para las compañías suecas en la ventana `Examinar Customer_remote_view`.



## Vista que muestra los registros cuyo país coincide con el parámetro proporcionado



Cust_id	Company	Address	City	Country	Contact
BERGS	Berglunds snabbköp	Berguvsvägen 8	Luleå	Suecia	Christina Berglund
FOLKO	Folk och få HB	Åkergatan 24	Bräcke	Suecia	Maria Larsson

**Sugerencia** Si su [parámetro](#) es una [expresión](#), escriba la expresión entre paréntesis. Esto permite que toda la expresión se evalúe como parte del parámetro.

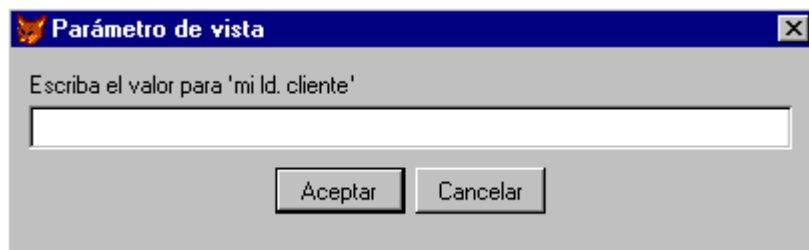
## Pedir al usuario para que escriba un valor de parámetro

Si su [parámetro](#) no es una [variable](#) o una [expresión](#), quizá desee pedir al usuario que suministre el valor del parámetro mediante la utilización de una cadena entre comillas como parámetro de la vista. Cuando cree un parámetro de vista con una cadena entre comillas después del símbolo "?", Visual FoxPro no interpretará la cadena como una expresión. En su lugar, le pedirá que introduzca el valor del parámetro en [tiempo de ejecución](#). Por ejemplo, el código siguiente crea una vista remota parametrizada que pide al usuario que suministre un valor para el parámetro '?mi id cliente':

```
OPEN DATABASE testdata
CREATE SQL VIEW customer_remote_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM customer ;
    WHERE customer.cust_id = '?mi id cliente'
USE customer_remote_view
```

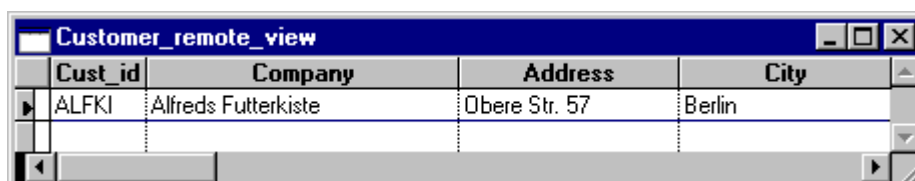
Cuando utilice la vista del ejemplo anterior aparecerá el cuadro de diálogo Parámetro de vista.

## El cuadro de diálogo Parámetro de vista pide el valor de la cadena entre comillas



Cuando haya introducido un identificador (Id.) de usuario válido, Visual FoxPro recuperará el registro que coincide con ese Id.. Si introduce el valor 'ALFKI' en el ejemplo anterior y examina Customer\_remote\_view, verá el registro del cliente en la ventana Examinar.

## Ventana Examinar que muestra el registro para el cust\_id ALFKI



Cust_id	Company	Address	City
ALFKI	Alfreds Futterkiste	Obere Str. 57	Berlin

Mediante la utilización de una cadena entre comillas como parámetro de vista, se asegurará de que Visual FoxPro pedirá siempre al usuario el valor del parámetro.

## Abrir múltiples instancias de una vista

Puede abrir múltiples instancias de una vista en [áreas de trabajo](#) distintas, del mismo modo que puede abrir una tabla en más de un área de trabajo. A diferencia de las [tablas](#), las vistas recopilan un nuevo conjunto de datos predeterminado cada vez que usted las utiliza.

### Para abrir una vista en múltiples áreas de trabajo

- En el [Administrador de proyectos](#), elija el nombre de la vista y después **Examinar** para mostrar la vista en una ventana **Examinar**. Repita este proceso para abrir la vista en áreas de trabajo adicionales.

–O bien–

- En la ventana [Sesión de datos](#), elija **Abrir** y el nombre de la vista. Repita este proceso para abrir la vista en áreas de trabajo adicionales.

–O bien–

- Tenga acceso a la vista mediante programación con el comando [USE](#).

Cuando tiene acceso a la vista mediante programación con el comando USE, puede elegir abrir otra [instancia](#) de la vista sin necesidad de volver a consultar el [origen de datos](#). Esto resulta especialmente útil cuando quiere abrir una vista remota en múltiples [áreas de trabajo](#) sin esperar a que los datos se descarguen desde un origen de datos remoto.

### Para usar una vista de nuevo sin descargar datos

- Use la cláusula NOREQUERY con el comando [USE](#).

–O bien–

- Use la cláusula AGAIN con el comando [USE](#).

El código siguiente utiliza la cláusula NOREQUERY para mostrar el [cursor](#) buscado de la primera [instancia](#) de product\_remote\_view en dos ventanas Examinar sin volver a consultar el [origen de datos](#) remoto:

```
OPEN DATABASE testdata
CREATE SQL VIEW product_remote_view ;
CONNECTION remote_01 ;
```

```
AS SELECT * FROM products
USE product_remote_view
BROWSE
SELECT 0
USE product_remote_view NOREQUERY
BROWSE
```

Puede especificar un número de sesión mediante la cláusula NOREQUERY. Si no especifica un número de sesión, Visual FoxPro buscará en todas las [sesiones](#). Si se encuentra un conjunto de resultados ya abierto para la vista, se volverá a abrir un [cursor](#) con el mismo conjunto de resultados. Si no se encuentra un conjunto de resultados ya abierto, se obtendrá un nuevo conjunto para la vista. Del mismo modo que en las tablas, si no se encuentra la vista se abrirá un nuevo cursor de vista.

Si desea que Visual FoxPro busque sólo en la sesión actual un conjunto de resultados abierto para su vista, puede utilizar la cláusula AGAIN. El código siguiente muestra `product_remote_view` en dos ventanas Examinar:

```
OPEN DATABASE testdata
USE product_remote_view
BROWSE
USE product_remote_view AGAIN in 0
BROWSE
```

Cuando utilice la cláusula AGAIN, Visual FoxPro buscará un [cursor](#) de vista existente en la sesión actual y abrirá un [alias](#) adicional que apunta a este cursor de vista. Abrir otra instancia de una vista con la cláusula AGAIN equivale a ejecutar [USE](#) con la cláusula NONQUERY con el número de sesión actual..

## Mostrar la estructura de una vista

Puede abrir y mostrar únicamente la estructura de una vista mediante la cláusula NODATA del comando USE. Esta opción resulta especialmente útil cuando se quiere ver la estructura de una [vista remota](#) sin tener que esperar a la descarga de datos.

### Para abrir una vista sin datos

- Tenga acceso a la vista mediante programación con el comando [USE](#) y la cláusula NODATA.

El código siguiente muestra `customer_remote_view` sin datos en una ventana Examinar:

```
OPEN DATABASE testdata
USE customer_remote_view NODATA in 0
BROWSE
```

El uso de una vista con la cláusula NODATA siempre abre un nuevo [cursor](#) de la vista. La cláusula NODATA es siempre el modo más rápido de obtener la estructura de una vista, porque crea el menor cursor posible en el origen de datos remoto. Cuando utilice la cláusula NODATA, Visual FoxPro creará una cláusula WHERE para la vista que siempre devolverá un valor falso. Puesto que ningún registro del origen de datos puede cumplir la condición de la cláusula WHERE, no se seleccionará ninguna fila en el cursor del [origen de datos](#) remoto. Su vista se recuperará rápidamente ya que no espera que el origen de datos remoto cree un cursor potencialmente grande.

espera que el origen de datos remoto cree un cursor potencialmente grande.

**Sugerencia** El uso de la cláusula NODATA es más eficaz que usar un valor 0 para la propiedad MaxRecords en su vista o en el cursor. Cuando utilice la propiedad MaxRecords, deberá esperar mientras el origen de datos remoto construye un cursor para la vista que contiene todas las filas de datos que cumplen las condiciones normales de la cláusula WHERE. Las filas del cursor completo de la vista remota se descargarán de acuerdo con el valor de la propiedad MaxRecords.

## Crear un índice en una vista

Puede crear [índices](#) locales en una vista, igual que en una tabla, mediante el comando [INDEX](#) ON. A diferencia de los índices generados para una [tabla](#), los índices locales que cree en una vista no se almacenarán definitivamente: desaparecerán cuando cierre la vista.

**Sugerencia** Considere el tamaño del conjunto de resultados de su vista cuando decida si va a crear un índice local en una vista. Indexar un conjunto de resultados grande puede llevar un tiempo considerable y disminuir el rendimiento de su vista.

Para obtener más información acerca de la creación de índices, consulte el capítulo 7, [Trabajar con tablas](#), o vea [INDEX](#).

## Crear relaciones temporales en las vistas

Puede crear [relaciones temporales](#) entre índices de vistas o entre índices de vistas e índices de tablas mediante el comando [SET RELATION](#).

Para obtener un mejor rendimiento cuando utilice el comando SET RELATION para relacionar una vista y una tabla, haga que la vista sea el objeto primario y la tabla el objeto secundario en la relación. Hacer que la tabla sea el objeto secundario es más eficaz porque el índice estructural de la tabla se mantiene constantemente, se tiene acceso al mismo de forma más rápida y lo puede usar el [entorno de datos](#) para ordenar los registros. Hay que volver a generar el índice de la vista cada vez que ésta se activa y tarda más tiempo que el índice de la tabla. Un índice de una vista no forma parte de la definición de la vista; por lo tanto, si usa un entorno de datos, la vista no puede ser el objeto secundario porque el índice del objeto secundario tiene que existir como parte de la definición y esto no lo admiten las vistas.

## Establecer las propiedades de vistas y conexiones

Cuando cree una vista, ésta heredaré los valores de las propiedades, como UpdateType y UseMemoSize, del cursor de entorno o cursor 0 de la [sesión](#) actual. Puede cambiar estos valores predeterminados de las propiedades mediante la función [CURSORSETPROP\(\)](#) con 0 como número de cursor. Una vez creada la vista y almacenada en una base de datos, puede cambiar las propiedades de la vista mediante la función [DBSETPROP\(\)](#). Los cambios que realice a las propiedades de una vista en una base de datos se almacenarán definitivamente en la base de datos.

Cuando utilice una vista, el cursor activo de la vista heredaré los valores de las propiedades almacenados para la vista en la base de datos. Puede cambiar estas propiedades en el cursor activo mediante la función [CURSORSETPROP\(\)](#) para el cursor de vista. Los cambios realizados por la función CURSORSETPROP() son temporales. El valor temporal para la vista activa desaparece al cerrar la vista, mientras que el valor temporal para el cursor 0 desaparece al cerrar la sesión de Visual

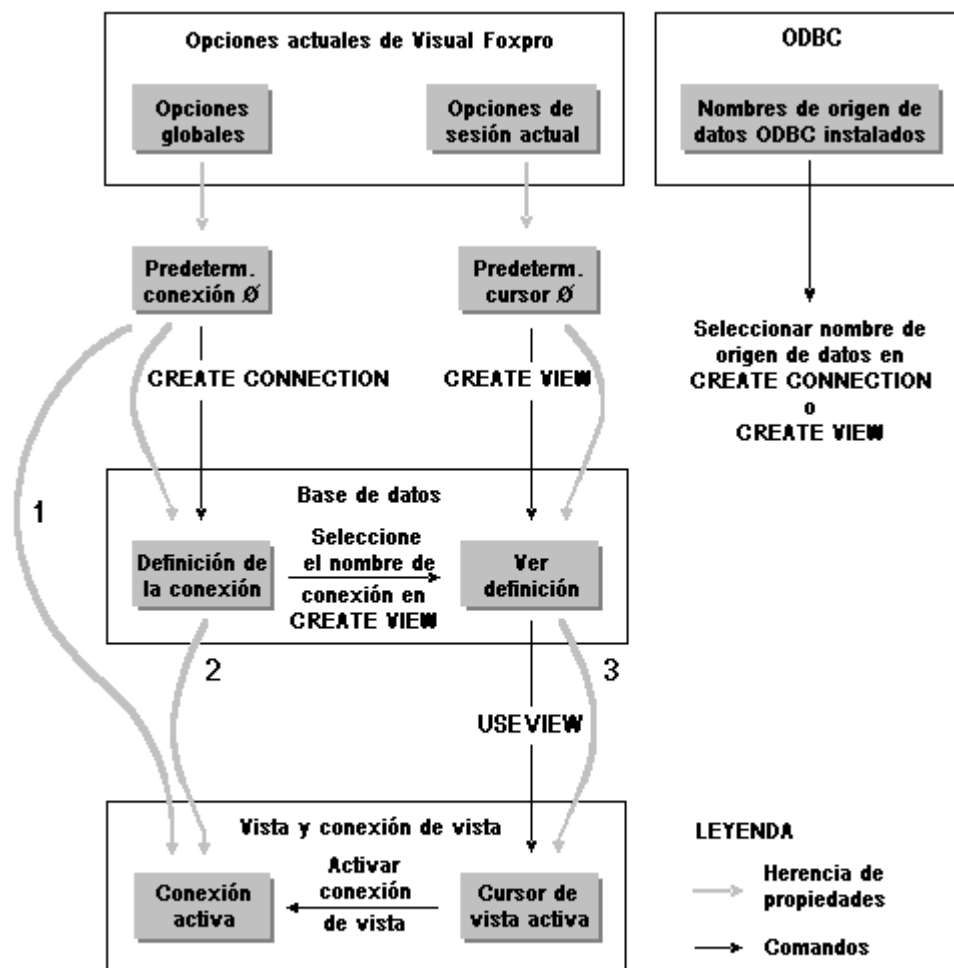
cerrar la vista, mientras que el valor temporal para el cursor o desaparece al cerrar la sesión de Visual FoxPro.

Las [conexiones](#) heredan las propiedades de una forma similar. Las propiedades predeterminadas de la conexión 0 se heredan cuando usted crea y almacena una [conexión con nombre](#) en una base de datos. Puede cambiar estos valores predeterminados de propiedades para la conexión 0 mediante la función [SQLSETPROP\(\)](#). Después de haber creado la conexión y haberla almacenado en una base de datos, puede cambiar las propiedades de conexión mediante la función [DBSETPROP\(\)](#). Cuando utilice una conexión, la conexión activa heredará los valores de las propiedades almacenados para la conexión en la base de datos. Puede cambiar estas propiedades en la conexión activa con la función [SQLSETPROP\(\)](#) para el controlador de la conexión.

Tanto las [vistas](#) como las [conexiones](#) pueden utilizar un [origen de datos](#) ODBC con nombre. Si utiliza un origen de datos ODBC en una vista, la conexión heredará las propiedades de la configuración predeterminada de la [sesión](#).

El diagrama siguiente ilustra la herencia de propiedades para las vistas y conexiones. Las líneas en gris representan el flujo de herencia de propiedades, mientras que las líneas negras representan los comandos de Visual FoxPro.

### Propiedades de vistas y conexiones, y sus herencias



<sup>1</sup> Si la vista se basa en un origen de datos ODBC, las propiedades de la conexión activa se heredarán en la conexión 0.

<sup>2</sup> Si la vista se basa en una conexión de base de datos, las propiedades de la conexión activa se heredarán de la conexión 0.

<sup>1</sup> Si la vista se basa en una conexión de bases de datos, las propiedades de la conexión activa se heredan de la definición de conexión de la base de datos.

<sup>2</sup> Las propiedades del cursor de vista activa se heredan de la definición de vista de la base de datos.

## Cambiar tipos de datos predeterminados al descargar vistas remotas

Cuando cree una vista, se establece la propiedad `DataType` para todos los campos a un valor predeterminado. El valor es la letra del tipo de datos (D, G, I, L, M, P, T, Y) para tipos de datos de longitud fija y la letra seguida por parámetros de precisión y escala entre paréntesis (B(*d*), C(*n*), N(*n*,*d*)) para tipos de longitud variable. Esta propiedad es de sólo lectura para vistas locales. Para ver una lista de tipos predeterminados, consulte "Descargar y cargar datos de vistas remotas" en el capítulo 21, [Implementar una aplicación cliente-servidor](#).

Puede modificar el valor de la propiedad `DataType` para el campo vista remota con la función [DBSETPROP\(\)](#) como se muestra en esta tabla.

Tipos de datos ODBC para campos remotos	Tipos de datos posibles de un cursor de Visual FoxPro
SQL_CHAR SQL_VARCHAR SQL_LONGVARCHAR	Character o Memo <sup>1</sup> (predeterminado); también General o Picture
SQL_BINARY SQL_VARBINARY SQL_LONGVARBINARY	Memo (predeterminado); también Character, General o Picture
SQL_DECIMAL SQL_NUMERIC	Numeric o Currency <sup>2</sup> (predeterminado); también Character, Integer o Double
SQL_BIT	Logical (predeterminado); también Character
SQL_TINYINT SQL_SMALLINT SQL_INTEGER	Integer (predeterminado); también Character, Numeric, Double o Currency
SQL_BIGINT	Character (predeterminado); también Integer, Numeric, Double o Currency
SQL_REAL SQL_FLOAT SQL_DOUBLE	Double (predeterminado); el número de caracteres decimales es el valor de SET DECIMALS en Visual FoxPro; también Character, Integer, Numeric o Currency
SQL_DATE	Date (predeterminado); también Character o DateTime
SQL_TIME	DateTime <sup>3</sup> (predeterminado); también Character
SQL_TIMESTAMP	DateTime <sup>4</sup> (predeterminado); también Character o Date

1. Si el ancho del campo ODBC es menor que el valor de la propiedad de cursor `UseMemoSize`, se convierte en un campo Character en el cursor de Visual FoxPro; si no, se convierte en un campo Memo.

2. Si el campo de servidor es un tipo de datos monetario, se convierte en un tipo de datos Currency en Visual FoxPro.
3. El valor predeterminado del día es 1/1/1900.
4. Si el valor del campo SQL\_TIMESTAMP contiene fracciones de segundo, las fracciones se truncan cuando el valor se convierte en un tipo de datos DateTime de Visual FoxPro.

### Usar la propiedad DataType

Puede usar la propiedad DataType para elegir un [tipo de datos](#) diferente que el predeterminado. Por ejemplo, es posible que quiera descargar un campo marca de hora del servidor a Visual FoxPro, pero la asignación de tipo de datos predeterminada a un campo DateTime de Visual FoxPro truncaría cualquier fracción de segundo almacenada en la marca de hora de servidor. Puede usar la propiedad DataType para asignar el campo marca de hora remota a un campo Character de Visual FoxPro para conservar las fracciones de segundo.

### Cerrar las tablas base de una vista

Las [tablas base](#) locales abiertas automáticamente cuando usa una vista no se cierran automáticamente cuando cierra una vista; debe cerrarlas explícitamente. Esto es coherente con el comando SELECT - SQL.

## Actualizar datos en una vista

Puede actualizar los datos en una vista del mismo modo que lo haría en una tabla. Con una vista también puede actualizar las [tablas base](#) de la vista. Las vistas se almacenan de forma predeterminada en un búfer de filas con el método optimista. Puede cambiarlo para que se almacenen en un búfer de tablas; para obtener más información acerca del almacenamiento en búfer, consulte el capítulo 17, [Programar para acceso compartido](#).

Puede actualizar datos en una vista a través de la interfaz o mediante el lenguaje de programación. El primer paso para actualizar los datos de una vista consiste en hacer actualizable la vista. En la mayoría de los casos, los valores predeterminados de las propiedades preparan automáticamente la vista para que sea actualizable, pero las actualizaciones no se envían al origen de datos hasta que usted indique a Visual FoxPro que lo haga estableciendo a On la propiedad SendUpdates.

Una vista utiliza cinco propiedades para controlar las actualizaciones. Estas propiedades se relacionan a continuación, junto con sus valores predeterminados:

### Propiedades de actualización de vistas y sus valores predeterminados

Propiedad de vista	Valores predeterminados
Tables	Incluye todas las tablas que tienen campos actualizables y al menos un campo clave principal.
KeyField	Campos clave de la base de datos y <a href="#">claves principales</a> y remotas de la



	tabla.
UpdateName	Nombre_tabla.nombre_columna para todos los campos.
Updatable	Todos los campos excepto los campos de clave principal.
SendUpdates	De forma predeterminada son los valores predeterminados de la sesión, que originalmente se establece en falso (.F.); si cambia el valor a verdadero (.T.), se convertirá en el valor predeterminado para todas las vistas creadas en la sesión.
CompareMemo	El valor predeterminado es verdadero (.T.), lo cual significa que los campos memo se incluyen en la cláusula WHERE y se usan para detectar conflictos de actualización.

Si bien las cinco propiedades son necesarias para actualizar datos, la propiedad SendUpdates actúa como "conmutador principal" que controla si se envían o no las actualizaciones. A medida que desarrolla su aplicación, puede desactivar la propiedad SendUpdates y configurar las demás propiedades para permitir actualizaciones de los campos que desee actualizar. Cuando vaya a probar su aplicación, podrá activar la propiedad SendUpdates para iniciar el flujo de actualizaciones.

En algunas situaciones más complejas, puede que los valores predeterminados de actualización no proporcionen actualizaciones para una vista que usted creó mediante programación. Para permitir las actualizaciones, examine el valor predeterminado de cada una de las propiedades de actualización y ajústelo según sea necesario. También puede especificar propiedades adicionales, tales como UpdateType, WhereType, etc., de acuerdo con sus preferencias. Para ver una lista completa de las propiedades de vista, vea [DBGETPROP\(\)](#).

### Para hacer que una vista sea actualizable desde el Diseñador de vistas

- En el [Diseñador de vistas](#), seleccione la ficha **Criterios de actualización** y active el valor predeterminado.

El valor predeterminado de las vistas que usted crea mediante el Diseñador de vistas normalmente prepara la vista para que sea actualizable; sólo necesita activar la casilla de verificación "Enviar actualizaciones SQL" para activar las actualizaciones. Además puede modificar las tablas, los campos, la cláusula SQL WHERE y las opciones de actualización como desee.

### Para hacer que una vista sea actualizable estableciendo las propiedades de actualización de la vista

- Examine el valor predeterminado actual con el comando [DISPLAY DATABASE](#) y modifique después las propiedades de la definición de vista como desee con la función [DBSETPROP\(\)](#).

El ejemplo siguiente muestra los pasos que podría seguir para especificar las cinco propiedades de actualización de vista mediante programación:

**Nota** Las propiedades View predeterminadas pueden suministrar toda la información necesaria para actualizar la vista.



1. Establecer la propiedad Tables con al menos el nombre de una tabla.

Por ejemplo, si tuviera una vista basada en la tabla `customer`, podría configurar el nombre de la tabla con la siguiente función:

```
DBSETPROP('cust_view','View','Tables','customer')
```

**Sugerencia** Si una tabla aparece como un cualificador en la propiedad UpdateName pero no está incluida en la lista predeterminada de la propiedad Tables, quizá no tenga especificado un campo clave principal. Haga que la tabla sea actualizable; para ello, agregue a la lista de la propiedad el campo que considere que es un campo clave y, a continuación, agregue la tabla a la lista de la propiedad Tables.

2. Establezca la propiedad KeyField con uno o más nombres de campos locales de Visual FoxPro que, juntos, definan una clave única para la tabla de actualización.

Con el mismo ejemplo, podría hacer que `cust_id` fuese el campo clave mediante el código siguiente:

```
DBSETPROP('cust_view.cust_id','Field','KeyField',.T.)
```

**Precaución** Asegúrese de que el campo o los campos clave que especifique definan una clave única tanto en la tabla base que desee actualizar como en la vista.

3. Asigne los campos de la vista a los campos de su tabla base con la propiedad UpdateName. Esta propiedad resulta especialmente útil cuando la vista está basada en una combinación de dos tablas con un nombre de campo común o cuando los campos tienen alias en la vista. Para actualizar la tabla base deseada, asigne el nombre del campo de la vista de Visual FoxPro al campo de la tabla base y al nombre de la tabla.

```
DBSETPROP('cust_view.cust_id','Field','UpdateName',;  
'customer.cust_id')
```

**Sugerencia** Para evitar la creación de campos sinónimos en su vista, puede cualificar nombres de campo en la instrucción SQL que utilice para generar la vista. Después, utilice la propiedad UpdateName de la vista de Visual FoxPro para asociar cada campo cualificado a la tabla base y al campo correspondiente.

4. Especifique el alcance de los campos que desea actualizar mediante la propiedad UpdateField. Debe especificar solamente aquellos campos también especificados con la propiedad UpdateName.

```
DBSETPROP('cust_view.cust_id','Field','Updatable',;  
.T.)
```

5. Establezca la propiedad SendUpdates como verdadera (.T.). Se trata del conmutador principal que indica a Visual FoxPro que debe crear y enviar actualizaciones a cualquiera de las tablas y campos que usted especificó como actualizables.

```
DBSETPROP('cust_view','View','SendUpdates',.T.)
```

Cuando utilice [DBSETPROP\(\)](#) para establecer las propiedades de una vista antes de usarla, el valor se almacenará en la base de datos y se usará automáticamente siempre que active la vista. Cuando la vista está activa, puede utilizar [CURSORSETPROP\(\)](#) para cambiar el valor de las propiedades de la vista activa. El valor de las propiedades que establezca para una vista activa mediante [CURSORSETPROP\(\)](#) no se almacenará al cerrar la vista.

## Actualizar múltiples tablas en una vista

Puede actualizar múltiples tablas base desde una vista. Cuando su vista combine dos o más tablas, debe establecer las propiedades para asegurarse de que sólo sea actualizable el lado varios de la consulta de la vista.

Las vistas se actualizan tabla por tabla. Debe asegurarse de que para cada tabla a la que se tiene acceso en una vista, el conjunto de campos clave sea una clave única tanto para el conjunto resultante de la vista como para la tabla base.

### Para hacer actualizable una vista de múltiples tablas

- En el [Diseñador de vistas](#), elija la ficha **Criterios de actualización** y seleccione las tablas y los nombres de los campos que desea actualizar.

–O bien–

- Use la función [DBSETPROP\(\)](#).

En la mayoría de los casos, los valores predeterminados proporcionados por Visual FoxPro preparan una vista de múltiples tablas para que sea actualizable, aunque cree la vista mediante programación. El ejemplo de código siguiente crea y establece explícitamente propiedades para actualizar una vista de dos tablas. Puede usar este ejemplo como guía para personalizar las propiedades de actualización de una vista.

### Actualización de múltiples tablas en una vista

#### Código

---

```
CREATE SQL VIEW emp_cust_view AS ;
  SELECT employee.emp_id, ;
  employee.phone, customer.cust_id, ;
  customer.emp_id, customer.contact, ;
  customer.company ;
FROM employee, customer ;
WHERE employee.emp_id = customer.emp_id
```

---

```
DBSETPROP('emp_cust_view', 'View', 'Tables',
'employee, customer')
```

---

```
DBSETPROP('emp_cust_view.emp_id', 'Field', ;
DBSETPROP('emp_cust_view.phone', 'Field', ;
'UpdateName', 'employee'
'UpdateName', 'customer')
```

---

```

DBSETPROP('emp_cust_view.phone', 'Field', ;
DBSETPROP('emp_cust_view.cust_id', 'Field', ;
DBSETPROP('emp_cust_view.emp_id1', 'Field', ;
DBSETPROP('emp_cust_view.contact', 'Field', ;
DBSETPROP('emp_cust_view.company', 'Field', ;
'UpdateName', 'employee.
'UpdateName', 'customer.
'UpdateName', 'customer.
'UpdateName', 'customer.
'UpdateName', 'customer.

DBSETPROP('emp_cust_view.emp_id', 'Field', ;
'KeyField', .T.)

DBSETPROP('emp_cust_view.cust_id', 'Field', ;
'KeyField', .T.)
DBSETPROP('emp_cust_view.emp_id1', 'Field', ;
'KeyField', .T.)

DBSETPROP('emp_cust_view.phone', 'Field', ;
'UpdatableField', .T.)
DBSETPROP('emp_cust_view.contact', 'Field', ;
'UpdatableField', .T.)
DBSETPROP('emp_cust_view.company', 'Field', ;
'UpdatableField', .T.)

DBSETPROP('emp_cust_view', 'View', ;
'SendUpdates', .T.)

GO TOP
REPLACE employee.phone WITH "(206)111-2222"
REPLACE customer.contact WITH "John Doe"

TABLEUPDATE( )

```

## Personalizar vistas con el diccionario de datos

Como las vistas están almacenadas en una base de datos, puede crear:

- [Títulos](#)
- Comentarios para la vista y los campos de vista
- [Valores predeterminados](#) para campos de vista
- [Reglas](#) a nivel de campo y a nivel de fila para mensajes de error de regla

Las características del diccionario de datos para vistas son similares en su funcionamiento a sus homólogos para tablas de base de datos. Sin embargo, se usa el lenguaje en lugar del Diseñador de tablas para crear títulos, comentarios, valores predeterminados y reglas para vistas.

## Crear valores predeterminados para campos de vista

De la misma manera que los valores predeterminados para campos de tabla, los [valores predeterminados](#) de campo de vista se almacenan en la base de datos y están disponibles cada vez que usa la vista. Visual FoxPro no compara los valores predeterminados que cree localmente con ningún

valor predeterminado establecido en el [origen](#) de datos remoto. Debe crear valores predeterminados aceptables para el origen de datos.

### Para asignar un valor predeterminado a un campo predeterminado

- En la ficha [Campos](#) del **Diseñador de vistas**, seleccione un campo y, a continuación, elija **Propiedades** y escriba el valor predeterminado para el campo.

–O bien–

- Use la propiedad DefaultValue de la función [DBSETPROP\(\)](#).

Por ejemplo, es posible que su aplicación limite la cantidad de mercancías que un cliente nuevo puede encargar hasta que haya tenido tiempo de terminar un cheque de crédito y de determinar la cantidad de crédito que quiere conceder al cliente. El siguiente ejemplo crea un campo `maxordamt` con un valor predeterminado de 1000:

```
OPEN DATABASE testdata
USE VIEW customer_view
?DBSETPROP ('Customer_view.maxordamt', 'Field', 'DefaultValue', 1000)
```

También puede usar [valores predeterminados](#) para llenar automáticamente algunas filas para los usuarios. Por ejemplo, puede agregar un [control Grid](#) para un formulario de entrada de pedidos basado en una vista remota de una tabla de elementos de pedido. El campo `order_id` es el campo clave que asigna cada fila de la cuadrícula a su homóloga de la tabla de elementos de pedido remota. Como el Id. de pedido para cada fila de la cuadrícula será el mismo para un pedido, puede usar un valor predeterminado para guardar las pulsaciones de teclas y llenar el campo `order_id` automáticamente.

**Sugerencia** Si una de las reglas comerciales de su aplicación requiere que un campo contenga una entrada, el hecho de proporcionar un valor predeterminado ayuda a asegurar que no se va a infringir una regla concreta a [nivel de campo](#) o a [nivel de registro](#).

### Crear reglas en campos y filas de vista

Puede crear versiones locales de reglas de orígenes de datos remotos para:

- Reducir el tiempo de respuesta.
- Reducir el impacto en recursos de red.
- Probar datos antes de enviarlos al origen de datos remotos.
- Evitar el envío de datos defectuosos al origen de datos remotos.

Visual FoxPro no compara las reglas creadas localmente con reglas remotas. Debe crear [reglas](#) aceptables para el [origen de datos](#). Si las reglas remotas cambian, debe cambiar las reglas locales para que se cumplan.

### Para crear una regla en un campo o una fila de vista

- En la ficha [Campos](#) del **Diseñador de vistas**, seleccione un campo y, a continuación, elija

**Propiedades** y escriba la expresión de regla y el texto de mensaje para el campo.

–O bien–

- Use las propiedades RuleExpression y RuleText de la función [DBSETPROP\(\)](#).

Por ejemplo, el código siguiente crea una regla a nivel de campo en `orditems_view` que evita la entrada de una cantidad inferior a 1:

```
OPEN DATABASE testdata
USE VIEW orditems_view
DBSETPROP('Orditems_view.quantity','Field', ;
          'RuleExpression', 'quantity >= 1')
DBSETPROP('Orditems_view.quantity','Field', ;
          'RuleText', ;
          '"Las cantidades deben ser mayores o iguales que 1"')
```

También puede usar la función [DBSETPROP\(\)](#) para crear reglas a nivel de fila.

## Combinar vistas

Puede generar una vista basada en otras vistas. Puede que quiera hacerlo si necesita un subconjunto de la información disponible en varias vistas diferentes o si desea combinar datos locales y remotos en una única vista. Una vista basada en otras vistas, o en una combinación de tablas locales y vistas locales o remotas, se denomina una [vista multicapa](#). La vista que combina otras vistas es la vista de nivel superior. Puede tener múltiples niveles de vistas entre la vista de nivel superior y las tablas base locales o remotas. Cuando utilice una vista multicapa, las vistas en las que se basa la vista de nivel superior y cualquier tabla base de Visual FoxPro usada en vistas de nivel superior o intermedio se mostrarán en la [ventana Sesión de datos](#). Las tablas remotas no aparecen en la ventana Sesión de datos.

### Combinar datos locales y remotos en una vista

Puede combinar datos locales y remotos en una vista si crea una nueva vista local basada en una vista local y una vista remota.

#### Para crear una vista que combina datos locales y remotos

- En el [Administrador de proyectos](#), seleccione una base de datos, elija **Vistas locales** y, a continuación, elija **Nuevo** para abrir el [Diseñador de vistas](#). Agregue cualquier combinación de tablas, vistas locales y vistas remotas a su vista.

–O bien–

- Use el comando [CREATE SQL VIEW](#).

Por ejemplo, para crear una vista local que combine información de la tabla local `Employee` y la tabla remota `Orders`, puede utilizar el código siguiente:

```
OPEN DATABASE testdata
CREATE SQL VIEW remote_orders_view ;
```

```
CONNECTION remote_01 ;
AS SELECT * FROM orders
CREATE SQL VIEW local_employee_remote_orders_view ;
AS SELECT * FROM testdata!local_employee_view, ;
testdata!remote_orders_view ;
WHERE local_employee_view.emp_id = ;
remote_orders_view.emp_id
```

## Actualizar datos locales y remotos en una vista

Cuando actualice datos en una [vista multicapa](#), las actualizaciones bajarán un nivel, hasta la vista en que está basada la vista de nivel superior. Si desea actualizar las [tablas base](#) desde las que se construye una vista multicapa, deberá ejecutar un comando [TABLEUPDATE](#) para cada vista de la estructura.

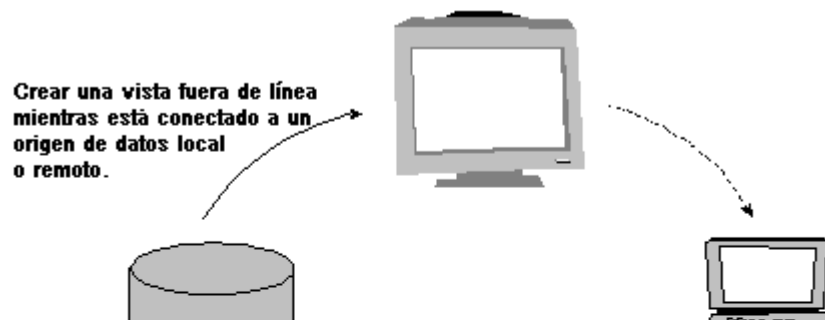
## Trabajar con datos fuera de línea

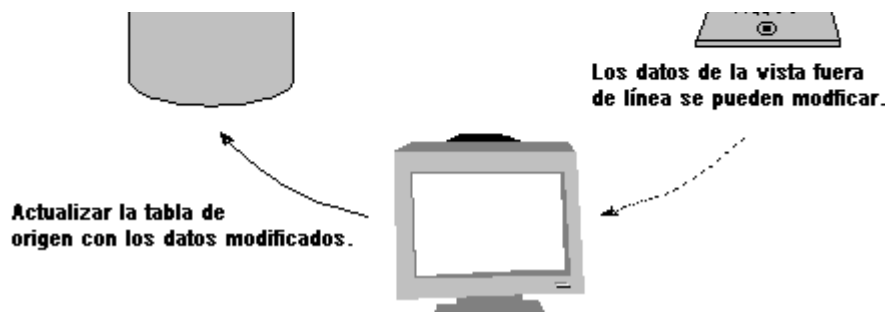
Hay veces en las que quiere mostrar, reunir o modificar datos independientemente de la base de datos host. Con las características de la vista fuera de línea de Visual FoxPro, puede usar vistas para conectarse a una base de datos host y crear un subconjunto de datos para trabajar fuera de línea. Entonces, al trabajar fuera de línea podrá usar la vista directamente o mediante una aplicación que cree. Cuando haya terminado, puede cargar las modificaciones almacenadas en la vista a la base de datos host.

Algunos escenarios en los que son útiles las vistas fuera de línea son:

- Una situación de almacenamiento de datos, en la que se mantienen grandes bases de datos centralizadas en servidores MIS. Si sólo está interesado en datos pertenecientes a, por ejemplo, el departamento de Marketing, puede crear una vista que incluya sólo los datos que le importen. Entonces puede usar los datos fuera de línea, permitir que varios usuarios del departamento de Marketing actualicen los datos y después devolver los datos modificados a la base de datos de origen.
- Una ubicación geográfica remota que requiere que lleve un subconjunto de datos en su portátil, modificar los datos de forma independiente en la base de datos host y después actualizar la base de datos con los datos modificados.
- Datos sensibles al tiempo. Por ejemplo, puede querer actualizar datos que reflejen los aumentos de sueldo de los empleados antes de que se apliquen los nuevos sueldos.

## Trabajar con vistas fuera de línea





Para crear y usar datos de vista fuera de línea, puede usar las siguientes características del lenguaje:

- La función [CREATEOFFLINE\(\)](#)
- El comando [USE SQLNombreVista](#) con las cláusulas ADMIN y ONLINE
- El comando [TABLEUPDATE](#)
- La función [DROPOFFLINE\(\)](#)

Si pretende usar la vista fuera de línea en un equipo distinto del utilizado para crear la vista fuera de línea, debe preparar el destino fuera de línea creando una copia del archivo de base de datos host (.dbc); asegúrese de que el origen de datos ODBC usado por la vista existe en el equipo de destino; y analice los requisitos de datos para determinar el contenido de la vista que necesita.

**Nota** Use el programa Administrador ODBC para instalar orígenes de datos en un equipo. Puede tener acceso al programa Administrador ODBC desde el grupo de programas Visual FoxPro o desde el Panel de control.

## Crear vistas fuera de línea

Como con los datos en línea, analice sus requisitos antes de crear vistas fuera de línea para determinar el diseño de vistas que necesita en la base de datos fuera de línea. Cuando determine el subconjunto de datos que desea usar fuera de línea, puede partir de una vista existente o crear una vista nueva. Si ya existe una vista que devuelve los registros que quiere usar fuera de línea, puede usarla, o puede crear una vista mediante programación. La vista que usa fuera de línea se almacena en un archivo .dbf en el contenedor de base de datos.

**Nota** Si pretende modificar datos en una vista fuera de línea, asegúrese de hacer la vista actualizable antes de usarla fuera de línea. Cuando una vista está fuera de línea sólo puede establecer sus propiedades de actualización a través de programación; no puede modificar una vista fuera de línea en el [Diseñador de vistas](#).

### Para usar una vista existente fuera de línea

- Use la función [CREATEOFFLINE\(\)](#) y el nombre de la vista.

Por ejemplo, si quiere ir a sitios cliente para actualizar cuentas, agregar clientes y registrar nuevas ventas, necesita la información de los clientes así como los pedidos actuales y las descripciones de productos en pantalla. Puede tener una vista llamada `customerinfo` que combine información de la tabla Customers, la tabla Orders y la tabla OrderItems. Para crear la vista, use este código:

```
CREATEOFFLINE("customerinfo")
```

## Para crear una vista fuera de línea mediante programación

- Use el comando [CREATE SQL VIEW](#), seguido por el comando [CREATEOFFLINE\(\)](#).

Por ejemplo, el código siguiente crea una vista que muestra datos de la tabla `Products` y la tabla `Inventory` desde la base de datos en línea. Como no se especifica ningún criterio de actualización, esta vista es de sólo lectura.

```
CREATE SQL VIEW showproducts ;  
    CONNECTION dsources ;  
    AS SELECT * FROM Products INNER JOIN Inventory ;  
    ON Products.ProductID = Inventory.ProductID ;  
CREATEOFFLINE("showproducts")
```

## Mostrar y modificar datos fuera de línea

Después de crear la vista para sus datos fuera de línea, puede usarla como cualquier vista de su aplicación: puede agregar, cambiar y eliminar registros. Múltiples usuarios pueden tener acceso a la vista fuera de línea simultáneamente usando la misma base de datos en modo compartido. Si decide no conservar las modificaciones, puede revertir la información para que refleje la información original.

### Usar datos fuera de línea

Si usa los datos fuera de línea, puede mostrar y actualizar datos de la misma forma que lo hace en línea con los mismos formularios, informes o aplicaciones. Por ejemplo, el código siguiente abre la vista `Showproducts`:

```
USE Showproducts
```

**Sugerencia** Si no obtiene el subconjunto de datos que esperaba, compruebe la configuración de optimización para la vista remota. Si establece la propiedad `MaxRecords` mediante la función [DBSETPROP\(\)](#), sólo aparece ese número de registros en las vistas fuera de línea. Sin embargo, si incluye un campo `Memo` en la lista de campos de la vista, se incluye automáticamente en el conjunto



inserta un campo memo en la lista de campos de la vista, se inserta automáticamente en el conjunto de resultados incluso si FetchMemo está establecido como falso (.F.).

## Administrar datos fuera de línea

En algunos casos (especialmente en un entorno de múltiples usuarios en el que muchas personas modifican los datos) puede querer examinar las modificaciones realizadas a la vista fuera de línea antes de introducir las modificaciones en la base de datos de origen. Con el comando [USE](#) y la cláusula ADMIN, puede ver todas las modificaciones que se han introducido en una vista desde que se usó fuera de línea. Entonces puede revertir de forma selectiva las modificaciones que se hayan realizado sin estar conectado al origen de datos. Por ejemplo, el código siguiente abre la vista Showproducts en modo administrador:

```
USE Showproducts ADMIN
```

## Actualizar datos en línea

Después de terminar de trabajar fuera de línea, puede actualizar los datos en el servidor usando las mismas transacciones de actualización de tabla que utiliza normalmente con los datos en línea. Al trabajar con datos remotos, recuerde las sugerencias siguientes:

- Para actualizaciones de registros individuales, use transacciones automáticas.
- Para actualizaciones de proceso por lotes, use transacciones manuales.
- Si es necesario, incluya código para detectar conflictos, cree un registro de conflictos y resuélvalos.

Antes de poder procesar sus actualizaciones, tiene que usar el comando [USE](#) y la palabra clave ONLINE para volver a conectarse a la base de datos host. Después de ejecutar el comando, Visual FoxPro intenta localizar la base de datos host con la información de origen de datos almacenada en la vista. Cuando la conexión está establecida, puede usar [TABLEUPDATE\(\)](#) para procesar las actualizaciones almacenadas en los datos fuera de línea.

Para asegurarse de que la información de conexión es correcta independientemente de la ubicación del host y las tablas de vista, tiene que usar sintaxis de cadena de conexión en lugar de una conexión con nombre.

## Actualizar lotes de registros en tablas locales

Para procesar un lote de cambios en tablas locales, puede usar transacciones manuales que le permiten procesar el lote de cambios en una única transacción en lugar de una serie de transacciones independientes.

## Actualización de tablas locales con vistas fuera de línea

Código	Comentarios
USE myofflineview ONLINE EXCLUSIVE	Volver a abrir la vista

```

BEGIN TRANSACTION
IF TABLEUPDATE (2, .F., "myofflineview")
    END TRANSACTION
ELSE
    MESSAGEBOX("Ha ocurrido un error: La actualización no ha tenido éxito")
    ROLLBACK
ENDIF

```

Compro  
actualiz  
apropia

## Actualizar lotes de registros en tablas remotas

Para procesar un lote de cambios en tablas remotas, use transacciones manuales: empiece con [TABLEUPDATE\(\)](#) y termine el procesamiento con [SQLCOMMIT\(\)](#) o [SQLROLLBACK\(\)](#).

Para establecer la conexión para administrar sus transacciones manualmente, tiene que usar [CURSORGETPROP\(\)](#) en el cursor de la vista para obtener el controlador de conexión y, a continuación, establecer la propiedad Transactions a modo manual.

En el código siguiente, la identificación de conexión actual para la vista, myview, se almacena en hConn1. hConn1 se usa para establecer la propiedad Transactions a "2" para transacciones manuales.

```

hConn1 = CURSORGETPROP("CONNECTHANDLE", "myview") ;
SQLSETPROP(hConn1, "TRANSACTIONS", 2)

```

Después de establecer la conexión para controlar las actualizaciones, puede usar [TABLEUPDATE\(\)](#) para controlar sus transacciones.

Si las tablas host residen en un servidor remoto, como SQL Server, puede usar el siguiente código como directiva.

## Actualización de tablas remotas con vistas fuera de línea

Código	Comentari
USE myofflineview ONLINE EXCLUSIVE	Volver a co abrir la vist
SQLSetProp(liviewhandle, "transacciones", 2) SQLSetProp(custviewhandle, "transacciones", 2) SQLSetProp(ordviewhandle, "transacciones", 2)	Establecer l las vistas pa manualmen
IF NOT TABLEUPDATE(.T., .F., "lineitemsview") =SQLROLLBACK(ordviewhandle) =MESSAGEBOX("No se puede actualizar la tabla de elementos de línea") IF NOT TableUpdate(.T., .F., "ordersview") =SQLROLLBACK(liviewhandle) =MESSAGEBOX("No se puede actualizar la tabla de pedidos") IF NOT TABLEUPDATE(.T., .F., "customerview") =SQLROLLBACK(custviewhandle) =MESSAGEBOX("No se puede actualizar la tabla de clientes") Else *# comprobar escenarios de error IF NOT SQLCOMMIT(liviewhandle)	Controlar a conflictos d

```

        =SQLROLLBACK(liviewhandle)
    IF NOT SQLCOMMIT(ordviewhandle)
        =SQLROLLBACK(ordviewhandle)
    IF NOT SQLCOMMIT(custviewhandle)
        =SQLROLLBACK(custviewhandle)
    ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

```

## Actualizar un registro

Si va a actualizar una única fila, puede usar transacciones automáticas. Como cada instrucción para procesar una actualización, eliminación o inserción se controla como una transacción independiente, no puede deshacer instrucciones de transacciones anteriores.

```

USE customerview ONLINE EXCLUSIVE
GO TO 3
    IF TABLEUPDATE (0, .F. workarea)
        * conflicto controlando el código
    ENDIF

```

**Sugerencia** Para actualizar un único registro en una tabla local, use la función [GETNEXTMODIFIED\(\)](#).

## Cancelar actualizaciones fuera de línea

Si decide que quiere eliminar los datos fuera de línea y convertir la vista en una vista en línea, puede usar la función [DROPOFFLINE\(\)](#).

### Para cancelar actualizaciones fuera de línea

- Use [DROPOFFLINE\(\)](#) con el nombre de la vista.

Asegúrese de comprobar los valores devueltos. Verdadero (.T.) indica que se ha conseguido y falso (.F.) indica que la vista no se ha cerrado antes de ejecutar el comando.

El código siguiente pasa por alto todas las modificaciones realizadas al subconjunto de datos de myview. La vista sigue formando parte de la base de datos, pero se ignora el conjunto de datos actual.

```
DROPOFFLINE("myview")
```

Puede eliminar registros fuera de línea, pero no puede usar los comandos [PACK](#), [ZAP](#) o [INSERT](#) con una vista fuera de línea.

## Optimizar el rendimiento de una vista

Puede optimizar el rendimiento de sus vistas estableciendo sus propiedades.

## Controlar el tamaño de búsqueda de la búsqueda progresiva

Puede controlar el número de filas que Visual FoxPro busca progresivamente en el origen de datos remoto mediante la propiedad `FetchSize` de la vista y del cursor activo. Utilice [DBSETPROP\(\)](#) y [CURSORSETPROP\(\)](#) para establecer estas propiedades.

## Controlar la búsqueda memo

Puede utilizar la característica de búsqueda diferida de memo de Visual FoxPro para acelerar la recuperación de los datos de la vista. Cuando elija esta característica, Visual FoxPro no recuperará el contenido de un campo Memo hasta que usted no abra y muestre el campo. Como Visual FoxPro necesita el campo clave y el nombre de la tabla para encontrar una fila en el origen de datos remoto, debe establecer la propiedad `UpdateName` o `UpdateFieldList`, la propiedad `KeyField` o `KeyFieldList` y la propiedad `Tables` para que la búsqueda diferida de memo funcione. No obstante, no tiene que establecer las propiedades `SendUpdates` o `Updatable` para que la búsqueda memo funcione.

## Establecer el número máximo de registros que se descargarán

Puede controlar la cantidad de datos que se descargan al abrir una vista si establece la propiedad `MaxRecords`. Cuando Visual FoxPro envía una instrucción SQL al origen de datos para crear una vista, el origen de datos genera y almacena un conjunto resultante. La propiedad `MaxRecords` especifica el número máximo de filas que se buscan del conjunto resultante remoto para la vista. El valor predeterminado es -1, que descarga todas las filas en el conjunto resultante.

### Para controlar el número de filas que se descargan en una vista

- Elija **Opciones** en el menú **Herramientas** y seleccione la ficha **Datos remotos**. En el área **Opciones predeterminadas de vista remota**, junto a **Máximo de registros para buscar**, desactive **Todos**, introduzca un valor en el cuadro de texto y elija **Aceptar**.

-O bien-

- Use la propiedad `MaxRecords` de las funciones [DBSETPROP\(\)](#) o [CURSORSETPROP\(\)](#).

Por ejemplo, el código siguiente modifica la definición de la vista para limitar a 50 el número de filas descargadas en la vista, independientemente del tamaño del conjunto resultante generado en el origen de datos remoto:

```
OPEN DATABASE testdata
USE VIEW remote_customer_view
?DBSETPROP ('Remote_customer_view', ;      'View','MaxRecords', 50)
```

Puede utilizar la función [CURSORSETPROP\(\)](#) para establecer el límite de `MaxRecords` para una vista activa.

**Sugerencia** No puede usar la propiedad `MaxRecords` para detener una consulta en ejecución, ya que la propiedad `MaxRecords` no controla la creación del conjunto resultante. Use la propiedad `QueryTimeout` para controlar el tiempo de ejecución en el origen de datos remoto.

## Optimizar filtros y combinaciones

Para tomar decisiones de optimización para una vista o una consulta, es posible que tenga que conocer el [plan de ejecución](#): el orden en que se evalúan las cláusulas de [combinaciones](#) y [filtros](#). Con la función [SYS\(3054\)](#), puede mostrar uno de los tres niveles de optimización [Rushmore](#). Los tres niveles indican el grado en que las condiciones de filtro o las condiciones de combinación pudieron utilizar la optimización Rushmore. Los niveles son completamente (Full), parcialmente (Partial) o nada (None).

### Para mostrar el plan de ejecución para filtros

1. En la ventana [Comandos](#), escriba **SYS(3054,1)** para activar SQL ShowPlan.
2. Escriba la instrucción SQL SELECT.

Por ejemplo, puede escribir:

```
SELECT * FROM customer, orders ;  
AND Upper(country) = "Méjico"
```

3. Lea el plan de ejecución que aparece en pantalla.

Para este ejemplo, la pantalla podría tener esta apariencia:

```
Using Index Tag Country to optimize table customer  
Rushmore Optimization Level for table customer: Full  
Rushmore Optimization level for table orders: none
```

4. En la ventana **Comandos**, escriba **SYS(3054,0)** para desactivar SQL ShowPlan.

Entonces puede pasar 11 a la función SYS para evaluar combinaciones en las cláusulas FROM o WHERE.

### Para mostrar el plan de ejecución para combinaciones

1. En la ventana [Comandos](#), escriba **SYS(3054,11)** para activar SQL ShowPlan.
2. Escriba su instrucción SQL SELECT.

Por ejemplo, puede escribir:

```
SELECT * ;  
FROM customer INNER JOIN orders ;  
ON customer.cust_id = orders.cust_id ;  
WHERE Upper(country) = "Méjico"
```

3. Lea el plan de ejecución que aparece en pantalla.

Para este ejemplo, la pantalla puede tener la siguiente apariencia:

```
Using Index Tag Country to optimize table customer
Rushmore Optimization Level for table customer: Full
Rushmore Optimization level for table orders: none
Joining table customer and table orders using Cust_id
```

4. En la ventana **Comandos**, escriba **SYS(3054,0)** para desactivar SQL ShowPlan.

## Controlar la evaluación de combinaciones

Si el [plan de ejecución](#) para sus [combinaciones](#) no satisface sus necesidades específicas, puede hacer que el orden de combinación se ejecute exactamente como está escrito sin optimización del procesador. Para forzar el orden de evaluación de la combinación, debe agregar la palabra clave **FORCE** y colocar sus condiciones de combinación en la cláusula **FROM**. Las condiciones de combinación que se colocan dentro de una cláusula **WHERE** no se incluyen en una evaluación de combinación forzada.

**Nota** No puede usar la palabra clave **FORCE** en instrucciones de paso a través **SQL** ni en vistas remotas porque la palabra clave es una extensión de Visual FoxPro del estándar [ANSI](#) y no está admitida en otros diccionarios **SQL**.

La cláusula **FORCE** es global, por lo que se aplica a todas las tablas de la cláusula **JOIN**. Asegúrese de que el orden en el que aparecen las tablas de combinación es exactamente el orden en el que se deben combinar. También puede usar los paréntesis para controlar el orden de evaluación de combinaciones.

En este ejemplo, la primera combinación especificada también es la primera combinación evaluada. La tabla **Customer** se combina con la tabla **Orders** en primer lugar. El resultado de la combinación se combina entonces con la tabla **OrdItems**:

```
SELECT * ;
FROM FORCE Customers ;
INNER JOIN Orders ;
    ON Orders.Company_ID = Customers.Company_ID ;
INNER JOIN OrItems;
    ON OrItems.Order_NO = Orders.Order_NO
```

En el ejemplo, la combinación entre paréntesis para la tabla **Orders** y **OrdItems** se evalúa primero. El resultado de la combinación se utiliza en la evaluación de la combinación con **Customers**:

```
SELECT * ;
FROM FORCE Customers ;
INNER JOIN (orders INNER JOIN OrItems ;
    ON OrItems.Order_No = Orders.Order_No) ;
    ON Orders.Company_ID = Customers.Company_ID
```

## Conexiones compartidas para múltiples vistas remotas

Puede utilizar una conexión activa como canal de información para múltiples vistas remotas si comparte una [conexión](#). Cuando comparte una conexión activa, usted:

- Reduce el número de conexiones de un servidor remoto.
- Reduce los costes por conexiones a servidores que se cobran por conexión.

Para compartir conexiones establece la definición de la vista para utilizar una conexión compartida al activarse. Cuando se utiliza la vista, Visual FoxPro se conecta al origen de datos remoto con la conexión compartida existente (en caso de que haya alguna). Si no hay ninguna conexión compartida en uso, Visual FoxPro creará una conexión exclusiva cuando se abra la vista, que puede compartirse con otras vistas.

Sólo una instancia activa de una definición de conexión con nombre se comparte durante una sesión de Visual FoxPro. Si hay varias instancias activas de la misma definición de conexión, la primera de ellas que se use como conexión compartida se convertirá en la conexión compartida designada. Todas las vistas que utilicen esa definición de conexión y empleen conexiones compartidas tendrán acceso al servidor remoto a través de la conexión compartida designada.

No se compartirán otras conexiones distintas de la conexión compartida designada. El uso compartido de conexiones no está ligado a las [sesiones](#).

### Para compartir una conexión

- Elija **Opciones** en el menú **Herramientas**. Elija la ficha **Datos remotos**; active la casilla de verificación **Compartir conexión** en el área **Opciones predeterminadas de vista remota** y elija **Aceptar**.

–O bien–

- Use el [Diseñador de vistas](#).

–O bien–

- Use el comando [CREATE SQL VIEW](#) con la cláusula **SHARE**.

El código siguiente crea una vista que, cuando se activa con el comando [USE](#), comparte una conexión:

```
CREATE SQL VIEW product_view_remote ;  
    CONNECTION remote_01 SHARE AS ;  
    SELECT * FROM products  
USE product_view_remote
```

### Comprobar si una conexión está ocupada

Cuando una conexión está ocupada, como ocurre cuando Visual FoxPro busca datos de forma progresiva en un [cursor](#), no querrá comenzar otra búsqueda o enviar actualizaciones por la misma conexión. Puede determinar si una conexión está ocupada mediante la propiedad **ConnectBusy**, que devuelve un valor verdadero (.T.) si la conexión está ocupada. Puede utilizar esta propiedad en su aplicación para comprobar si la conexión está ocupada o no antes de enviar una solicitud a un origen de datos remoto a través de una conexión compartida.

### Para determinar si una conexión está ocupada

Visual FoxPro proporciona la propiedad [ConnectBusy](#) para determinar si una conexión está ocupada.

- Use la propiedad ConnectBusy de la función [SQLGETPROP\(\)](#).

Necesita el controlador de la conexión para utilizar la función [SQLGETPROP\(\)](#). Puede identificar el controlador de la conexión para una vista activa mediante la propiedad ConnectHandle de la función [CURSORGETPROP\(\)](#). El código siguiente identifica un controlador de conexión y lo utiliza para comprobar si la conexión está ocupada:

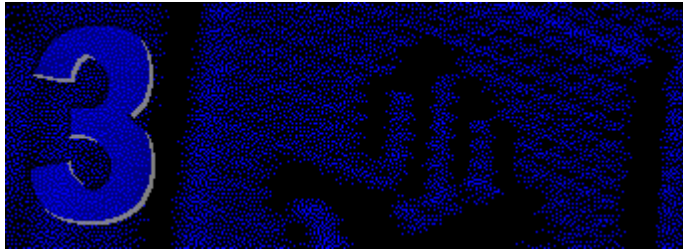
```
nConnectionHandle=CURSORGETPROP('ConnectHandle')
SQLGETPROP(nConnectionHandle, "ConnectBusy")
```







# Manual del programador, Parte 3: Crear la interfaz



Una interfaz bien diseñada puede guiar a los usuarios por su aplicación. Los formularios, clases, controles, menús y barras de herramientas ofrecen un conjunto variado de herramientas para diseñar una excelente interfaz de usuario.

## Capítulo 9 [Crear formularios](#)

Su aplicación necesita formularios para permitir a los usuarios ver e introducir datos. Pero también puede personalizar formularios estándar visualmente y mediante programación para crear un entorno especializado para sus usuarios.

## Capítulo 11 [Usar controles](#)

Los controles administran las interacciones entre los usuarios y su aplicación. Visual FoxPro ofrece diversos controles para mejorar la interfaz de su aplicación.

## Capítulo 12 [Diseñar menús y barras de herramientas](#)

Un buen sistema de menús dice mucho a los usuarios acerca del diseño y la estructura de su aplicación. Si diseña detenidamente los menús puede mejorar la facilidad de uso de su aplicación, proporcionar acceso inmediato a tareas comunes y agregar una apariencia de tipo Windows a su aplicación.

# Capítulo 9: Crear formularios

Los formularios no sólo sirven para ofrecer a los usuarios una interfaz familiar para ver e introducir datos en una base de datos, sino que también ofrecen un amplio conjunto de [objetos](#) que pueden responder a los [eventos](#) del usuario (o del sistema) permitiéndoles realizar las tareas de administración de información de la forma más sencilla e intuitiva posible.

En este capítulo se tratan los temas siguientes:

- [Diseñar formularios](#)
- [Crear un formulario nuevo](#)
- [Agregar objetos a formularios](#)

- [Manipular objetos](#)
- [Administrar formularios](#)

## Diseñar formularios

Visual FoxPro proporciona un eficaz Diseñador de formularios para que el diseño de formularios resulte rápido y sencillo. Puede disponer de:

- Diversos tipos de objetos en los formularios.
- Datos dependientes del formulario.
- Formularios de nivel superior o formularios secundarios.
- Múltiples formularios que pueden manipularse conjuntamente.
- Formularios basados en sus propias plantillas personalizadas.

Los formularios y los conjuntos de formularios son objetos con sus propias [propiedades](#), [eventos](#) y [métodos](#) que pueden establecerse en el Diseñador de formularios. Un conjunto de formularios consta de uno o más formularios que pueden manipularse como una unidad. Por ejemplo, si tiene cuatro formularios en su conjunto de formularios, podrá mostrarlos u ocultarlos como uno solo mediante un único comando en tiempo de ejecución.

## Crear un formulario nuevo

Puede crear formularios nuevos en el [Diseñador de formularios](#), y ver a medida que lo diseña cómo verá el usuario cada objeto.

### Para crear un formulario nuevo

- En el [Administrador de proyectos](#), seleccione **Formularios** y elija **Nuevo**.

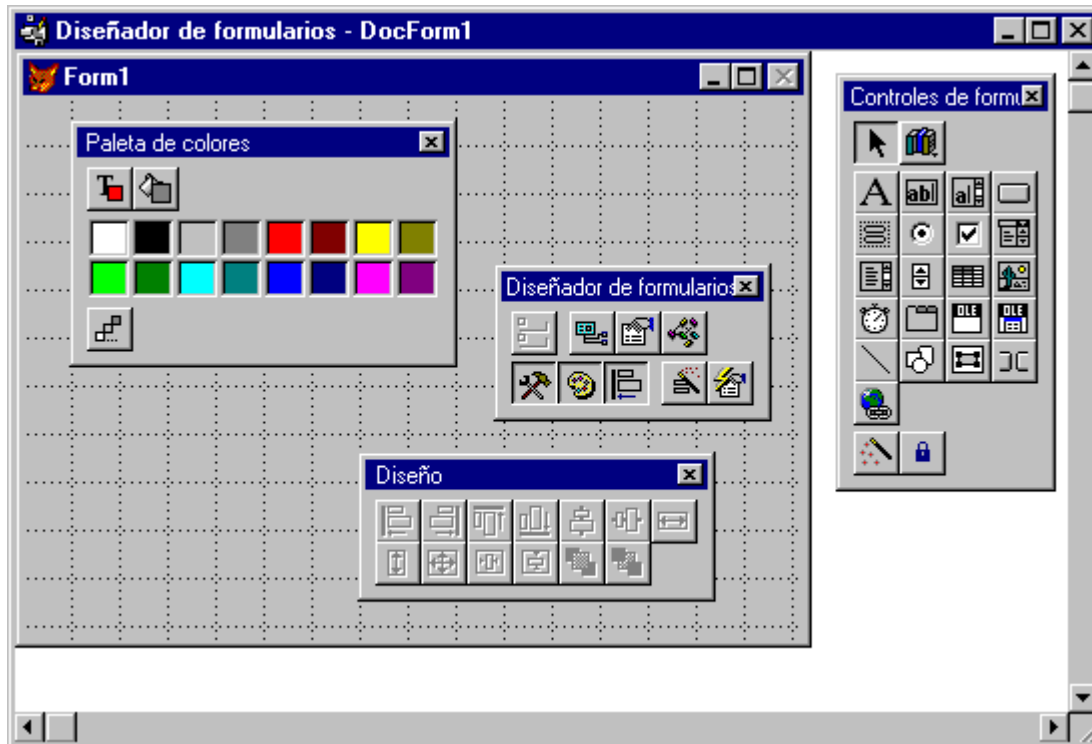
–O bien–

En el menú **Archivo**, elija **Nuevo**, seleccione **Formulario** y, a continuación, elija **Nuevo archivo**.

–O bien–

- Utilice el comando [CREATE FORM](#).

**El Diseñador de formularios con barras de herramientas:** [Diseñador de formularios](#), [Controles de formularios](#), [Diseño](#) y [Paleta](#)



Para obtener una descripción más detallada del Diseñador de formularios, consulte el capítulo 8, [Administrar datos mediante formularios](#), del *Manual del usuario*. Si desea más información sobre las barras de herramientas, busque el nombre de la barra de herramientas en la Ayuda.

## Establecer el entorno de datos

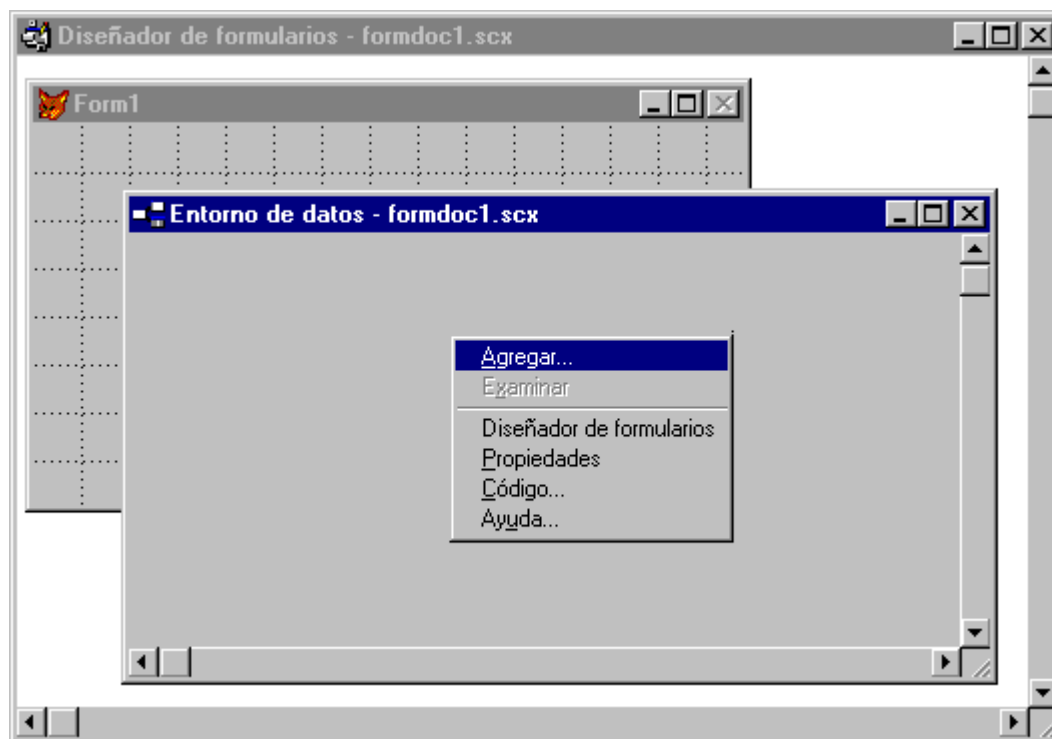
El [entorno de datos](#) de un formulario o un conjunto de formularios incluye las [tablas](#) o [vistas](#) con las que interactúa el formulario y las relaciones entre tablas que espera el formulario. Puede diseñar visualmente el entorno de datos en el Diseñador de entornos de datos y guardarlo con el formulario o con el conjunto de formularios.

El entorno de datos puede automatizar la apertura y el cierre de tablas y vistas cuando se ejecuta el formulario. Además, el entorno de datos le ayuda a establecer la propiedad [ControlSource](#) para controles al rellenar el cuadro del valor de la propiedad ControlSource de la [ventana Propiedades](#) con todos los campos del entorno de datos.

### Para abrir el Diseñador de entornos de datos

1. En el menú **Ver**, elija **Entorno de datos**.
2. En el cuadro **Agregar tabla o vista**, elija **Agregar**.
3. En el cuadro de diálogo **Abrir**, elija la tabla o vista que desea agregar al entorno de datos.

### El Diseñador de entornos de datos



### Propiedades habituales del entorno de datos

Las siguientes propiedades del entorno de datos suelen establecerse en la ventana Propiedades:

Propiedad	Descripción	Valor predeterminado
<a href="#">AutoCloseTables</a>	Controla si las tablas y las vistas se cierran cuando se libera el formulario o el conjunto de formularios.	Verdadero (.T.)
<a href="#">AutoOpenTables</a>	Controla si las tablas y las vistas del entorno de datos se abren cuando se ejecuta el formulario.	Verdadero (.T.)
<a href="#">InitialSelectedAlias</a>	La tabla o la vista que se selecciona cuando se ejecuta el formulario.	" " en tiempo de diseño. Si no se especifica, en tiempo de ejecución se seleccionará inicialmente el primer cursor agregado a DataEnvironment.

### Agregar una tabla o vista al Diseñador de entornos de datos

Cuando agregue [tablas](#) o [vistas](#) al Diseñador de entornos de datos, puede ver los [campos](#) y los [índices](#) que pertenecen a la tabla o a la vista.

### Para agregar una tabla o una vista al entorno de datos

1. En el [Diseñador de entornos de datos](#), elija **Agregar** en el menú [Entorno de datos](#).
2. En el cuadro de diálogo [Agregar tabla o vista](#), elija una tabla o una vista de la lista.

–O bien–

Si no hay ninguna base de datos ni ningún proyecto abierto, elija **Otros** para seleccionar una tabla.

También puede arrastrar una tabla o una vista desde un proyecto abierto hasta el [Diseñador de entornos de datos](#).

Cuando el Diseñador de entornos de datos está activo, la [ventana Propiedades](#) muestra [objetos](#) y [propiedades](#) asociadas al entorno de datos. Cada tabla o vista del entorno de datos, cada relación entre tablas y el mismo entorno de datos son objetos distintos en el cuadro Objeto de la ventana Propiedades.

### Eliminar una tabla del Diseñador de entornos de datos

Al quitar una tabla del entorno de datos también se quitan las [relaciones](#) en las que interviene la tabla.

### Para quitar una tabla o una vista del Diseñador de entornos de datos

1. En el [Diseñador de entornos de datos](#), seleccione la tabla o la vista.
2. En el menú **Entorno de datos**, elija **Quitar**.

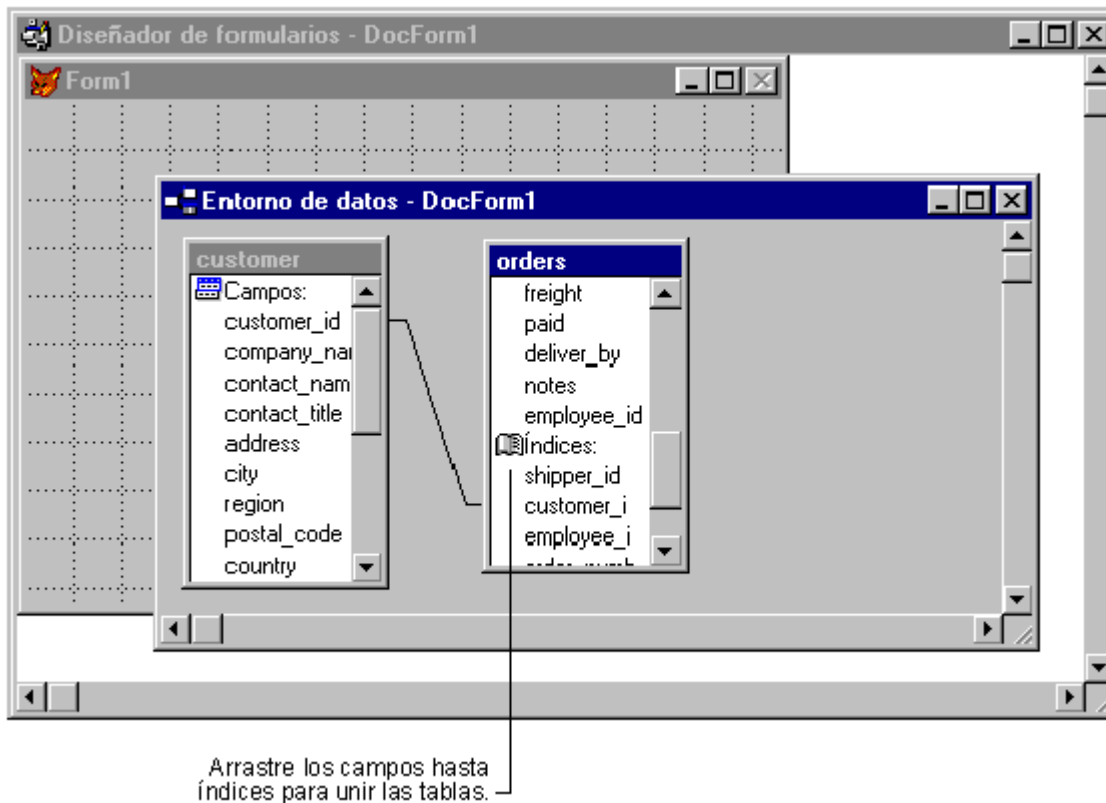
### Establecer relaciones en el Diseñador de entornos de datos

Si agrega al Diseñador de entornos de datos tablas que tienen [relaciones persistentes](#) establecidas en una base de datos, las relaciones se agregarán automáticamente al entorno de datos. Si las tablas no tienen relaciones persistentes, podrá relacionarlas en el Diseñador de entornos de datos.

### Para establecer relaciones en el Diseñador de entornos de datos

- Arrastre un campo desde la [tabla principal](#) hasta la etiqueta de índice correspondiente de la [tabla relacionada](#).

### El Diseñador de entornos de datos con relaciones establecidas entre tablas



También puede arrastrar un campo desde la [tabla principal](#) hasta un campo de la [tabla relacionada](#). Si no hay ninguna etiqueta de índice en la tabla relacionada correspondiente al campo de la tabla principal, se le pedirá que cree la etiqueta de índice.

### Modificar relaciones en el Diseñador de entornos de datos

Cuando establezca una [relación](#) en el Diseñador de entornos de datos, una línea entre las tablas indicará la relación.

### Para modificar las propiedades de la relación

- En la ventana [Propiedades](#), seleccione la relación en el cuadro **Objeto**.

Las propiedades de la relación corresponden a cláusulas y palabras clave de los comandos [SET RELATION](#) y [SET SKIP](#).

La propiedad [RelationalExpr](#) se establece de forma predeterminada con el nombre del campo [clave principal](#) de la tabla primaria. Si la tabla relacionada está indexada en una [expresión](#), deberá establecer la propiedad RelationalExpr con esta expresión. Por ejemplo, si la tabla relacionada está indexada en `UPPER(cust_id)`, deberá establecer RelationalExpr como `UPPER(cust_id)`.

Si la relación no es de [uno a varios](#), establezca la propiedad [OneToMany](#) como falsa (.F.). Esto equivale a utilizar el comando [SET RELATION](#) sin ejecutar [SET SKIP](#).

Establecer la propiedad OneToMany de una relación como verdadera (.T.) equivale a ejecutar el



comando SET SKIP. Si omite la [tabla primaria](#), el puntero de registro permanecerá en el mismo registro primario hasta pasar a través de todos los registros relacionados de la [tabla secundaria](#).

**Nota** Si desea crear una relación de uno a varios en el formulario o el conjunto de formularios, establezca como verdadera (.T.) la propiedad OneToMany, aunque se haya establecido una relación persistente de uno a varios en la base de datos.

## Crear interfaces de un único documento e interfaces de documentos múltiples

Visual FoxPro le permite crear dos tipos de aplicaciones:

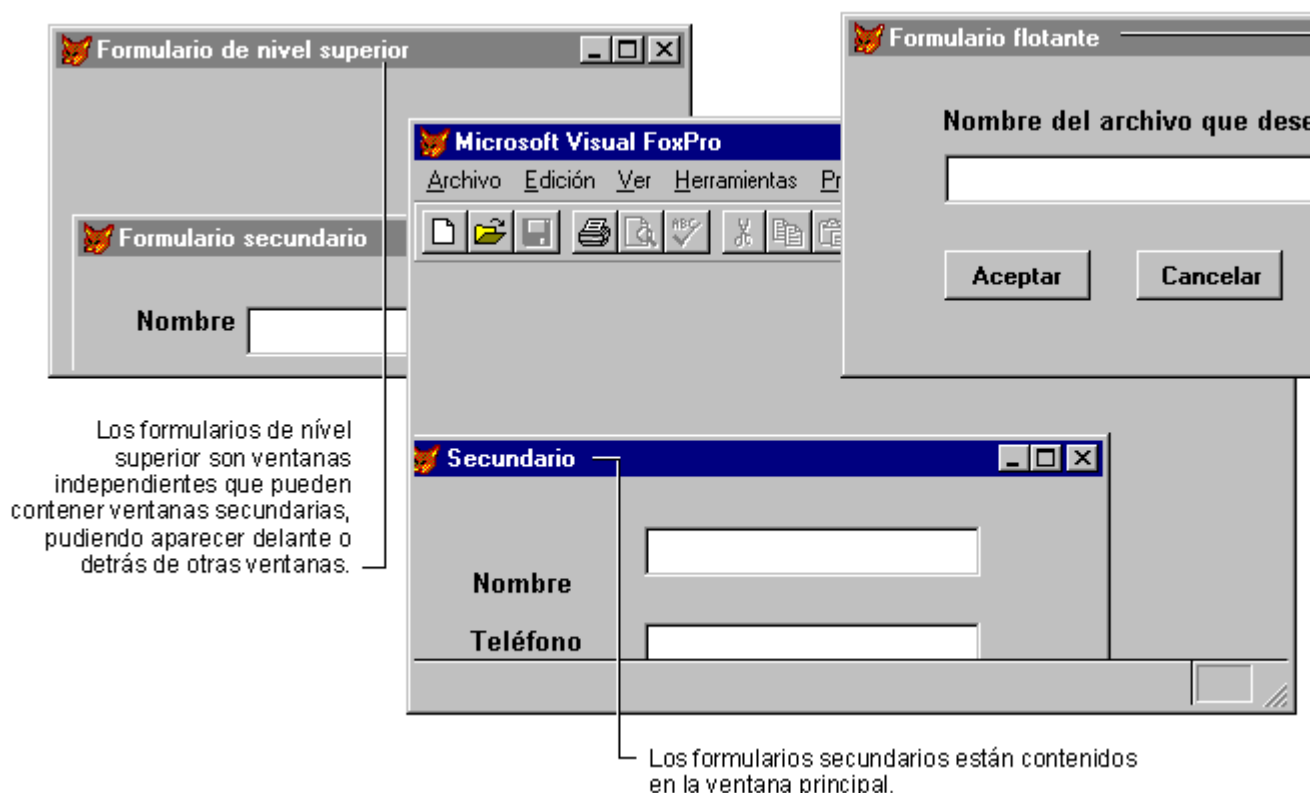
- Las aplicaciones con [interfaz de documentos múltiples\(Multiple-document interface, MDI\)](#) están formadas por una ventana principal única y las ventanas de la aplicación están contenidas en la ventana principal. Visual FoxPro es fundamentalmente una aplicación MDI, con la ventana Comandos, las ventanas de edición y las ventanas de diseñadores contenidas en la ventana principal de Visual FoxPro.
- Las aplicaciones con [interfaz de un único documento \(SDI\)](#) están formadas por una o más ventanas independientes, cada una de las cuales aparece como ventana independiente en el escritorio de Windows. Microsoft Exchange es un ejemplo de una aplicación SDI, en la que cada mensaje que abra aparece en su propia ventana independiente.

Una aplicación formada por ventanas sencillas es generalmente una aplicación SDI, pero algunas aplicaciones mezclan elementos SDI y MDI. Por ejemplo, Visual FoxPro muestra su depurador como aplicación SDI, que a su vez contiene ventanas MDI.

Para admitir ambos tipos de interfaz, Visual FoxPro le permite crear varios tipos de formularios:

- *Formulario secundario.* Un formulario contenido en otra ventana, utilizada en la creación de aplicaciones MDI. Los formularios secundarios no se pueden mover fuera de los límites de su formulario primario y cuando se minimizan aparecen en la parte inferior del formulario primario. Si el formulario primario está minimizado, también se minimizan los formularios secundarios.
- *Formulario flotante.* Formulario que pertenece a un formulario primario, pero no está contenido en él. Los formularios flotantes se pueden mover por toda la pantalla. No se pueden mover por detrás de su ventana primaria. Si están minimizados, aparece un formulario flotante en la parte inferior del escritorio. Si su formulario primario está minimizado, los formularios flotantes también se minimizan. Los formularios flotantes también se usan en la creación de aplicaciones MDI.
- *Formulario de nivel superior.* Un formulario independiente sin formulario primario, utilizado para crear una aplicación SDI o como formulario primario de otros formularios secundarios de una aplicación MDI. Los formularios de nivel superior funcionan al mismo nivel que otras aplicaciones de Windows y pueden aparecer por delante o por detrás de ellas. Aparecen en la barra de estado de Windows.

## Formularios secundarios, flotantes y de nivel superior



## Especificar un tipo de formulario

Todos los tipos de formularios se crean de forma similar, pero se establecen [propiedades](#) específicas para indicar cuál debe ser el comportamiento de los formularios.

Si va a crear un formulario secundario, debe especificar no sólo qué debe aparecer dentro de otro formulario, sino también si es un formulario secundario compatible con [MDI](#), que indica el comportamiento del formulario cuando se maximiza. Si el formulario secundario es compatible MDI, se combina con el formulario primario, compartiendo la barra de título, el título, los menús y las barras de herramientas del formulario primario. Un formulario secundario no compatible con MDI se maximiza en toda la zona cliente del formulario primario, pero conserva su título y su barra de título.

## Para especificar un formulario secundario

1. Cree o modifique el formulario mediante el [Diseñador de formularios](#).
2. Establezca la propiedad [ShowWindow](#) del formulario a uno de los valores siguientes:
  - **0 – En pantalla.** El formulario primario del formulario secundario será la ventana principal de Visual FoxPro.
  - **1 – En formulario de nivel superior.** El formulario primario del formulario secundario será el formulario de nivel superior que esté activo cuando se muestre la ventana secundaria. Use este valor si quiere que la ventana secundaria aparezca dentro de cualquier ventana de nivel superior distinta de la ventana principal de Visual FoxPro.
3. Establezca la propiedad [MDIForm](#) del formulario a .T. (verdadero) si quiere que el formulario

secundarios se combine con el formulario primario cuando se maximice, o a .F. (falso) si la ventana secundaria debe seguir siendo una ventana independiente cuando se maximice.

Un formulario flotante es una variación de un formulario secundario.

### Para especificar un formulario flotante

1. Cree o modifique el formulario con el [Diseñador de formularios](#).
2. Establezca la propiedad [ShowWindow](#) del formulario a uno de los valores siguientes:
  - **0 – En pantalla.** El formulario primario del formulario flotante será la ventana principal de Visual FoxPro.
  - **1 – En formulario de nivel secundario.** El formulario primario del formulario flotante será el formulario de nivel superior que esté activo cuando se muestre la ventana flotante.
3. Establezca la propiedad [Desktop](#) del formulario a .T. (verdadero).

### Para especificar un formulario de nivel superior

1. Cree o modifique el formulario con el [Diseñador de formularios](#).
2. Establezca la propiedad [ShowWindow](#) del formulario a **2 – Como formulario de nivel superior**.

### Mostrar un formulario secundario dentro de un formulario de nivel superior

Si ha creado un formulario secundario en el que la propiedad [ShowWindow](#) esté establecido como **1 – En formulario de nivel superior**, no especifique directamente el formulario de nivel superior que actúa como formulario primario del formulario secundario. En lugar de esto, Visual FoxPro asigna el formulario secundario a un formulario primario cuando se muestre la ventana secundaria.

### Para mostrar un formulario secundario dentro de un formulario de nivel superior

1. Cree un formulario de nivel superior.
2. En el código de evento del formulario de nivel superior, incluya el comando [DO FORM](#) y especifique el nombre del formulario secundario que va a mostrar.

Por ejemplo, cree un botón en el formulario de nivel superior y, a continuación, incluya en el código del evento [Click](#) para el botón un comando como este:

```
DO FORM MiSecundario
```

**Nota** El formulario de nivel superior debe estar visible y activo cuando se muestre el formulario secundario. Por lo tanto, no puede usar el [evento Init](#) del formulario de nivel superior para mostrar un formulario secundario, porque el formulario de nivel superior aún no estará activo.

3. Active el formulario de nivel superior y, si es necesario, desencadene el evento que muestra el

formulario secundario.

## Ocultar la ventana principal de Visual FoxPro

Si está ejecutando un formulario de nivel superior, es posible que quiera que la ventana principal de Visual FoxPro esté visible. Puede usar la propiedad [Visible](#) del [objeto Application](#) para ocultar y mostrar la ventana principal de Visual FoxPro cuando sea necesario.

### Para ocultar la ventana principal de Visual FoxPro

1. En el evento [Init](#) del formulario, incluya la siguiente línea de código:

```
Application.Visible = .F.
```

2. En el evento [Destroy](#) del formulario, incluya la siguiente línea de código:

```
Application.Visible = .T.
```

Asegúrese de que también proporciona una forma de cerrar el formulario con `THISFORM.Release` en algún método o evento.

**Nota** También puede incluir la línea siguiente en un archivo de configuración para ocultar la ventana principal de Visual FoxPro:

```
SCREEN = OFF
```

Para obtener más información sobre la configuración de Visual FoxPro, consulte el capítulo 3, [Configurar Visual FoxPro](#), de la *Guía de instalación e Índice principal*.

## Agregar un menú a un formulario de nivel superior

### Para agregar un menú a un formulario de nivel principal

1. Cree un menú de formulario de nivel superior. Para obtener más información acerca de la creación de menús para formularios de nivel superior, consulte el capítulo 11, [Diseñar menús y barras de herramientas](#).
2. Establezca la propiedad [ShowWindow](#) del formulario a **2 – Como formulario de nivel superior**.
3. En el evento [Init](#) del formulario, ejecute el programa de menú y pásele dos [parámetros](#):

```
DO menuname.mpr WITH oForm, lAutoRename
```

*oForm* es una referencia de objeto al formulario. En el evento `Init` del formulario, pásele `THIS` como primer parámetro.

*lAutoRename* especifica si se genera o no un nombre único para el menú. Si pretende ejecutar varias instancias del formulario, pase `.T.` como *lAutoRename*.

Por ejemplo, puede llamar a un menú llamado `mySDImenu` con este código:

```
DO mySDImenu.mpr WITH THIS, .T.
```

## Ampliar formularios con conjuntos de formularios

Puede manipular múltiples formularios como si fueran un grupo incluyéndolos en un [conjunto de formularios](#). Un conjunto de formularios tiene estas ventajas:

- Permite mostrar u ocultar todos los formularios de un conjunto al mismo tiempo.
- Permite organizar visualmente múltiples formularios al mismo tiempo para controlar sus posiciones relativas.
- Como todos los formularios de un conjunto de formularios se definen en un único archivo .SCX con un único [entorno de datos](#), puede sincronizar automáticamente punteros de registros en múltiples formularios. Si cambia el puntero de registro de una [tabla primaria](#) de un formulario, los registros secundarios de otro formulario se actualizan y se muestran.

**Nota** Todos los formularios de un conjunto de formularios se cargan cuando se ejecuta el conjunto. Si necesita cargar muchos formularios con muchos [controles](#), puede tardar varios segundos.

### Crear un nuevo conjunto de formularios

Un conjunto de formularios es un contenedor primario para uno o varios formularios. Cuando tenga activo el [Diseñador de formularios](#), podrá crear un conjunto de formularios.

#### Para crear un conjunto de formularios

- En el menú [Formulario](#), elija **Crear conjunto de formularios**.

Si no desea trabajar con múltiples formularios como si fueran un grupo de formularios, no será necesario que cree un conjunto. Después de crear un [conjunto de formularios](#), podrá agregarle formularios.

### Agregar y eliminar formularios

Cuando haya creado un conjunto de formularios, podrá agregarle formularios nuevos y eliminar formularios existentes.

#### Para agregar formularios adicionales a un conjunto de formularios

- En el menú [Formulario](#), elija **Agregar nuevo formulario**.

#### Para eliminar un formulario de un conjunto de formularios

1. En el cuadro **Formulario** situado en la parte inferior del [Diseñador de formularios](#), seleccione el formulario.
2. En el menú **Formulario**, elija **Quitar formulario**.

Si en un conjunto de formularios sólo tiene un formulario, podrá eliminar el conjunto de formularios para quedarse sólo con el formulario.

### Para eliminar un conjunto de formularios

- En el menú [Formulario](#), elija **Quitar conjunto de formularios**.

Los formularios se guardan con formato de tabla en un archivo que tiene la extensión .scx. Cuando cree un formulario, la tabla .scx contendrá un registro para el formulario, un registro para el [entorno de datos](#) y dos registros de uso interno. Se agregará un registro para cada objeto que agregue al formulario o al entorno de datos. Si crea un conjunto de formularios, se agregará un registro adicional para el conjunto de formularios y para cada formulario nuevo. El contenedor principal de cada formulario es el conjunto de formularios, mientras que el contenedor primario de cada control es el formulario en el que está situado.

**Sugerencia** Cuando ejecute un conjunto de formularios, no deseará que todos los formularios del conjunto sean visibles inicialmente. Establezca la propiedad [Visible](#) como falsa (.F.) para los formularios que no desee mostrar cuando se ejecute el conjunto de formularios. Establezca la propiedad Visible como verdadera (.T.) si desea mostrar los formularios.

## Agregar objetos a formularios

Para diseñar la funcionalidad que desea en un formulario, agregue los controles apropiados, establezca las propiedades del formulario y de los controles, y escriba el código de evento.

Puede agregar los siguientes tipos de objetos a un formulario:

- [Controles](#)
- [Contenedores](#)
- [Clases definidas por el usuario](#)
- [Objetos OLE](#)

### Descripción de los objetos contenedores y de control

Los [objetos](#) de Visual FoxPro pueden corresponder a dos categorías, según la naturaleza de la clase en la que se basen:

- Los [contenedores](#) pueden ser el objeto primario para otros objetos. Por ejemplo, un formulario, como contenedor, es un objeto primario para una casilla de verificación de ese formulario.
- Los [controles](#) puede estar contenidos en contenedores, pero no pueden ser objetos primarios de otros objetos. Por ejemplo, una casilla de verificación no puede contener ningún otro objeto.

El [Diseñador de formularios](#) permite diseñar contenedores y controles.

Contenedor	Puede contener
Columna	Encabezados y cualquier objeto excepto conjunto de formularios, formularios, barras de herramientas, cronómetros y otras columnas
Grupo de botones de comando	Botones de comando
Conjunto de formularios	Formularios, barras de herramientas
Formulario	Marcos de página, cuadrículas, cualquier control
Cuadrícula	Columnas
Grupo de botones de opción	Botones de opción
Marco de página	Páginas
Página	Cuadrículas, cualquier control

## Agregar contenedores de Visual FoxPro

Además de conjuntos de formularios y formularios, Visual FoxPro proporciona cuatro clases de contenedores de base.

### Clases de contenedores de Visual FoxPro

<a href="#">Grupo de botones de comando</a>	<a href="#">Grupo de botones de comando</a>
<a href="#">Cuadrícula</a>	<a href="#">Marco de páginas</a>

### Para agregar objetos contenedores a un formulario

- En la barra de herramientas [Controles de formularios](#), seleccione el botón del objeto contenedor deseado (cuadrícula, marco de páginas o grupo de botones) y arrástrelo para ajustar su tamaño en el formulario.

Al agregar un grupo de botones de comando o un grupo de botones de opción a un formulario, el grupo contiene dos botones de forma predeterminada. Cuando agregue un marco de páginas a un formulario, el marco de páginas contendrá dos páginas de forma predeterminada. Puede agregar más botones o páginas si establece las propiedades [ButtonCount](#) o [PageCount](#) con el número que desee.

Cuando agregue una cuadrícula a un formulario, la propiedad [ColumnCount](#) se establecerá de forma predeterminada como - 1, que indica AutoFill (relleno automático). En tiempo de ejecución, la cuadrícula mostrará tantas columnas como campos haya en la tabla RowSource. Si no desea AutoFill, puede especificar el número de columnas si establece la [propiedad ColumnCount](#) de la cuadrícula.

Para obtener más información sobre estos objetos contenedores, consulte el capítulo 10, [Usar controles](#).

## Propiedades Collection y Count

Todos los objetos contenedores de Visual FoxPro tienen asociadas una propiedad contador y una propiedad colección. La propiedad colección es una [matriz](#) que hace referencia a cada objeto contenido. La propiedad contador es una propiedad numérica que indica el número de objetos contenidos.

Se da nombre a las propiedades colección y contador para cada contenedor de acuerdo con el tipo de objeto que se puede contener en el contenedor. La tabla siguiente muestra los contenedores y las correspondientes propiedades colección y contador.

Contenedor	Propiedad colección	Propiedad contador
<a href="#">Application</a>	<a href="#">Objects</a> <a href="#">Forms</a>	<a href="#">Count</a> <a href="#">FormCount</a>
<a href="#">FormSet</a>	<a href="#">Forms</a>	<a href="#">FormCount</a>
<a href="#">Form</a>	<a href="#">Objects</a> <a href="#">Controls</a>	<a href="#">Count</a> <a href="#">ControlCount</a>
<a href="#">PageFrame</a>	<a href="#">Pages</a>	<a href="#">PageCount</a>
<a href="#">Page</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Grid</a>	<a href="#">Columns</a>	<a href="#">ColumnCount</a>
<a href="#">CommandGroup</a>	<a href="#">Buttons</a>	<a href="#">ButtonCount</a>
<a href="#">OptionGroup</a>	<a href="#">Buttons</a>	<a href="#">ButtonCount</a>
<a href="#">Column</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">ToolBar</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Container</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Control</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>

Estas propiedades le permiten usar un bucle para manipular mediante programación todos los objetos contenidos o algunos objetos específicos. Por ejemplo, las líneas de código siguientes establecen la propiedad [BackColor](#) de columnas de una cuadrícula a verde y rojo de forma alterna:

```
o = THISFORM.grd1
FOR i = 1 to o.ColumnCount
  IF i % 2 = 0 && Columna par
    o.Columns(i).BackColor = RGB(0,255,0) && Verde
  ELSE
    o.Columns(i).BackColor = RGB(255,0,0) && Rojo
  ENDIF
ENDFOR
```

## Agregar controles de Visual FoxPro a un formulario



La barra de herramientas Controles de formularios permite agregar fácilmente al formulario cualquier control estándar de Visual FoxPro.

### Controles estándar de Visual FoxPro

[Check box](#)

[Hyperlink](#)

[List box](#)

[Spinner](#)

[Combo box](#)

[Image](#)

[OLE Bound Control](#)

[Text box](#)

[Command button](#)

[Label](#)

[OLE Container Control](#)

[Timer](#)

[Edit box](#)

[Line](#)

[Shape](#)

### Para agregar controles a un formulario

- En la barra de herramientas [Controles de formularios](#), seleccione el botón del control deseado y haga clic o arrástrelo para ajustar su tamaño en el formulario.

Para obtener más información sobre los controles que puede elegir, consulte el capítulo 10, [Usar controles](#).

### Agregar controles vinculados a datos a un formulario

Puede vincular controles a datos de una tabla, una vista, un campo de tabla o un campo de vista si establece la propiedad [ControlSource](#) de un control a un campo de la [propiedad RecordSource](#) de una [cuadrícula](#), a una tabla o una vista. Pero también puede crear controles vinculados a datos si arrastra campos o tablas al formulario directamente desde:

- El [Administrador de proyectos](#)
- El [Diseñador de bases de datos](#)
- El [Diseñador de entorno de datos](#)

La clase de control creado de esta forma depende de la configuración de **Asignaciones de campos** de la ficha **Propiedades** del **Diseñador de tablas** o de la ficha **Asignación de campos** del cuadro de diálogo **Opciones**.

Para obtener más información sobre cómo establecer clases de controles predeterminadas, vea el [Diseñador de tablas](#) o la ficha [Asignación de campos](#) del cuadro de diálogo Opciones.

### Agregar objetos definidos por el usuario a un formulario

Una de las características más potentes de Visual FoxPro es su capacidad para crear [clases](#) que se pueden utilizar y reutilizar fácilmente en distintas partes de las aplicaciones. Cuando haya creado las clases, podrá agregarlas a los formularios o conjuntos de formularios.

### Para agregar un objeto basado en una clase personalizada

- En el [Administrador de proyectos](#), arrastre la clase hasta el formulario o la página.

Las clases se pueden agregar directamente desde la barra de herramientas [Controles de formularios](#) cuando las registra.

### Agregar bibliotecas de clases a la barra de herramientas Controles

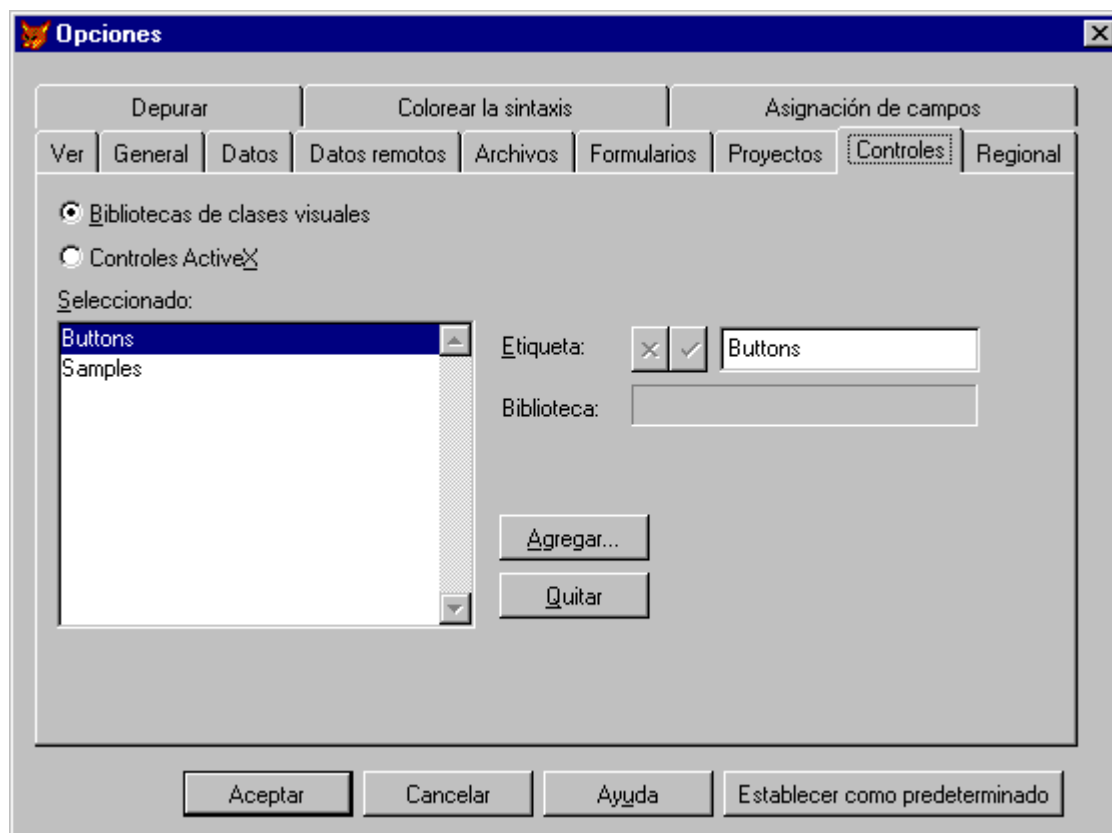
Debe registrar las [bibliotecas de clases](#) antes de poder mostrarlas en la barra de herramientas [Controles de formularios](#).

### Para registrar una biblioteca de clases

1. En el menú [Herramientas](#), elija **Opciones**.
2. En el cuadro de diálogo [Opciones](#), elija la ficha **Controles**.
3. Elija **Agregar**.
4. En el cuadro de diálogo **Abrir**, elija una biblioteca de clases que desee agregar a la lista **Seleccionado** y elija **Abrir**.
5. Repita los pasos 3 y 4 hasta agregar todas las bibliotecas que desee registrar.

Las clases de las bibliotecas que se hayan incluido en la lista Seleccionado pueden emplearse en el [Diseñador de formularios](#) con la misma facilidad que las [clases de base](#) de Visual FoxPro.

### Ficha Controles del cuadro de diálogo Opciones



**Sugerencia** Si quiere que las bibliotecas de clases estén disponibles en la barra de herramientas Controles de formularios cada vez que ejecute Visual FoxPro, elija Establecer como predeterminado en el cuadro de diálogo [Opciones](#).

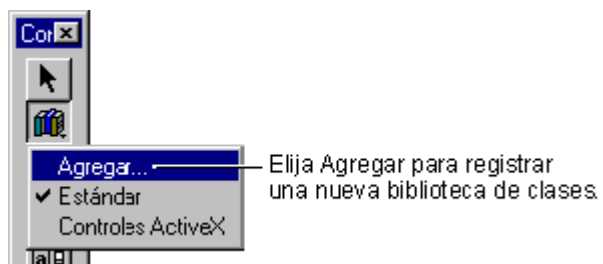
También puede registrar bibliotecas directamente en el [Diseñador de formularios](#).

### Para registrar una biblioteca de clases en el Diseñador de formularios



1. En la barra de herramientas [Controles de formularios](#), elija el botón **Ver clases**.
2. En el submenú, elija **Agregar**.

#### Submenú del botón Ver clases



3. En el cuadro de diálogo **Abrir**, elija una biblioteca de clases que desee agregar a la barra de herramientas **Controles de formularios** y elija **Abrir**.

### Agregar objetos a un formulario desde una biblioteca de clases

Cuando haya agregado bibliotecas de clases en la ficha Clases del cuadro de diálogo [Opciones](#) o desde el submenú Ver clases, tendrá acceso a las mismas en el [Diseñador de formularios](#).

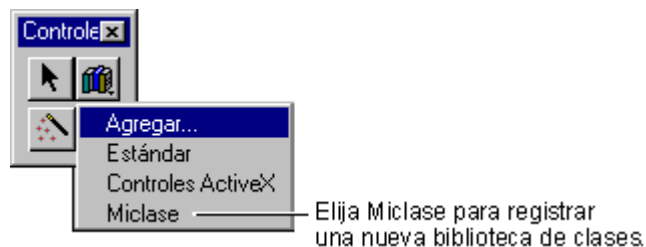
### Para agregar un objeto personalizado desde la barra de herramientas Controles de formularios



1. En la barra de herramientas [Controles de formularios](#), elija el botón **Ver clases**.
2. En la lista de bibliotecas de clases registradas, seleccione la biblioteca que contiene el control que desea agregar al formulario.

La barra de herramientas está formada por los controles de la biblioteca que ha seleccionado.

### La biblioteca de clases definida por el usuario se agrega al submenú Ver clases



3. Haga clic en el control que desee y arrastre para ajustar su tamaño en el formulario.

**Nota** Puede eliminar una biblioteca de clases visuales desde el menú de la barra de herramientas Ver clases si selecciona la biblioteca en la lista Seleccionado de la ficha Controles de formularios del cuadro de diálogo Opciones y elige Quitar.

Cuando agregue a un formularios objetos que no estén basados en las [clases de base](#) de Visual FoxPro, se almacenará una ruta relativa a la biblioteca de clases (archivo .vcx) en el archivo .scx del formulario. Si mueve el formulario o la [biblioteca de clases](#) a una ubicación diferente, Visual FoxPro muestra un cuadro de diálogo cuando intenta ejecutar el formulario de forma que puede localizar manualmente la biblioteca de clases.

### Determinar los controles que hay en un formulario

Para determinar cuántos controles hay en un formulario, puede utilizar la propiedad [ControlCount](#). La propiedad [Controls\[n\]](#) del formulario permite hacer referencia a cada control del formulario. El programa siguiente imprime la propiedad [Name](#) de todos los controles que hay actualmente en el formulario activo.

```
ACTIVATE SCREEN  && imprimir en la ventana principal de Visual FoxPro
FOR nCnt = 1 TO Application.ActiveForm.ControlCount
    ? Application.ActiveForm.Controls[nCnt].Name
ENDFOR
```

## Agregar propiedades y métodos a un formulario

Puede agregar a un formulario tantas [propiedades](#) y [métodos](#) nuevos como desee. Las propiedades contienen un valor, mientras que los métodos contienen código de procedimientos que se ejecuta al llamar al método. Las propiedades y los métodos nuevos tienen como alcance el formulario y se hace referencia a los mismos del mismo modo que a otras propiedades o métodos.

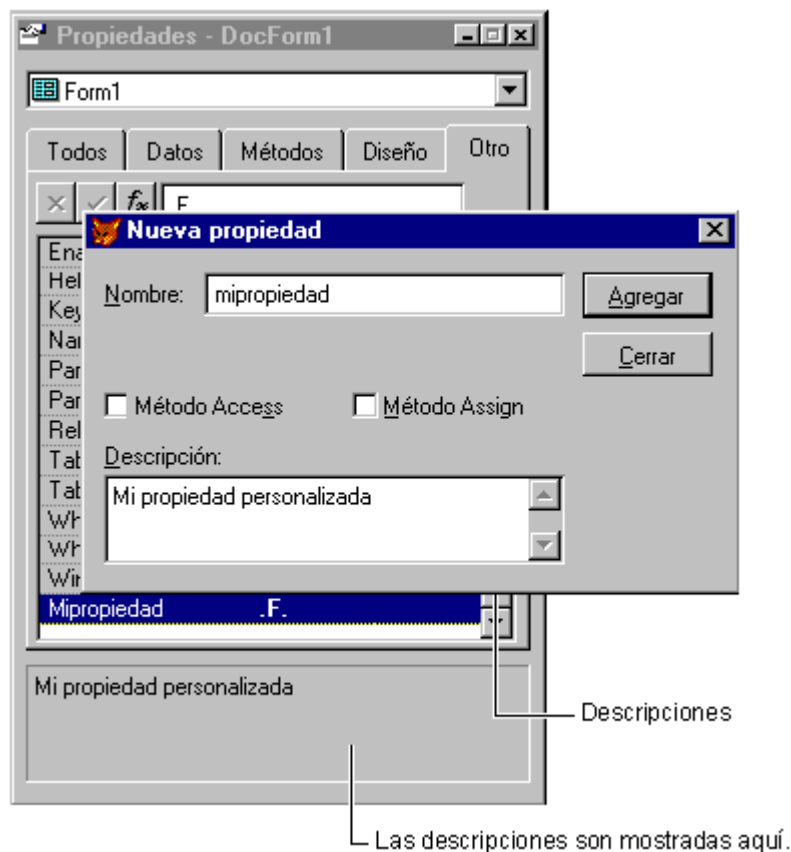
### Crear propiedades nuevas

Si dispone de un conjunto de formularios, las propiedades y los métodos que agregue en el [Diseñador de formularios](#) tendrán como alcance el conjunto de formularios. Si no dispone de ningún conjunto de formularios, las propiedades y los métodos tendrán como alcance el formulario.

### Para agregar una propiedad nueva a un formulario

1. En el menú [Formulario](#), elija **Nueva propiedad**.
2. En el cuadro de diálogo [Nueva propiedad](#), escriba el nombre de la propiedad. También puede incluir una descripción de la propiedad que se mostrará en la parte inferior de la ventana [Propiedades](#).

### Agregar una propiedad a un formulario



### Crear una propiedad de matriz

Una propiedad de matriz se incluye en el alcance del formulario o del conjunto de formularios como cualquier otra propiedad, pero puede manipularse mediante los comandos y las funciones de matriz de Visual FoxPro.

#### Para crear una propiedad de matriz

1. Agregue una nueva propiedad al formulario.
2. En el cuadro **Nombre** del cuadro de diálogo [Nueva propiedad](#), escriba el nombre de la propiedad de matriz e incluya el tamaño y las dimensiones de la matriz.

Por ejemplo, podría escribir **arrayprop[10,2]** en el cuadro Nombre del cuadro de diálogo Nueva propiedad para crear una matriz bidimensional con 10 filas.

Las propiedades de matriz son de sólo lectura en modo de diseño, pero puede administrar, redimensionar y asignar valores a los elementos de la propiedad matriz en [tiempo de ejecución](#). Para ver un ejemplo del uso de una propiedad de matriz, consulte [Administrar múltiples instancias de un formulario](#).

### Crear nuevos métodos

Puede agregar al formulario [métodos](#) a los que se puede llamar del mismo modo que a los métodos de

clase de formulario.

### Para crear un nuevo método para un formulario

1. En el menú [Formulario](#), elija **Nuevo método**.
2. En el cuadro de diálogo [Nuevo método](#), escriba el nombre del método. Opcionalmente, puede incluir una descripción del método.

Para llamar a los métodos definidos por el usuario, como sucede con los métodos de clase de base, utilice la [sintaxis](#) siguiente:

*NombreObjeto.NombreMétodo*

El método también puede aceptar [parámetros](#) y devolver valores. En este caso, se llama al método en una instrucción de asignación:

*cVariable = NombreObjeto.NombreMétodo(cParámetro, nParámetro)*

### Incluir constantes predefinidas

Para utilizar [constantes](#) predefinidas en sus métodos, puede incluir un archivo de encabezado en un formulario o un conjunto de formularios con [#INCLUDE](#). Un archivo de encabezado suele contener constantes de tiempo de compilación definidas con la directiva del preprocesador [#DEFINE](#).

### Para incluir un archivo en un formulario

1. En el menú **Formulario**, elija **Incluir archivo**.
2. En el cuadro de diálogo [Incluir archivo](#), especifique el archivo deseado en el cuadro de texto **Incluir archivo**.

–O bien–

Elija el botón de tres puntos para abrir el cuadro de diálogo **Incluir** y elija el archivo.

3. Elija **Incluir**.

## Manipular objetos

Hay varias formas de manipular los [objetos](#) en [tiempo de diseño](#):

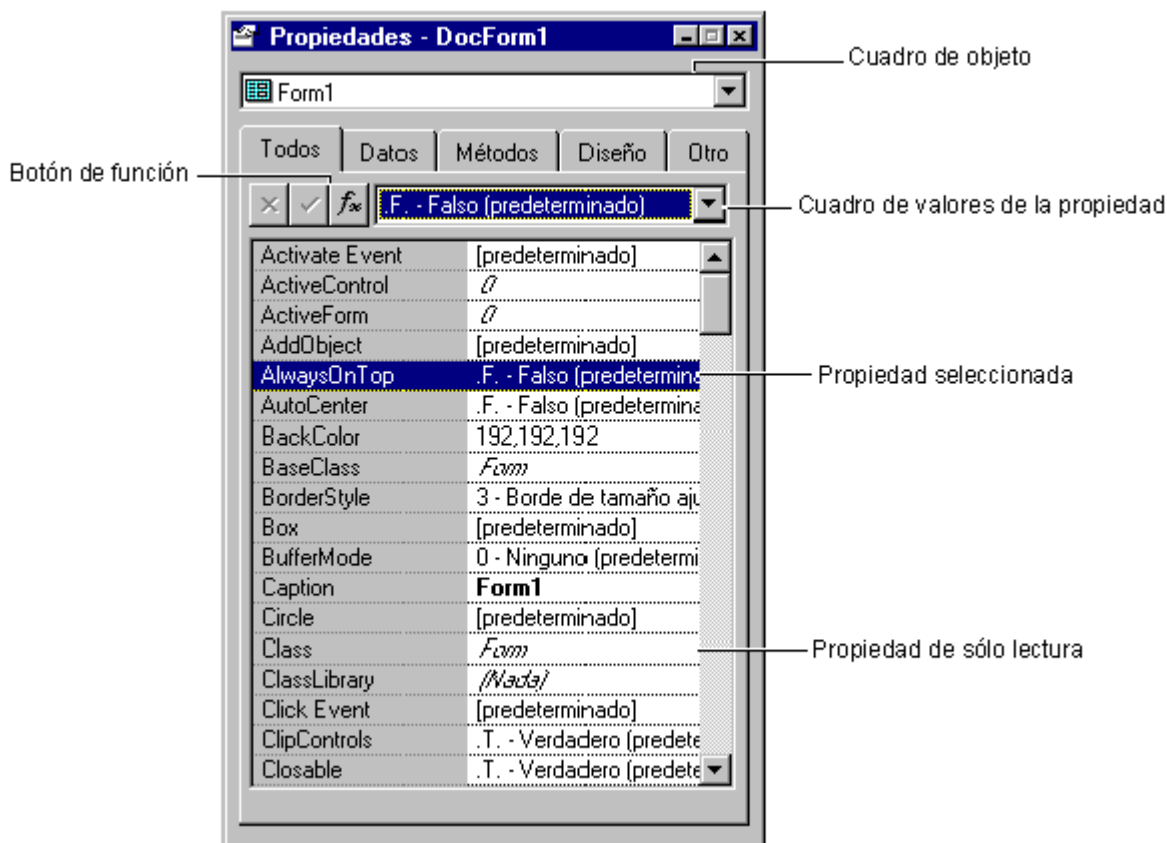
- Establezca el tamaño y la posición de los objetos arrastrándolos en la ventana [Diseñador de formularios](#).
- Alinee los controles eligiendo las herramientas de alineación de la barra de herramientas [Diseño](#) o los comandos del menú [Formato](#).
- Establezca los colores de primer plano y de fondo en la barra de herramientas [Paleta](#).
- Establezca las propiedades en la ventana [Propiedades](#). El punto principal de control para todos

los objetos del formulario o del conjunto de formularios es la ventana Propiedades.

## Establecer propiedades en tiempo de diseño

Cuando se abre la ventana Propiedades, muestra las [propiedades](#) o los [eventos](#) del objeto seleccionado. Si hay más de un objeto seleccionado, las propiedades que los objetos tienen en común se muestran en la ventana Propiedades. Para modificar las propiedades o los eventos de otro objeto, elija el objeto correspondiente en el cuadro Objeto o bien seleccione otro control en el formulario.

### La ventana Propiedades



### Para establecer una propiedad

1. En la ventana [Propiedades](#), seleccione una propiedad en la lista **Propiedades y eventos**.
2. En el cuadro **Valores de propiedades**, escriba o elija el valor que desee para la propiedad seleccionada.

**Nota** Las propiedades de sólo lectura en [tiempo de diseño](#), como la propiedad [Class](#) de un objeto, se muestran en cursiva en la lista **Propiedades y eventos** de la ventana **Propiedades**.

Si la propiedad necesita un valor de carácter, no será necesario que incluya el valor entre comillas. Si desea que el título de un formulario sea CLIENTE, escriba **CLIENTE** en el cuadro **Valores de propiedades**. Si desea que el título de un formulario sea "CLIENTE", mostrando las comillas en el título de la ventana, escriba "CLIENTE" en el cuadro **Valores de propiedades**.



## Establecer propiedades con expresiones

También puede establecer propiedades con los resultados de [expresiones](#) o [funciones](#) a través de la ventana Propiedades.

### Para establecer una propiedad con una expresión

- En la ventana [Propiedades](#), elija el botón **Función** para abrir el **Generador de expresiones**.  
–O bien–
- En el cuadro **Valores de propiedades**, escriba = seguido de una expresión.

Por ejemplo, puede establecer la propiedad [Caption](#) de un formulario para indicar la tabla activa actualmente cuando se ejecuta el formulario; para ello, escriba =**ALIAS( )** en el cuadro **Valores de propiedades**.

La expresión de una propiedad se evalúa cuando se establece en la ventana Propiedades y cuando el objeto se inicializa en [tiempo de ejecución](#) o de [diseño](#). Una vez creado el objeto, el valor de la propiedad no cambiará hasta que usted o un usuario la modifique explícitamente.

**Solución de problemas** Si establece una propiedad con el resultado de una [función definida por el usuario](#), la función se evaluará cuando establezca la propiedad o cuando modifique o ejecute el formulario. Si hay un error en la función definida por el usuario, es posible que no pueda abrir el formulario.

También puede establecer la propiedad para la función definida por el usuario en el [evento Init](#) del objeto, como en el ejemplo siguiente:

```
THIS.Caption = mifunción( )
```

Si hay un error en la función definida por el usuario, no podrá ejecutar el formulario de este modo, pero sí podrá modificarlo.

## Definir el comportamiento de un formulario

Cuando diseña un formulario en el [Diseñador de formularios](#), verá los cambios en directo: excepto para establecer la propiedad [Visible](#) a falso (.F.), los cambios visuales y de comportamiento que efectúe se reflejarán inmediatamente en el formulario. Si establece la propiedad [WindowState](#) como 1 (Minimizada) o 2 (Maximizada), el formulario del Diseñador de formularios reflejará inmediatamente este cambio. Si establece la propiedad [Movable](#) como falso (.F.), el usuario no podrá mover el formulario en [tiempo de ejecución](#) y usted tampoco podrá moverlo en [tiempo de diseño](#). Es conveniente diseñar la funcionalidad del formulario y agregar todos los controles apropiados antes de establecer algunas propiedades que determinan su comportamiento.

Las siguientes propiedades de formulario suelen establecerse en tiempo de diseño para definir la apariencia y el comportamiento del formulario:

<b>Propiedad</b>	<b>Descripción</b>	<b>Opción predeterminada</b>
<a href="#"><u>AlwaysOnTop</u></a>	Controla si un formulario siempre está situado sobre las demás ventanas abiertas.	Falso (.F.)
<a href="#"><u>AutoCenter</u></a>	Controla si el formulario se centra automáticamente en la ventana principal de Visual FoxPro cuando se inicializa el formulario.	Falso (.F.)
<a href="#"><u>BackColor</u></a>	Determina el color de la ventana del formulario.	255,255,255
<a href="#"><u>BorderStyle</u></a>	Controla si el formulario no tiene borde, tiene un borde de una sola línea, de doble ancho o del sistema. Si BorderStyle es 3 (Sistema), el usuario podrá cambiar el tamaño del formulario.	3
<a href="#"><u>Caption</u></a>	Determina el texto que aparece en la barra de título del formulario.	Form1
<a href="#"><u>Closable</u></a>	Controla si el usuario puede cerrar el formulario haciendo doble clic en el cuadro de cierre.	Verdadero (.T.)
<a href="#"><u>DataSession</u></a>	Controla si las tablas del formulario o el conjunto de formularios están abiertas en áreas de trabajo accesibles globalmente o privadas para el formulario o el conjunto de formularios.	1
<a href="#"><u>MaxButton</u></a>	Controla si el formulario tiene o no un botón de maximizar.	Verdadero (.T.)
<a href="#"><u>MinButton</u></a>	Controla si el formulario tiene o no un botón de minimizar.	Verdadero (.T.)
<a href="#"><u>Movable</u></a>	Controla si el formulario puede moverse o no a una nueva ubicación de la pantalla.	Verdadero (.T.)
<a href="#"><u>ScaleMode</u></a>	Controla si la unidad de medida para las propiedades de tamaño y posición de los objetos es fóxeles o píxeles.	Determinado por los valores del cuadro de diálogo <b>Opciones</b> .
<a href="#"><u>Scrollbars</u></a>	Controla el tipo de barras de desplazamiento que tiene un formulario.	0 – Ninguna
<a href="#"><u>TitleBar</u></a>	Controla si aparece una barra de título en la parte superior del formulario.	1 – Activo
<a href="#"><u>ShowWindow</u></a>	Controla si la ventana es una ventana secundaria (en la pantalla), flotante o de nivel superior.	0 - En pantalla
<a href="#"><u>WindowState</u></a>	Controla si el formulario está minimizado, maximizado o es normal.	0 – Normal

---

<a href="#">WindowType</a>	Controla si el formulario es de tipo sin modo (opción predeterminada) o modal. Si es modal, el usuario deberá cerrar el formulario antes de tener acceso a ningún otro elemento de la interfaz de usuario de la aplicación.	0 – Sin modo
----------------------------	---	--------------

---

Utilice la propiedad [LockScreen](#) para que el ajuste en tiempo de ejecución de las propiedades de diseño de controles parezca más limpio.

### Asignar iconos a formularios

En Visual FoxPro para Windows, puede asignar un icono al formulario; el icono se muestra cuando la ventana está minimizada en Windows NT<sup>®</sup> y en la barra de título en Windows 95. Para asignar un icono a un formulario, establezca la propiedad Icon del formulario al nombre de un archivo .ICO.

#### Para asignar un icono a un formulario

1. Abra el formulario.
2. Abra la ventana [Propiedades](#).
3. Establezca la propiedad [Icon](#) al archivo .ICO que quiera mostrar.

### Modificar código de evento y método

Los eventos son acciones del usuario, como clics o movimientos del *mouse*, o bien acciones del sistema, como el avance del reloj del sistema. Los métodos son procedimientos asociados al objeto que se invocan específicamente mediante programación. Si desea ver una explicación de los eventos y métodos, consulte el capítulo 3, [Programación orientada a objetos](#). Puede especificar el código que desea procesar cuando se desencadene un evento o se invoque un método.

#### Para modificar código de evento o método

1. En el menú **Ver**, elija **Código**.
2. Seleccione el evento o el método en el cuadro **Procedimiento**.
3. En la ventana **Modificar**, escriba el código que desea procesar cuando se desencadene el evento o se invoque el método.

Por ejemplo, podría tener en un formulario un botón de comando con el título Salir. En el evento Click del botón, incluya la línea:

```
THISFORM.Release
```

**Sugerencia** Para moverse entre los procedimientos en la ventana Edición de código, presione AV PÁG o RE PÁG.

Cuando el usuario hace clic en el botón de comando, se elimina el formulario de la pantalla y de la memoria. Si no desea liberar el formulario de la memoria, puede incluir en su lugar la línea siguiente en el evento Click:

```
THISFORM.Hide
```

**Nota** Si el código asociado al evento Init de un [conjunto de formularios](#), un [formulario](#) o cualquier objeto de un conjunto de formularios devuelve falso (.F.), no se creará el formulario o el conjunto de formularios

## Guardar formularios

Es necesario guardar el formulario antes de poder ejecutarlo. Si intenta ejecutar el formulario o cerrar el Diseñador de formularios cuando aún no ha guardado el formulario, Visual FoxPro le pedirá que guarde o que descarte los cambios realizados.

### Para guardar un formulario

- En el [Diseñador de formularios](#), elija **Guardar** en el menú **Archivo**.

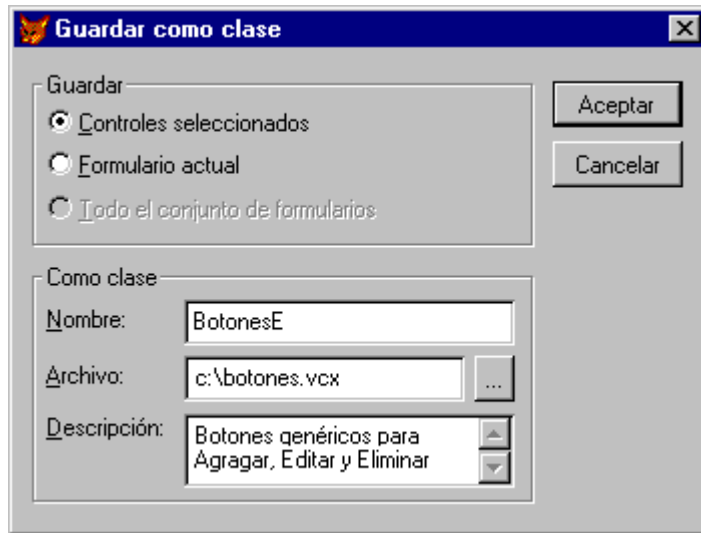
### Guardar formularios y controles como clases

También puede guardar un formulario o un subconjunto de los controles de un formulario, como una definición de clase. Si piensa crear [subclases](#) basadas en el formulario o volver a utilizar los controles en otros formularios, guarde el formulario como una definición de clase.

### Para guardar un formulario o los controles seleccionados como una definición de clase

1. En el menú **Archivo**, elija **Guardar como clase**.
2. En el cuadro de diálogo [Guardar como clase](#), elija **Formulario actual** o **Controles seleccionados**.

### El cuadro de diálogo Guardar como clase



3. Escriba un nombre para la clase en el cuadro **Nombre**.
4. En el cuadro **Archivo**, introduzca un nombre de archivo en el que se almacenará la clase.
5. Elija **Aceptar**.

Si no incluye una extensión con el nombre de archivo, se agregará la extensión predeterminada .vcx cuando se guarde el archivo. Cuando se haya guardado un formulario como una definición de clase, podrá modificarla mediante el comando MODIFY CLASS. Para obtener más información sobre la creación de clases, consulte el capítulo 3, [Programación orientada a objetos](#).

## Ejecutar un formulario

Un formulario puede ejecutarse directamente desde la interfaz o mediante programación.

### Ejecutar un formulario de forma interactiva

Hay varios modos de ejecutar el formulario que ha diseñado.



Si está trabajando en el [Diseñador de formularios](#), puede probar el formulario haciendo clic en el botón Ejecutar de la barra de herramientas [Diseñador de formularios](#). Para volver a abrir el formulario en el Diseñador de formularios, ciérrelo o elija el botón Modificar formulario de la barra de herramientas.

También puede ejecutar un formulario desde un proyecto o mediante programación.

### Para ejecutar un formulario

- En el [Administrador de proyectos](#), seleccione el formulario y elija **Ejecutar**.

–O bien–

- Escriba [DO FORM](#) en la ventana [Comandos](#).

También puede ejecutar el formulario si elige Ejecutar en el menú Programa, a continuación elige Formulario en el cuadro Tipo de archivo y finalmente selecciona el formulario y elige Ejecutar.

### Ejecutar un formulario desde un programa

Para ejecutar un formulario mediante programación, incluya el comando [DO FORM](#) en el código asociado a un evento, en código de método o en un programa o un procedimiento.

### Dar nombre al objeto formulario

De forma predeterminada, cuando usa el comando DO FORM, el nombre del objeto formulario es el mismo que el del archivo .scx. Por ejemplo, la siguiente línea de código ejecuta Customer.scx. Visual FoxPro crea automáticamente una [variable](#) objeto para el formulario llamada `customer`:

```
DO FORM Customer
```

### Para dar nombre a un objeto formulario

- Use la cláusula NAME del comando [DO FORM](#).

Por ejemplo, los comandos siguientes ejecutan un formulario y crean dos nombres de [variable](#) objeto formulario:

```
DO FORM Customer NAME frmCust1  
DO FORM Customer NAME frmCust2
```

### Manipular el objeto formulario

Si ejecuta el comando [DO FORM](#) desde la [ventana Comandos](#), el objeto formulario se asociará a una [variable](#). Puede tener acceso al objeto formulario a través del nombre de variable. Por ejemplo, los comandos siguientes, ejecutados en la ventana Comandos, abren un formulario llamado `Customer` y cambian su título.

```
DO FORM Customer  
Customer.Caption = "Hola"
```

Si a continuación ejecuta el comando siguiente en la ventana Comandos, aparecerá O en la ventana de salida activa, lo que indica que `Customer` es un objeto:

```
? TYPE( "Customer" )
```

Si ejecuta el comando [DO FORM](#) en un programa, el objeto formulario se incluirá en el alcance del programa. Si se completa el programa o el procedimiento, el objeto desaparecerá, pero el formulario permanecerá visible. Por ejemplo, podría ejecutar el programa siguiente:

```
*formtest.prg  
DO FORM Customer
```

Después de ejecutar el programa, el formulario permanece visible y todos los controles del formulario están activos, pero `TYPE("Customer")` devuelve `U`, lo que indica que `Customer` es una [variable](#) no definida. El comando siguiente, ejecutado en la ventana Comandos, generaría un error:

```
Customer.Caption = "Hola"
```

Sin embargo, sí puede tener acceso al formulario mediante las propiedades [ActiveForm](#), [Forms](#) y [FormCount](#) del objeto de la aplicación.

### Ajustar el alcance del formulario a la variable de objeto formulario

La palabra clave `LINKED` del comando [DO FORM](#) permite vincular el formulario al objeto formulario. Si incluye la palabra clave `LINKED`, cuando la [variable](#) asociada al objeto formulario salga del alcance, se liberará el formulario.

Por ejemplo, el comando siguiente crea un formulario vinculado a la variable de objeto `frmCust2`:

```
DO FORM Customer NAME frmCust2 LINKED
```

Cuando se libera `frmCust2`, el formulario se cierra.

### Cerrar un formulario activo

Para permitir que los usuarios cierren el formulario activo al hacer doble clic en el cuadro de control o elegir Cerrar en el menú Control del formulario, establezca la propiedad `Closable` del formulario.

### Para permitir a un usuario cerrar el formulario activo

- En la ventana [Propiedades](#), establezca la propiedad [Closable](#) como verdadero (.T.).

–O bien–

- Utilice el comando [RELEASE](#).

Por ejemplo, puede cerrar y liberar el formulario `frmCustomer` si ejecuta el comando siguiente en un programa o en la [ventana Comandos](#):

```
RELEASE frmCustomer
```

También puede permitir que un usuario cierre y libere un formulario si incluye el comando siguiente en el código de evento [Click](#) de un control, como un botón de comando con el título Salir:

```
THISFORM.Release
```

También puede usar el comando [RELEASE](#) en el código asociado a un objeto del formulario, pero no se ejecutará el código que haya incluido en el método `Release`.

**Solución de problemas** Al liberar un formulario, se libera de la memoria la [variable](#) de objeto creada para el formulario. Hay una única variable para un conjunto de formularios, por lo que no podrá liberar formularios de un conjunto de formularios sin liberar el conjunto. Si desea liberar el conjunto de formularios, puede utilizar `RELEASE THISFORMSET`. Si desea liberar un formulario de la pantalla de modo que un usuario no pueda verlo o interactuar con él, utilice `THISFORM.Hide`.

## Establecer propiedades en tiempo de ejecución

El modelo de objetos de Visual FoxPro proporciona mucho control sobre las propiedades en [tiempo de ejecución](#).

### Referencias a objetos en la jerarquía de objetos

Para manipular un objeto, deberá identificarlo en relación con la jerarquía contenedora. En el nivel superior de la jerarquía de contenedores (el conjunto de formularios o el formulario) debe hacer referencia a la [variable](#) de objeto. A no ser que use la cláusula `NAME` del comando [DO FORM](#), la variable de objeto tiene el mismo nombre que el archivo `.scx`.

Para manipular las propiedades haga referencia al control y a la propiedad, separadas por un punto (`.`):

*objetovariable.[formulario.]control.propiedad = Configuración*

La tabla siguiente muestra las propiedades o las palabras clave que facilitan el establecimiento de referencias a un objeto en la jerarquía de objetos.

Propiedad o palabra clave	Referencia
<a href="#">ActiveControl</a>	El control del formulario activo actualmente que tiene el enfoque
<a href="#">ActiveForm</a>	El formulario activo actualmente
<a href="#">ActivePage</a>	La página activa del formulario activo actualmente
<a href="#">Parent</a>	El contenedor más cercano al objeto
<a href="#">THIS</a>	El objeto o un procedimiento o evento del objeto
<a href="#">THISFORM</a>	El formulario que contiene el objeto
<a href="#">THISFORMSET</a>	El conjunto de formularios que contiene al objeto

Por ejemplo, para cambiar el título de un botón de comando del formulario `frmCust` de un conjunto de formularios almacenado en `Custview.scx`, use el siguiente comando en un programa o en la ventana Comandos:

```
CustView.frmCust.cmdButton1.Caption = "Modificar"
```

Utilice las palabras clave `THIS`, `THISFORM` y `THISFORMSET` para hacer referencia a objetos que están en un formulario o un conjunto de formularios. Por ejemplo, para cambiar el título de un botón



de comando cuando hace clic en éste, incluya el comando siguiente en el código de evento Click para el botón de comando:

```
THIS.Caption = "Modificar"
```

La tabla siguiente ofrece ejemplos del uso de THISFORMSET, THISFORM, THIS y Parent para establecer propiedades de objeto.

Comando	Dónde debe incluir el comando
<code>THISFORMSET.frm1.cmd1.Caption = 'Aceptar'</code>	En el código de evento o de método de cualquier control de un formulario del conjunto de formularios salvo <code>frm1</code> .
<code>THISFORM.cmd1.Caption = 'Aceptar'</code>	En el código de evento o de método de cualquier control salvo <code>cmd1</code> en el mismo formulario en que está <code>cmd1</code> .
<code>THIS.Caption = 'Aceptar'</code>	En el código de evento o de método del control cuyo título desea cambiar.
<code>THIS.Parent.BackColor = RGB(192,0,0)</code>	En el código de evento o de método de un control de un formulario. El comando cambia a rojo oscuro el color de fondo del formulario.

### Establecer propiedades en tiempo de ejecución mediante expresiones

También puede establecer propiedades en [tiempo de ejecución](#) si utiliza [expresiones](#) o [funciones](#).

### Para establecer propiedades a expresiones en tiempo de ejecución

- Asigne una expresión a la propiedad.
- O bien–
- Asigne a la propiedad el resultado de una [función definida por el usuario](#).

Por ejemplo, podría establecer el título de un botón como Modificar o Guardar, según el valor de una [variable](#). Declare la variable en el programa que llama a su formulario:

```
PUBLIC glModificar
glModificar = .F.
```

A continuación, utilice una expresión IIF para configurar la propiedad Caption:

```
frsSet1.frmForm1.cmdButton1.Caption = ;
IIF(glModificar = .F., "Modificar", "Guardar")
```

Podría determinar el tamaño de un botón y establecer el título mediante expresiones con campos de

una tabla:

```
* establecer ancho del botón como longitud de 'Llamada ' + nombre y
* apellido
frmForm1.cmdButton1.Width = 5 + ;
    LEN(ALLTRIM(employee.first_name + " " + employee.last_name))
* establecer título del botón como 'Llamada ' + nombre y apellido
frmForm1.cmdButton1.Caption = "Llamada " ;
    + ALLTRIM(employee.first_name + " " + employee.last_name)
```

También podría establecer el título mediante una función definida por el usuario:

```
frsSet1.frmForm1.cmdButton1.Caption = esttítulo()
```

## Establecer múltiples propiedades

Puede establecer múltiples propiedades de una sola vez.

### Para establecer múltiples propiedades

- Utilice la estructura [WITH ... ENDWITH](#).

Por ejemplo, para establecer múltiples propiedades de una columna en una [cuadrícula](#) de un [formulario](#), podría incluir la instrucción siguiente en el código de cualquier evento o método del formulario:

```
WITH THISFORM.grdGrid1.grcColumn1
.Width = 5
.Resizable = .F.
.ForeColor = RGB(0,0,0)
.BackColor = RGB(255,255,255)
.SelectOnEntry = .T.
ENDWITH
```

## Llamar a métodos en tiempo de ejecución

La [sintaxis](#) para llamar a los métodos de un objeto es la siguiente:

*Primario.Objeto.Método*

Cuando crea un [objeto](#), puede llamar a los [métodos](#) de ese objeto desde cualquier lugar de la aplicación. Los comandos siguientes llaman a métodos para mostrar un formulario y establecer el enfoque en un botón de comando:

```
* conjunto de formularios guardado en MYF_SET.SCX
myf_set.frmForm1.Show
myf_set.frmForm1.cmdButton1.SetFocus
```

Para ocultar el formulario, ejecute este comando:

```
myf_set.frmForm1.Hide
```

## Responder a eventos

El código incluido en un procedimiento de evento se ejecuta cuando se produce el [evento](#). Por ejemplo, el código que incluya en el procedimiento de evento Click de un botón de comando se ejecutará cuando el usuario haga clic en el botón de comando.

Una llamada al código de procedimiento asociado a un evento no hace que se produzca un evento. Por ejemplo, la instrucción siguiente hace que se ejecute el código del evento Activate de frmGuíaTel, pero no activa el formulario:

```
frmGuíaTel.Activate
```

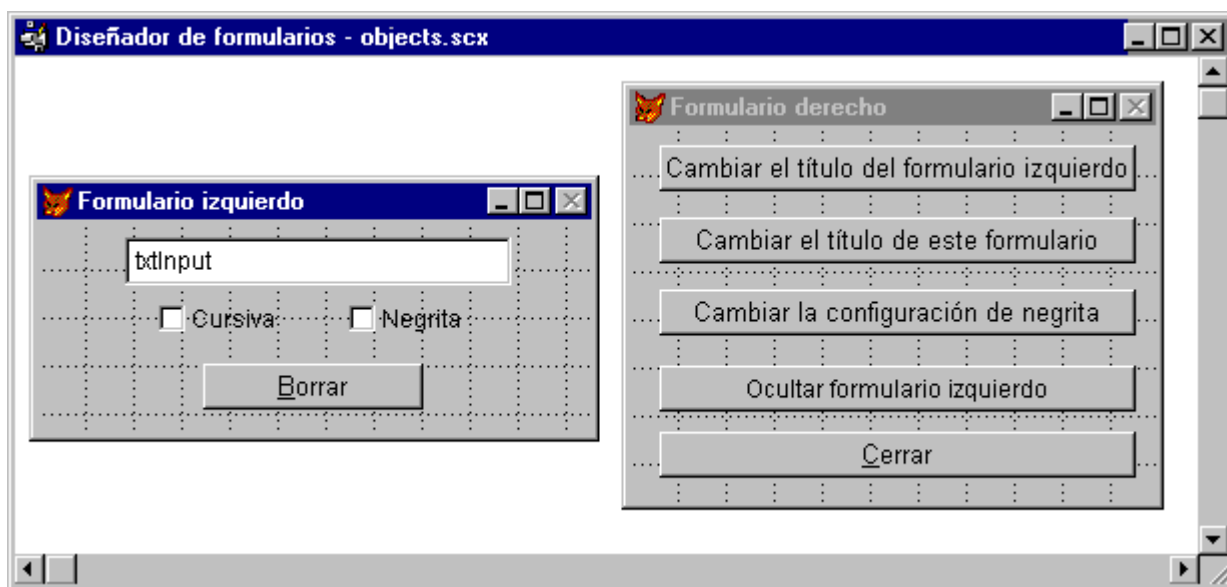
Al llamar al [método Show](#) de un formulario se mostrará y activará el formulario. En este momento se ejecutará el código del evento Activate:

```
frmGuíaTel.Show
```

## Ejemplo de manipulación de objetos

El ejemplo siguiente establece propiedades y llama a código de evento desde diversos objetos de un [conjunto de formularios](#). El ejemplo incluye dos [formularios](#), frmLeft y frmRight, en un conjunto de formularios.

### Conjunto de formularios de ejemplo en el Diseñador de formularios



Las dos casillas de verificación y el botón de comando de Form1 están asociados a código de evento. El nombre del cuadro de texto de frmRight es txtInput.

### Código de evento para objetos de LeftForm

Objeto	Evento	Código
chkItalic	Click	THISFORM.txtInput.FontItalic = ; THIS.Value
chkBold	Click	THIS.txtInput.FontBold = THIS.Value
cmdClear	Click	THISFORM.txtInput.Value = "" THISFORM.txtInput.FontBold = .F. THISFORM.txtInput.FontItalic = .F. THISFORM.chkItalic.Value = .F. THISFORM.chkBold.Value = .F.

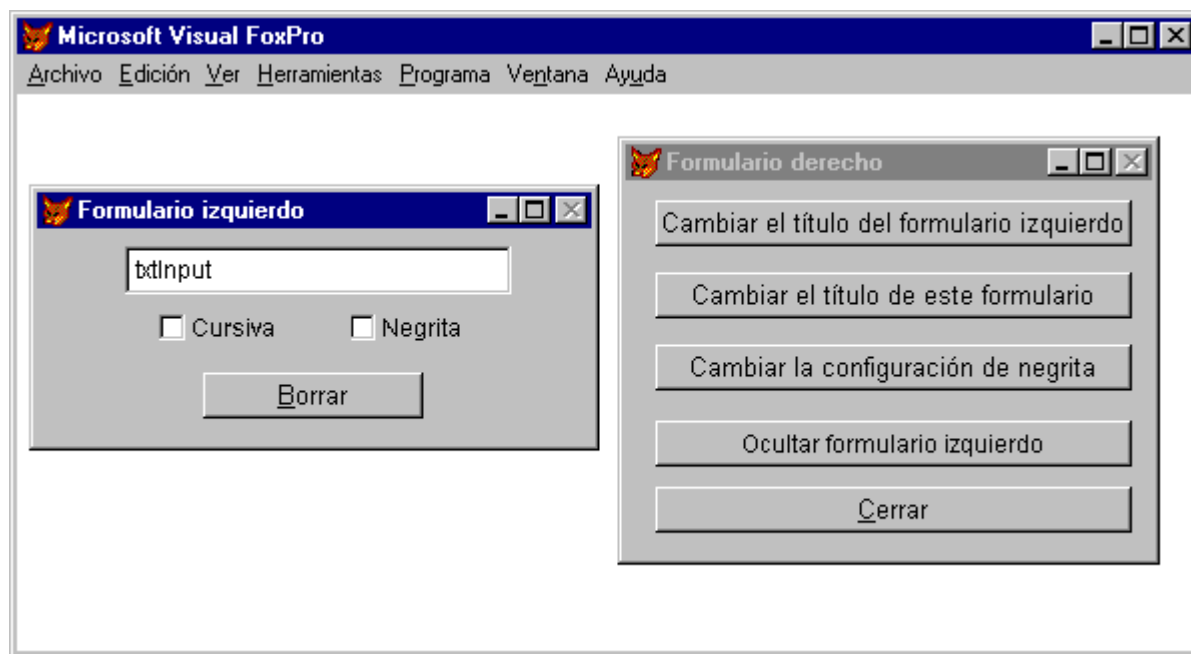
### Establecer una propiedad de otro control en el mismo formulario

Puede establecer las propiedades de un [control](#) desde el código de evento de otro con la palabra clave THISFORM o la propiedad [Parent](#). Los dos comandos siguientes se ejecutan cuando un usuario hace clic en las casillas de verificación Cursiva y Negrita, estableciendo las propiedades correspondientes del cuadro de texto:

```
THISFORM.txtInput.FontItalic = .T.  
THIS.Parent.txtInput.FontBold = .T.
```

En este caso, THISFORM y THIS.Parent pueden emplearse indistintamente.

### Conjunto de formularios de ejemplo en tiempo de ejecución



El código del evento Click para cmdClear utiliza THISFORM para restablecer los valores de los demás controles del formulario.

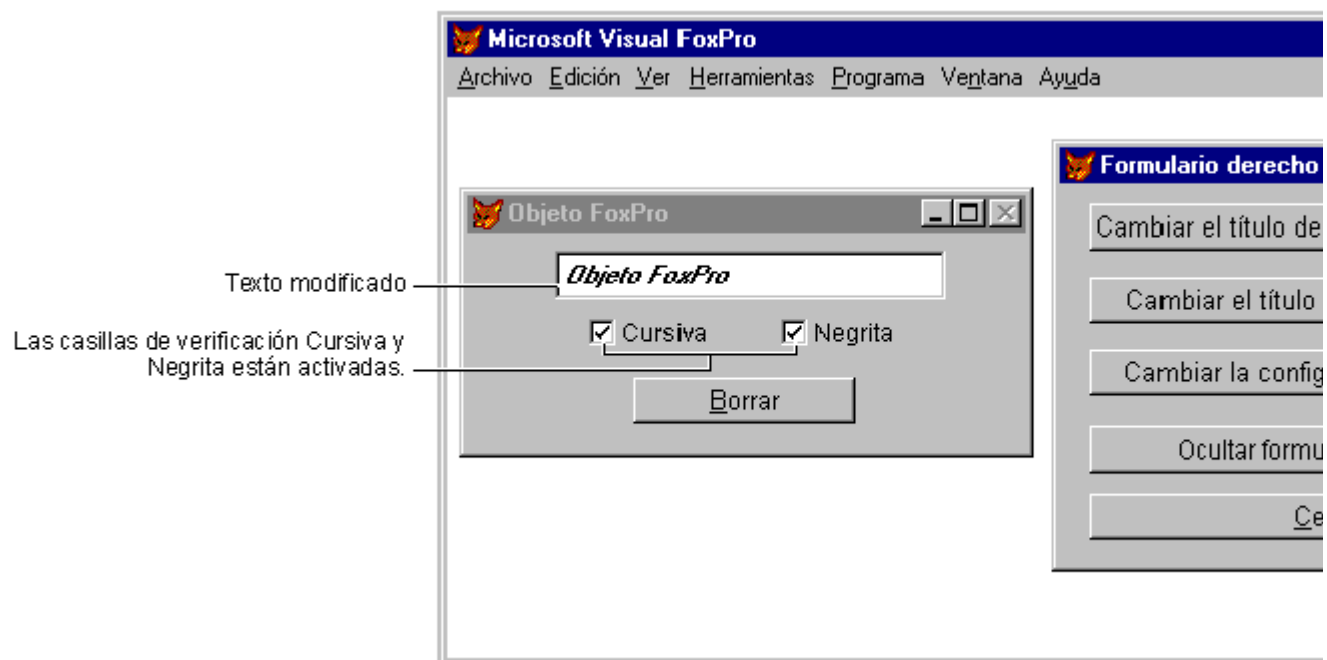
## Establecer las propiedades de otro formulario

También puede establecer [propiedades](#) de un [formulario](#) desde otro formulario distinto. Form2 contiene 5 botones de comando. El primer botón del formulario tiene este código en su evento [Click](#):

```
THISFORMSET.frmLeft.Caption = ;
    ALLTRIM(ThisFormSet.frmLeft.txtInput.Value)
```

Observe que debe hacer referencia al [conjunto de formularios](#) y al formulario cuando se establecen propiedades desde otro formulario distinto.

**El usuario hace clic en el botón de comando "Cambiar título del formulario de la izquierda" de frmRight**



El código de evento Click del segundo botón de comando de frmRight muestra el cómo se establece una propiedad de un formulario desde un objeto del formulario:

```
THISFORM.Caption = ;
    ALLTRIM(ThisFormSet.frmLeft.txtInput.Value)
```

Si el usuario elige este botón, el título de frmRight cambiará al valor del cuadro de texto de frmLeft.

## Acceso a objetos de formularios distintos

El código siguiente del evento [Click](#) del botón de comando Cambiar negrita cambia el valor de la casilla de verificación Negrita de Form1 y llama al código de evento asociado a este control.

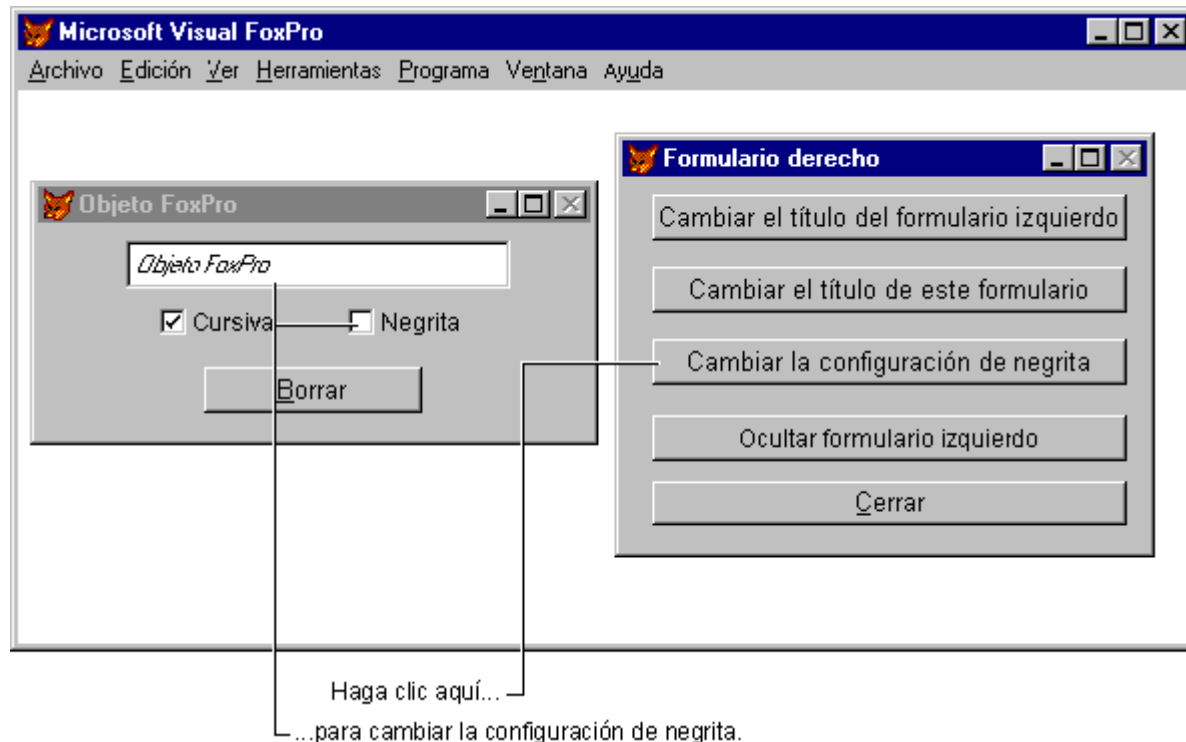
```
THISFORMSET.frmLeft.chkBold.Value = ;
    NOT THISFORMSET.frmLeft.chkBold.Value
THISFORMSET.frmLeft.chkBold.InteractiveChange
```

La última línea del ejemplo llama al evento [Interactive Change](#) de chkBold. También podría llamar a este procedimiento mediante el comando siguiente:

```
THISFORMSET.frmForm1.chkBold.InteractiveChange( )
```

Si se omite esta llamada al procedimiento, cambiará el valor de la casilla de verificación, pero no cambiará la propiedad [FontBold](#) del cuadro de texto.

**El usuario hace clic en el botón de comando Cambiar negrita del formulario de la derecha**



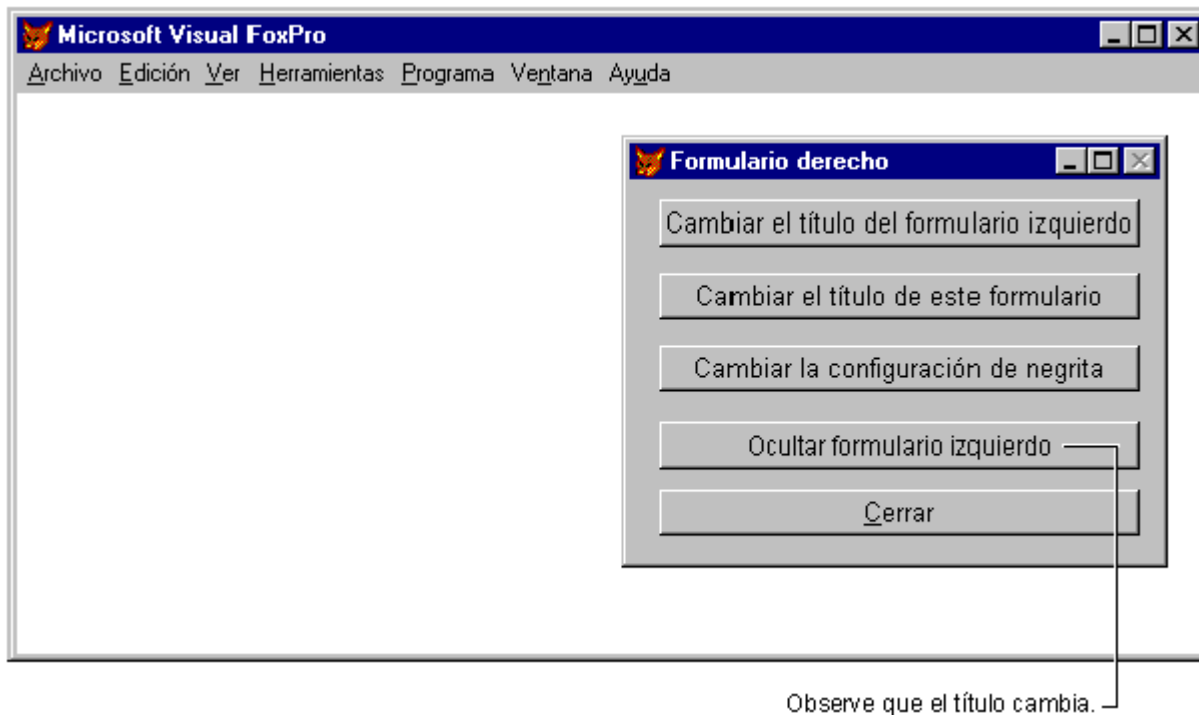
### Comprobar propiedades y llamar a código de método de otro formulario

El código siguiente del evento [Click](#) del botón de comando Ocultar formulario o Mostrar formulario, según el valor de la propiedad [Visible](#), y cambia el título del botón:

```
IF ThisFormSet.frmLeft.Visible
    ThisFormSet.frmLeft.Hide
    THIS.Caption = "Mostrar formulario"
ELSE
    ThisFormSet.frmLeft.Show
    THIS.Caption = "Ocultar formulario"
ENDIF
```

Observe que la palabra clave THIS se utiliza en el código de evento de un [control](#) para hacer referencia a propiedades del control.

**El usuario hace clic en el botón de comando Ocultar formulario izquierdo en el formulario de la derecha**



El comando siguiente del evento Click del botón de comando Salir libera el conjunto de formularios; esto hace que se cierren ambos formularios:

```
RELEASE ThisFormSet
```

## Administrar formularios

Los procedimientos siguientes describen tareas comunes relacionadas con la administración de formularios en una aplicación.

### Ocultar un formulario

Puede ocultar un formulario para que deje de estar visible para un usuario. Cuando el formulario está oculto, el usuario no puede tener acceso a sus controles, pero sigue teniendo control total sobre ellos mediante programación.

#### Para ocultar un formulario

- Utilice el método [Hide](#).

Por ejemplo, en el código asociado al evento [Click](#) de un botón de comando puede incluir la siguiente línea de código:

```
THISFORM.Hide
```

Cuando el usuario haga clic en el botón de comando, el formulario permanecerá en memoria, pero no será visible.

## Liberar un formulario

Puede permitir que un usuario libere un formulario cuando haya terminado de interactuar con él. Al liberar un formulario, ya no podrá tener acceso a sus propiedades y métodos.

### Para liberar un formulario

- Utilice el método [Release](#).

Por ejemplo, en el código asociado al evento [Click](#) de un botón de comando, podría incluir la línea de código siguiente:

```
THISFORM.Release
```

Cuando el usuario haga clic en el botón de comando, se cerrará el formulario.

## Transferir parámetros a un formulario

En algunos casos puede resultar conveniente transferir [parámetros](#) a formularios cuando se ejecutan para establecer valores de propiedad o para especificar valores operativos predeterminados.

### Para transferir un parámetro a un formulario creado en el Diseñador de formularios

1. Cree en el formulario propiedades para almacenar los parámetros, como ItemName e ItemQuantity.
2. En el código de evento [Init](#) para el formulario, incluya una instrucción [PARAMETERS](#) como ésta:

```
PARAMETERS cString, nNumber
```

3. En el código de evento Init para el formulario, asigne los parámetros a las propiedades, como en este ejemplo:

```
THIS.ItemName = cString  
THIS.ItemQuantity = nNumber
```

4. Cuando ejecute el formulario, incluya una cláusula WITH en el comando [DO FORM](#):

```
DO FORM miform WITH "Bagel", 24
```

## Devolver un valor desde un formulario

Puede utilizar formularios en su aplicación para permitir que los usuarios especifiquen un valor.

### Para devolver un valor desde un formulario

1. Establezca la propiedad [WindowType](#) del formulario como 1 para convertir el formulario en modal.



2. En el código asociado al evento Unload del formulario, incluya un comando [RETURN](#) con el valor devuelto.
3. En el programa o el método que ejecuta el formulario, incluya la palabra clave TO en el comando [DO FORM](#).

Por ejemplo, si `BuscarIDCliente` es un formulario modal que devuelve un valor de carácter, la línea de código siguiente almacenará el valor devuelto en una [variable](#) llamada `cIDCliente`:

```
DO FORM BuscarIDCliente TO cIDCliente
```

Para obtener más información al respecto, vea [RETURN](#) y [DO FORM](#).

**Solución de problemas** Si aparece un error, asegúrese de que `WindowType` está establecido a 1 (Modal).

## Guardar un formulario como HTML

Puede usar la opción **Guardar como HTML** del menú **Archivo** cuando cree un formulario para guardar el contenido de un formulario como un archivo HTML (Lenguaje de marcado de hipertexto).

### Para guardar un formulario como HTML

1. Abra el formulario.
2. Elija **Guardar como HTML** en el menú **Archivo**. (Se le pedirá que guarde el formulario si lo ha modificado.)
3. Escriba el nombre del archivo HTML que desea crear y elija **Guardar**.

## Administrar múltiples instancias de un formulario

Puede tener activas varias [instancias](#) de una definición de clase al mismo tiempo. Por ejemplo, puede diseñar un formulario de pedido pero tener varios pedidos abiertos en la aplicación, cada uno de los cuales emplea la misma definición de formulario, pero que se muestran y manipulan individualmente.

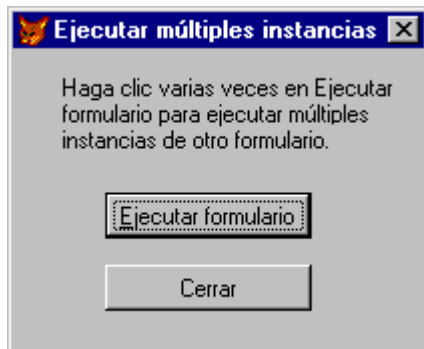
Si dispone de múltiples instancias de un formulario, los puntos clave que deberá tener en cuenta son los siguientes:

- Cree una propiedad de matriz en el formulario de inicio para almacenar las variables de objeto asociadas con cada instancia del formulario de múltiples instancias. El modo más sencillo de hacer un seguimiento de las variables de instancia cuando no se sabe con antelación cuántas va a tener consiste en utilizar una matriz.
- Para el formulario que va a tener múltiples instancias, establezca la propiedad [DataSession](#) como **2 – Sesión privada de datos**. Una sesión de datos privada proporciona un conjunto independiente de [áreas de trabajo](#) para cada instancia del formulario de forma que las tablas seleccionadas y las posiciones de puntero de registro sean todas independientes.

El ejemplo siguiente proporciona código que muestra cómo se crean múltiples instancias de un formulario. Para no hacerlo demasiado extenso, este código no está optimizado; está pensado únicamente para presentar los conceptos.

El formulario siguiente inicia múltiples instancias:

### Formulario Launch



### Valores de la propiedad para Launch.scx

Objeto	Propiedad	Valor
frmLaunch	aForms[1]	" "

### Código de evento para Launch.scx

Objeto	Evento	Código
cmdQuit	<a href="#">Click</a>	RELEASE THISFORM
cmdLaunch	Click	<pre>nInstance = ALLEN(THISFORM.aForms) DO FORM Multi ; NAME THISFORM.aForms[nInstance] ; LINKED DIMENSION ; THISFORM.aForms[nInstance + 1]</pre>

Para refinar el código de este ejemplo, podría administrar la [matriz](#) de objetos de formulario de modo que se cerraran los elementos vacíos de la matriz reutilizados como formularios y se abrieran nuevos formularios, en lugar de redimensionar siempre la matriz y aumentar en uno el número de elementos.

El formulario que puede tener múltiples instancias es Multi.scx. El entorno de datos para este formulario contiene la tabla Employee.

### Múltiples instancias de Multi.scx



### Establecer propiedad para Multi.scx

Objeto	Propiedad	Valor
txtFirstname	<a href="#">ControlSource</a>	Employee.first_name
txtLastName	<a href="#">ControlSource</a>	Employee.last_name
frmMulti	<a href="#">DataSession</a>	2 - Sesión predeterminada de datos

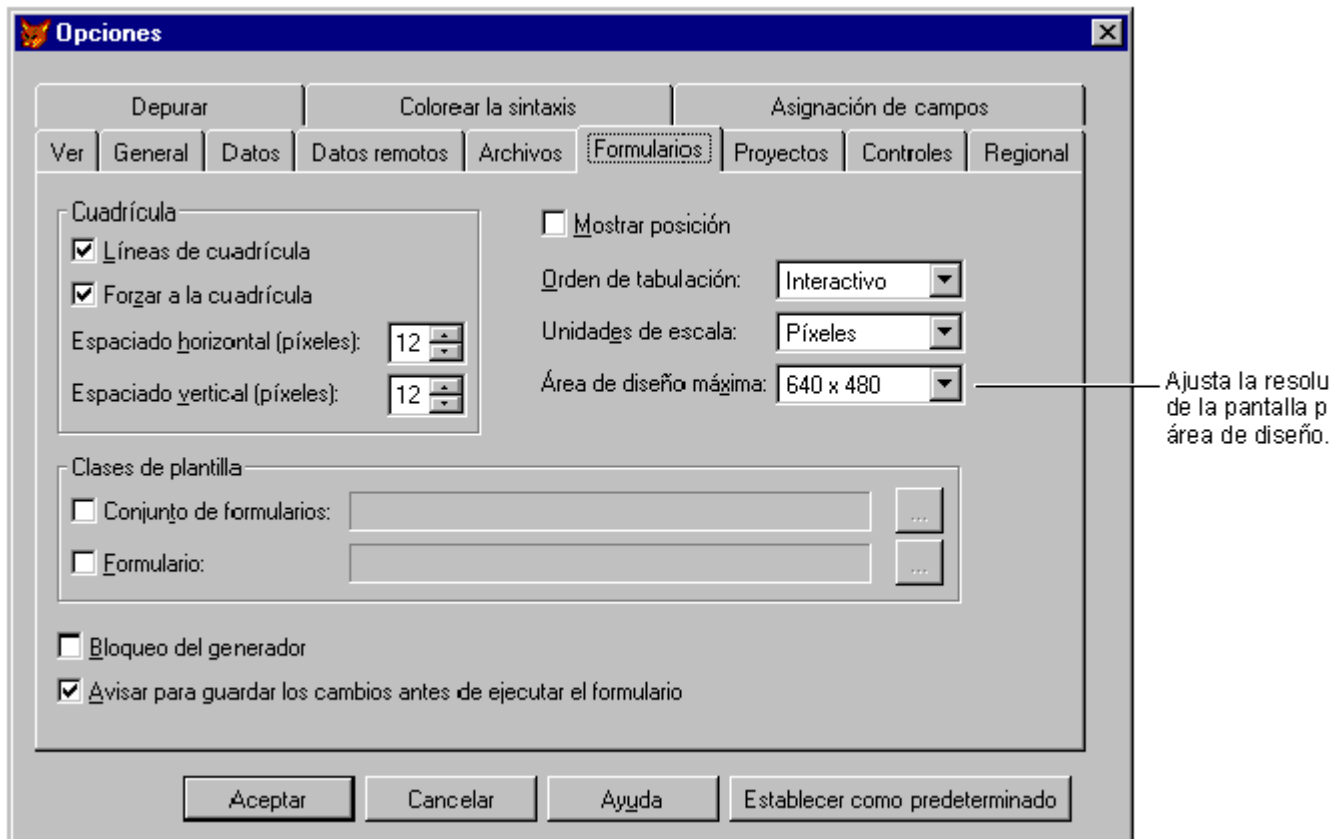
Al elegir Iniciar formulario en el formulario Launch, se crea una instancia del formulario Multi. Al cerrar el formulario Launch se libera la matriz de propiedades aForms y se destruyen todas las instancias de Multi.

Visual FoxPro proporciona algunas [funciones](#) y [propiedades](#) que le ayudan a administrar múltiples instancias de objetos. Para obtener información al respecto, vea [AINSTANCE\(\)](#), [AUSED\(\)](#) y [DataSessionID](#) en la *Referencia del lenguaje*.

### Establecer el área de diseño para un formulario

Puede establecer el área máxima de diseño para el [Diseñador de formularios](#) en el cuadro de diálogo Opciones.

### Ficha Formularios del cuadro de diálogo Opciones



### Para establecer el área de diseño máxima para un formulario

1. En el menú **Herramientas**, elija **Opciones**.
2. En el cuadro de diálogo [Opciones](#), elija la ficha **Formularios**.
3. En el cuadro **Área de diseño máxima**, elija las coordenadas en píxeles del área máxima de diseño.

Cuando establezca el área máxima de diseño, el fondo del Diseñador de formularios tendrá color blanco dentro de los límites del área de diseño, y gris fuera del área máxima de diseño. Por ejemplo, si desarrolla aplicaciones en un monitor con una resolución de 1024 x 768, podrá establecer la resolución de diseño en 640 x 480 y saber con seguridad que los formularios que diseñe siempre cabrán en pantallas de 640 x 480.

En el área de diseño, deje espacio para los atributos estándar de las ventanas, como las [barras de herramientas](#). Por ejemplo, en una pantalla de 640 x 480, un formulario con una barra de estado y una barra de herramientas acoplada en la parte superior o inferior de la pantalla puede tener una altura máxima de 390 píxeles.

Atributo de la ventana principal de Visual FoxPro	Píxeles necesarios
Título y menú	38
Barra de estado	23
Barra de herramientas acoplada	29

## Usar datos locales y remotos en un formulario

Puede crear formularios que pueden usar alternativamente [datos locales](#) y datos almacenados de forma remota (por ejemplo, en un servidor de base de datos). Esto le permite crear una aplicación prototipo con datos locales o de prueba, y después cambiar a datos [remotos](#) sin cambios importantes en sus formularios.

Por ejemplo, si su aplicación de Visual FoxPro es un cliente para una tabla grande de clientes almacenada en un servidor de base de datos, puede crear un archivo .dbf local que contiene una muestra pequeña pero representativa de datos. Entonces puede crear, probar y depurar sus formularios en base a este pequeño conjunto de datos. Cuando esté preparado para distribuir la aplicación, puede vincular el formulario al conjunto grande de datos.

La clave para poder cambiar entre datos remotos y locales es asegurarse de que usa [vistas](#) en lugar de vincular directamente el formulario (y sus controles) a una tabla. Para tener acceso a datos remotos, debe usar una vista en cualquier evento. Por lo tanto, para facilitar el cambio entre datos locales y remotos, cree también una vista para los datos remotos. Cuando cree el formulario, puede agregar ambas vistas a sus [entornos de datos](#) y, a continuación, cambie de un tipo a otro según sea necesario.

### Para crear un formulario que pueda cambiar entre datos locales y datos remotos

1. Cree dos [vistas](#) de los datos, una que apunte a los [datos remotos](#), y otra que apunte a los [datos locales](#).
2. Cree un nuevo formulario.
3. Abra el [Diseñador de entornos de datos](#) para el formulario y, a continuación, agregue ambas vistas.
4. Haga clic con el botón secundario en el **Diseñador de entornos de datos** y, a continuación, elija **Propiedades**.
5. En la [ventana Propiedades](#), establezca la propiedad [Alias](#) para ambos cursores al mismo nombre.
6. Establezca la propiedad [OpenViews](#) del entorno de datos a **1 – Sólo local** o **2 – Sólo remota**, en función de qué vista quiera usar al ejecutar el formulario.

**Nota** Como usa el mismo alias para ambas vistas, no elija **0 – Local y remota**

(predeterminada).

7. En el formulario, agregue los [controles](#) que necesite y establezca sus propiedades [ControlSource](#) a los campos apropiados de la vista. Como las dos vistas tienen el mismo [alias](#), los controles responderán automáticamente a la vista que esté activa cuando se ejecute el formulario.

Después de crear el formulario, puede cambiar los alias de vistas si cambia la propiedad OpenViews del entorno de datos. Puede hacerlo en el Entorno de datos mientras usa el [Diseñador de formularios](#). De forma alternativa, puede escribir código y adjuntarlo a un evento, lo cual es útil si quiere cambiar vistas en [tiempo de ejecución](#). Por ejemplo, puede colocar el código en el evento [Activate](#) del formulario:

```
THISFORM.DataEnvironment.OpenViews = 2 && Usar vista remota
```

Si crea un formulario que puede alternar entre datos locales y remotos, también debe diseñar el código de desplazamiento para acomodar ambas vistas, en particular si diseña formularios con [relaciones uno a varios](#). Por ejemplo, su formulario sólo tiene acceso a una tabla o vista local, puede usar código como el siguiente en un botón de comando Siguiente para mover al siguiente registro en un cursor:

```
SKIP 1  
THISFORM.Refresh()
```

Sin embargo, este código es ineficaz cuando se desplaza en una vista remota, porque supone que el [cursor](#) contiene todos los datos que requiere el formulario. Como regla general, debe minimizar la cantidad de datos que transfiere desde el origen de datos remoto.

La solución consiste en usar una vista parametrizada. Por ejemplo, la definición para una vista usada para modificar información de clientes podría ser:

```
SELECT * FROM CUSTOMERS WHERE ;  
CUSTOMERS.COMPANY_NAME = ?pCompanyName
```

Cuando el formulario se ejecuta, puede pedir al usuario un nombre de cliente mediante un cuadro de diálogo o permitiendo al usuario escribir un nombre en un cuadro de texto. El código para un botón Mostrar sería similar al siguiente:

```
pCompanyName = THISFORM.txtCompanyName.Value  
REQUERY("customer")  
THISFORM.Refresh()
```

Para obtener más información sobre vistas parametrizadas, consulte Crear una vista parametrizada en el capítulo 8, [Crear vistas](#).

## Establecer plantillas de formularios

Puede crear su propia clase de formulario con el fin de utilizar una plantilla para todos los formularios nuevos o bien puede utilizar una de las clases de ejemplo que se incluyen con Visual FoxPro.

Cuando cree un formulario nuevo, éste se basará en el formulario de [plantilla](#) que se establece en el cuadro de diálogo Opciones. Si no se especifica ninguna plantilla, el nuevo formulario se basará en la [clase de base](#) Form de Visual FoxPro. Para obtener más información sobre las clases de Visual FoxPro, consulte el capítulo 3, [Programación orientada a objetos](#).

### **Ventajas del uso de plantillas de formulario**

Las plantillas de formulario permiten establecer [propiedades](#) predeterminadas para los formularios, de forma que todos los formularios de la aplicación tengan una apariencia parecida y se utilicen de forma similar. Podría incluir un logotipo de su compañía, por ejemplo, y utilizar un mismo esquema de colores en todos los formularios diseñando una clase de formulario de plantilla con estos atributos. Si cambiara el logotipo de la compañía, podría cambiar la imagen que aparece en la clase de formulario de plantilla y todos los formularios creados con esa plantilla heredarían automáticamente el nuevo logotipo.

Puede agregar propiedades y [métodos](#) personalizados a la clase de formulario de Visual FoxPro de modo que estas propiedades y estos métodos estén disponibles para todos los formularios de la aplicación. Si normalmente crea [variables](#) y procedimientos definidos por el usuario cuyo alcance es un formulario, el uso de propiedades y métodos personalizados le proporcionará la funcionalidad necesaria, a la vez que le permitirá disponer de un modelo de encapsulado más limpio.

### **Especificar la plantilla predeterminada de formulario**

Puede especificar una clase de formulario procedente de una [biblioteca de clases](#) registrada para la plantilla de formulario.

### **Para especificar una plantilla predeterminada de formulario**

1. En el menú **Herramientas**, elija **Opciones**.
2. En el cuadro de diálogo [Opciones](#), elija la ficha **Formularios**.
3. En el área **Clases de plantilla**, active la casilla de verificación **Formulario**.

Si no se ha seleccionado ninguna plantilla de formulario, aparecerá el cuadro de diálogo **Plantilla de formulario** en el que podrá elegir una clase de formulario. Si ha seleccionado alguna plantilla de formulario, puede cambiarla si elige el botón con tres puntos y selecciona otra clase.

4. Elija **Establecer como predeterminado** si desea usar la plantilla en posteriores sesiones de Visual FoxPro.
5. Elija **Aceptar**.

### **Ficha Formularios del cuadro de diálogo Opciones**



### Usar plantillas de formularios

Puede especificar plantillas de [conjuntos de formularios](#) del mismo modo que las plantillas de formularios. Se permiten las combinaciones siguientes:

- Especificar plantillas de conjuntos de formularios y de formularios.

Al elegir Formulario en el cuadro de diálogo [Nuevo](#) (y en los restantes modos de crear un nuevo formulario) se creará automáticamente un conjunto de formularios basado en la clase de conjunto de formularios de plantilla. Cuando elija Agregar nuevo formulario en el menú [Formulario](#) del [Diseñador de formularios](#), se agregará al conjunto de formularios un formulario basado en la plantilla de formulario.

- Especificar sólo la plantilla de conjunto de formularios.

Al elegir Formulario en el cuadro de diálogo Nuevo (y en los restantes modos de crear un nuevo formulario) se creará automáticamente un conjunto de formularios basado en la clase de conjunto de formularios de plantilla. Cuando elija Agregar nuevo formulario en el menú Formulario del [Diseñador de formularios](#), se agregará al conjunto de formularios un formulario basado en la clase de base Form de Visual FoxPro.

- Especificar sólo la plantilla de formulario.

Al elegir Formulario en el cuadro de diálogo Nuevo (y en los restantes modos de crear un



nuevo formulario) se creará automáticamente un formulario basado en la clase de formulario de plantilla.

- No especificar plantillas.

Al elegir Formulario en el cuadro de diálogo Nuevo (y en los restantes modos de crear un nuevo formulario) se creará automáticamente un formulario basado en la clase de base Form de Visual FoxPro.

## Capítulo 10: Usar controles

Los controles son el medio fundamental de interacción de los usuarios. Para manipular sus datos y llevar a cabo tareas los usuarios escriben y hacen clic en los controles, y se desplazan por los controles de los [formularios](#) de su aplicación.

En este capítulo se tratan los temas siguientes:

- [Descripción de los controles y los datos](#)
- [Elegir el control adecuado para la tarea](#)
- [Simplificar el uso de los controles](#)
- [Ampliar formularios](#)

Para obtener información adicional, vea [Controles y objetos](#) en la *Referencia del lenguaje*.

### Descripción de los controles y los datos

Los formularios pueden tener dos tipos de controles: controles que dependen de datos de tablas y controles que no dependen de tablas y controles. Cuando los usuarios interactúan con controles dependientes, los valores que introducen o eligen se almacenan en el origen de datos, que puede ser un campo de tabla, un campo de cursor o una [variable](#). Un control se vincula a datos al establecer su propiedad [ControlSource](#) o, en el caso de las cuadrículas, su propiedad [RecordSource](#).

Si no establece la propiedad ControlSource de un control, el valor que introduzca o elija el usuario en el control sólo se almacenará como el valor de una propiedad; no se escribirá en disco ni se almacenará en memoria más allá de la vida del control.

#### Efecto de un valor de ControlSource sobre los controles

Control	Efecto
<a href="#">Casilla de verificación</a>	Si el ControlSource es un campo de una tabla, los valores lógicos verdadero (.T.) o falso (.F.), o numéricos 0, 1 ó 2 del campo ControlSource harán que la casilla de verificación se active, desactive o atenúe a medida que el puntero de registro se mueva por la tabla.
<a href="#">Columna</a>	Si el ControlSource es un campo de tabla, el usuario modificará

directamente el campo cuando modifique valores en la columna. Para vincular una cuadrícula completa a datos, establezca la propiedad RecordSource de la cuadrícula.

<a href="#">Cuadro de lista</a> o <a href="#">cuadro combinado</a>	Si el ControlSource es una variable de memoria, el valor que elija el usuario en la lista se almacenará en la variable de memoria. Si el ControlSource es un campo de una tabla, el valor se almacenará en el campo, en el puntero de registro. Si un elemento de la lista coincide con el valor del campo de la tabla, se seleccionará el elemento de la lista cuando el puntero de registro se desplace por la tabla.
<a href="#">Botón de opción</a>	<p>Si ControlSource es un campo numérico, 0 ó 1 se escribe en el campo, en función de si se elige el botón.</p> <p>Si ControlSource es un campo de caracteres, en el campo se escribirá (.T.) o (.F.), en función de si se elige el botón o no. Sin embargo, si el puntero de registro se mueve en la tabla, el valor del botón de opción no se actualizará para reflejar el valor de carácter del campo.</p> <p>Si el ControlSource del control OptionGroup (no del mismo botón de opción) es un campo de caracteres, el título del botón de opción se almacena en el campo si se elige el botón de opción. Observe que el origen de control para un botón de opción (a diferencia de un control OptionGroup) no puede ser un campo de caracteres o Visual FoxPro informará de <a href="#">tipo de datos</a> incorrecto cuando se ejecute el formulario.</p>
<a href="#">Control numérico</a>	El control numérico refleja y escribe valores numéricos en el campo o la variable subyacente.
<a href="#">Cuadro de texto</a> o <a href="#">cuadro de edición</a>	El valor del campo de la tabla se muestra en el cuadro de texto. Los cambios que realiza el usuario en este valor vuelven a escribirse en la tabla. Al mover el puntero de registro se verá afectada la <a href="#">propiedad Value</a> del cuadro de texto.

Algunas de las tareas que se llevan a cabo con controles, aunque no todas, exigen disponer de datos dependientes del control.

## Elegir el control apropiado para la tarea

Los controles de Visual FoxPro son flexibles y versátiles. Si bien hay múltiples controles que pueden emplearse para llevar a cabo una determinada tarea, necesitará emplear una estrategia coherente en relación con los controles que utiliza de modo que los usuarios sepan lo que pueden esperar cuando vean la interfaz que ha creado. Por ejemplo, una etiqueta tiene un evento [Click](#) del mismo modo que un botón de comando, pero los usuarios suelen hacer clic en los botones de comando para realizar acciones.

La mayor parte de la funcionalidad que querrá incorporar a sus formularios corresponderá a una de estas categorías:

- Proporcionar a los usuarios una serie de opciones predeterminadas

- Aceptar entradas del usuario que no se pueden determinar de forma previa
- Aceptar entradas del usuario en un determinado intervalo
- Permitir a los usuarios realizar acciones específicas
- Realizar determinadas acciones a intervalos específicos
- Mostrar información

## Proporcionar una serie de opciones predeterminadas

Uno de los modos más sencillos de asegurar la validez de los datos de una base de datos consiste en proporcionar a los usuarios una serie de opciones predeterminadas. Si controla las opciones del usuario puede asegurarse de que en la base de datos no se almacenan datos no válidos. Los controles siguientes permiten proporcionar a los usuarios una serie de opciones predeterminadas:

- Grupos de botones de opción
- Cuadros de lista y listas desplegables
- Casillas de verificación

### Usar grupos de botones de opción



Los grupos de botones de opción son contenedores que alojan botones de opción. Generalmente, los botones de opción permiten a los usuarios especificar una opción entre varias en un cuadro de diálogo, en lugar de introducir los datos. Los botones de opción también se pueden emplear para especificar la salida a un archivo, a una impresora o para realizar una vista previa, como se describe en el capítulo 12, [Agregar consultas e informes](#).

### Establecer el número de botones de opción en un grupo

Cuando crea un grupo de botones de opción en un formulario, se incluyen dos botones de opción de forma predeterminada. Puede determinar el número de botones de opción que hay en un grupo si cambia la propiedad `ButtonCount`.

### Para establecer el número de botones de opción de un grupo

- Establezca la propiedad [ButtonCount](#) con el número deseado de botones de opción.

Por ejemplo, para tener un grupo de seis botones de opción, establezca a 6 la propiedad `ButtonCount` del grupo de botones de opción.

La propiedad [Value](#) del grupo indica qué botón se ha elegido. Por ejemplo, si un usuario elige el cuarto botón de opción de un grupo de seis, el valor del grupo de botones de opción será 4.

Si la propiedad [ControlSource](#) del grupo es un campo de caracteres o si la propiedad `Value` se establece a un valor de tipo `Character` antes de que se ejecute el formulario, la propiedad `Value` del grupo es el título del botón de opción seleccionado.

### Establecer propiedades de botones de opción

Para ajustar manualmente los elementos individuales de un grupo de botones de opción o de botones de comando en el [Diseñador de formularios](#), elija Modificar en el menú de método abreviado del grupo.

Es posible establecer [propiedades](#) de botones individuales en la [ventana Propiedades](#). También puede establecer estas propiedades en tiempo de ejecución si especifica el nombre del botón de opción y el valor deseado de la propiedad. Por ejemplo, la línea de código siguiente establece el título de `optCust` en el grupo de botones de opción `opgChoices`:

```
THISFORM.opgChoices.optCust.Caption = "Ordenar por Customer"
```

También puede establecer estas propiedades en [tiempo de ejecución](#) mediante la propiedad [Buttons](#) y especificando el número de índice del botón de opción del grupo. Por ejemplo, si `optCust` es el tercer botón del grupo, la línea de código siguiente establecerá el título de `optCust`:

```
THISFORM.opgChoices.Buttons(3).Caption = "Ordenar por Customer"
```

### Para establecer propiedades en todos los botones de un grupo

- Utilice el método [SetAll](#) del grupo.

Por ejemplo, la línea de código siguiente deshabilita todos los botones de un grupo de botones de opción llamado `opgMyGroup` en un formulario:

```
THISFORM.opgMyGroup.SetAll("Enabled", .F., "OptionButton")
```

### Activar y desactivar botones de un grupo

El ejemplo anterior muestra cómo desactivar mediante programación todos los botones de opción de un grupo. Cuando los botones están desactivados, se muestran en los colores especificados en las propiedades [DisabledForeColor](#) y [DisabledBackColor](#) de los botones de opción. También puede establecer la propiedad [Enabled](#) del grupo de botones de opción como falso (.F.) para desactivar el grupo; sin embargo, no habría ninguna pista visual para el usuario.

### Determinar el botón de opción seleccionado actualmente

La propiedad [Value](#) del [grupo de botones de opción](#) permite determinar el botón de opción que está seleccionado en el grupo. Si el [origen de control](#) para el botón es numérico, tiene cinco botones de opción en un grupo. Si hay cinco botones de opción en un grupo y el tercero está seleccionado, la propiedad `Value` del grupo de botones de opción será 3. Si ningún botón de opción está seleccionado, la propiedad `Value` del grupo será 0.

También puede determinar el título del botón de opción seleccionado con las propiedades [Value](#) y [Buttons](#) del grupo. Por ejemplo, la línea de código siguiente almacena en una variable `cSelected` la propiedad [Caption](#) del botón de opción seleccionado.

```
oGroup = THISFORM.opg1  
cSelected = oGroup.Buttons(oGroup.Value).Caption
```

## Filtrar listas con botones de opción

Si tiene un pequeño grupo de [filtros](#) de tabla predeterminados, puede usar botones para permitir al usuario cambiar entre los filtros.

El ejemplo siguiente utiliza un formulario con un cuadro de lista (`lstCustomers`) y un grupo de botones de opción que contiene tres botones de opción.

### Valores de las propiedades del cuadro de lista

Objeto	Propiedad	Valor
LstCustomers	<a href="#">RowSourceType</a>	2 - Alias
LstCustomers	<a href="#">RowSource</a>	Customer

Los filtros se establecen en el código del evento [Click](#) de los botones de opción.

### Código de evento para filtrar una lista cuando el usuario elige un botón de opción

Objeto	Evento	Código
OptAll	Click	<pre>SET FILTER TO GO TOP THISFORM.lstCustomers.Requery</pre>
OptCanada	Click	<pre>SET FILTER TO customer.country = "Canadá" GO TOP THISFORM.lstCustomers.Requery</pre>
OptUK	Click	<pre>SET FILTER TO customer.country = "Reino Unido" GO TOP THISFORM.lstCustomers.Requery</pre>

Cuando el usuario cierre el formulario, no olvide restablecer el filtro incluyendo SET FILTER TO en el evento Click del botón de cierre o en el evento [Destroy](#).

**Sugerencia** Para actualizar una lista cuando el origen de la lista puede haber cambiado, utilice el método [Requery](#).

### Usar botones de opción para almacenar opciones de usuario en una tabla

Aunque no es tan común, puede utilizar botones de opción para obtener información de un usuario y almacenarla en una tabla guardando la propiedad [Caption](#). Por ejemplo, si dispone de una aplicación de prueba normalizada, puede utilizar los botones de opción para permitir que el usuario elija entre múltiples opciones A, B, C o D. También puede utilizar botones de opción para indicar el género en una tabla de empleados.

### Para almacenar en una tabla la propiedad Caption de un botón de opción

1. Establezca la propiedad [Value](#) del grupo de botones de opción como una cadena vacía.
2. Establezca la propiedad [ControlSource](#) del grupo de botones de opción como un campo de tipo Character de una tabla.

Por ejemplo, si los títulos de los botones de opción de un grupo son "A", "B", "C" y "D", y el ControlSource del grupo de botones de opción es un campo de caracteres, cuando el usuario elija el botón con el título "B", en el campo se almacenará "B".

### Para ver un ejemplo de una prueba de elección múltiple con botones de opción

1. Ejecute Solution.app en el directorio Visual Studio ...\\Samples\\Vfp98\\Solution.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Botones de opción**.
3. Haga clic en **Presentar múltiples opciones a un usuario**.

### Usar cuadros de lista y cuadros de lista desplegable



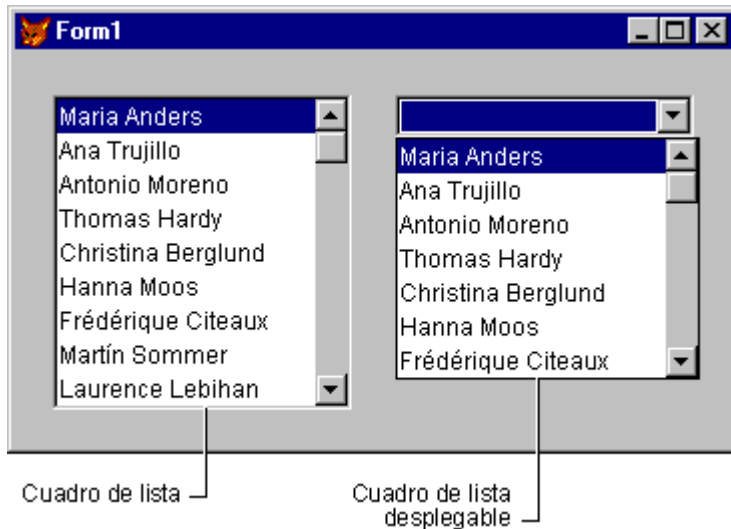
Los cuadros de lista y los cuadros de lista desplegable (controles de tipo cuadro combinado con la propiedad Style establecida como 2–Lista desplegable) proporcionan al usuario una lista por la que puede desplazarse y que contiene una serie de opciones o información. En un cuadro de lista, puede haber múltiples elementos visibles en todo momento. En un cuadro de lista desplegable sólo se ve un elemento, aunque el usuario puede hacer clic en el botón de flecha abajo para mostrar una lista desplegable con todos los elementos del cuadro de lista desplegable.



Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio para ver varios ejemplos que muestran el uso de cuadros de lista y cuadros de lista desplegable, como los siguientes:

- Agregar imágenes a una lista.
- Seleccionar varios elementos de una lista.
- Llenar a una lista con valores de orígenes distintos.
- Mostrar múltiples columnas en una lista.
- Ordenar elementos de una lista.
- Mover elementos entre listas.

### Cuadro de lista y cuadro de lista desplegable con los mismos valores para la propiedad RowSource



**Sugerencia** Cuadro de lista y cuadro de lista desplegable con los mismos valores para la propiedad RowSource.

### Propiedades y métodos comunes de las listas

Las propiedades siguientes de cuadros de lista suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">ColumnCount</a>	Indica el número de columnas del cuadro de lista.
<a href="#">ControlSource</a>	Indica dónde se almacena el valor que elige un usuario en la lista.
<a href="#">MoverBars</a>	Indica si las barras de movimiento se muestran a la izquierda de los elementos de la lista de modo que el usuario pueda reorganizar fácilmente los elementos de la lista.
<a href="#">Multiselect</a>	Indica si el usuario puede seleccionar o no más de un elemento de la lista al mismo tiempo.
<a href="#">RowSource</a>	Indica de dónde provienen los valores que se muestran en la lista.
<a href="#">RowSourceType</a>	Indica si RowSource es un valor, una tabla, una instrucción SQL, una consulta, una matriz, una lista de archivos o una lista de campos.

**Nota** La propiedad [Value](#) de una lista puede ser numérica o de caracteres. El valor predeterminado es numérico. Establezca la propiedad Value como una cadena vacía si RowSource es un valor de tipo Character y desea que la propiedad Value refleje la cadena de caracteres del elemento seleccionado en la lista. Puede presionar la barra espaciadora y, a continuación, la tecla retroceso para introducir una cadena vacía para una propiedad en la [ventana Propiedades](#).

Los siguientes métodos de cuadro de lista suelen utilizarse con frecuencia:

Método	Descripción
<a href="#">AddItem</a>	Agrega un elemento a una lista con un RowSourceType de 0.
<a href="#">RemoveItem</a>	Quita un elemento de una lista con un RowSourceType de 0.
<a href="#">Requery</a>	Actualiza la lista si han cambiado los valores de RowSource.

### Rellenar un cuadro de lista o un cuadro combinado

Puede rellenar un cuadro de lista con elementos procedentes de diversos orígenes si establece las propiedades [RowSourceType](#) y [RowSource](#).

### Elegir un tipo de datos para un cuadro de lista o un cuadro combinado

La propiedad [RowSourceType](#) determina qué tipo de origen rellena el cuadro de lista o el cuadro combinado, como una [matriz](#) o una [tabla](#). Una vez establecida la propiedad RowSourceType, especifique el origen de los elementos de la lista; para ello, establezca la propiedad [RowSource](#).

RowSourceType	Origen de los elementos de lista
0	Ninguno. Agrega elementos a la lista mediante programación.
1	Valor
2	Alias
3	Instrucción SQL
4	Consulta (.qpr)
5	Matriz
6	Campos
7	Archivos
8	Estructura
9	Emergente. Se incluye por compatibilidad con versiones anteriores.

Las secciones siguientes describen los distintos valores de RowSourceType.

**Nota** Si establece la propiedad RowSourceType como 0, el valor predeterminado, la lista no se rellenará automáticamente. Puede agregar elementos a la lista mediante el método [AddItem](#):

```
frmForm1.lstMyList.RowSourceType = 0
frmForm1.lstMyList.AddItem("Primer elemento")
frmForm1.lstMyList.AddItem("Segundo elemento")
frmForm1.lstMyList.AddItem("Tercer elemento")
```

El método [RemoveItem](#) permite quitar elementos de la lista. Por ejemplo, la línea de código siguiente



quita "Segundo elemento" de la lista:

```
frmForm1.lstMyList.RemoveItem(2)
```

**Valor** Si establece la propiedad RowSourceType a 1, podrá especificar múltiples valores en la propiedad RowSource que se mostrarán en la lista. Si establece la propiedad RowSource a través de la [ventana Propiedades](#), incluya una lista de elementos delimitados por comas. Si establece RowSource mediante programación, incluya entre comillas la lista delimitada por comas:

```
Form1.lstMyList.RowSourceType = 1  
Form1.lstMyList.RowSource = "uno,dos,tres,cuatro"
```

**Alias** Si establece la propiedad RowSourceType a 2, podrá incluir valores de uno o más campos de una tabla abierta.

Si la propiedad [ColumnCount](#) es 0 ó 1, la lista mostrará valores del primer campo de la tabla. Si establece la propiedad ColumnCount como 3, la lista mostrará valores de los tres primeros campos de la tabla. Para mostrar campos en un orden distinto del que tienen en la tabla, establezca la propiedad RowSourceType como 3 –Instrucción SQL o 6 – Campos.

**Nota** Cuando RowSourceType es 2 – Tabla o 6 – Campos, el puntero de registro de la tabla se mueve al registro que contiene el valor del elemento elegido por el usuario.

**Instrucción SQL** Si establece la propiedad RowSourceType a 3 – Instrucción SQL, incluya una instrucción [SELECT - SQL](#) en la propiedad RowSource. Por ejemplo, la instrucción siguiente selecciona todos los campos y todos los registros de la tabla customer en un [cursor](#):

```
SELECT * FROM Customer INTO CURSOR mylist
```

Si establece RowSource mediante programación, no olvide incluir la instrucción SELECT entre comillas.

**Nota** De forma predeterminada, las instrucciones SELECT de Visual FoxPro sin cláusulas INTO muestran inmediatamente el cursor resultante en una [ventana Examinar](#). Puesto que este comportamiento no suele ser conveniente en una instrucción SQL de RowSource, incluya una cláusula INTO CURSOR en la instrucción SELECT.

**Consulta** Si establece la propiedad RowSourceType a 4, podrá rellenar el cuadro de lista con los resultados de una [consulta](#) diseñada en el [Diseñador de consultas](#). Cuando RowSourceType está establecida a 4, establezca RowSource en el archivo .qpr. Por ejemplo, la línea de código siguiente establece la propiedad RowSource de una lista en una consulta.

```
THISFORM.List1.RowSource = "region.qpr"
```

Si no especifica una extensión de archivo, Visual FoxPro adoptará la extensión .qpr.

**Matriz** Si establece la propiedad RowSourceType en 5, la lista se rellenará con los elementos de una [matriz](#). Puede crear una propiedad de matriz del [formulario](#) o el [conjunto de formularios](#) para RowSource o bien utilizar una matriz creada en otra parte de la aplicación.

Para obtener información sobre la creación de propiedades de matrices, consulte el capítulo 9, [Crear formularios](#).

**Solución de problemas** Cuando es necesario, Visual FoxPro evalúa el valor de RowSource de una lista en la aplicación, no sólo en el [método](#) en que se establece la propiedad RowSource. Debe tener en cuenta este alcance. Si crea una matriz local en un método, esa matriz tendrá como alcance el método y no estará disponible en todos los casos en que Visual FoxPro necesite evaluar el valor de la propiedad. Si establece la propiedad RowSource de una lista como una propiedad de matriz del formulario o el conjunto de formularios, deberá hacer referencia a la propiedad en relación a la lista, no en relación al método en el que se ha establecido la propiedad. Por ejemplo, si tiene una propiedad de matriz de formulario llamada `arrayprop`, las líneas de código siguientes del Init del formulario producirán resultados distintos:

```
THIS.lst1.RowSource = "THIS.arrayprop"      && Error  
THIS.lst1.RowSource = "THISFORM.arrayprop"  && No hay error.
```

### Para rellenar una lista con los elementos de una matriz multidimensional

1. Establezca la propiedad [RowSourceType](#) a 5.
2. Establezca la propiedad [RowSource](#) a la matriz multidimensional.
3. Establezca la propiedad [ColumnCount](#) al número de columnas que desea mostrar.
4. Establezca la propiedad [ColumnWidths](#) a los anchos deseados para cada columna.

**Campos** Si establece la propiedad RowSourceType en 6, podrá especificar un campo o una lista de campos delimitados por comas para rellenar la lista, como:

```
contact,company,country
```

Puede incluir los siguientes tipos de información en la propiedad RowSource de una lista cuya propiedad RowSourceType tenga el valor 6—Campos:

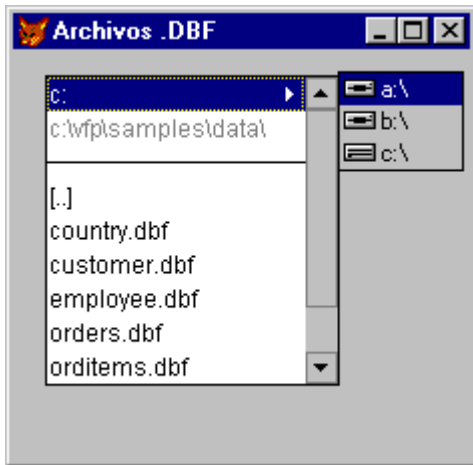
- campo
- alias.campo
- alias.campo, campo, campo, ...

Si en la lista desea incluir campos de múltiples tablas, establezca la propiedad RowSourceType como 3—Instrucción SQL.

A diferencia de un valor RowSourceType de 2—Tabla, un valor RowSourceType de 6—Campos permite mostrar campos independientemente de sus posiciones reales en la tabla.

**Archivos** Si establece la propiedad RowSourceType en 7, la lista se llenará con los archivos del directorio actual. Además, las opciones de la lista permiten elegir una unidad y un directorio distintos para los nombres de archivos que se muestran en la lista.

### Lista rellena con archivos de un directorio



Establezca RowSource como la estructura del tipo de archivos que desea mostrar en la lista. Por ejemplo, para mostrar tablas de Visual FoxPro en la lista, establezca la propiedad RowSource como **\*.dbf**.

**Estructura** Si establece la propiedad RowSourceType a 8, la lista se rellenará con los campos de la tabla que especifique al establecer la propiedad RowSource. Este valor de RowSourceType resulta útil para presentar al usuario una lista de campos en la que buscar valores o una lista de campos por la que ordenar una tabla.

**Emergente** Si establece la propiedad RowSourceType a 9, podrá rellenar una lista a partir de un emergente definido anteriormente. Esta opción se incluye por compatibilidad con versiones anteriores.

### Crear cuadros de lista de múltiples columnas

En Visual FoxPro un cuadro de lista puede contener tantas columnas como desee, aunque su número predeterminado es uno. Un cuadro de lista multicolumna se diferencia de una [cuadrícula](#) en que en el primero se selecciona una fila cada vez, mientras que en el segundo pueden seleccionarse celdas individuales de una cuadrícula y los datos de la lista no se pueden modificar directamente.

### Para mostrar múltiples columnas en un cuadro de lista

1. Establezca la propiedad [ColumnCount](#) como el número de columnas deseadas.
2. Establezca la propiedad [ColumnWidths](#). Por ejemplo, si hay tres columnas en el cuadro de lista, el comando siguiente establecería los anchos de columna a 10, 15 y 30, respectivamente:

```
THISFORM.listbox.ColumnWidths = "10, 15, 30"
```

3. Establezca la propiedad [RowSourceType](#) como **6 - Campos**.
4. Establezca la propiedad [RowSource](#) como los campos que se van a mostrar en columnas. Por ejemplo, el comando siguiente establece los orígenes de tres columnas en un cuadro de lista de tres columnas como los campos contact, city y country de la tabla customer:

```
form.listBox.RowSource = "contact,city,country"
```

**Nota** Para que las columnas se alineen correctamente, deberá establecer la propiedad `ColumnWidths` o cambiar la propiedad [FontName](#) a una fuente de espacio simple.

Cuando la propiedad `RowSourceType` de la lista se establece a 0 - Ninguno, puede usar el método `AddListItem` para agregar elementos a un cuadro de lista de múltiples columnas. Por ejemplo, el código siguiente agrega texto para especificar columnas en un cuadro de lista:

```
THISFORM.lst1.ColumnCount = 3
THISFORM.lst1.Columnwidths = "100,100,100"
THISFORM.lst1.AddListItem("fila1 col1", 1,1)
THISFORM.lst1.AddListItem("fila1 col2", 1,2)
THISFORM.lst1.AddListItem("fila1 col3", 1,3)
THISFORM.lst1.AddListItem("fila2 col2", 2,2)
```

### Permitir a los usuarios seleccionar múltiples elementos en un cuadro de lista

El comportamiento predeterminado de una lista permite seleccionar un elemento cada vez. Sin embargo, puede permitir que un usuario seleccione múltiples elementos de una lista.

#### Para permitir múltiples elementos seleccionados en una lista

- Establezca la propiedad [MultiSelect](#) de la lista como verdadera (.T.).

Para procesar los elementos seleccionados (para copiarlos a una [matriz](#) o incorporarlos en cualquier lugar de la aplicación), efectúe un bucle a través de los elementos de la lista y procese aquéllos para los que la propiedad [Selected](#) sea verdadera (.T.). Podría incluir el código siguiente en el [evento InteractiveChange](#) de un cuadro de lista para mostrar los elementos seleccionados en un cuadro combinado `cboSelected`, y el número de elementos seleccionados en un cuadro de texto, `txtNoSelected`:

```
nNumberSelected = 0  && variable para hacer un seguimiento del número
THISFORM.cboSelected.Clear  && borrar el cuadro combinado
FOR nCnt = 1 TO THIS.ListCount
  IF THIS.Selected(nCnt)
    nNumberSelected = nNumberSelected + 1
    THISFORM.cboSelected.Additem (THIS.List(nCnt))
  ENDIF
ENDFOR
THISFORM.txtNoSelected.Value = nNumberSelected
```

### Permitir a los usuarios agregar elementos a un cuadro de lista

Además de permitir a los usuarios seleccionar elementos de un cuadro de lista, puede permitir a los usuarios agregar elementos de forma interactiva a una lista.

#### Para agregar elementos a una lista de forma interactiva

- Utilice el método [AddItem](#).

En el ejemplo siguiente, el código del evento KeyPress de un cuadro de texto agrega el texto del cuadro de texto al cuadro de lista y borra el texto del cuadro cuando el usuario presiona entrar:

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF nKeyCode = 13      && Introducir tecla
    THISFORM.lstAdd.AddItem(This.Value)
    THIS.Value = ""
ENDIF
```

### Permitir a los usuarios introducir datos en una tabla desde una lista

Si la propiedad [ControlSource](#) está establecida como un campo, lo que seleccione el usuario en la lista se escribirá en la tabla. Éste es un modo sencillo de asegurar la integridad de los datos de la tabla. Aunque el usuario puede introducir datos erróneos, no podrá introducir valores no válidos.

Por ejemplo, si tiene una lista de regiones o países para que elija un usuario, el usuario no podrá introducir una abreviatura no válida de país o región.

### Permitir a los usuarios ir a un registro seleccionando un valor en una lista

A menudo resultará útil dejar que los usuarios seleccionen el [registro](#) que quieren ver o modificar. Por ejemplo, puede proporcionar a los usuarios una lista de nombres de clientes. Cuando el usuario seleccione un cliente de la lista, se seleccionará el registro del cliente en la tabla y se mostrará la información de ese cliente en los cuadros de texto del formulario. Esto puede llevarse a cabo de varias formas, según el origen de datos del formulario.

RowSourceType	Selección del registro adecuado
2 - Alias 6 - Campos	Cuando el usuario elige un valor de la lista, el puntero de registro se establece automáticamente en el registro deseado. Ejecute THISFORM.Refresh en el código de evento InteractiveChange de la lista para mostrar los nuevos valores de otros controles del formulario.
0 - Ninguno 1 - Valor 3 - Instrucción SQL 4 - QPR 5 - Matriz	En el código de evento InteractiveChange, seleccione la tabla que tiene el registro con los valores deseados y, a continuación, busque el valor deseado. Por ejemplo, si RowSource almacena números de identificación de cliente de la tabla de clientes, utilice este código:  <pre>SELECT customer LOCATE FOR THIS.Value = cust_id THISFORM.Refresh</pre>

### Actualizar una presentación uno a varios según un valor de lista

Cuando el usuario decide ir a un registro eligiendo un valor de una lista, puede tener una relación de [uno a varios](#) que necesite reflejar el puntero de registro modificado en la [tabla primaria](#). Puede implementar esta funcionalidad con tablas locales y vistas [locales](#) o [remotas](#).

### Tablas locales

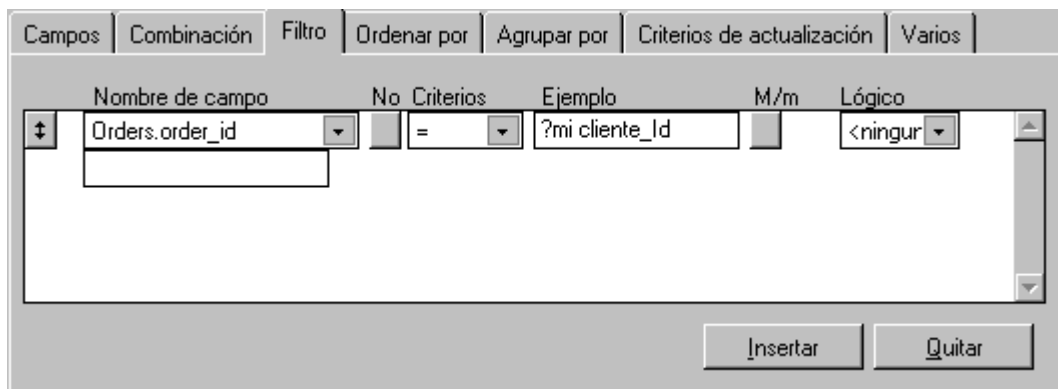
Si la propiedad RowSourceType de la lista es 2–Tabla o 6–Campos y la propiedad RowSource es una tabla local con una relación establecida en el [entorno de datos](#) del formulario, ejecute `THISFORM.Refresh` cuando el usuario elija un nuevo valor. La parte varios de una [relación uno a varios](#) muestra de forma automática únicamente aquellos registros que coinciden con la expresión de la tabla primaria que participa en la [relación](#).

## Vistas

La actualización de una presentación uno a varios es ligeramente diferente si la propiedad RowSource del cuadro de lista es una vista [local](#) o [remota](#). El siguiente ejemplo describe la creación de un formulario con un [cuadro de lista](#) y una [cuadrícula](#). El cuadro de lista muestra los valores del campo `cust_id` en la tabla `TESTDATA!Customer`. La cuadrícula muestra los pedidos asociados al campo `cust_id` seleccionado en el cuadro de lista.

En primer lugar, en el [Diseñador de vistas](#) cree una vista parametrizada para los pedidos. Cuando cree la vista en el Diseñador de vistas, establezca el criterio de selección para la [clave principal](#) a una [variable](#). En el ejemplo siguiente, la variable se llama `m.cCust_id`.

### Vista parametrizada con una variable



A continuación, cuando diseñe el formulario, siga los pasos del procedimiento siguiente. Observe que la vista requiere un valor para el [parámetro](#) que no está disponible cuando se carga el formulario. Si establece la propiedad [NoDataOnLoad](#) del objeto [cursor](#) de la vista como verdadero (.T.), impide que la vista se ejecute hasta que se llame a la función [REQUERY\(\)](#), momento en que el usuario habría seleccionado un valor para la variable utilizada en la vista parametrizada.

### Para diseñar una lista de uno a varios basada en vistas locales o remotas

1. Agregue la tabla y la vista parametrizada al [entorno de datos](#).
2. En la ventana [Propiedades](#) para el objeto cursor de vista del **Entorno de datos**, establezca la propiedad `NoDataOnLoad` como verdadero (.T.).
3. Establezca la propiedad `RowSourceType` del cuadro de lista como **6 - Campos** y establezca su propiedad `RowSource` al campo al que se hace referencia como clave externa en el parámetro de la vista.

En el ejemplo, establecería la propiedad `RowSource` a `customer.cust_id`.

4. Establezca la propiedad `RecordSource` de la cuadrícula al nombre de la vista que ha creado antes.
5. En el código de evento `InteractiveChange` del cuadro de lista, almacene el valor del cuadro de lista en la variable `y`, a continuación, vuelva a consultar la vista, como en este ejemplo:

```
m.cCust_id = THIS.Value
*suponemos que el nombre de la vista es orders_view
=REQUERY("orders_view")
```

Para obtener más información sobre vistas locales y remotas, consulte el capítulo 8, [Crear vistas](#).

### Mostrar registros secundarios en una lista

Se pueden mostrar registros de una [relación uno a varios](#) en una lista de modo que la lista muestre los registros secundarios de la relación a medida que el puntero de registro se mueve a través de la [tabla primaria](#).

#### Para mostrar registros secundarios en una lista

1. Agregue una lista al formulario.
2. Establezca la propiedad [ColumnCount](#) de la lista como el número de columnas que desea mostrar.

Por ejemplo, si desea mostrar los campos `Order_id`, `Order_net`, y `Shipped_on` en la lista, establezca la propiedad `ColumnCount` a 3.

3. Establezca la propiedad [ColumnWidths](#) a los anchos apropiados para mostrar los campos seleccionados.
4. Establezca la propiedad [RowSourceType](#) de la lista a **3 – Instrucción SQL**.
5. Establezca la propiedad [RowSource](#) a la instrucción [SELECT](#). Por ejemplo, la instrucción siguiente selecciona tres campos de la tabla `orders` para el registro actual en la tabla `customer`:

```
SELECT order_id, order_net, shipped_on from orders ;
WHERE order.cust_id = customer.cust_id ;
INTO CURSOR temp
```

6. En el [evento Init](#) del formulario y en el código que mueve el puntero de registro a través de la tabla, vuelva a consultar la lista:

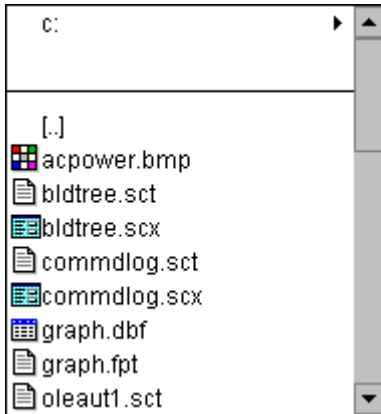
```
THISFORM.lstChild.Requery
```

### Agregar imágenes a elementos de una lista

Puede establecer la [propiedad Picture](#) de la lista como el archivo.bmp que desea mostrar junto a los elementos de la lista.

Por ejemplo, podría tener un cuadro de lista lleno de archivos, con mapas de bits distintos junto a cada archivo según se trate de tablas, programas u otro tipo de archivos.

### Cuadro de lista con imágenes



El código siguiente está asociado al [evento Click](#) del cuadro de lista:

```
FOR iItem = 5 TO THIS.ListCount      && los archivos empiezan en el 5º elemento
  cExtension = UPPER(RIGHT(THIS.List(iItem),3))
  DO CASE
    CASE cExtension = "DBF"
      THIS.Picture(iItem) = "tables.bmp"
    CASE cExtension = "BMP"
      THIS.Picture(iItem) = "other.bmp"
    CASE cExtension = "PRG"
      THIS.Picture(iItem) = "programs.bmp"
    CASE cExtension = "SCX"
      THIS.Picture(iItem) = "form.bmp"
    OTHERWISE
      THIS.Picture(iItem) = IIF("]" $ cExtension, ;
        "", "textfile.bmp")
  ENDCASE
ENDFOR
```

### Usar casillas de verificación



Las [casillas de verificación](#) pueden emplearse para permitir que un usuario especifique un estado booleano: Verdadero o Falso, Activado o Desactivado, Abierto o Cerrado. Sin embargo, en algunos casos la evaluación de algo como Verdadero o Falso no es muy precisa, como preguntas no contestadas en un cuestionario de tipo Verdadero o Falso.

### Para ver ejemplos de uso de casillas de verificación

1. Ejecute Solution.app en el directorio Visual Studio ...\\Samples\\Vfp98\\Solution.



- En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Casilla de verificación**.

Hay cuatro estados posibles para una casilla de verificación, determinados por la [propiedad Value](#).

Presentación	Propiedad Value
<input type="checkbox"/> Check1	0 o .F.
<input checked="" type="checkbox"/> Check2	1 o .T.
<input checked="" type="checkbox"/> Check3	2
<input type="checkbox"/> Check4	.NULL.

La propiedad Value de la casilla de verificación refleja el [tipo de datos](#) de la última asignación. Si establece la propiedad como verdadera (.T.) o falsa (.F.), el tipo será Logical hasta que establezca la propiedad en un valor numérico.

**Sugerencia** Un usuario puede mostrar un valor nulo en una casilla de verificación si presiona CTRL+0.

### Almacenar o mostrar campos lógicos

Si establece la propiedad [ControlSource](#) de la casilla de verificación como un campo lógico de una tabla, la casilla de verificación se mostrará activada cuando el valor del registro actual sea verdadero (.T.), como no activada cuando el registro actual sea falso (.F.) y como atenuada cuando haya un [valor nulo](#) (.NULL.) en el registro actual.

### Aceptar entradas que no se pueden determinar previamente

No siempre es posible anticipar todos los valores posibles que un usuario necesita introducir en un control. Los controles siguientes permiten aceptar entradas de usuarios que no se pueden determinar previamente:

- Cuadros de texto
- Cuadros de edición
- Cuadros combinados

### Usar cuadros de texto



El cuadro de texto es el control básico que permite a los usuarios agregar o modificar datos almacenados en un campo no memo de una tabla.

### Para ver ejemplos de uso de cuadros de texto

1. Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Cuadro de texto**.

### Para manipular mediante programación el texto que se muestra en el cuadro de texto

- Establezca o haga referencia a la propiedad [Value](#).

Si establece la propiedad [ControlSource](#) para el cuadro de texto, el valor que aparece en el cuadro de texto se almacenará en la propiedad Value del cuadro de texto y en el campo de la tabla o del cursor que se especifique en la propiedad ControlSource.

### Validar datos en un cuadro de texto

Para comprobar el valor del cuadro de texto, incluya código en el [método](#) asociado al [evento Valid](#). Si el valor no es válido, se devolverá falso (.F.) o 0. Si Valid devuelve falso (.F.) se muestra el mensaje "La entrada no es válida". Si desea mostrar su propio mensaje, incluya el comando WAIT WINDOW o la función [MESSAGEBOX\(\)](#) en el código Valid y devuelva 0.

Por ejemplo, si tiene un cuadro de texto que permite a un usuario escribir la fecha de una cita, puede asegurarse de que la fecha no ha pasado si incluye el código siguiente en el evento Valid del cuadro de texto:

```
IF CTOD(THIS.Value) < DATE( )
    = MESSAGEBOX("Debe escribir una fecha futura",1)
    RETURN 0
ENDIF
```

### Seleccionar texto cuando el cuadro de texto recibe el enfoque

Para seleccionar todo el texto cuando el usuario escribe en el cuadro de texto usando el teclado, establezca la propiedad [SelectOnEntry](#) a verdadero (.T.).

### Formato de texto en un cuadro de texto

Puede utilizar la propiedad [InputMask](#) para determinar los valores que se pueden escribir en el cuadro de texto y la propiedad [Format](#) para determinar cómo se muestran los valores en el cuadro de texto.

### Usar la propiedad InputMask

La propiedad InputMask determina las características de cada carácter escrito en el cuadro de texto. Por ejemplo, puede establecer la propiedad InputMask en 999.999,99 para limitar la entrada del usuario a valores numéricos inferiores a 1.000.000 con dos posiciones decimales. La coma y el punto se mostrarán en el cuadro de texto antes de que el usuario pueda introducir algún valor. Si el usuario presiona una tecla de carácter, el carácter no aparecerá en el cuadro de texto.

Si tiene un campo lógico y desea que un usuario puede introducir "S" o "N", pero no "T" o "F", establezca la propiedad InputMask como "S".

## Aceptar contraseñas de usuario en un cuadro de texto

Con frecuencia, en una aplicación es conveniente obtener información segura de un usuario, como una contraseña. Puede utilizar un [cuadro de texto](#) para obtener esta información sin que aparezca en la pantalla.

### Para aceptar la entrada del usuario sin mostrar el valor real

- Establezca la propiedad [PasswordChar](#) del cuadro de texto como \* o algún otro carácter genérico.

Si establece la propiedad PasswordChar como algo que no sea una cadena vacía, las propiedades [Value](#) y [Text](#) del cuadro de texto contendrán el valor real que el usuario escribió en el cuadro de texto, pero éste mostrará un carácter genérico para cada tecla que haya presionado el usuario.

### Escribir fechas en un cuadro de texto

Los cuadros de texto tienen varias propiedades que se pueden establecer para facilitar a los usuarios escribir valores de fecha.

Propiedad	Descripción
<a href="#">Century</a>	Especifica si los dos primeros dígitos del año se muestran o no.
<a href="#">DateFormat</a>	Formato de la fecha en el cuadro de texto entre quince formatos predefinidos, como Americano, Alemán, Japonés.
<a href="#">StrictDateEntry</a>	Si se establece StrictDateEntry a 0 - Libre, permite al usuario escribir fechas en formatos más flexibles que el predeterminado 99/99/99.

### Propiedades comunes de los cuadros de texto

Las siguientes propiedades de cuadros de texto suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">Alignment</a>	Especifica si el contenido del cuadro de texto está alineado a la izquierda, a la derecha, centrado o alineado automáticamente. La alineación automática depende del tipo de datos. Los números, por ejemplo, se alinean a la derecha y los caracteres se alinean a la izquierda.
<a href="#">ControlSource</a>	El campo de tabla o variable cuyo valor se muestra en el cuadro de texto.
<a href="#">InputMask</a>	Especifica la regla de entrada de datos que cada carácter escrito debe seguir. Para obtener información específica sobre InputMask, vea la Ayuda.

---

<a href="#">SelectOnEntry</a>	Especifica si el contenido del cuadro de texto se selecciona automáticamente cuando el cuadro de texto recibe el enfoque.
<hr/>	
<a href="#">TabStop</a>	Especifica si el usuario puede llegar al control mediante tabulaciones. Si TabStop está establecido a .F., un usuario puede seleccionar el cuadro de texto si hace clic en él.

---

## Usar cuadros de edición



Puede permitir que los usuarios modifiquen texto de campos de caracteres o de campos memo largos en [cuadros de edición](#). Los cuadros de edición permiten el ajuste automático de línea y ofrecen la posibilidad de moverse por el texto con las teclas de dirección, las teclas de avance y retroceso de página, y las barras de desplazamiento.

### Para ver ejemplos del uso de cuadros de edición

1. Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Cuadro de edición**.

### Permitir a los usuarios modificar un campo memo en un cuadro de edición

Lo único que debe hacer para permitir a un usuario modificar un campo memo en un cuadro de edición es establecer la propiedad [ControlSource](#) del cuadro de edición como el campo memo. Por ejemplo, si tiene un campo memo llamado `comentarios` en una tabla llamada `registro`, podrá establecer la propiedad `ControlSource` de un cuadro de edición como `registro.comentarios` para permitir que un usuario modifique el campo memo en el cuadro de edición.

### Permitir a los usuarios modificar un archivo de texto en un cuadro de edición

También puede permitir a un usuario modificar un archivo de texto en un cuadro de edición. El formulario siguiente lo demuestra.

### Formulario de ejemplo para modificar un archivo de texto en un cuadro de edición



Un botón "Aceptar" en el formulario cierra el formulario con el siguiente comando en el código de evento Click:

```
RELEASE THISFORM
```

Los otros dos botones de este ejemplo, `cmdOpenFile` y `cmdSave`, permiten a un usuario abrir un archivo de texto y guardar el archivo después de modificarlo.

### Código asociado al evento Click de `cmdOpenFile`

Código	Comentarios
<pre>CREATE CURSOR textfile ;   (filename c(35), mem m) APPEND BLANK</pre>	<p>Crea un <a href="#">cursor</a> con un campo de caracteres para guardar el nombre del archivo de texto y un campo memo para guardar el contenido del archivo de texto.</p> <p>Agrega un registro en blanco al cursor.</p>
<pre>REPLACE textfile.FileName WITH ;   GETFILE("TXT")</pre>	<p>Usa la función <a href="#">GETFILE()</a> para devolver el nombre del archivo a abrir. Almacena el nombre en el campo <code>FileName</code> del cursor.</p>
<pre>IF EMPTY(textfile.FileName)   RETURN ENDIF</pre>	<p>Si el usuario elige Cancelar en el cuadro de diálogo Obtener archivo, el campo <code>FileName</code> estará vacío y no habrá ningún archivo que abrir.</p>
<pre>APPEND MEMO mem FROM ;   (textfile.FileName) OVERWRITE</pre>	<p>Llena el campo memo con el texto del archivo.</p>
<pre>THISFORM.edtText.ControlSource = ;   "textfile.mem" THISFORM.Refresh</pre>	<p>Establece la propiedad <a href="#">ControlSource</a> del cuadro de edición en el formulario.</p>
<pre>THISFORM.cmdSave.Enabled = .T.</pre>	<p>Activa el botón Guardar.</p>

---

Una vez abierto y modificado el archivo, el botón "Guardar" permite que el usuario vuelva a escribir los cambios en el archivo.

### Código asociado al [evento Click](#) de cmdSave

Código	Comentarios
<pre>COPY MEMO textfile.mem TO ;     (textfile.filename)</pre>	Sobrescribe el valor antiguo en el archivo con el texto del campo memo.

---

### Manipular texto seleccionado en un cuadro de edición

Los cuadros de edición y los cuadros de texto tienen tres propiedades que le permiten trabajar con texto seleccionado: SelLength, SelStart y SelText.

Puede seleccionar texto mediante programación con las propiedades [SelStart](#) y [SelLength](#). Por ejemplo, las líneas de código siguientes seleccionan la primera palabra de un cuadro de edición.

```
Form1.edtText.SelStart = 0  
Form1.edtText.SelLength = AT(" ", Form1.edtText.Text) - 1
```

**Sugerencia** Cuando cambia la propiedad SelStart, el cuadro de edición desplaza el texto para mostrar el nuevo valor de SelStart. Si cambia SelStart en un bucle, por ejemplo, al buscar texto, el código se ejecutará más rápido si incluye THISFORM.LockScreen = .T. antes de procesar y THISFORM.LockScreen = .F. después de procesar.

Para tener acceso al texto seleccionado en un cuadro de edición o en un cuadro de texto, utilice la propiedad [SelText](#). Por ejemplo, la línea de código siguiente escribirá en mayúsculas el texto seleccionado:

```
Form1.edtText.SelText = UPPER(Form1.edtText.SelText)
```

### Propiedades comunes de los cuadros de edición

Las propiedades siguientes de los cuadros de edición suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">AllowTabs</a>	Si el usuario puede insertar tabulaciones en el cuadro de edición en lugar de moverse al control siguiente. Si permite tabulaciones, asegúrese de indicar que los usuarios pueden moverse al control siguiente presionando CTRL+TAB.
<a href="#">HideSelection</a>	Si el texto seleccionado en el cuadro de edición está seleccionado de forma visible cuando el cuadro de edición no tiene el enfoque.
<a href="#">ReadOnly</a>	Si el usuario puede cambiar el texto en el cuadro de edición.

---

[ScrollBars](#)

---

Si hay barras de desplazamiento verticales.

---

## Usar cuadros combinados



El control cuadro combinado tiene la funcionalidad de un cuadro de lista y un cuadro de texto. Hay dos estilos para un [cuadro combinado](#): cuadro combinado desplegable y cuadro de lista desplegable. Puede especificar cuál desea si cambia la propiedad [Style](#) del control. Las listas desplegables se describieron en la sección "[Usar cuadros de lista y cuadros de lista desplegables](#)" de este mismo capítulo.

### Cuadro combinado desplegable

Un usuario puede hacer clic en el botón para ver una lista de opciones o introducir un nuevo elemento directamente en el cuadro situado junto al botón. La propiedad `Style` predeterminada de un cuadro combinado es 0 – Cuadro desplegable.

### Agregar elementos de usuario a listas combinadas desplegables

Para agregar el nuevo valor de usuario al cuadro combinado desplegable, puede utilizar la línea de código siguiente en el [método](#) asociado al [evento Valid](#) del cuadro combinado:

```
THIS.AddItem(THIS.Text)
```

Sin embargo, antes de agregar un elemento, es conveniente asegurarse de que el valor no está ya en el cuadro combinado desplegable:

```
lItemExists = .F. && se supone que el valor no está en la lista.
FOR i = 1 to THIS.ListCount
    IF THIS.List(i) = THIS.Text
        lItemExists = .T.
        EXIT
    ENDIF
ENDFOR

IF !lItemExists
    THIS.AddItem(THIS.Text)
ENDIF
```

### Propiedades comunes de los cuadros combinados

Las siguientes propiedades de los cuadros combinados suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">ControlSource</a>	Especifica el campo de la tabla en el que se almacena el valor que elige o escribe el usuario.
<a href="#">DisplayCount</a>	Especifica el número máximo de elementos mostrados en la lista.

<a href="#">InputMask</a>	Para cuadros combinados desplegables, especifica el tipo de valores que se pueden escribir.
<a href="#">IncrementalSearch</a>	Especifica si el control intenta hacer coincidir un elemento de la lista a medida que el usuario escribe cada letra.
<a href="#">RowSource</a>	Especifica el origen de los elementos del cuadro combinado.
<a href="#">RowSourceType</a>	Especifica el tipo de origen del cuadro combinado. Los tipos de origen de fila de un cuadro combinado son iguales que los de una lista. Para ver una explicación de cada uno de ellos, vea la Ayuda o la sección sobre cuadros de lista y cuadros de lista desplegable en este capítulo.
<a href="#">Style</a>	Especifica si el cuadro combinado es un cuadro combinado desplegable o una lista desplegable.

## Aceptar entradas numéricas en un determinado intervalo

Aunque puede establecer la [propiedad InputMask](#) e incluir código en el [evento Valid](#) para comprobar que los valores numéricos introducidos en los cuadros de texto quedan dentro de un determinado intervalo, el modo más sencillo de comprobar el intervalo de valores consiste en utilizar un [control numérico](#).

### Usar controles numéricos



Los controles numéricos pueden emplearse para permitir a los usuarios realizar selecciones mostrando los valores o escribiendo directamente el valor en el cuadro del control numérico.

### Establecer el intervalo de valores que pueden elegir los usuarios

Establezca las propiedades [KeyboardHighValue](#) y [SpinnerHighValue](#) como el número más alto que desea que los usuarios puedan escribir en el control numérico.

Establezca las propiedades [KeyboardLowValue](#) y [SpinnerLowValue](#) como el número más bajo que desea que los usuarios puedan introducir en el control numérico.

### Disminuir un control numérico cuando el usuario hace clic en el botón Arriba

En algunos casos, si el control numérico refleja un valor como "prioridad", será conveniente que el usuario pueda aumentar la prioridad de 2 a 1 haciendo clic en el botón "Arriba". Para hacer que el número del control numérico disminuya cuando el usuario haga clic en el botón "Arriba", establezca la [propiedad Increment](#) como  $-1$ .

### Recorrido por valores no numéricos

Si bien el valor de un control numérico es numérico, puede utilizar el control Spinner y un cuadro de texto para que los usuarios puedan utilizar diversos tipos de datos. Por ejemplo, si desea que un



usuario pueda recorrer un intervalo de fechas, puede ajustar el tamaño del control numérico de modo que sólo estén visibles los botones y situar un cuadro de texto junto a los botones del control numérico. Establezca la [propiedad Value](#) del cuadro de texto como una fecha y en los eventos [UpClick](#) y [DownClick](#) del control numérico, incremente o disminuya la fecha.

**Sugerencia** Puede usar la función `GetSystemMetrics` de la API de Windows para establecer el ancho del control numérico de forma que sólo los botones estén visibles y tengan la anchura óptima para mostrar los mapas de bits flecha hacia arriba y flecha hacia abajo.

1. Establezca la [propiedad BorderStyle](#) del control numérico a 0.
2. Incluya el código siguiente en el evento Init del control numérico:

```
DECLARE INTEGER GetSystemMetrics IN Win32api INTEGER
THIS.Width = GetSystemMetrics(2) && SM_CXVSCROLL
```

## Propiedades comunes de los controles numéricos

Las siguientes propiedades de los controles numéricos suelen establecerse en [tiempo de diseño](#).

Propiedad	Descripción
<a href="#">Interval</a>	Cuánto se incrementa o disminuye el valor cada vez que el usuario hace clic en los botones "Arriba" o "Abajo".
<a href="#">KeyboardHighValue</a>	El valor más alto que puede escribirse en el cuadro de texto del control numérico.
<a href="#">KeyboardLowValue</a>	El valor más bajo que puede escribirse en el cuadro de texto del control numérico.
<a href="#">SpinnerHighValue</a>	El valor más alto que muestra el control numérico cuando el usuario hace clic en el botón "Arriba".
<a href="#">SpinnerLowValue</a>	El valor más bajo que muestra el control numérico cuando el usuario hace clic en el botón "Abajo".

## Permitir acciones específicas

Es posible que en numerosas ocasiones desee permitir que los usuarios realicen determinadas acciones que no tienen nada que ver con la manipulación de valores. Por ejemplo, puede permitir que un usuario cierre un formulario, abra otro formulario, se mueva por una tabla, guarde o cancele modificaciones, ejecute un informe o una consulta, salte a una dirección de un destino de Internet o una intranet o realice alguna otra acción.

## Usar botones de comando y grupos de botones de comando



Uno de los lugares más frecuentes para situar el código para acciones específicas es el [evento Click](#)

de un botón de comando.

### Convertir un botón de comando en la opción predeterminada

Establezca la [propiedad Default](#) como verdadera (.T.) para convertir el botón de comando en la opción predeterminada. La opción predeterminada tiene un borde más grueso que otros botones de comando. Si un botón de comando es la opción predeterminada, cuando el usuario presione ENTRAR, se ejecutará el evento Click del botón de comando.

**Nota** Si el objeto seleccionado en un formulario es un [cuadro de edición](#) o una [cuadrícula](#), el código asociado al evento Click de la opción predeterminada no se ejecutará cuando el usuario presione ENTRAR. Si se presiona entrar en un cuadro de edición, se agregará un retorno de carro y un avance de línea al valor del cuadro de edición. Si se presiona ENTRAR en una cuadrícula, se seleccionará un campo adyacente. Para ejecutar el evento Click del botón predeterminado, presione CTRL+ENTRAR.

### Propiedades comunes de los botones de comando

Las siguientes propiedades de los botones de comando suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">Cancel</a>	Especifica que el código asociado al evento Click del botón de comando se ejecuta cuando el usuario presiona ESC.
<a href="#">Caption</a>	Texto que se muestra en el botón.
<a href="#">DisabledPicture</a>	Imagen .bmp que se muestra cuando se desactiva el botón.
<a href="#">DownPicture</a>	Imagen .bmp que se muestra cuando se presiona el botón.
<a href="#">Enabled</a>	Indica si puede elegirse o no el botón.
<a href="#">Picture</a>	Imagen .bmp que se muestra en el botón.

También puede incluir botones de comando en un grupo de modo que pueda manipularlos individualmente o como un grupo.

### Administrar opciones de botones de comando a nivel de grupo

Si desea trabajar con un único procedimiento de método para todo el código de los [eventos Click](#) de botones de comando de un grupo, podrá adjuntar el código al evento Click del [grupo de botones de comando](#). La [propiedad Value](#) del grupo de botones de comando indica en qué botones se ha hecho clic, como demuestra el ejemplo de código siguiente:

```
DO CASE
  CASE THIS.Value = 1
    WAIT WINDOW "Ha hecho clic en " + THIS.cmdCommand1.Caption NOWAIT
    * realizar alguna acción
  CASE THIS.Value = 2
    WAIT WINDOW "Ha hecho clic en " + THIS.cmdCommand2.Caption NOWAIT
    * realizar otra acción
```

```
CASE THIS.Value = 3
    WAIT WINDOW "Ha hecho clic en " + THIS.cmdCommand3.Caption NOWAIT
    * realizar una tercera acción
ENDCASE
```

**Nota** Si el usuario hace clic en el grupo de botones de comando pero no en un determinado botón, la propiedad Value seguirá reflejando el último botón de comando seleccionado.

Si ha escrito código para el evento Click de un determinado botón del grupo, cuando el usuario elija ese botón se ejecutará ese código en lugar del código del evento Click del grupo.

### Propiedades comunes de los grupos de botones de comando

Las siguientes propiedades de los grupos de botones de comando suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">ButtonCount</a>	Número de botones del grupo de comandos.
<a href="#">BackStyle</a>	Especifica si el grupo de botones de comando tiene un fondo transparente u opaco. Un fondo transparente parece tener el mismo color que el que tiene el objeto subyacente, normalmente el formulario o una página.

### Utilizar el objeto Hyperlink

Puede utilizar el objeto Hyperlink para saltar a una dirección de un destino de Internet o de una intranet. El objeto Hyperlink se puede utilizar para iniciar una aplicación que admita hipervínculos, generalmente un explorador de Internet como Microsoft Internet Explorer, y abrir la página especificada en la dirección. El método Hyperlink NavigateTo( ) le permite especificar la dirección de destino a la que se salta.

Por ejemplo, para ir al sitio Internet de Microsoft en World Wide Web desde un formulario, agregue en primer lugar el control Hyperlink al formulario. Agregue un comando al formulario y, a continuación, agregue el código siguiente al evento Click del botón de comando:

```
THISFORM.Hyperlink1.NavigateTo('www.microsoft.com')
```

Cuando se ejecute el formulario puede hacer clic en el botón de comando para saltar al sitio Web de Microsoft.

### Realizar acciones específicas a intervalos regulares

El control Cronómetro permite realizar acciones o comprobar valores a intervalos regulares.

#### Usar el control Cronómetro



Los controles Cronómetro responden al paso del tiempo independientemente de la interacción con el usuario, de modo que pueden programarse para que realicen acciones a intervalos regulares. Suelen emplearse para comprobar el reloj del sistema y ver si es hora de llevar a cabo una determinada tarea. Los cronómetros también resultan útiles para otros tipos de procesamiento en segundo plano.

### Para ver ejemplos del uso de cronómetros

1. Ejecute Solution.app en el directorio Visual Studio ...\\Samples\\Vfp98\\Solution.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Cronómetro**.

Cada cronómetro tiene una [propiedad Interval](#), que especifica el número de milisegundos que transcurren entre un evento de cronómetro y el siguiente. A menos que se desactive, un cronómetro continúa recibiendo un [evento](#) (denominado de acuerdo con el evento Timer) a intervalos de tiempo aproximadamente iguales. La propiedad Interval tiene algunas limitaciones que deben tenerse en cuenta cuando se programa un cronómetro:

- El intervalo puede estar entre 0 y 2.147.483.647, inclusive, lo que significa que el intervalo más largo es de unas 596,5 horas (más de 24 días).
- No se garantiza que el intervalo tenga una duración exacta. Para asegurar la precisión, el cronómetro debe comprobar el reloj del sistema cuando lo necesita, en lugar de intentar realizar un seguimiento interno del tiempo acumulado.
- El sistema genera 18 impulsos de reloj por segundo por lo que, aunque la propiedad Interval se mide en milisegundos, la precisión real de un intervalo no es superior a la decimoctava parte de un segundo.
- Si su aplicación u otra distinta sobrecarga el sistema (por ejemplo, a través de bucles largos, cálculos intensivos o acceso al disco, a la red o al puerto), es posible que la aplicación no obtenga eventos de cronómetro con la frecuencia que especifica la propiedad Interval.

### Colocar un control Timer en un formulario

Colocar un control Timer en un [formulario](#) es como dibujar cualquier otro control: se elige la herramienta Timer en la barra de herramientas [Controles de formularios](#) y se arrastra a un formulario.

### Un control Timer



El cronómetro aparece en el formulario en [tiempo de diseño](#) de forma que puede seleccionarlo, ver sus propiedades y escribir un procedimiento de evento para el mismo. En tiempo de [ejecución](#), el cronómetro es invisible y su posición y tamaño son irrelevantes.

### Inicializar un control Timer

Un control Timer tiene dos propiedades clave.

Propiedad	Valor
<a href="#">Enabled</a>	Si desea que el cronómetro comience a funcionar en cuanto se cargue el formulario, establézcala a verdadero (.T.). De lo contrario, deje esta propiedad establecida a falso (.F.). Puede elegir un evento externo (como un clic en un botón de comando) para que se inicie la operación del cronómetro.
<a href="#">Interval</a>	Número de milisegundos entre los eventos del cronómetro.

Observe que la propiedad Enabled del cronómetro es distinta que la de otros objetos. Con la mayoría de los objetos, la propiedad Enabled determina si el objeto puede responder o no a un evento causado por el usuario. Con el control Timer, al establecer Enabled a falso (.F.) se suspende el funcionamiento del cronómetro.

Recuerde que el evento Timer es periódico. La propiedad Interval no determina "cuánto tiempo", sino más bien "con qué frecuencia". La duración del intervalo debe depender de la precisión que desee. Puesto que existen posibilidades inherentes de error, cree el intervalo con la mitad de la precisión deseada.

**Nota** Cuanto más frecuentemente se genere un evento de cronómetro, más tiempo de procesador se consumirá para responder al evento. Esto puede hacer más lento el rendimiento global. No establezca un intervalo excesivamente pequeño a menos que lo necesite.

### Responder al evento Timer

Cuando transcurre el intervalo del control Timer, Visual FoxPro genera el [evento Timer](#). La respuesta a este evento suele consistir en comprobar alguna condición general, como el reloj del sistema.

Un reloj digital es una aplicación muy sencilla pero de gran utilidad que interviene en un control Timer. Cuando comprenda cómo funciona la aplicación, podrá mejorarla para que funcione como un despertador, un cronómetro u otro dispositivo de temporización.

La aplicación de reloj digital incluye un cronómetro y una etiqueta con un borde. En [tiempo de diseño](#), la aplicación tiene esta apariencia:

### La aplicación reloj digital



En [tiempo de ejecución](#), el cronómetro es invisible.

Control	Propiedad	Valor
LblTime	<a href="#">Caption</a>	
Timer1	<a href="#">Interval</a>	500 (medio segundo)
Timer1	<a href="#">Enabled</a>	Verdadero

El único procedimiento de la aplicación es el procedimiento de evento Timer:

```
IF THISFORM.lblTime.Caption != Time()
    THISFORM.lblTime.Caption = Time()
ENDIF
```

La propiedad Interval del cronómetro está establecida a 500, siguiendo la regla de establecer el intervalo en la mitad del período más corto que desea distinguir (en este caso, un segundo). Esto puede hacer que el código de cronómetro actualice la etiqueta con la misma hora dos veces en un segundo, lo que podría producir un parpadeo visible. Por ello, el código comprueba si la hora es distinta de lo que aparece en la etiqueta antes de cambiar el título.

## Mostrar información

Uno de los principios de un buen diseño consiste en que la información relevante esté visible. Puede utilizar los controles siguientes para mostrar información a los usuarios:

- Imágenes
- Etiquetas
- Cuadros de texto
- Cuadros de edición
- Formas

### Usar imágenes



El control Image permite agregar imágenes (archivos .bmp) al formulario. Un control Image tiene la gama completa de [propiedades](#), [eventos](#) y [métodos](#) que tienen otros controles, por lo que puede cambiarse dinámicamente en [tiempo de ejecución](#). Los usuarios pueden interactuar con imágenes haciendo clic, haciendo doble clic, etc.

La tabla siguiente muestra algunas de las propiedades clave de un control Image.

Propiedad	Descripción
<a href="#">Picture</a>	La imagen (archivo .bmp) que se muestra.
<a href="#">BorderStyle</a>	Indica si la imagen tiene o no un borde visible.
<a href="#">Stretch</a>	Si Stretch se establece a 0 – Recortar, no se mostrarán las partes de la

imagen que superen las dimensiones del control Image. Si Stretch se establece a 1 – Isométrico, el control Image conservará las dimensiones originales de la imagen y mostrará la imagen en la medida que lo permitan las dimensiones del control Image. Si Stretch se establece a 2 – Estirar, la imagen se ajustará para que coincida exactamente con el alto y el ancho del control Image.

---

## Usar etiquetas



Las etiquetas se diferencian de los cuadros de texto en los siguientes aspectos:

- No pueden tener un origen de datos.
- No pueden modificarse directamente.
- No puede tener acceso a las mismas mediante la tecla tab.

Se pueden cambiar las propiedades [Caption](#) y [Visible](#) de las etiquetas mediante programación para adaptar la etiqueta a la situación concreta.

## Propiedades comunes de las etiquetas

Las siguientes propiedades de las etiquetas suelen establecerse en [tiempo de diseño](#).

Propiedad	Descripción
<a href="#">Caption</a>	El texto que muestra la etiqueta.
<a href="#">AutoSize</a>	Indica si el tamaño de la etiqueta se ajusta a la longitud del título.
<a href="#">BackStyle</a>	Indica si la etiqueta es opaca o transparente.
<a href="#">WordWrap</a>	Indica si el texto que se muestra en la etiqueta puede ajustarse automáticamente a líneas adicionales.

## Usar cuadros de texto y cuadros de edición para mostrar información

Establezca la [propiedad ReadOnly](#) de cuadros de texto y cuadros de edición para mostrar información que el usuario puede ver pero no modificar. Si sólo desactiva un cuadro de edición, el usuario no podrá desplazarse por el texto.

## Usar formas y líneas

Las [formas](#) y las [líneas](#) ayudan a agrupar visualmente elementos de los formularios. Se ha comprobado que la asociación de elementos relacionados ayuda a los usuarios a comprender y utilizar una interfaz, lo que facilita el uso de la aplicación.

Las siguientes propiedades del control Shape suelen establecerse en [tiempo de diseño](#).



Propiedad	Descripción
<a href="#">Curvature</a>	Un valor entre 0 (ángulos de 90 grados) y 99 (círculo o elipse).
<a href="#">FillStyle</a>	Indica si la forma es transparente o tiene un determinado modelo de relleno del fondo.
<a href="#">SpecialEffect</a>	Indica si la forma es sencilla o tridimensional. Sólo tiene efecto cuando la propiedad Curvature se establece a 0.

Las siguientes propiedades de Line suelen establecerse en [tiempo de diseño](#).



Propiedad	Descripción
<a href="#">BorderWidth</a>	Indica cuántos píxeles de ancho tiene la línea.
<a href="#">LineSlant</a>	Cuando la línea no es horizontal ni vertical, indica el sentido de la inclinación. Los valores válidos para esta propiedad son una barra diagonal ( / ) y una barra inversa ( \ ).

### Usar gráficos de formulario para mostrar información

Puede mostrar información gráficamente en un formulario con los siguientes métodos de formulario.

Método	Descripción
<a href="#">Circle</a>	Dibuja una figura circular o un arco en un formulario.
<a href="#">Cls</a>	Borra gráficos y texto de un formulario.
<a href="#">Line</a>	Dibuja una línea en un formulario.
<a href="#">Pset</a>	Establece un punto de un formulario con un determinado color.
<a href="#">Print</a>	Imprime una cadena de caracteres en un formulario.

### Para ver ejemplos que muestran gráficos de formularios

1. Ejecute Solution.app en el directorio ... \Samples\Vfp98\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Formularios** y, a continuación, haga clic en **Gráficos de formulario**.

### Mejorar la presentación de controles



Los [botones de comando](#), las [casillas de verificación](#) y los [botones de opción](#) pueden mostrar una imagen además de un título. Todos estos controles tienen propiedades que permiten especificar imágenes que se muestran en los controles.

Propiedad	Descripción
<a href="#">DisabledPicture</a>	Imagen que se muestra en el botón cuando éste está desactivado.
<a href="#">DownPicture</a>	Imagen que se muestra en el botón cuando éste está presionado.
<a href="#">Picture</a>	Imagen que se muestra en el botón cuando éste está activado y no presionado.

Si no especifica una propiedad DisabledPicture, Visual FoxPro mostrará la imagen atenuada cuando se desactive el control. Si no especifica DownPicture, Visual FoxPro mostrará la imagen con los colores del fondo cambiados de modo que el botón aparezca presionado cuando se presione el botón.

Si no desea que se muestre un título además de la imagen, establezca la propiedad [Caption](#) como una cadena vacía eliminando el título predeterminado en el cuadro "Edición de propiedades" de la [ventana Propiedades](#).

### Usar máscaras de imagen

En muchos casos, una imagen .bmp contiene espacio en blanco que no conviene que aparezca en los controles. Un borde blanco alrededor de una imagen de forma irregular puede dar una mala apariencia al control. Para evitar este problema, Visual FoxPro crea una máscara temporal predeterminada para el .bmp. Las áreas en blanco reciben un atributo transparente de modo que sea transparente el color subyacente del botón o el fondo. Para mantener en blanco algunas áreas del .bmp, cree una máscara para el .bmp que no aplique el valor predeterminado.

### Para crear una máscara para un .bmp

1. Abra el archivo .BMP en Paint u otra utilidad de mapa de bits.
2. Pinte de negro todas las áreas de la imagen que desea que aparezcan tal como son en el .bmp. Deje en blanco todas las áreas que desea que sean transparentes.
3. Guarde el archivo en el mismo directorio y con el mismo nombre que el archivo .bmp pero con la extensión .msk.

Cuando Visual FoxPro cargue un archivo .bmp especificado por la propiedad [Picture](#) para un botón de comando, un botón de opción o una casilla de verificación, buscará en el mismo directorio un archivo .msk equivalente. Si en el directorio hay un archivo .msk con el mismo nombre que el .bmp, Visual FoxPro lo utilizará como máscara para el .bmp. Todas las áreas en blanco de la imagen .msk se convierten en transparentes en el .bmp, mientras que las áreas negras de la imagen .msk se muestran tal como aparecen en el .bmp.

**Nota** La imagen .bmp y la imagen .msk deben tener las mismas dimensiones para que la máscara

pueda representar el área del .bmp.

## Manipular múltiples filas de datos

Visual FoxPro proporciona una herramienta muy potente, el objeto Grid, para mostrar y manipular múltiples filas de datos.

### Usar cuadrículas



La cuadrícula es un objeto contenedor. Del mismo modo que un conjunto de formularios puede contener formularios, una cuadrícula puede contener columnas. Las columnas, a su vez, contienen encabezados y controles, cada uno de los cuales tiene su propio conjunto de [propiedades](#), [eventos](#) y [métodos](#), lo que proporciona un gran control sobre los elementos de la cuadrícula.

Contenedor	Puede contener
Cuadrícula	Columnas
Columna	Encabezados, controles

El objeto Grid permite mostrar y manipular filas y columnas de datos de un [formulario](#) o una [página](#). Una aplicación especialmente útil del control Grid consiste en crear formularios de uno a varios, como un formulario de facturas.

### Para ver ejemplos del uso de cuadrículas

1. Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Cuadrícula**.

### Un formulario con una cuadrícula llena

Producto	Precio	Almacenado	Eliminado
Té Dharamsala	18.0000	39.000	<input type="checkbox"/>
Cervez tibetana Barley	19.0000	17.000	<input type="checkbox"/>
Sirope de regaliz	10.0000	13.000	<input type="checkbox"/>
Espicias Cajun del chef Anton	22.0000	53.000	<input type="checkbox"/>
Mezcla Gumbo del chef Anton	21.3500	0.000	<input checked="" type="checkbox"/>
Mermelada de grosellas de la abuela	25.0000	120.000	<input type="checkbox"/>

Configuración de Sparse: ☒ Precio ☒ Almacenado ☐ Eliminado

Cerrar

### Para agregar un control Grid a un formulario

- En la barra de herramientas [Controles de formularios](#), elija el botón **Cuadrícula** y arrástrelo hasta obtener el tamaño deseado en la ventana **Formulario**.

Si no especifica ningún valor [RecordSource](#) para la cuadrícula y hay una tabla abierta en el [área de trabajo](#) actual, la cuadrícula mostrará todos los campos de esa tabla.

### Establecer el número de columnas de una cuadrícula

Una de las primeras propiedades que puede establecer para el control Grid es el número de columnas.

### Para establecer el número de columnas de una cuadrícula

1. Seleccione la propiedad ColumnCount en la lista de **Propiedades y métodos**.
2. En el cuadro **Propiedad**, escriba el número de columnas que desea.

Si la propiedad ColumnCount está establecida a -1 (el valor predeterminado), la cuadrícula contendrá, en [tiempo de ejecución](#), tantas columnas como campos haya en la tabla asociada a la cuadrícula.

### Ajustar de forma manual la presentación de la cuadrícula en tiempo de diseño

Cuando haya agregado columnas a la cuadrícula, podrá cambiar el ancho de las columnas y el alto de las filas. Podrá ajustar de forma manual las propiedades de alto y ancho de los objetos columna y fila en la ventana Propiedades o bien establecer visualmente estas propiedades en modo de diseño de cuadrícula.

### Para cambiar al modo de diseño de cuadrícula

- Elija **Modificar** en el menú de método abreviado de la cuadrícula.  
–O bien–
- En el cuadro **Objeto** de la [ventana Propiedades](#) seleccione una columna de la cuadrícula.

Cuando esté en modo de diseño de cuadrícula, aparecerá un borde grueso alrededor de la cuadrícula. Para salir de este modo, seleccione el formulario u otro control.

### Para ajustar el ancho de las columnas de una cuadrícula

1. En modo de diseño de cuadrícula, sitúe el puntero del *mouse* entre los encabezados de columna de la cuadrícula de modo que el puntero cambie a una barra con las flechas que apunta a la izquierda y la derecha.
2. Seleccione la columna y arrástrela hasta que tenga el ancho deseado

–O bien–

Establezca la propiedad Width de la columna en la [ventana Propiedades](#).

### Para ajustar el alto de las filas de una cuadrícula

1. En modo de diseño de cuadrícula, sitúe el puntero del *mouse* entre el primer botón y el segundo botón de la parte izquierda del control **Grid** de modo que el puntero cambie a una barra con las flechas que apunta hacia arriba y hacia abajo.
2. Seleccione la fila y arrástrela hasta que tenga el alto deseada.

–O bien–

Establezca la propiedad Height de la columna en la [ventana Propiedades](#).

**Sugerencia** Puede evitar que un usuario cambie la altura de las filas de cuadrícula en tiempo de ejecución si establece [AllowRowSizing](#) a falso (.F.).

### Establecer el origen de los datos que se muestran en la cuadrícula

Puede establecer el origen de los datos para la cuadrícula y para cada columna individualmente.

### Para establecer el origen de datos para una cuadrícula

1. Seleccione la cuadrícula y, a continuación, haga clic en la propiedad RecordSourceType en la [ventana Propiedades](#).
2. Establezca la propiedad RecordSourceType como **0 - Tabla**, si desea que Visual FoxPro abra la tabla o como **1 - Alias** si desea que la cuadrícula se llene con los campos de una tabla que ya está abierta.
3. Haga clic en la propiedad RecordSource de la ventana **Propiedades**.
4. Escriba el nombre del [alias](#) o la [tabla](#) que va a servir de [origen de datos](#) para la cuadrícula.

Si desea especificar determinados campos para que aparezcan en columnas específicas, también puede establecer el [origen de datos](#) para una columna.

### Para establecer el origen de datos para una columna

1. Seleccione la columna y, a continuación, haga clic en la propiedad ControlSource de la [ventana Propiedades](#).
2. Escriba el nombre del [alias](#) o la [tabla](#) y el [campo](#) que va a servir como origen para los valores que se muestran en la columna. Por ejemplo, puede escribir:

```
Orders.order_id
```

### Agregar registros a una cuadrícula

Puede permitir a los usuarios agregar nuevos registros a una tabla mostrada en una cuadrícula si establece la [propiedad AllowAddNew](#) de la cuadrícula a verdadero (.T.). Cuando la propiedad AllowAddNew está establecida a verdadero, se agregan nuevos registros a la tabla cuando el último registro está seleccionado y el usuario presiona la tecla FLECHA ABAJO.

Si quiere tener más control sobre cuándo un usuario agrega nuevos registros a una tabla, puede establecer la propiedad AllowAddNew a falso (.F.), el valor predeterminado, y usar los comandos [APPEND BLANK](#) o [INSERT](#) para agregar nuevos registros.

### Establecer un formulario de uno a varios mediante el control Grid

Uno de los usos más comunes de una cuadrícula consiste en mostrar los registros secundarios de una tabla, mientras que los [cuadros de texto](#) muestran los datos de los registros primarios. Cuando el usuario se mueve por los registros de la [tabla primaria](#), la cuadrícula muestra los registros secundarios correspondientes.

Si tiene un [entorno de datos](#) para su formulario que incluye una [relación de uno a varios](#) entre dos tablas, le resultará muy fácil mostrar la relación de uno a varios en el formulario.

### Para establecer un formulario de uno a varios con un entorno de datos

1. Arrastre los campos deseados desde la tabla primaria del [Diseñador de entorno de datos](#) hasta el formulario.
2. Arrastre la tabla relacionada desde el **Diseñador de entornos de datos** hasta el formulario.

En casi todos los casos, será conveniente crear un entorno de datos para el formulario o el conjunto de formularios. Sin embargo, no es mucho más complicado crear un formulario de uno a varios sin utilizar el Diseñador de entornos de datos.

### Para establecer un formulario de uno a varios sin crear un entorno de datos

1. Agregue cuadros de texto al formulario para mostrar los campos deseados de la [tabla principal](#).
2. Establezca la propiedad [ControlSource](#) de los cuadros de texto como la tabla principal.
3. Agregue una cuadrícula al formulario.
4. Establezca la propiedad [RecordSource](#) de la cuadrícula con el nombre de la tabla relacionada.
5. Establezca la propiedad [LinkMaster](#) de la cuadrícula con el nombre de la tabla principal.
6. Establezca la propiedad [ChildOrder](#) de la cuadrícula con el nombre de la etiqueta de índice de la tabla relacionada que corresponde a la expresión relacional de la tabla principal.
7. Establezca la propiedad [RelationalExpr](#) de la cuadrícula con la expresión que combina la [tabla](#)

[relacionada](#) con la tabla principal. Por ejemplo, si la etiqueta ChildOrder está indexada en "apellido + nombre", establezca RelationalExpr con la misma expresión.

De cualquiera de las formas que establezca el formulario uno a varios, podrá agregar controles de desplazamiento para moverse por la tabla primaria y actualizar los objetos del formulario. Por ejemplo, el código siguiente puede incluirse en el [evento Click](#) de un botón de comando:

```
SELECT orders && si orders es la tabla primaria  
SKIP  
IF EOF( )  
    GO BOTTOM  
ENDIF  
THISFORM.Refresh
```

### Mostrar controles en columnas de cuadrícula

Además de mostrar datos de campos en una cuadrícula, puede tener controles en las columnas de una cuadrícula para poder mostrar a un usuario [cuadros de texto](#), [casillas de verificación](#), [controles desplegables](#), [controles numéricos](#) y otros tipos de controles incrustados. Por ejemplo, si tiene un campo lógico en una tabla, cuando ejecute el formulario un usuario podrá distinguir qué valores de registro son verdaderos (.T.) y cuáles son falsos (.F.) si ve si la casilla de verificación está activada. Cambiar el valor es tan fácil como activar o desactivar la casilla de verificación.

Puede agregar controles a columnas de la cuadrícula de forma interactiva en el [Diseñador de formularios](#) o bien puede escribir código para agregar los controles a las columnas en [tiempo de ejecución](#).

### Para agregar de forma interactiva controles a una columna de cuadrícula

1. Agregue una cuadrícula a un formulario.
2. En la [ventana Propiedades](#), establezca la propiedad ColumnCount de la cuadrícula como el número de columnas deseadas.

Por ejemplo, escriba **2** para una cuadrícula de dos columnas.

3. En la ventana **Propiedades** seleccione la columna primaria para el control en el cuadro Objeto.

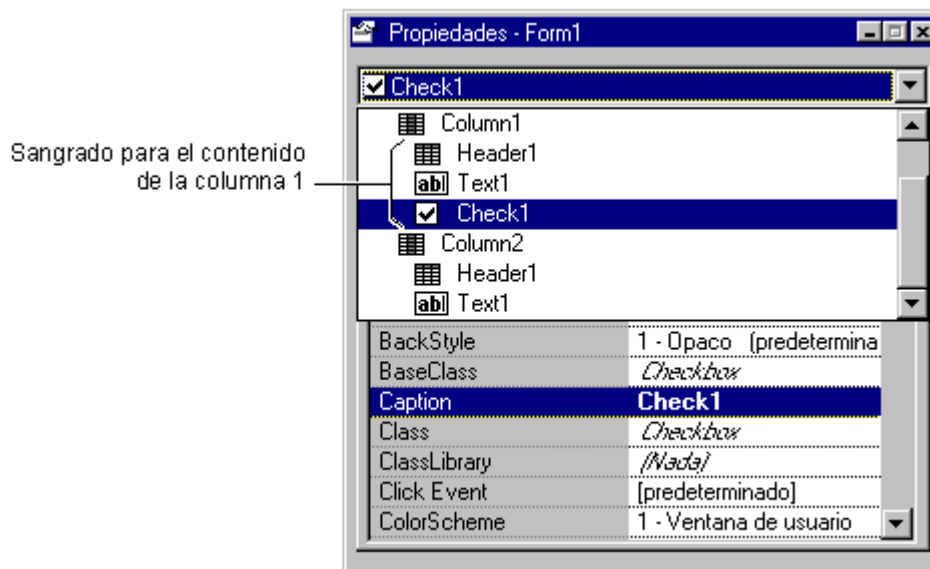
Por ejemplo, seleccione Columna1 para agregar un control a Columna1. El borde de la cuadrícula cambiará para indicar que está modificando un objeto contenido cuando seleccione la columna.

4. Seleccione el control deseado en la barra de herramientas [Controles de formularios](#) y haga clic en la columna primaria.

El nuevo control no aparecerá ahora en la columna de cuadrícula dentro del **Diseñador de formularios**, pero será visible en tiempo de ejecución.

5. En la ventana **Propiedades** asegúrese de que el control se muestra sangrado bajo la columna primaria en el cuadro **Objeto**.

### Una casilla de verificación agregada a una columna de cuadrícula



Si el nuevo control es una casilla de verificación, establezca la propiedad Caption de la casilla como " " y la propiedad Sparse de la columna como falso (.F.).

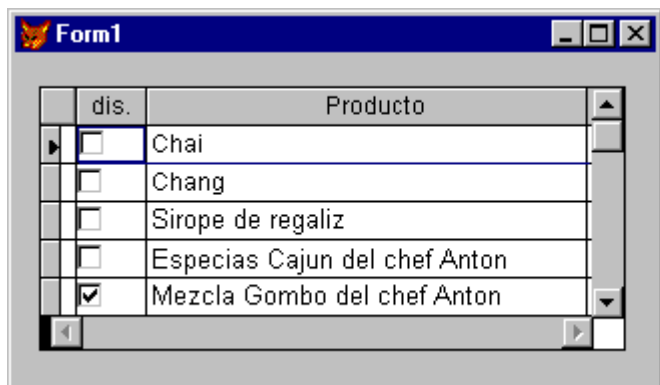
- Establezca la propiedad ControlSource de la columna primaria como el campo de tabla deseado.

Por ejemplo, el ControlSource de la columna en la siguiente ilustración es `products.discontinu` de `Testdata.dbc` en el directorio `...\Samples\Vfp98\Data de Visual Studio`.

- Establezca la propiedad CurrentControl de la columna primaria como el nuevo control.

Cuando ejecute el formulario, el control aparecerá en la columna de cuadrícula.

**La casilla de verificación se muestra en la columna en tiempo de ejecución.**



**Sugerencia** Si desea poder centrar una casilla de verificación en una columna de cuadrícula, cree una [clase de contenedor](#), agregue una casilla de verificación a la clase de contenedor y ajuste la

posición de la casilla en dicha clase. Agregue la clase de contenedor a la columna de cuadrícula y establezca la propiedad `ControlSource` de la casilla de verificación como el campo deseado.

### Para quitar controles de columnas de cuadrícula en el Generador de formularios

1. En el cuadro **Objeto** de la [ventana Propiedades](#), seleccione el control.
2. Active el [Diseñador de formularios](#).

Si la ventana **Propiedades** está visible, el nombre del control aparecerá en el cuadro **Objeto**.

3. Presione la tecla `supr`.

También puede agregar controles a una columna de cuadrícula con el [método `AddObject`](#) en el código.

### Para agregar controles a una columna de cuadrícula mediante programa

- En el [evento `Init`](#) de la cuadrícula, use el [método `AddObject`](#) para agregar el control a la columna de cuadrícula y establezca la [propiedad `CurrentControl`](#) de la columna.

Por ejemplo, las líneas de código siguientes del evento `Init` de una cuadrícula agregan dos controles a una columna de cuadrícula y especifican una de ellas como el control actual:

```
THIS.grcColumn1.AddObject("spnQuantity", "SPINNER")
THIS.grcColumn1.AddObject("cboQuantity", "COMBOBOX")
THIS.grcColumn1.CurrentControl = "spnQuantity"
* Las siguientes líneas de código aseguran que el control está visible
* y se muestra en cada fila de la cuadrícula
THIS.grcColumn1.spnQuantity.Visible = .T.
THIS.grcColumn1.Sparse = .F.
```

En este ejemplo, `Column1` tiene tres valores actuales de control posibles:

- `spnQuantity`
- `cboQuantity`
- `Text1` (el control predeterminado)

**Nota** Las propiedades establecidas a nivel de cuadrícula no se transfieren a las columnas o los encabezados. Del mismo modo, deberá establecer directamente las propiedades de los encabezados y los controles contenidos, ya que no heredan sus propiedades de los valores a nivel de columna.

**Sugerencia** Para presentar mejor los cuadros combinados en columnas de cuadrícula, establezca las siguientes propiedades de cuadro combinado:

```
BackStyle = 0          && Transparente
Margin = 0
SpecialEffect = 1 && Plano
BorderStyle = 0       && Ninguno
```



## Usar formato condicional en una cuadrícula

El formato especial de una cuadrícula puede facilitar al usuario el examen de registros y la localización de cierta información. Para proporcionar formato condicional, utilice las propiedades dinámicas de fuentes y colores de una columna.

Por ejemplo, puede agregar una cuadrícula a un formulario y establecer la propiedad [ColumnCount](#) a 2. Establezca la propiedad [ControlSource](#) de la primera columna como `orders.to_name` y la propiedad [ControlSource](#) de la segunda columna como `orders.order_net`. Para mostrar totales de pedido inferiores a 500,00 con negro como color de primer plano y los totales de pedido mayores o iguales que 500,00 con rojo como color de primer plano, incluya la línea siguiente en el código de evento Init de la cuadrícula:

```
THIS.Column2.DynamicForeColor = ;
    "IIF(orders.order_net >= 500, RGB(255,0,0), RGB(0,0,0))"
```

## Propiedades comunes de las cuadrículas

Las siguientes propiedades de las cuadrículas suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">ChildOrder</a>	La <a href="#">clave externa</a> de la <a href="#">tabla secundaria</a> que se combina con la <a href="#">clave principal</a> de la <a href="#">tabla primaria</a> .
<a href="#">ColumnCount</a>	Número de columnas. Si <a href="#">ColumnCount</a> está establecida a - 1, la columna tendrá tantas columnas como campos haya en la propiedad <a href="#">RecordSource</a> de la cuadrícula.
<a href="#">LinkMaster</a>	La tabla primaria para registros secundarios que se muestran en la cuadrícula.
<a href="#">RecordSource</a>	Los datos que se muestran en la cuadrícula.
<a href="#">RecordSourceType</a>	Indica de dónde provienen los datos que se muestran en la cuadrícula: una <a href="#">tabla</a> , un <a href="#">alias</a> , una <a href="#">consulta</a> o una tabla seleccionada por el usuario como respuesta a una petición.

## Propiedades comunes de las columnas

Las siguientes propiedades de las columnas suelen establecerse en [tiempo de diseño](#).

Propiedad	Descripción
<a href="#">ControlSource</a>	Los datos que se muestran en la columna. Suele ser un campo de una tabla.
<a href="#">Sparse</a>	Si <a href="#">Sparse</a> se establece como verdadero (.T.), los controles de una

cuadrícula sólo se mostrarán como controles cuando se seleccione la celda de la columna. Otras celdas de la columna muestran el valor de datos subyacente en un cuadro de texto. Si establece Sparse como verdadero (.T.), la actualización será más rápida si un usuario se desplaza por una cuadrícula con muchas filas visibles.

---

#### [CurrentControl](#)

Indica cuál es el control activo de la cuadrícula. El valor predeterminado es Text1, pero si agrega un control a la columna, podrá especificarlo como CurrentControl.

---

**Nota** La propiedad [ReadOnly](#) de un control de una columna queda anulada por la propiedad ReadOnly de la columna. Si establece la propiedad ReadOnly del control de una columna en el código asociado al [evento AfterRowColChange](#), el nuevo valor será válido mientras se encuentre en esa celda.

## Simplificar el uso de los controles

Es conveniente hacer todo lo posible para facilitar a los usuarios la comprensión y el uso de los controles. Las teclas de acceso, el orden de tabulación, el texto de Información sobre herramientas y la desactivación selectiva contribuyen a un diseño más fácil de usar.

### Establecer teclas de acceso

Un usuario puede elegir un control en cualquier lugar del [formulario](#); para ello debe presionar ALT y la tecla correspondiente.

#### Para especificar una tecla de acceso para un control

- Sitúe una barra inversa y un signo menor que (<) delante de la letra deseada en la [propiedad Caption](#) del control.

Por ejemplo, el valor de la propiedad siguiente para el título de un botón de comando convierte la letra A en la tecla de acceso.

\<Abrir

Un usuario puede elegir el [botón de comando](#) desde cualquier lugar del formulario presionando ALT+A.

#### Para especificar una tecla de acceso para un cuadro de texto o un cuadro de edición

1. Cree una [etiqueta](#) con una barra invertida y un signo menos (<) delante de la letra deseada, como C\<liente.
2. Asegúrese de que la etiqueta es el control que precede en el [orden de tabulación](#) el cuadro de texto o el cuadro de edición que quiere que reciba el enfoque.

### Establecer el orden de tabulación de los controles

El [orden de tabulación](#) predeterminado de los controles del formulario es el orden en que se agregan los controles al formulario.

**Sugerencia** Establezca el orden de tabulación de los controles de forma que el usuario pueda moverse fácilmente por los controles en un orden lógico.

### Para cambiar el orden de tabulación de controles

1. En la [barra de herramientas Diseñador de formularios](#), elija el botón **Establecer orden de tabulación**.
2. Haga doble clic en el cuadro situado junto al control que desea que tenga el enfoque inicial cuando se abra el formulario
3. Haga clic en el cuadro situado junto a los otros controles en el orden en que quiere que se llegue a ellos mediante tabulaciones.
4. Haga clic en cualquier lugar fuera de los cuadros de orden de tabulación para terminar.

También puede establecer el orden de tabulación para los objetos del formulario por lista, según los valores de la ficha **Formularios** en el [cuadro de diálogo Opciones](#).

Puede establecer el orden de selección para los botones de opción y de comando de un grupo de controles. Para mover un grupo de controles con el teclado, el usuario tabula al primer botón del grupo de controles y, a continuación, usa las flechas para seleccionar otros botones del grupo.

### Para cambiar el orden de selección de botones de un grupo de controles

1. En la [ventana Propiedades](#), seleccione el grupo en la lista **Objeto**. Un borde grueso indica que el grupo está en modo de edición.
2. Seleccione la ventana del **Diseñador de formularios**.
3. En el menú **Ver**, elija **Orden de tabulación**.
4. Establezca el orden de selección de la misma forma que lo haría para el orden de tabulación para controles.

### Establecer el texto de Información sobre herramientas

Cada control tiene una propiedad [ToolTipText](#) que permite especificar el texto que se muestra cuando el usuario detiene el puntero del *mouse* sobre el control. La Información sobre herramientas es especialmente útil para los botones que tienen iconos en lugar de texto.

### Para especificar el texto de Información sobre herramientas

- En la [ventana Propiedades](#), seleccione la propiedad **ToolTipText** y escriba el texto deseado.

La propiedad [ShowTips](#) del formulario determina si se muestra o no el texto de Información sobre herramientas.

## Cambiar la presentación del puntero del *mouse*

Puede cambiar la presentación del puntero del *mouse* para que proporcione pistas visuales a sus usuarios sobre los distintos estados en los que puede estar la aplicación.

Por ejemplo, en la clase `tsBaseForm` de la aplicación de ejemplo Importadores Tasmanian, un método `WaitMode` cambia el puntero del *mouse* al cursor de estado de espera predeterminado. Antes de ejecutar código que puede tardar cierto tiempo en ser procesado, la aplicación Importadores Tasmanian pasa el valor verdadero (.T.) al método `WaitMode` para cambiar el puntero e informar así al usuario de que se está procesando. Cuando se haya terminado el procesamiento, una llamada a `WaitMode` con falso (.F.) restablece el puntero del *mouse* predeterminado.

```
* Método WaitMode de la clase tsBaseForm
LPARAMETERS tlWaitMode

lnMousePointer = IIF(tlWaitMode, MOUSE_HOURLASS, MOUSE_DEFAULT)
thisform.MousePointer = lnMousePointer
thisform.SetAll('MousePointer', lnMousePointer)
```

Si quiere cambiar el puntero del *mouse* a otro que no sea uno de los punteros predeterminado, establezca la propiedad [MousePointer](#) a 99 - Personalizado y establezca la propiedad [MouseIcon](#) como su propio archivo de cursor (.cur) o de icono (.ico).

## Habilitar y deshabilitar controles

Establezca la [propiedad Enabled](#) del control como falsa (.F.) si la funcionalidad del control no está disponible en una determinada situación.

## Habilitar y deshabilitar botones de grupo

Puede habilitar o deshabilitar botones de opción o botones de comando individuales de un grupo si establece la [propiedad Enabled](#) de cada botón como verdadera (.T.) o falsa (.F.). También puede deshabilitar o habilitar todos los botones de un grupo si establece la propiedad `Enabled` del grupo, como en la línea de código siguiente:

```
frmForm1.cmdGroup1.Enabled = .T.
```

Cuando se establece como falsa (.F.) la propiedad `Enabled` de un [grupo de botones de opción](#) o de un [grupo de botones de comando](#), se deshabilitan todos los botones del grupo, pero no se muestran con las propiedades `ForeColor` y `BackColor` deshabilitadas. Al establecer la propiedad `Enabled` del grupo no cambia la propiedad `Enabled` de los botones individuales del grupo. De este modo, se puede deshabilitar un grupo de botones con algunos de los botones ya deshabilitados. Cuando se habilita el grupo, los botones que estaban deshabilitados originalmente permanecen deshabilitados.

Si desea deshabilitar todos los botones de un grupo para que aparezcan deshabilitados y no desea conservar información sobre qué botones estaban deshabilitados o habilitados originalmente, puede

utilizar el método [SetAll](#) del grupo, de este modo:

```
frmForm1.opgOptionGroup1.SetAll("Enabled", .F.)
```

## Permitir a los usuarios arrastrar y colocar

Cuando se diseñan aplicaciones de Visual FoxPro, se puede arrastrar texto, archivos y objetos desde la [Galería de componentes](#), el [Administrador de proyectos](#), el [Diseñador de bases de datos](#) y el [Diseñador de entornos de datos](#) hasta las ubicaciones deseadas de formularios e informes. Las características de [arrastrar y colocar](#) de Visual FoxPro permiten ampliar esta capacidad al usuario en [tiempo de ejecución](#).

Esta capacidad de arrastrar y colocar se extiende a operaciones de múltiples formularios. El usuario puede arrastrar texto, archivos y controles a cualquier lugar de la pantalla, incluidos otros formularios.

Ahora, Visual FoxPro admite dos tipos de operaciones de arrastrar y colocar: arrastrar y colocar de OLE y arrastrar y colocar controles. Arrastrar y colocar de OLE, le permite mover datos entre otras aplicaciones que admiten arrastrar y colocar de OLE (como Visual FoxPro, Visual Basic, el Explorador de Windows, Microsoft Word y Excel, etc.). En una aplicación distribuida de Visual FoxPro puede mover datos entre controles de la aplicación o entre controles y otras aplicaciones para Windows que admitan arrastrar y colocar de OLE.

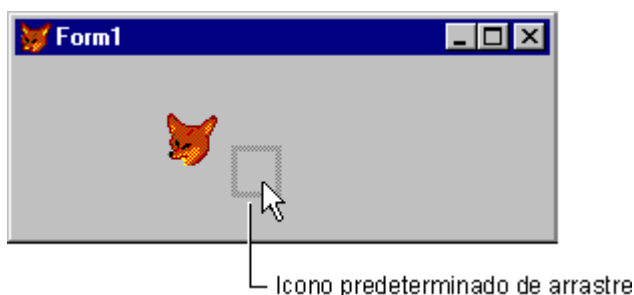
Arrastrar y colocar controles le permite arrastrar y colocar controles de Visual FoxPro en sus aplicaciones de Visual FoxPro. También se admite en versiones anteriores de Visual FoxPro. Cuando el usuario arrastra un control, Visual FoxPro proporciona un contorno gris del mismo tamaño que el objeto y que se mueve con el puntero del *mouse*. Puede pasar por alto este comportamiento predeterminado si especifica un archivo de cursor (.cur) para la [propiedad DragIcon](#) de un control.

Esta sección describe el proceso de arrastrar y colocar controles. Para obtener más información acerca de arrastrar y colocar de OLE, vea [Técnica arrastrar y colocar de OLE](#) en el capítulo 31, "Interoperabilidad e Internet".

Para ver ejemplos de arrastrar y colocar controles

1. Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **General**.

### Arrastrar un control Image en tiempo de ejecución



**Nota** Al arrastrar un control en tiempo de ejecución no se cambia su ubicación automáticamente. Puede hacerlo, pero deberá programar la reubicación, como se describirá en la sección "[Mover los controles en una operación de arrastrar y colocar](#)", más adelante en este mismo capítulo. En muchas ocasiones se suele arrastrar para indicar que debe llevarse a cabo alguna acción; el control mantiene su posición original cuando el usuario suelta el botón del *mouse*.

Si emplea las siguientes propiedades, eventos y métodos de [arrastrar y colocar](#), podrá especificar el significado de una operación de arrastrar y cómo puede iniciarse el arrastre (si se inicia) para un determinado control.

Para	Utilice esta característica
Activar el arrastre automático o manual de un control.	<a href="#">Propiedad DragMode</a>
Especificar qué icono se muestra cuando se arrastra el control.	<a href="#">Propiedad DragIcon</a>
Reconocer cuándo se coloca un control en el objeto.	<a href="#">Evento DragDrop</a>
Reconocer cuándo se arrastra un control sobre el objeto.	<a href="#">Evento DragOver</a>
Iniciar o detener un arrastre manual.	<a href="#">Método Drag</a>

Todos los controles visuales pueden arrastrarse en [tiempo de ejecución](#) y todos los controles comparten las propiedades que se indican en la tabla anterior. Los formularios reconocen los eventos DragDrop y DragOver, pero no tienen propiedades DragMode y DragIcon.

### Habilitar el modo de arrastre automático

Para permitir que el usuario arrastre un control siempre que haga clic en el control, establezca su [propiedad DragMode](#) como 1. De este modo se habilitará el arrastre automático del control. Cuando se establece como Automático, el arrastre siempre está activado.

**Nota** Mientras se produce una operación de arrastre automático, el control que se arrastra no reconoce otros eventos del *mouse*.

### Respuestas cuando el usuario coloca el objeto

Cuando el usuario suelta el botón del *mouse* después de arrastrar un control, Visual FoxPro genera un [evento DragDrop](#). Hay muchas formas de responder a este evento. Se puede colocar el control en la nueva ubicación (indicada por la última posición del contorno gris). Recuerde que el control no se mueve automáticamente a la nueva ubicación.

Hay dos términos muy importantes relativos a las operaciones de arrastrar y colocar: *origen* y *destino*.

Término	Significado
Origen	El control que se arrastra.
Destino	El objeto sobre el que el usuario coloca el control. Este objeto, que puede ser un formulario o un control, reconoce el evento DragDrop.

Un [control](#) se convierte en el destino si la posición del *mouse* se encuentra dentro de sus bordes cuando se suelta el botón. Un [formulario](#) es el destino si el puntero está en una parte en blanco del formulario.

El evento DragDrop recibe tres [parámetros](#): *oSource*, *nXCoord* y *nYCoord*. El parámetro *oSource* es una referencia al control que se colocó sobre el destino. Los parámetros *nXCoord* y *nYCoord* contienen, respectivamente, las coordenadas horizontal y vertical del puntero del *mouse* dentro del destino.

Puesto que *oSource* es un [objeto](#), se utiliza del mismo modo que un control; se puede hacer referencia a sus [propiedades](#) o llamar a uno de sus [métodos](#). Por ejemplo, las instrucciones siguientes del código asociado al evento DragDrop comprueban si el usuario ha colocado un control sobre sí mismo:

```
LPARAMETERS oSource, nXCoord, nYCoord
IF oSource.Name != THIS.Name
    * Realizar alguna acción.
ELSE
    * El control se ha colocado sobre sí mismo.
    * Realizar otra acción.
ENDIF
```

Todos los tipos posibles de control para *oSource* tienen una [propiedad Visible](#). Por lo tanto, un control puede hacerse invisible cuando se coloca en una determinada parte de un formulario o sobre otro control. La siguiente línea del código asociado al evento DragDrop de un [control Image](#) hace que un control arrastrado desaparezca cuando se coloca sobre una imagen:

```
LPARAMETERS oSource, nXCoord, nYCoord
oSource.Visible = .F.
```

### Indicar zonas de colocación válidas

Al habilitar arrastrar y colocar, puede ayudar a los usuarios; para ello, incluya claves visuales sobre dónde pueden y dónde no pueden colocar un control. Para ello, lo mejor que se puede hacer es cambiar la propiedad [DragIcon](#) del origen en el código asociado al [evento DragOver](#).

El siguiente código del evento DragOver de un control indica a un usuario que el control no es un destino de colocación válido. En este ejemplo, *cOldIcon* es una propiedad del formulario definida por el usuario.

```
LPARAMETERS oSource, nXCoord, nYCoord, nState
DO CASE
    CASE nState = 0 && Entrar
        THISFORM.cOldIcon = oSource.DragIcon
        oSource.DragIcon = "NODROP01.CUR"
```

```
CASE nState = 1 && Salir
    oSource.DragIcon = THISFORM.cOldIcon
ENDCASE
```

## Control del inicio y el final de arrastre

Visual FoxPro tiene un valor Manual para la [propiedad DragMode](#) que proporciona más control que el valor Automático. El valor Manual permite especificar cuándo se puede y cuándo no se puede arrastrar un control. (Cuando DragMode se establece como Automático, el control siempre se puede arrastrar a condición de que no se cambie el valor).

Por ejemplo, puede que necesite habilitar el arrastre como respuesta a eventos MouseDown y MouseUp o como respuesta a un comando del teclado o de menú. La configuración Manual también permite reconocer un evento MouseDown antes de comenzar a arrastrar, de modo que pueda registrar la posición del *mouse*.

Para habilitar el arrastre desde código, deje DragMode en su valor predeterminado (0 – Manual). A continuación, utilice el [método Drag](#) cuando quiera comenzar o detener el arrastre de un objeto:

```
container.control.Drag(nAction)
```

Si *nAction* es 1, el método Drag iniciará el arrastre del control. Si *nAction* es 2, se colocará el control, lo cual producirá un evento DragDrop. El valor 0 para *nAction* cancela el arrastre. El efecto es similar a dar el valor 2, salvo que no se produce ningún evento DragDrop.

**Nota** Para activar la capacidad de arrastrar y colocar desde un [cuadro de lista](#), el mejor lugar para llamar al método Drag es el código asociado al evento MouseMove del cuadro de lista de origen, después de determinar que el botón del *mouse* está presionado. Para ver un ejemplo, consulte Lmover.scx en el directorio ...\\Samples\\Vfp98\\Solution\\Controls\\Lists de Visual Studio.

## Mover los controles en una operación de arrastrar y colocar

Puede interesar que el control de origen cambie de posición cuando el usuario libere el botón del *mouse*. Para mover un control a la nueva ubicación del *mouse*, use el [método Move](#). Por ejemplo, el siguiente código de evento DragDrop de un formulario mueve el control que se arrastra a la nueva ubicación:

```
LPARAMETERS oSource, nXCoord, nYCoord
oSource.Move(nXCoord, nYCoord)
```

Es posible que este código no produzca con precisión los efectos deseados, ya que la esquina superior izquierda del control está situada en la ubicación del *mouse*. El código siguiente sitúa el centro del control en la ubicación del *mouse*:

```
LPARAMETERS oSource, nXCoord, nYCoord
oSource.Move ((nXCoord - oSource.Width / 2), ;
    (nYCoord - oSource.Height / 2))
```

El código funciona mejor cuando se establece [DragIcon](#) con un valor distinto del predeterminado (rectángulo gris). Cuando se usa el rectángulo gris, normalmente se desea mover el control de forma precisa a la posición final del rectángulo gris. Para ello, registre la posición inicial del *mouse* en el



control de origen. A continuación, use esta posición como referencia cuando mueva el control. Para ver un ejemplo, vea Ddrop.scx en el directorio ...\\Samples\\Vfp98\\Solution\\Forms de Visual Studio.

### Para registrar la posición inicial del *mouse*

1. Especifique el arrastre manual del control.
2. Declare dos [variables](#) a nivel de formulario, nDragX y nDragY.
3. Active el arrastre cuando se produzca un [eventoMouseDown](#). Además, almacene el valor de *nXCoord* y *nYCoord* en las variables a nivel de formulario de este evento.
4. Desactive el arrastre cuando se produzca el [eventoMouseUp](#).

## Ampliar formularios

Los marcos de página permiten ampliar la superficie de los formularios y los [controles ActiveX](#) permiten ampliar la funcionalidad de los formularios.

### Usar marcos de página



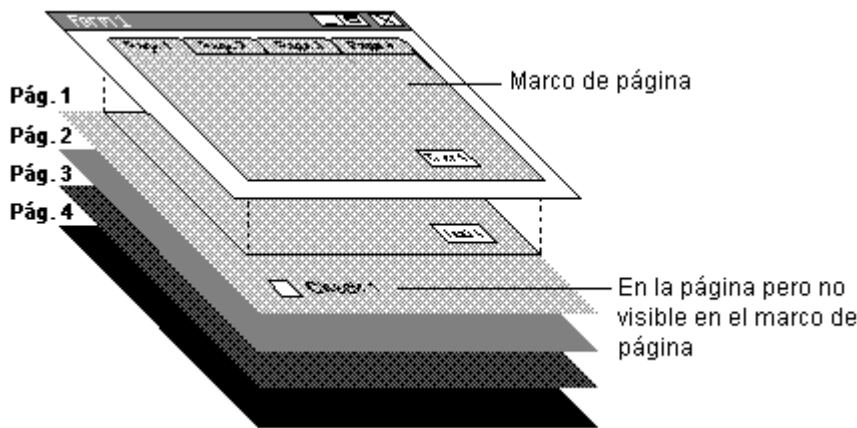
Un marco de página es un objeto contenedor que contiene páginas. A su vez, las páginas contienen controles. Las propiedades pueden establecerse a nivel de marco de página, de página o de control.

### Para ver ejemplos del uso de marcos de páginas

1. Ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.
2. En la vista de árbol, haga clic en **Controles** y, a continuación, haga clic en **Marco de páginas**.

El marco de página puede considerarse como un contenedor tridimensional que presenta [páginas](#) en capas. Sólo los controles de la página superior (o sobre el marco de página) pueden estar visibles y activos.

### Múltiples páginas en un marco de página de un formulario



El marco de página define la ubicación de las páginas y la cantidad de página que está visible. La esquina superior izquierda de una página está acoplada a la esquina superior izquierda del marco de página. Los controles pueden situarse en páginas que van más allá de las dimensiones del marco de página. Estos controles están activos, pero no son visibles a menos que se cambien mediante programación las propiedades [Height](#) y [Width](#) del marco de página para hacer visibles los controles.

### Usar páginas en una aplicación

Con marcos de página y páginas, se pueden crear formularios o cuadros de diálogo con fichas con los mismos tipos de capacidades de interfaz que se ven en el Administrador de proyectos.

Asimismo, los marcos de página permiten definir una región del formulario en la que pueden intercambiarse fácilmente controles. Por ejemplo, en los [Asistentes](#), la mayor parte del formulario permanece constante, pero un área del formulario cambia con cada paso. En lugar de crear cinco formularios con pasos de asistente, puede crear un formulario con un marco de página y cinco páginas.

Solution.app, en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio, contiene dos ejemplos que demuestran el uso de marcos con y sin [fichas](#).

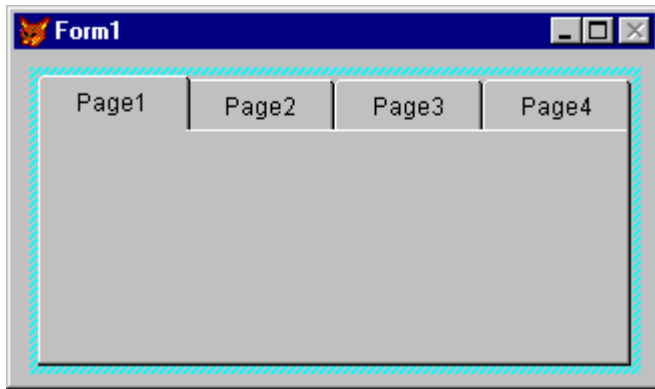
### Agregar marcos de página a un formulario

Puede incluir uno o más marcos de página en cualquier formulario.

#### Para agregar un marco de página a un formulario

1. En la [barra de herramientas Controles de formularios](#), elija el botón **Marco de página** y arrástrelo para ajustar su tamaño en la ventana **Formulario**.
2. Establezca la propiedad PageCount para indicar el número de páginas que se van a incluir en el marco.

#### Marco de página con cuatro páginas



3. En el menú contextual del marco, elija **Modificar** para activar el marco como contenedor. El borde del marco de página se amplía para indicar que está activo.
4. Agregue controles del mismo modo que en un formulario.

**Nota** Al igual que otros controles de contenedor, seleccione el marco de página y elija **Modificar** en el menú que aparece al presionar el botón secundario del *mouse* o seleccione el contenedor en la lista desplegable "Objeto" de la [ventana Propiedades](#), de modo que se seleccione el contenedor (es decir, tenga un borde más amplio) antes de agregar controles a la página que está diseñando. Si no activa el marco de página como contenedor antes de agregar controles, los controles se agregarán al [formulario](#) en lugar de a la [página](#), aunque puede parecer que están en la página.

#### Para seleccionar otra página en el marco de página

1. Active el marco de páginas como contenedor; para ello, haga clic con el botón secundario del *mouse* y elija **Modificar**.
2. Seleccione la ficha de la página que quiere usar.
  - O bien–
  - Seleccione la página en el cuadro **Objeto** de la [ventana Propiedades](#).
  - O bien–
  - Seleccione la página en el cuadro **Página** en la parte inferior del [Diseñador de formularios](#).

#### Agregar controles a una página

Cuando se agregan controles a una página, sólo están visibles y activos cuando su página está activa.

#### Para agregar controles a una página

1. En el cuadro **Objeto** de la ventana [Propiedades](#), seleccione la página. Aparecerá un borde alrededor del marco de página que indica que puede manipular los [objetos](#) contenidos en ella.

2. En la barra de herramientas [Controles de formularios](#), elija el botón del control que desea y arrástrelo para ajustarlo a la página.

### Administrar títulos largos en fichas de página

Si los [títulos](#) de las [fichas](#) son más largos de lo que puede mostrarse en la ficha dado el ancho del marco de página y el número de páginas, dispone de dos opciones:

- Establezca la propiedad [TabStretch](#) como **1 - Recortar** para mostrar sólo los caracteres de los títulos que se ajustan a las fichas. Recortar es el valor predeterminado.
- Establezca la propiedad [TabStretch](#) como **0 - Pila** para apilar las fichas de modo que sea visible el título completo de todas las fichas.

### Cambiar páginas mediante programación

Tanto si el marco de página se muestra con [fichas](#) como si no, es posible convertir en activa una página mediante programación con la [propiedad ActivePage](#). Por ejemplo, el siguiente código del procedimiento de evento Click de un botón de comando de un formulario cambia la página activa de un marco de páginas del formulario en la tercera página:

```
THISFORM.pgfOptions.ActivePage = 3
```

### Propiedades comunes de los marcos de páginas

Las propiedades siguientes de los marcos de páginas suelen establecerse en tiempo de diseño.

Propiedad	Descripción
<a href="#">Tabs</a>	Especifica si las fichas son visibles o no para las páginas.
<a href="#">TabStyle</a>	Especifica si las fichas tienen o no el mismo tamaño y si juntas ocupan el mismo ancho que el marco de páginas.
<a href="#">PageCount</a>	El número de páginas del marco de página.

### Control contenedor OLE



Agregue un objeto OLE a un formulario; para ello haga clic en esta herramienta y arrástrela para ajustar su tamaño en la ventana Formulario. Esta herramienta puede representar un objeto servidor como Microsoft Excel o Microsoft Word, o puede representar un control ActiveX si el directorio SYSTEM de Windows contiene controles ActiveX (archivos con una extensión .ocx). Para obtener información general sobre los controles ActiveX, consulte el capítulo 16, [Agregar OLE](#).

### Control ActiveX dependiente



Puede crear un objeto ActiveX dependiente en un formulario; para ello, haga clic en esta herramienta y arrástrela para ajustar su tamaño en la ventana Formulario. Después de crear el objeto, conéctelo a un campo General de la tabla. A continuación, utilice el objeto para mostrar el contenido del campo. Por ejemplo, si almacena documentos de Word en un campo de tipo General, podrá mostrar el contenido de estos documentos con un objeto OLE dependiente en un formulario.

### Para crear un objeto ActiveX dependiente

1. Cree o abra un formulario.
2. En la barra de herramientas [Controles de formularios](#), elija el botón **Control ActiveX dependiente** y arrástrelo para ajustar su tamaño en el formulario.
3. Vincule el objeto ActiveX a un campo de tipo General; para ello, establezca la propiedad ControlSource del objeto.

Para ver un ejemplo de cómo se usa el control ActiveX dependiente, consulte el capítulo 16, [Agregar OLE](#).

## Capítulo 11: Diseñar menús y barras de herramientas

Los [menús](#) y [barras de herramientas](#) proporcionan una forma estructurada y accesible para que los usuarios aprovechen los comandos y las herramientas contenidas en sus aplicaciones. La preparación y el diseño apropiados de menús y barras de herramientas asegurarán que se ofrece la funcionalidad clave de su aplicación y que los usuarios no se sentirán frustrados al utilizar la aplicación.

Para obtener más información acerca de la personalización de barras de herramientas de Visual FoxPro, consulte el Capítulo 3, [Configurar Visual FoxPro](#), en la *Guía de instalación*.

En este capítulo se tratan los temas siguientes:

- [Usar menús en sus aplicaciones](#)
- [Crear barras de herramientas personalizadas](#)
- [Probar y depurar un sistema de menús](#)
- [Personalizar un sistema de menús](#)

### Usar menús en sus aplicaciones

A menudo, los usuarios examinan los [menús](#) antes de buscar información sobre la aplicación en cualquier otro lugar. Si los menús están bien diseñados, los usuarios podrán comprender la aplicación y desarrollar un modelo mental basado únicamente en la organización de los menús y su contenido. El Diseñador de menús de Visual FoxPro permite crear menús con los que mejorar la calidad de sus aplicaciones.

Cada componente de una aplicación de Visual FoxPro puede tener su propio [sistema de menús](#) o conjunto de menús. En las secciones siguientes se describe la forma de crear un sistema de menús, pero no se indica cómo incorporarlo a una aplicación. Si desea información al respecto, consulte el capítulo 13, [Compilar una aplicación](#).

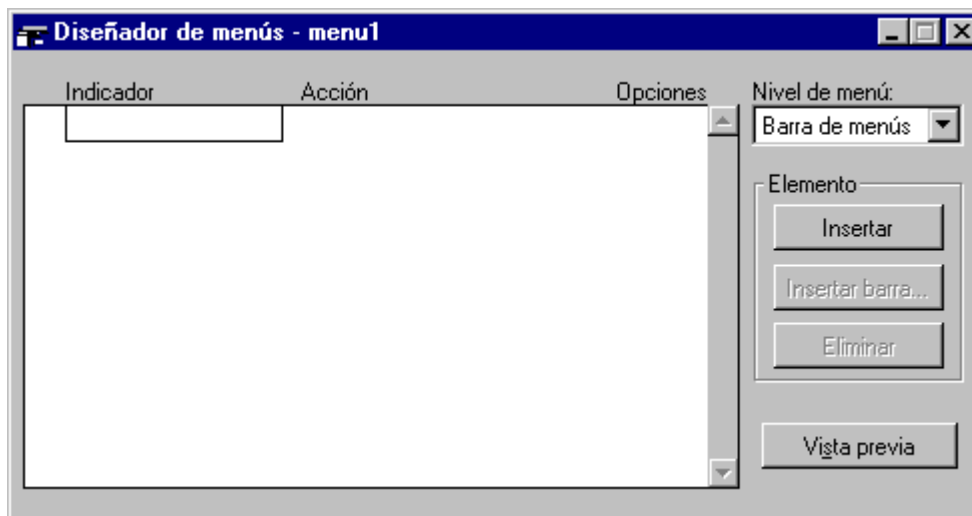
En las secciones siguientes se describe:

- [Crear un sistema de menús](#)
- [Diseñar un sistema de menús](#)
- [Crear menús, elementos de menú y submenús](#)
- [Asignar tareas a un sistema de menús](#)

## Crear un sistema de menús

La mayor parte del trabajo en la creación de un sistema de menús se realiza en el Diseñador de menús, en el que crea los menús, los submenús y las opciones de menú.

### Diseñador de menús



La creación de un sistema de menús implica varios pasos. Independientemente del tamaño de la aplicación y la complejidad de los menús que pretenda usar, debe:

- Preparar y diseñar el sistema.

Decida los menús que necesita, dónde deben aparecer en la interfaz, cuáles requieren submenús, etc. Si desea más información sobre el diseño de sistemas de menús, consulte [Diseñar un sistema de menús](#), más adelante en este mismo capítulo.

- Crear los menús y submenús.

Defina los títulos, elementos de menú y submenús mediante el Diseñador de menús.

- Asignar tareas al sistema de menús.

Especifique las tareas que los menús deben realizar, como mostrar formularios y cuadros de diálogo. Incluya también un [preprograma](#) y [postprograma](#) si es conveniente. El preprograma se ejecuta antes de que se defina el sistema de menús y puede incluir código para abrir archivos, declarar [variables](#) de memoria o colocar el sistema de menús en una [pila](#) de forma que se pueda recuperar después. El postprograma contiene los procedimientos que usted escribe, se ejecuta después del código de definición de los menús y determina los elementos de menú disponibles o no disponibles para su selección.

- Generar el programa de menú.
- Ejecutar el programa para probar el sistema de menús.

## Diseñar un sistema de menús

La utilidad de una aplicación puede depender de la calidad de sus [sistemas de menús](#). Si dedica tiempo de diseño a los menús, los usuarios los aceptarán con facilidad y los aprenderán rápidamente.

Al diseñar un sistema de menús, considere las directrices siguientes:

- Organice el sistema según las tareas que vayan a realizar los usuarios y no según la jerarquía de los programas de la aplicación.

Los usuarios pueden formarse un modelo mental de la organización de la aplicación si observan los menús y sus elementos. Para diseñar estos menús y elementos de forma eficaz, debe saber cómo piensan los usuarios y cómo realizan su trabajo.

- Dé a cada menú un título significativo.
- Organice los elementos de los menús según su frecuencia de uso esperada, siguiendo su secuencia lógica o por orden alfabético.

Si no puede predecir la frecuencia de uso ni determinar un orden lógico, ordene los elementos de menú alfabéticamente. El orden alfabético es especialmente efectivo cuando un menú contiene más de ocho elementos. Con tantos elementos, el usuario dedica tiempo a la búsqueda y la ordenación alfabética la facilitará.

- Coloque líneas separadoras entre los grupos lógicos de elementos de menús.
- Limite a una pantalla el número de elementos de un menú.
- Si el número excede la longitud de una pantalla, cree submenús para los elementos en los que resulte conveniente.
- Elija [teclas de acceso](#) y métodos abreviados de teclado para los menús y sus elementos.

Por ejemplo, ALT+F podría ser una tecla de acceso para un menú Archivo.

- Utilice palabras que definan claramente los elementos de los menús.

Utilice palabras comunes, evitando la jerga informática y utilice verbos simples y activos para indicar las acciones que resultarán de elegir cada elemento de menú. No utilice nombres como verbos. Describa también los elementos de menús con construcciones paralelas. Por ejemplo, si utiliza una única palabra para todos los elementos, use el mismo tipo de palabra para todos

ellos.

- Utilice letras mayúsculas y minúsculas en los elementos de menú.

**Sugerencia** Para ver un buen sistema de menús, ejecute la aplicación Importadores Tasmanian (TASTRADE.APP), situada en el directorio SAMPLES\MAINSAMP.

## Crear menús, elementos de menú y submenús

Una vez diseñado el sistema de menús, puede crearlo con el Diseñador de menús. Puede crear menús, elementos de menú, submenús, líneas para separar grupos de elementos relacionados, etc. En las secciones siguientes se explica detalladamente la manera de hacerlo.

### Crear menú

Para crear menús, puede personalizar el sistema de menús de Visual FoxPro o bien desarrollar un sistema propio. Para partir del sistema de menús de Visual FoxPro, utilice la función Menú rápido.

### Para crear un sistema de menús con Menú rápido

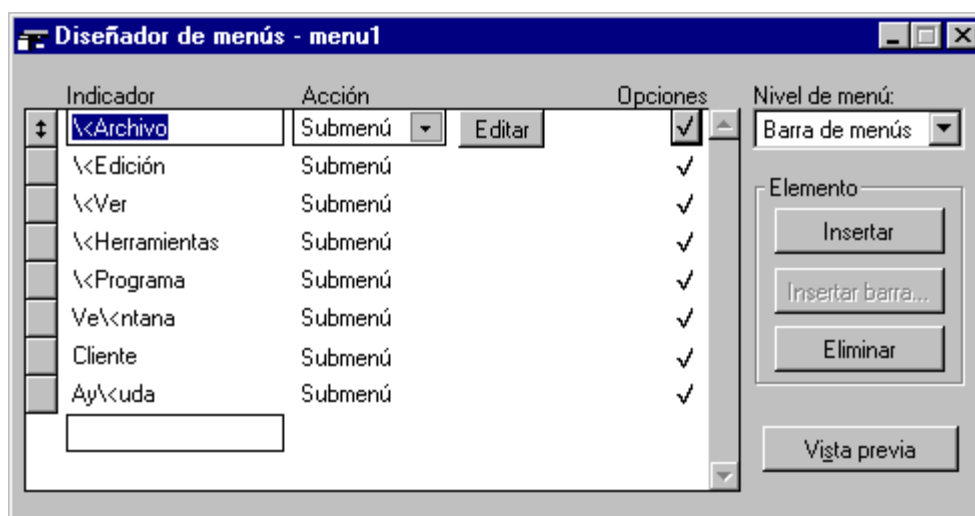
1. En el [Administrador de proyectos](#), seleccione la ficha **Otros**, seleccione **Menús** y haga clic en **Nuevo**.
2. Elija **Menú**.

Aparecerá el **Diseñador de menús**.

3. En el menú **Menú**, elija **Menú rápido**.

El **Diseñador de menús** contendrá ahora información sobre los menús principales de Visual FoxPro.

### Sistema de menús creado con la función Menú rápido

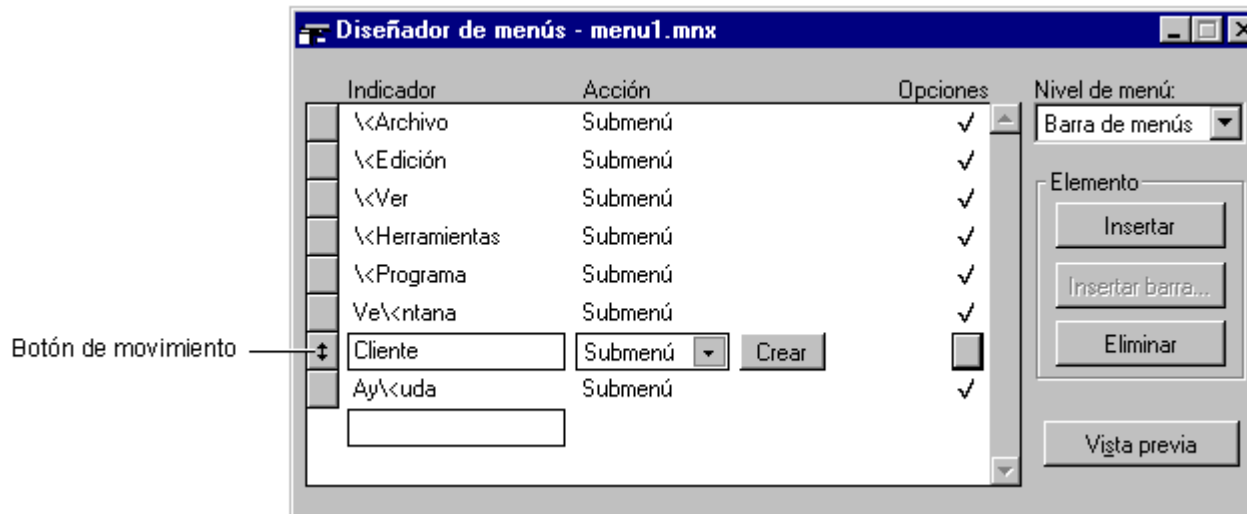




- Personalice el sistema agregando o modificando elementos de menú.

Por ejemplo, inserte un menú Cliente delante del menú Ayuda; para ello, haga clic en el botón de movimiento asociado al menú Ayuda, haga clic en el botón **Insertar** y escriba **Cliente** en la columna **Indicador**. El resultado tendrá el siguiente aspecto:

### Sistema de menús personalizado



**Sugerencia** Arrastre los botones de movimiento para cambiar la posición de los menús en la barra de menús.

Si necesita un menú Ayuda, sitúelo en último lugar en la barra de menús, de modo que los usuarios puedan encontrarlo rápidamente.

Antes de poder usar el menú en una aplicación debe generarlo.

### Para generar un menú

- En el [menú Menú](#), elija **Generar**.

Visual FoxPro le pedirá que guarde el sistema de menús en un archivo con la extensión .MNX. Este archivo es una tabla que almacena toda la información sobre el sistema de menús. Después de haber guardado el sistema de menús, Visual FoxPro le pedirá un archivo de salida con la extensión .MPR. Este archivo contendrá el programa de menú generado.

### Crear un menú contextual

Los menús contextuales aparecen cuando hace clic con el botón secundario del *mouse* en un [control](#) o un [objeto](#), y proporcionan una forma rápida de ofrecer todas las funciones que se aplican a ese objeto exclusivamente. Puede usar Visual FoxPro para crear menús contextuales y, a continuación, adjuntar estos menús a controles. Por ejemplo, puede crear un menú contextual que contenga los comandos Cortar, Copiar y Pegar, y que aparecerá cuando el usuario haga clic con el botón secundario en datos contenidos en un [control Grid](#).

### Para crear un menú contextual

1. En el [Administrador de proyectos](#), seleccione la ficha **Otros**, seleccione **Menús** y haga clic en **Nuevo**.
2. Elija **Menú contextual**.

Aparecerá el **Diseñador de menús contextuales**.

Cuando está en el Diseñador de menús contextuales, el proceso para agregar elementos de menú es el mismo que para la creación de menús.

Para ver un ejemplo de menús contextuales, ejecute Solution.app, ubicado en el directorio ...  
\\Samples\\Vfp98\\Solution de Visual Studio.

### Crear menús SDI

Los menús SDI son menús que aparecen en ventanas de [interfaz de un único documento](#) (SDI). Para crear un menú SDI debe indicar que el menú se va a usar en un formulario SDI cuando está diseñando el menú. Aparte de esto, el proceso de creación de un menú SDI es el mismo que el de creación de un menú normal.

### Para crear un menú SDI

- Mientras el [Diseñador de menús](#) está abierto, elija **Opciones generales** en el menú **Ver** y seleccione **Formulario de nivel superior**.

### Crear elementos de menú

Una vez creados los menús, puede colocar en ellos elementos de menú. Los elementos de menú pueden representar comandos o procedimientos de Visual FoxPro que tenga que ejecutar el usuario o bien pueden contener submenús que a su vez ofrezcan otros elementos.

### Para agregar elementos a un menú

1. En la columna **Indicador**, haga clic en el título del menú al que desee agregar elementos.
2. En el cuadro **Acción**, seleccione **Submenú**.

Aparecerá un botón **Crear** a la derecha de la lista.

3. Haga clic en el botón **Crear**.

Aparecerá una ventana de diseño vacía, en la que puede introducir los elementos de menú.

4. En la columna **Indicador**, escriba los nombres de los nuevos elementos de menú.

## Crear submenús

Para cada elemento de menú, puede crear un submenú con elementos adicionales.

### Para crear un submenú

1. En la columna **Indicador**, haga clic en el elemento al que desee agregar un submenú.
2. En el cuadro **Acción**, seleccione **Submenú**.

Aparecerá un botón **Crear** a la derecha de la lista. Si ya existe un submenú, aparecerá un botón **Modificar**.

3. Haga clic en el botón **Crear** o **Modificar**.
4. En la columna **Indicador**, escriba los nombres de los nuevos elementos de menú.

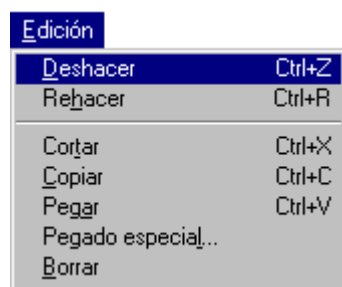
### Agregar menús mediante programa

Aunque normalmente los menús y sus elementos se crean con el Diseñador de menús, también es posible crearlos con comandos de Visual FoxPro. Por ejemplo, puede crear un menú con [DEFINE PAD](#), un submenú con [DEFINE POPUP](#) y elementos del submenú con una serie de comandos [DEFINE BAR](#).

### Agrupar elementos de menú

Para mejorar la legibilidad, puede separar grupos de elementos de menú similares con líneas divisorias. Por ejemplo, en Visual FoxPro, el menú Edición tiene una línea que separa los comandos Deshacer y Rehacer de los comandos Cortar, Copiar, Pegar y Pegado especial.

### Elementos de menú agrupados



### Para agrupar elementos de menú

1. En la columna **Indicador**, escriba \-. De esta forma se crea una línea divisoria.
2. Arrastre el botón situado a la izquierda del indicador \- para desplazar la línea divisoria hasta situarla en el lugar deseado.

## Guardar un menú como HTML

Puede usar la opción **Guardar como HTML** del menú **Archivo** cuando cree un menú para guardar su contenido como un archivo HTML (Lenguaje de marcado de hipertexto).

### Para guardar un formulario como HTML

1. Abra el menú.
2. Elija **Guardar como HTML** en el menú **Archivo**. (Se le pedirá que guarde el menú si lo ha modificado).
3. Escriba el nombre del archivo HTML que desea crear y elija **Guardar**.

## Incluir menús en una aplicación

Cuando haya creado un [sistema de menús](#), puede incluirlo en su aplicación.

### Para incluir un sistema de menús en la aplicación

- Agregue el archivo .MNX al proyecto y, a continuación, genere la aplicación desde el proyecto. Para obtener más información sobre la generación de aplicaciones, consulte el capítulo 13, [Compilar una aplicación](#).

### Adjuntar menús contextuales a controles

Cuando haya creado y generado un [menú contextual](#), puede adjuntarlo a un control. Los menús contextuales aparecen típicamente cuando el usuario hace clic con el botón secundario en un control. Puede adjuntar un menú contextual a un [control](#) específico si escribe una pequeña cantidad de código en el evento clic asociado al botón secundario del control.

1. Seleccione el control al que desea adjuntar el menú contextual.
2. En la ventana [Propiedades](#), elija la ficha **Métodos** y seleccione **RightClick Event**.
3. En la ventana de código, escriba **DO *menú*.MPR**, en donde *menú* es el nombre del menú contextual.

**Nota** Asegúrese de usar la extensión .mpr al hacer referencia a los menús contextuales.

### Adjuntar menús SDI a formularios

Cuando cree un menú SDI, puede adjuntarlo a un formulario SDI. Además, debe:

- Establecer la [propiedad ShowWindow](#) del formulario.
- Agregar una instrucción [DO](#) al [evento Init](#) del formulario.

## Para adjuntar un menú SDI a un formulario

1. En el [Diseñador de formularios](#), establezca la propiedad ShowWindow del formulario a **2 – Como formulario de nivel superior**.
2. En el evento Init del formulario, llame al menú.

Por ejemplo, si el menú se llama `SDIMENU.MPR`, agregue el código siguiente:

```
DO SDIMENU.MPR WITH THIS, .T.
```

## Asignar tareas a un sistema de menús

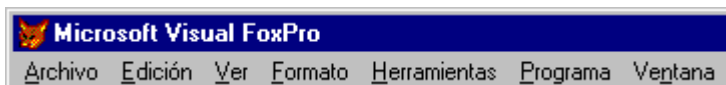
Al crear un [sistema de menús](#), debe considerar la facilidad de acceso al mismo y también asignarle tareas. Es conveniente definir teclas de acceso para facilitar la entrada al sistema de menús, y hay que asignar a los menús y a sus elementos tareas que deben realizar, como mostrar [formularios](#), [barras de herramientas](#) y otros sistemas de menús. Debe definir [teclas de acceso](#) para permitir la entrada al sistema de menús. También puede agregar métodos abreviados de teclado y activar o desactivar elementos de menú para conseguir un mayor control.

### Asignar teclas de acceso

Los menús bien diseñados cuentan con [teclas de acceso](#) para conseguir un acceso rápido por el teclado a la funcionalidad del menú. La tecla de acceso se representa en el título o elemento de menú con una letra subrayada. Por ejemplo, el menú Archivo de Visual FoxPro utiliza la "A" como tecla de acceso.

Si no asigna una tecla de acceso a un título o elemento de menú, Visual FoxPro asignará automáticamente la primera letra como tecla de acceso. Por ejemplo, para el menú Cliente creado anteriormente no se definió ninguna tecla de acceso y por ello Visual FoxPro le asignó su primera letra (C).

### Menús con teclas de acceso



## Para especificar la tecla de acceso de un menú o elemento de menú

- Escriba \< a la izquierda de la letra que desee que actúe como tecla de acceso.

Por ejemplo, para hacer que la tecla de acceso del título del menú Cliente sea "I", sustituya **Cliente** por **C\<liente** en la columna Indicador.

**Solución de problemas** Si una tecla de acceso a un menú no funciona, compruebe si hay teclas de acceso duplicadas.

### Asignar métodos abreviados de teclado

Además de asignar teclas de acceso, puede especificar métodos abreviados de teclado para los menús o sus elementos. Como ocurre con las teclas de acceso, los métodos abreviados de teclado permiten elegir un menú o un elemento de menú al mantener presionada una tecla mientras presiona otra. La diferencia entre las teclas de acceso y los métodos abreviados de teclado es que estos últimos pueden utilizarse para elegir un elemento de menú sin tener que mostrar primero el menú correspondiente.

Los métodos abreviados de teclado para los elementos de menú de Visual FoxPro son combinaciones de la tecla CTRL o ALT con otra tecla. Por ejemplo, en Visual FoxPro puede crear un archivo nuevo si presiona CTRL+N

### Para especificar un método abreviado de teclado para un menú o elemento de menú

1. En la columna **Indicador**, haga clic en el título o elemento de menú correspondiente.
2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.
3. En el cuadro **Etiqueta de tecla**, presione una combinación de teclas para crear un método abreviado de teclado.

Si un elemento de menú no tiene un método abreviado de teclado, Visual FoxPro mostrará "(presione la tecla)" en el cuadro **Etiqueta de tecla**.

4. En el cuadro **Texto de tecla**, agregue el texto que desee que aparezca junto al elemento de menú.

De forma predeterminada, Visual FoxPro repite el método abreviado de teclado del cuadro **Etiqueta de tecla** en el cuadro **Texto de tecla**. Sin embargo, puede modificar el contenido de **Texto de tecla**. Por ejemplo, si tanto en **Etiqueta de tecla** y **Texto de tecla** aparece CTRL+R, puede cambiar el contenido de **Texto de tecla** por ^R.

**Nota** CTRL+J es un método abreviado de teclado no válido porque se usa para cerrar ciertos cuadros de diálogo en Visual FoxPro.

### Activar y desactivar elementos de menú

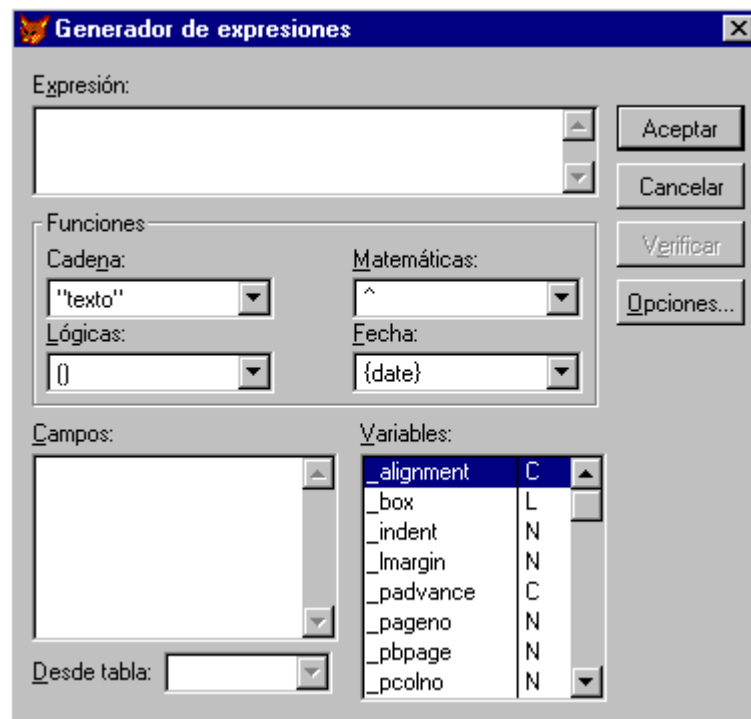
Puede activar o desactivar un menú o un elemento de menú basándose en una condición lógica.

### Para activar o desactivar un menú o elemento de menú

1. En la columna **Indicador**, haga clic en el título o elemento de menú correspondiente.
2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.
3. Haga clic en la casilla de verificación **Saltar por**.

Aparecerá el **Generador de expresiones**.

## Generador de expresiones



4. En el cuadro **Saltar por**, escriba la expresión que determinará si el menú está activado o desactivado.

Si la expresión da como resultado falso (.F.), el menú o elemento quedará activado. Si la expresión da como resultado verdadero (.T.), el menú o elemento se desactivará y no se podrá seleccionar o elegir. Si desea más información al respecto, vea los temas [DEFINE BAR](#) y [DEFINE PAD](#).

**Nota** Una vez mostrado el sistema de menús, puede activar y desactivar los menús y sus elementos con el comando [SET SKIP OF](#).

### Marcar el estado de un elemento de menú

En un menú, una marca de verificación junto a un elemento indica que está activo. Por ejemplo, si coloca una marca de verificación junto al elemento Crédito del menú Cliente antes creado, esta opción quedará activada.

En tiempo de ejecución puede colocar una marca de verificación junto a un elemento de menú mediante el comando [SET MARK OF](#).

Para ver un ejemplo sobre cómo desactivar y marcar el estado de elementos de menú, ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.

### Asignar tareas a menús o a elementos de menú

Cuando se selecciona un menú o un elemento de menú, éste realiza una tarea, como mostrar un [formulario](#), una [barra de herramientas](#) o bien otro sistema de menús. Para realizar una tarea, el menú o elemento deberá ejecutar un comando de Visual FoxPro. Este comando puede estar contenido en una línea o puede ser una llamada a procedimiento.

**Sugerencia** Escriba un procedimiento si va a utilizar el mismo conjunto de comandos en varios lugares. Debe dar nombre al procedimiento explícitamente y debe escribir un [postprograma](#) en el menú o en algún lugar en el que menú u [objeto](#) puede hacer referencia al mismo.

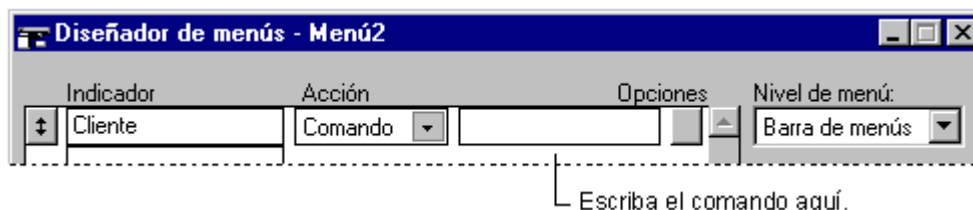
### Realizar tareas con comandos

Para realizar una tarea, puede asignar un comando a un menú o elemento de menú. Puede tratarse de cualquier comando válido de Visual FoxPro, incluyendo una llamada a un programa que exista en la ruta de acceso o a un procedimiento definido en la opción Postprograma del [cuadro de diálogo Opciones generales](#). Para obtener más información, consulte [Crear un procedimiento predeterminado para un sistema de menús](#), más adelante en este mismo capítulo.

### Para asignar un comando a un menú o elemento de menú

1. En la columna **Indicador**, haga clic en el título o elemento de menú correspondiente.
2. En el cuadro **Acción**, elija **Comando**.
3. En el cuadro situado a la derecha del cuadro **Acción**, escriba el comando correspondiente:

### Asignar un comando a un menú



Si el comando llama a un procedimiento del postprograma del menú, utilice el comando [DO](#) con la [sintaxis](#) siguiente:

`DO nombreproc IN nombremenú`

En esta sintaxis, *nombremenú* especifica la ubicación del procedimiento. Se trata del nombre del archivo de menú y debe tener la extensión .mpr. Si no especifica la ubicación en *nombremenú*, deberá hacerlo con [SET PROCEDURE](#) a *nombremenú.mpr*, si el procedimiento se encuentra en el postprograma del menú.

### Mostrar formularios y cuadros de diálogo

Para mostrar un formulario, conjunto de formularios o cuadro de diálogo compilados desde un menú o un elemento de menú, puede llamarlos con un comando o con un [procedimiento](#). Por ejemplo, para

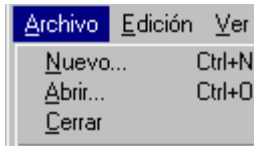


mostrar un formulario llamado "Pedidos", utilice el comando siguiente:

```
DO FORM Pedidos
```

**Sugerencia** Al crear un menú o un elemento de menú que muestre un formulario, un conjunto de formularios o un cuadro de diálogo, indique con tres puntos al final del texto que el usuario debe proporcionar más información.

**Los puntos en un elemento de menú indican que el usuario debe proporcionar más información.**



### Mostrar barras de herramientas

Si crea una barra de herramientas personalizada para una aplicación, puede mostrarla llamándola desde un menú o un elemento de menú. Si desea más información al respecto, consulte [Crear barras de herramientas personalizadas](#) más adelante en este mismo capítulo.

### Realizar tareas con procedimientos

Puede asignar un [procedimiento](#) a un menú o elemento de menú. La forma de hacerlo dependerá de si el menú o el elemento tiene submenús.

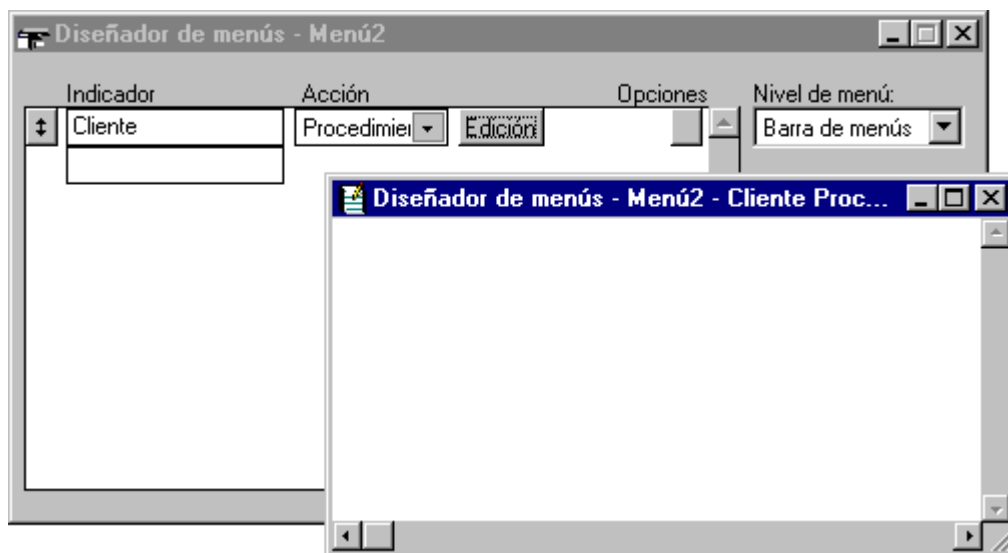
#### Para asignar un procedimiento a un menú o a un elemento de menú sin submenús

1. En la columna **Indicador**, haga clic en el título o elemento de menú correspondiente.
2. En el cuadro **Acción**, seleccione **Procedimiento**.

Aparecerá un botón **Crear** a la derecha de la lista. Si ha definido previamente un procedimiento, en lugar del botón Crear aparecerá un botón **Modificar**.

3. Haga clic en el botón **Crear** o **Modificar**.

#### Asignar un procedimiento a un menú con submenús



4. Escriba el código apropiado en la ventana.

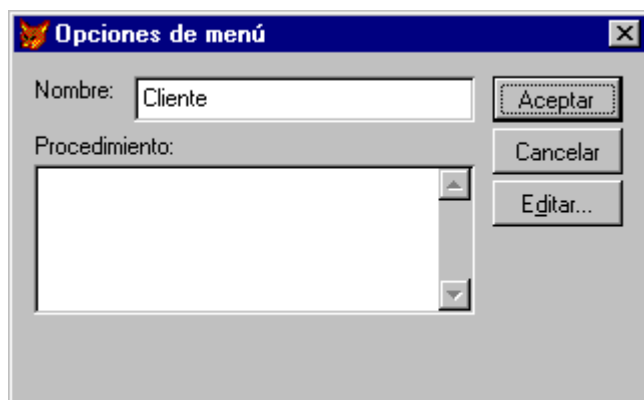
**Nota** No es necesario escribir el comando [PROCEDURE](#) en la ventana de edición del procedimiento, ya que Visual FoxPro lo generará automáticamente. El único lugar donde es necesario el comando PROCEDURE es en el [postprograma](#).

#### Para asignar un procedimiento a un menú o a un elemento de menú con submenús

1. En el cuadro **Nivel de menú**, seleccione el nivel en el que se encuentre el menú o elemento de menú correspondiente. Por ejemplo, suponga que el sistema de menús incluye el menú Cliente creado anteriormente. Para asignar un procedimiento al menú Cliente, seleccione el nivel "Barra de menú" en el cuadro "Nivel de menú". Del mismo modo, para asignar un procedimiento a un elemento del menú Cliente, elija el nivel "Cliente" en la lista.
2. En el menú **Ver**, elija **Opciones de menú**.

Visual FoxPro mostrará el cuadro de diálogo **Opciones de menú**.

#### Asignar un procedimiento a un menú sin submenús



3. Asigne el procedimiento mediante uno de los métodos siguientes:
  - Escriba un procedimiento o llámelo en el cuadro **Procedimiento**.

–O bien–

- Haga clic en el botón **Modificar** y luego en **Aceptar**, para abrir una ventana de edición independiente para escribir o llamar a un procedimiento.

### Agregar un preprograma a un sistema de menús

Para personalizar un [sistema de menús](#) puede agregarle un [preprograma](#). El preprograma puede incluir código para establecer el entorno, definir variables de memoria, abrir los archivos necesarios y guardar o restaurar sistemas de menús con los comandos [PUSH MENU](#) y [POP MENU](#).

#### Para agregar un preprograma a un sistema de menús

1. En el menú **Ver**, elija **Opciones generales**.
2. En el área **Código del menú**, elija **Instalación** y, a continuación, elija **Aceptar**.
3. En la ventana del preprograma, escriba el código adecuado.

Sus cambios se guardan cuando se cierra el Diseñador de menús.

### Agregar un postprograma a un sistema de menús

Para ajustar el sistema de menús a sus necesidades, puede agregarle un [postprograma](#). Normalmente, el postprograma contiene código que activa o desactiva inicialmente los menús y sus elementos. Cuando genera y ejecuta el programa de menú, el [preprograma](#) y el código de definición de menú se procesan antes que el postprograma.

#### Para agregar un postprograma a un sistema de menús

1. En el menú **Ver**, elija **Opciones generales**.
2. En el área **Código del menú**, seleccione la casilla de verificación **Postprograma** y, a continuación, elija **Aceptar**.
3. En la ventana del postprograma, escriba el código adecuado.

Sus cambios se guardan cuando cierra el **Diseñador de menús**.

**Sugerencia** Si el menú es el programa principal de una aplicación, incluya un comando [READ EVENTS](#) en el postprograma y asigne un comando [CLEAR EVENTS](#) al comando de menú utilizado para salir del sistema de menús. Esto evita que sus aplicaciones terminen de forma prematura en tiempo de ejecución.

### Controlar los menús en tiempo de ejecución

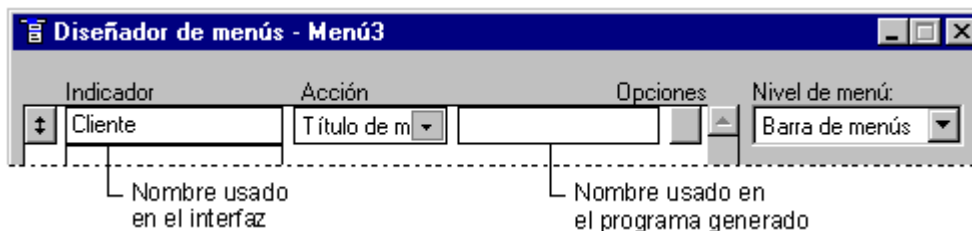
Todos los menús de Visual FoxPro tienen dos nombres y cada elemento de menú tiene un nombre y un número. Visual FoxPro utiliza un nombre en la interfaz de usuario y el otro nombre o el número en el programa de menú generado (.mpr). Puede usar estos nombres o números para hacer referencia a menús de control y a elementos de menú en [tiempo de ejecución](#). Si no proporciona un nombre o un número al crear menús y elementos de menú, Visual FoxPro crea uno cuando genera el programa de menú.

Para ver un ejemplo de adición y eliminación de menús en tiempo de ejecución, vea Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.

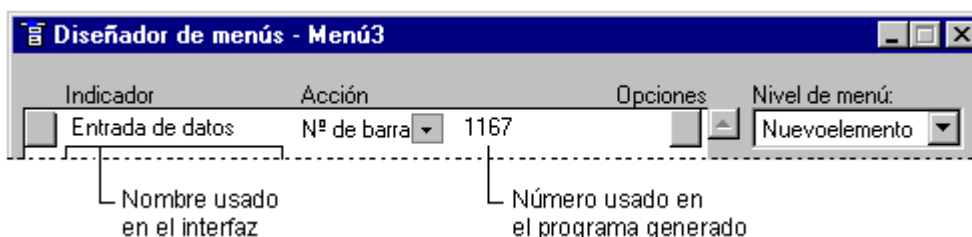
**Precaución** Evite utilizar nombres y números generados en el código, ya que cambiarán cada vez que se genere el programa de menú. Si hace referencia a un nombre o número generado, el código puede fallar.

En el [Diseñador de menús](#), la columna Indicador muestra lo que aparece en la interfaz de usuario y en la columna situada a la derecha del cuadro Acción se indica lo que aparece en el programa generado.

### Usar Título de menú para hacer referencia a un título de menú en el programa de menú generado



### Usar N° de barra para hacer referencia a un elemento de menú en el programa de menú generado



### Para especificar un nombre para un título de menú

1. En la columna **Indicador**, haga clic en el título de menú correspondiente.

**Nota** La columna **Acción** debe mostrar **Comando**, **Submenú** o **Procedimiento** y no **Título de menú**.

2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.

3. En el cuadro de diálogo **Título de menú**, escriba el nombre que prefiera.
4. Elija **Aceptar** para volver al **Diseñador de menús**.

### Para especificar un número de barra para un elemento de menú

1. En la columna **Indicador**, haga clic en el elemento de menú correspondiente.

**Nota** La columna **Acción** debe mostrar **Comando**, **Submenú** o **Procedimiento** y no **Nº de barra**.

2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.
3. En el cuadro **Barra nº**, escriba el número de barra que desee.
4. Elija **Aceptar** para volver al **Diseñador de menús**.

**Sugerencia** Si utiliza la función Menú rápido, no cambie los nombres o números que proporciona Visual FoxPro para los menús y elementos de menú, pues de lo contrario podría obtener resultados impredecibles al ejecutar el programa de menú generado.

## Crear barras de herramientas personalizadas

Si su aplicación incluye tareas repetitivas que los usuarios realizan con frecuencia, puede agregar [barras de herramientas](#) personalizadas para simplificar o acelerar dichas tareas. Por ejemplo, si los usuarios suelen imprimir un informe mediante un comando de menú, podrá simplificar esa tarea si incluye una barra de herramientas con un botón para imprimir.

Las siguientes secciones describen la forma de crear barras de herramientas personalizadas para sus aplicaciones. Para ver más información acerca de la personalización de las barras de herramientas incluidas con Visual FoxPro, busque [Personalizar barras de herramientas](#).

En las secciones siguientes se describe lo siguiente:

- [Definir una clase ToolBar](#)
- [Agregar objetos a una clase ToolBar personalizada](#)
- [Agregar barras de herramientas personalizadas a sus conjuntos de formularios](#)

### Definir una clase ToolBar

Para crear una [barra de herramientas](#) personalizada, debe definir primero una [clase](#) para la misma. Visual FoxPro ofrece una [clase de base](#) ToolBar a partir de la cual puede crear la clase que necesite.

Después de definir una clase de barra de herramientas, puede agregar un [objeto](#) a la clase barra de herramientas y, a continuación, definir las propiedades, eventos y métodos para la barra de herramientas personalizada. Finalmente, puede agregar la barra de herramientas a un [conjunto de formularios](#).

### Para definir una clase ToolBar personalizada

1. Desde el [Administrador de proyectos](#), seleccione **Clases** y elija **Nuevo**.
2. En el cuadro **Nombre de clase**, escriba el nombre de su clase.
3. En el cuadro **Basado en**, seleccione **Toolbar** para utilizar la clase de base Toolbar.

–O bien–

Elija el botón del diálogo para elegir otra clase de barra de herramientas.

4. En el cuadro **Almacenar en**, escriba el nombre de la [biblioteca](#) en la que desea guardar la nueva clase.

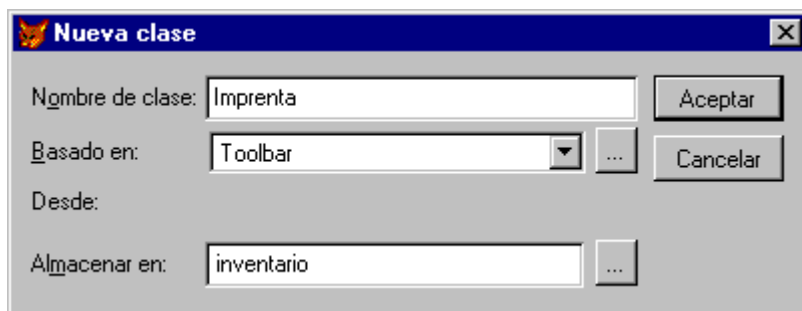
–O bien–

Haga clic en el botón con tres puntos para seleccionar una biblioteca existente.

5. Agregue objetos a la nueva clase Toolbar.

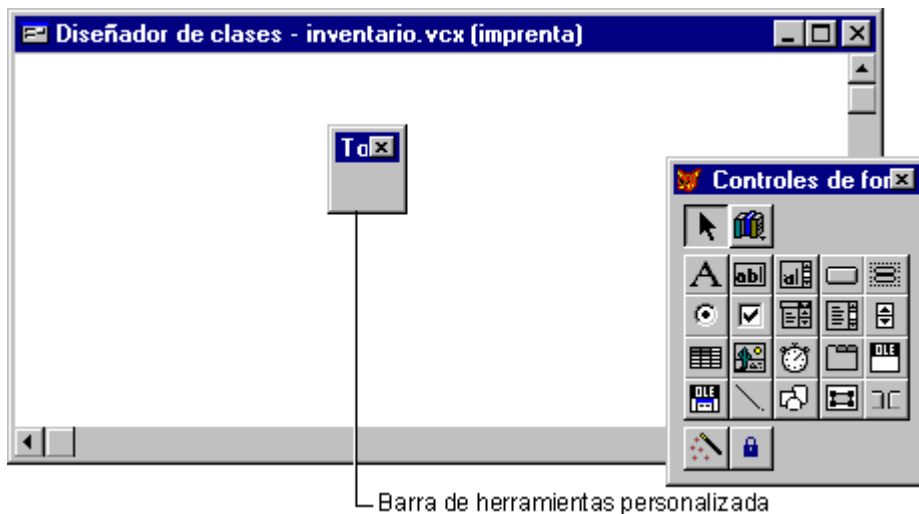
Por ejemplo, podría almacenar una clase de impresión basada en la clase de base Toolbar de una biblioteca de inventario.

### Crear una nueva clase en el cuadro de diálogo Nueva clase



Cuando complete el [cuadro de diálogo Nueva clase](#), aparecerá el Diseñador de clases.

### Una nueva barra de herramientas personalizada en el Diseñador de clases



También puede definir una clase de barra de herramientas de las maneras siguientes:

- Elija Nuevo en el menú Archivo y, a continuación, elija Clase.
- Use el comando [CREATE CLASS](#) o [MODIFY CLASS](#).
- Defina la clase mediante el comando [DEFINE CLASS](#).

### Agregar objetos a una clase de barra de herramientas personalizada

Después de crear una [clase](#) de barra de herramientas personalizada puede agregarle [objetos](#), incluyendo cualquier objeto admitido por Visual FoxPro. Por ejemplo, puede agregar objetos de la barra de herramientas Controles de formularios.

#### Para agregar objetos a su clase de barra de herramientas personalizada

1. Abra la [biblioteca de clase](#) que contiene la clase de barra de herramientas personalizada y, a continuación, abra la clase.
2. En la [barra de herramientas Controles de formularios](#), elija un objeto que desee agregar.
3. Ponga el objeto en la barra de herramientas personalizada; para ello, haga clic en dicha barra.
4. Repita los pasos 2 y 3 hasta que la barra de herramientas personalizada esté completa.
5. Reorganice los objetos de la barra de herramientas personalizada, si lo desea.

Por ejemplo, puede ajustar el tamaño de los objetos, arrastrarlos, eliminarlos haciendo clic en a tecla SUPR o agregar espacio adicional entre ellos insertando objetos separadores de la barra de herramientas Controles de formularios.

**Nota** Sólo puede mover un objeto cada vez.

6. Establezca las propiedades de la barra de herramientas en la ventana **Propiedades**.

7. Guarde la clase de barra de herramientas personalizada.

**Sugerencia** Puede agregar un mapa de bits o un icono a un botón de la barra de herramientas si establece su [propiedad Picture](#).

## Agregar barras de herramientas personalizadas a conjuntos de formularios

Después de definir una [clase](#) de barra de herramientas, puede crear una [barra de herramientas](#) a partir de la misma. Puede coordinar barras de herramientas y [formularios](#) con el Diseñador de formularios o mediante programación.

### Coordinar barras de herramientas y formularios en el Diseñador de formularios

Puede agregar una barra de herramientas a un [conjunto de formularios](#) de forma que dicha barra se abra junto con los formularios del conjunto de formularios. No puede agregar la barra de herramientas directamente al formulario.

### Para agregar una barra de herramientas a un conjunto de formularios mediante el Diseñador de formularios

1. Registre y seleccione la [biblioteca](#) que contiene la [clase](#) de barra de herramientas.
2. Abra el [conjunto de formularios](#) con el que desea utilizar la clase de barra de herramientas, haga clic en el botón **Ver clases** en la barra de herramientas **Controles de formulario** y seleccione la clase de barra de herramientas en la lista que aparecerá.
3. En la barra de herramientas **Controles de formulario**, elija la clase de barra de herramientas.
4. Haga clic en el **Diseñador de formularios** para agregar la barra de herramientas y después arrastre la barra de herramientas hasta su ubicación apropiada.

Visual FoxPro agregará la barra de herramientas al conjunto de formularios. Si un conjunto de formularios no está abierto, Visual FoxPro le pedirá que especifique uno.

5. Defina las acciones de la barra de herramientas y sus botones (consulte "[Definir acciones de barra de herramientas](#)" en la próxima sección).

**Sugerencia** Para determinar la clase de un objeto, examine su Información sobre herramientas en la [barra de herramientas Controles de formularios](#).

Para obtener más información sobre cómo registrar y seleccionar la biblioteca que contiene la clase de barra de herramientas, consulte "Adición de clases a formularios, conjuntos de formularios y barras de herramientas" en el capítulo 3, [Programación orientada a objetos](#).

### Coordinar barras de herramientas y formularios mediante programación

Además de utilizar el [Diseñador de formularios](#), puede agregar [barras de herramientas](#) a [conjuntos de formularios](#) mediante programación.



## Para agregar una barra de herramientas a un conjunto de formularios mediante código

- En el evento [Init](#) del conjunto de formularios, utilice el comando [SET CLASSLIB](#) para especificar la [biblioteca](#) que contiene la clase de barra de herramientas y después cree una barra de herramientas desde esa [clase](#) en el conjunto de formularios.

Por ejemplo, para agregar y ver la barra de herramientas `tbrPrint`, que está basada en la clase `printing` de la biblioteca de clases `inventory`, agregue el código siguiente al evento `Init` del conjunto de formularios:

```
SET CLASSLIB TO inventory
THIS.AddObject("tbrPrint","printing")
THIS.tbrPrint.Show
```

**Nota** Si la clase de barra de herramientas no define las acciones de la barra de herramientas y sus botones, deberá definir las acciones en los [procedimientos](#) de evento asociados a la barra de herramientas y sus botones. Para obtener más información al respecto, consulte "[Definir acciones de barra de herramientas](#)".

## Ejemplo: creación de una barra de herramientas personalizada

Puede definir todos los aspectos de una barra de herramientas en el código. Por ejemplo, si agrega el código siguiente al evento `Init` de un conjunto de formularios, cuando se cargue el conjunto de formularios Visual FoxPro creará y mostrará la barra de herramientas definida en el código. Esta barra contiene dos botones.

### Barra de herramientas con dos botones



Cuando se haga clic en estos botones, cambiarán los atributos de fuente del formulario `frmForm1` del conjunto de formularios.

## Código de evento `Init` del conjunto de formularios

Código	Comentarios
<pre>THIS.AddObject("tbrTool1","mibarra") THIS.tbrTool1.Show</pre>	<p>Agrega una barra de herramientas de la clase <code>mibarra</code> al conjunto de formularios actual y hace que dicha barra sea visible. Este código está en el evento <code>Init</code> del conjunto de formularios.</p>

## Código de definición de clase

Código	Comentarios
<pre> DEFINE CLASS mibarra AS TOOLBAR  ADD OBJECT cmdBold AS COMMANDBUTTON ADD OBJECT sep1 AS SEPARATOR ADD OBJECT cmdItalic AS COMMANDBUTTON </pre>	<p>Principio de la definición de clase: una barra de herramientas con un botón de comando, un separador y otro botón de comando.</p>
<pre> Left = 1 Top = 1 Width = 25 Caption = "Atributos de formulario" </pre>	<p>Establece las propiedades del objeto barra de herramientas.</p>
<pre> cmdBold.Caption = "B" cmdBold.Height = 1.7 cmdBold.Width = 10  cmdItalic.Caption = "I" cmdItalic.Height = 1.7 cmdItalic.Width = 10 cmdItalic.FontBold = .F. </pre>	<p>Establece las propiedades de los controles. Observe que no hay ningún valor en las propiedades Top o Left para los controles de una barra de herramientas. Los controles de una barra de herramientas se sitúan automáticamente en el orden en que se agregan.</p> <p>La propiedad FontBold de cmdItalic está establecida como falso (.F.) ya que FontBold es verdadero (.T.) de forma predeterminada.</p>
<pre> PROCEDURE Activate THIS.cmdBold.FontBold = ; THISFORMSET.frmForm1.FontBold THIS.cmdItalic.FontItalic = ; THISFORMSET.frmForm1.FontItalic ENDPROC </pre>	<p>Cuando se activa la barra de herramientas, los atributos de fuente de los dos botones de comando se establecen para reflejar las configuraciones de fuente Negrita Cursiva de frmForm1.</p>
<pre> PROCEDURE cmdBold.CLICK THISFORMSET.frmForm1.FontBold = ; !THISFORMSET.frmForm1.FontBold THIS.FontBold = ; THISFORMSET.frmForm1.FontBold ENDPROC </pre>	<p>Cuando el usuario hace clic en cmdBold se invierte la configuración FontBold de frmForm1 y la configuración FontBold de cmdBold se establece de forma que coincida.</p>
<pre> PROCEDURE cmdItalic.CLICK THISFORMSET.frmForm1.FontItalic = ; !THISFORMSET.frmForm1.FontItalic THIS.FontItalic = ; THISFORMSET.frmForm1.FontItalic ENDPROC </pre>	<p>Cuando el usuario hace clic en cmdItalic se invierte la configuración FontItalic de frmForm1 y la configuración FontItalic de cmdItalic se establece de forma que coincida.</p>
<pre> ENDDEFINE </pre>	<p>Fin de la definición de clase.</p>

## Establecer propiedades de barras de herramientas personalizadas

Mientras diseña una barra de herramientas personalizada puede establecer sus [propiedades](#). Por ejemplo, puede establecer la [propiedad Movable](#) para permitir al usuario mover la barra de herramientas.

Además, puede utilizar [métodos](#) y [eventos](#) para controlar las barras de herramientas personalizadas. Por ejemplo, puede utilizar el método [Dock](#) para acoplar o liberar una barra de herramientas y los eventos [BeforeDock](#) y [AfterDock](#) para controlar lo que ocurre antes y después de acoplar una barra de herramientas.

## Definir acciones de barra de herramientas

Después de crear una barra de herramientas debe definir las acciones asociadas a la barra de herramientas y sus [objetos](#). Por ejemplo, debe definir lo que ocurrirá cuando el usuario haga clic en la barra de herramientas o en uno de sus botones.

### Para definir una acción de barra de herramientas

1. Seleccione el objeto para el que desee definir una acción: la barra de herramientas o uno de sus botones.
2. En la [ventana Propiedades](#), haga clic en la ficha **Métodos**.
3. Modifique el evento apropiado.
4. Agregue el código que especifique la acción.

Además, puede establecer propiedades y métodos de la barra de herramientas y sus objetos.

## Coordinar menús y barras de herramientas personalizadas

Si crea una [barra de herramientas](#), deberá sincronizar sus botones con los correspondientes comandos de [menú](#). Por ejemplo, si activa un comando de menú, deberá activar su botón correspondiente.

Debe diseñar y crear su aplicación para:

- Realizar las mismas acciones cuando el usuario elige botones de barra de herramientas y elementos de menú asociados.
- Coordinar la activación y desactivación de botones de barra de herramientas y elementos de menú asociados.

Siga los pasos generales al coordinar elementos de menú y botones de barra de herramientas:

1. Cree una [barra de herramientas](#); para ello, defina una [clase](#) de barra de herramientas, agregue [botones de comando](#) e incluya el código operacional de los [métodos](#) asociados a los [eventos Click](#) de los botones de comando.
2. Cree el menú coordinado.

3. Agregue la barra de herramientas y el menú coordinados a un [conjunto de formularios](#).

### Crear un menú coordinado

Cuando coordine un [menú](#) con una barra de herramientas, los elementos de menú cumplen las mismas tareas que la [barra de herramientas](#) asociada y los elementos de menú asociados se desactivan automáticamente cuando el botón de barra de herramientas asociado se desactiva.

### Para crear un menú coordinado con una barra de herramientas

1. En el [Diseñador de menús](#), cree un submenú con un indicador descriptivo para cada botón de la barra de herramientas.
2. En la columna de acción para cada elemento de submenú, elija **Comando**.
3. Para cada elemento de submenú, llame al código asociado al evento [Click](#) del botón de comando de barra de herramientas apropiado.

Por ejemplo, si el nombre del botón de la barra de herramientas es `cmdA`, agregue la línea de código siguiente en el cuadro de edición para el comando de elemento de submenú:

```
Formset.toolbar.cmdA.Click
```

4. Elija el botón de la columna **Opciones** para abrir el cuadro de diálogo **Opciones de la acción** y elija **Saltar por**.
5. En el [Generador de expresiones](#), escriba una expresión que indique que la opción de menú se debe saltar cuando el botón de comando de la barra de herramientas no esté activado.

Por ejemplo, si el nombre del botón en la barra de herramientas es `cmdA`, escriba la siguiente expresión en el cuadro **Saltar por**:

```
NOT formset.toolbar.cmdA.Enabled
```

6. Genere el menú.
7. Agregue el menú al conjunto de formularios con la barra de herramientas y ejecute el conjunto de formularios.

Cuando el usuario abra el menú, Visual FoxPro evalúa la condición Saltar por y desactiva el elemento de menú si el botón de comando de barra de herramientas asociado está desactivado. Cuando el usuario elija un elemento en el menú, el código del [evento Click](#) del botón de comando de barra de herramientas asociado se ejecuta.

### Agregar una barra de herramientas y un menú coordinados a un conjunto de formularios

Cuando haya creado una [clase](#) de barra de herramientas y un [menú](#) que están diseñados para que funcionen conjuntamente, es fácil incorporarlos a un [conjunto de formularios](#).

## Para incorporar una barra de herramientas y un menú coordinados a un conjunto de formularios

1. Agregue la barra de herramientas al conjunto de formularios de una de las tres formas siguientes:
  - Arrastre la clase de barra de herramientas desde el [Administrador de proyectos](#) al [Diseñador de formularios](#).
  - Registre la [biblioteca de clases](#) de barras de herramientas al conjunto de formularios y agregue la barra de herramientas al conjunto de formularios desde la [barra de herramientas Controles de formularios](#).
  - En el evento [Init](#) del conjunto de formularios, incluya código con el método [AddObject](#) para agregar la barra de herramientas.
2. En el evento [Load](#) del conjunto de formularios, guarde el menú existente y ejecute su programa de menú.

Por ejemplo, si su nombre de menú es `mymenu`, incluya las siguientes líneas de código con los comandos [PUSH MENU](#) y [DO](#):

```
PUSH MENU _MSYSMENU  
DO mymenu.mpr
```

3. En el evento `Unload` del conjunto de formularios, restablezca el menú original con el comando [POP MENU](#):

```
POP MENU _MSYSMENU
```

Si algunos comandos de menú se usan más que otros, puede crear barras de herramientas personalizadas que contengan botones para esos comandos. Los usuarios podrán simplemente hacer clic en los botones siempre que necesiten los comandos. Sin embargo, si crea una barra de herramientas, debería sincronizar los comandos de menú con sus botones correspondientes. Por ejemplo, si activa un botón, debería activar su correspondiente comando de menú.

## Probar y depurar un sistema de menús

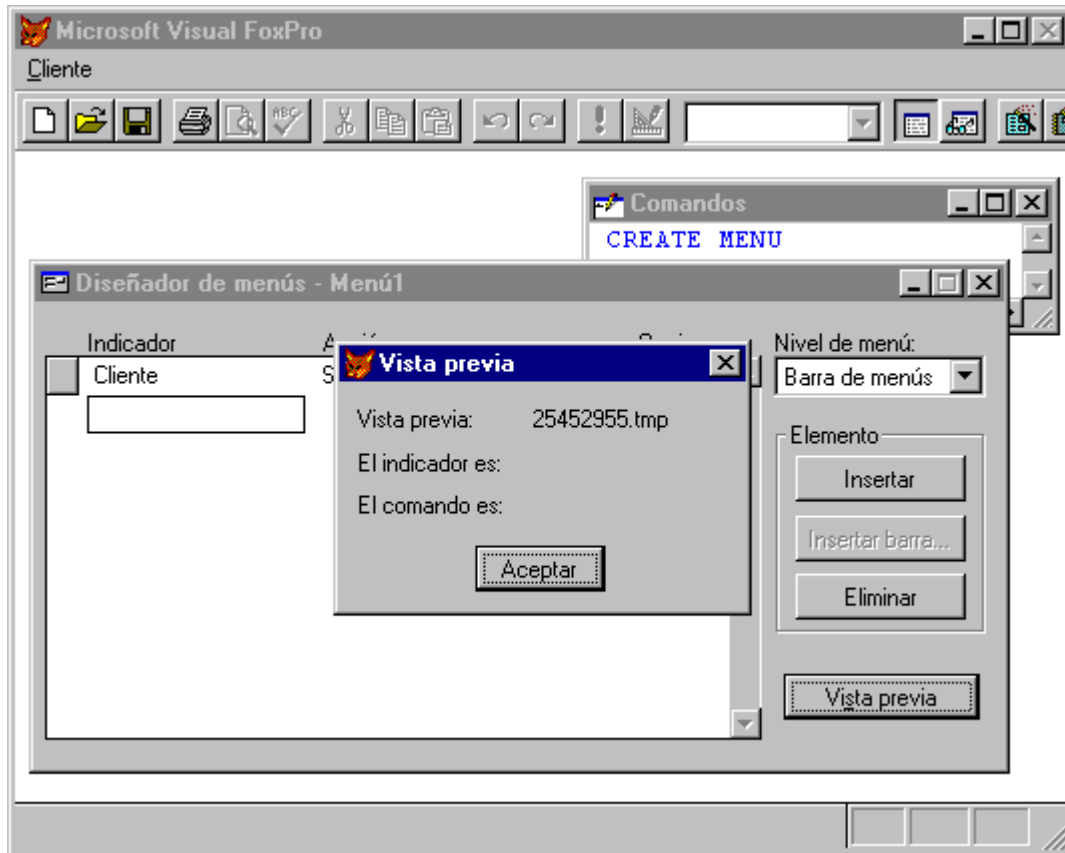
Al diseñar un [sistema de menús](#) puede ver una vista previa o también puede probarlo y depurarlo una vez generado el programa de menú.

### Para ver una vista previa de un sistema de menús durante su diseño

- En el [Diseñador de menús](#), elija **Vista previa**.

Al hacer clic en Vista previa aparecerá en el borde superior de la pantalla el sistema de menús que haya definido. Además, el cuadro de diálogo Vista previa mostrará el nombre de archivo (o nombre temporal) del sistema de menús.

### Vista previa de un sistema de menús



Si selecciona un título o un elemento de menú, también aparecerá en el cuadro de diálogo Vista previa, junto con el comando que tenga asignado, si existe.

### Para probar un sistema de menús

1. En el [menú Menú](#), elija **Generar**.

Si ha modificado el menú, Visual FoxPro le pedirá que guarde los cambios.

2. En el cuadro de diálogo **Generar menú**, escriba un nombre para el programa de menú generado en el cuadro **Archivo de salida** o elija el botón con tres puntos.
3. Elija **Generar** para crear un archivo de programa de menú con la extensión .mpr.
4. En el menú **Programa**, elija **Ejecutar** para ejecutar el programa.

**Precaución** Si modifica el programa de menú generado (el archivo .mpr), perderá todos los cambios al modificar el menú con el [Diseñador de menús](#) y volver a generar el programa de menú.

Si el programa de menú no funciona como se pretende, utilice las herramientas de diagnóstico que incluye Visual FoxPro. Si desea más información al respecto, consulte el capítulo 14, [Probar y depurar aplicaciones](#).

**Solución de problemas** Si dispone de la Edición profesional de Visual FoxPro y ejecuta una aplicación (.EXE) en la que el programa principal es un menú y la aplicación termina cuando muestra el menú, incluya el comando [READ EVENTS](#) en el postprograma y asigne un comando [CLEAR EVENTS](#) al comando de menú que permita al usuario salir del sistema de menús.

## Personalizar un sistema de menús

Una vez creado un [sistema de menús](#) básico, puede personalizarlo. Por ejemplo, puede crear mensajes de barra de estado, definir las posiciones de los menús y definir procedimientos predeterminados.

### Mostrar mensajes en la barra de estado

Cuando selecciona un menú o un elemento de menú, puede mostrar un mensaje en la barra de estado que lo describa. Estos mensajes ayudan al usuario ofreciéndole información adicional sobre la elección en el menú.

#### Para mostrar un mensaje cuando se selecciona un menú o un elemento de menú

1. En la columna **Indicador**, haga clic en el título o elemento de menú correspondiente.
2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.
3. Haga clic en el botón **Mensaje**.

Aparecerá el cuadro de diálogo **Generador de expresiones**.

4. En el cuadro **Mensaje**, escriba el mensaje que desee.

**Sugerencia** Indique las cadenas de caracteres entre comillas.

### Definir la posición de los títulos de los menús

Puede personalizar la ubicación de los títulos de menús definidos por el usuario en las aplicaciones. Puede personalizar la posición respecto al [sistema de menús](#) activo; para ello, elija las opciones del [cuadro de diálogo Opciones generales](#). También puede especificar la posición de los títulos de los menús cuando el usuario modifica visualmente un objeto.

#### Para especificar una posición relativa para los títulos de menú definidos por el usuario

1. En el menú **Ver**, elija **Opciones generales**.
2. Elija la opción **Ubicación** adecuada: **Reemplazar**, **Anexar**, **Antes** o **Después**.

Visual FoxPro volverá a colocar todos los títulos de menú que haya definido. Si solamente desea volver a colocar algunos, arrastre los botones de movimiento hasta los títulos de menú correspondientes en el [Diseñador de menús](#).

También puede especificar la posición de los títulos de los menús cuando el usuario modifica un objeto en la aplicación. Si incluye un [objeto](#) y el usuario lo activa, los títulos no aparecerán en la barra de menús resultante a menos que lo especifique explícitamente.

### Para controlar la posición de los títulos de los menús durante la modificación visual

1. En la columna **Indicador**, haga clic en el título de menú correspondiente.
2. Elija el botón de la columna **Opciones** para mostrar el cuadro de diálogo **Opciones de la acción**.
3. Active la casilla de verificación **Negociar**.
4. Elija uno de los botones de opción siguientes:
  - **Nada** no incluye ningún título en la barra de menús. Elegir **Nada** es como no elegir ninguna opción.
  - **Izquierda** coloca el título en el grupo izquierdo de títulos de la barra de menús.
  - **Centro** coloca el título en el grupo central de títulos de la barra de menús.
  - **Derecha** coloca el título en el grupo derecho de títulos de la barra de menús.

Si no elige Izquierda, Centro o Derecha, el título de menú no aparecerá en la barra de menús cuando el usuario modifique un objeto. Para obtener más información sobre la modificación visual de objetos, consulte el capítulo 16, [Agregar OLE](#).

### Guardar y restaurar menús

Para guardar y restaurar menús en la pila puede utilizar los comandos [PUSH MENU](#) y [POP MENU](#). Esta funcionalidad es útil cuando se desea quitar temporalmente un menú, sustituirlo por otro y después restaurar el menú original.

El número de menús que se pueden guardar sólo está limitado por la cantidad de memoria disponible.

**Sugerencia** Compruebe la memoria disponible con la función [SYS\(1016\)](#). Por ejemplo, para saber cuánta memoria utiliza el sistema de menús, llame a SYS(1016), guarde el menú en la pila y llame de nuevo a SYS(1016).

### Crear un procedimiento predeterminado para un sistema de menús

Puede crear un [procedimiento](#) global que se aplica a todo el [sistema de menús](#). Este procedimiento se ejecutará siempre que se elija un menú que no tenga asignado ningún procedimiento.

Por ejemplo, suponga que está programando una aplicación en la que algunos menús aún no cuentan con submenús, procedimientos, etc. Para estos menús, puede crear código general que se ejecute al



elegirlos. Por ejemplo, este procedimiento general puede incluir la función siguiente:

```
MESSAGEBOX("Característica no disponible")
```

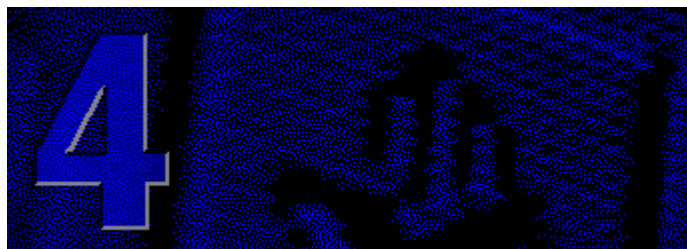
### Para crear un procedimiento predeterminado

1. Abra el sistema de menús que esté diseñando.
2. En el menú **Ver**, elija **Opciones generales**.
3. Asigne el procedimiento mediante alguno de los métodos siguientes:
  - Escriba un procedimiento o llámelo en el cuadro **Procedimiento**.  
  
–O bien–
  - Haga clic en el botón **Edición**, y luego en el botón **Aceptar**, para abrir una ventana de edición independiente en la que escribir o llamar al procedimiento.

### Establecer el menú del sistema

Puede manipular menús que utilizan el sistema de menús de Visual FoxPro mediante el comando [SET SYSMENU](#). Con SET SYSMENU puede desactivar los menús, agregarles o quitarles elementos, restaurar los menús predeterminados de Visual FoxPro y controlar el acceso a los menús durante la ejecución.

# Manual del programador, Parte 4: Agrupar todos los elementos



Ahora ya puede completar su aplicación y probar su funcionalidad. Agregue consultas e informes para ofrecer a los usuarios la información que necesitan y después ejecute la aplicación y busque áreas que pueda optimizar.

## Capítulo 12 [Agregar consultas e informes](#)

Después de crear datos y una interfaz para su aplicación, puede aumentar su eficacia con consultas e informes que proporcionen información importante para sus usuarios .

## Capítulo 13 [Compilar una aplicación](#)

Cree sus aplicaciones paso a paso y compruebe cada componente. Cuando haya incluido todos los componentes, será fácil compilarlos en una aplicación.

## Capítulo 14 [Probar y depurar aplicaciones](#)

Cuando programe una aplicación, necesitará comprobar si existen errores en los componentes. Visual FoxPro ofrece herramientas de depuración para ayudarle a buscar y corregir cualquier error que descubra en sus aplicaciones.

## Capítulo 15 [Optimizar aplicaciones](#)

Cuando disponga de una aplicación estable y en funcionamiento, considere las formas en que puede optimizar su rendimiento y convertirla en más pequeña y rápida.

# Capítulo 12: Agregar consultas e informes

Una vez creadas las tablas y formularios para su aplicación, puede agregar consultas e informes para seleccionar y mostrar datos a los usuarios. Las consultas pueden ir dirigidas hacia diversos destinos, de forma que puede usarlas en los otros componentes de su aplicación. De igual forma, puede ejecutar informes de manera independiente, sin necesidad de utilizar una consulta. Este capítulo resalta algunas de las formas en las que puede utilizar las consultas, agregar informes y exponer consultas e informes a los usuarios.

Cuando utiliza una consulta o una vista en su aplicación, en realidad utiliza una instrucción [SELECT - SQL](#). Este capítulo describe cómo puede utilizar en su aplicación una instrucción SELECT - SQL si la crea mediante una consulta definida en el Diseñador de consultas, una vista definida en el Diseñador de vistas o código introducido para un evento o procedimiento. Para obtener información detallada sobre las vistas, vea la parte 2, [Buscar información](#), del *Manual del usuario*.

Este capítulo trata los temas siguientes:

- [Agregar consultas](#)
- [Agregar informes y etiquetas](#)
- [Integrar consultas e informes](#)

## Agregar consultas

Cuando agrega consultas a su aplicación, puede combinar varios orígenes de datos, filtrar registros, manipular datos y ordenar los resultados, todo ello con la instrucción SELECT - SQL. Al utilizar instrucciones SQL, dispone de un control completo sobre los resultados obtenidos en sus consultas y sobre dónde se almacenan estos resultados.

**Una consulta es una instrucción SELECT - SQL.**



Puede agregar instrucciones SELECT - SQL a procedimientos o código de eventos. Para ver más información acerca de los eventos, consulte el capítulo 4, [Descripción del modelo de eventos](#).

## Crear una instrucción SELECT - SQL

### Para crear una instrucción SELECT - SQL

- Utilice el [Diseñador de consultas](#) o el [Diseñador de vistas](#) para crear la instrucción y copie el contenido de la ventana SQL a una ventana de código.
- O bien—
- En una ventana de código, escriba la instrucción [SELECT - SQL](#).

Por ejemplo, puede seleccionar todos los registros de la tabla `Customer` de la base de datos `TasTrade`

en los que el campo `country` contenga el valor "Canadá":

```
SELECT * ;
FROM tastrade!customer ;
WHERE customer.country = "Canadá"
```

Para ejecutar el comando inmediatamente, puede introducir la instrucción en la ventana [Comandos](#). Si desea que cada cláusula aparezca en una línea distinta dentro de la ventana, termine cada línea, excepto la última, con un punto y coma. De esta forma Visual FoxPro procesa el comando sólo después de la última línea.

## Seleccionar un número o porcentaje de registros

Si sólo necesita un cierto número o porcentaje de registros del conjunto de resultados que su consulta devolvería, puede usar la propiedad Registros incluidos de la ficha Varios de los Diseñadores de consultas o de vistas o puede agregar una cláusula TOP a su instrucción [SELECT - SQL](#). El número que proporcione en una cláusula TOP puede variar entre 1 y 32767. Para un porcentaje, puede usar un valor entre 0.001 y 99.99.

Por ejemplo, si quiere seleccionar los diez clientes principales con las cantidades de pedidos más altas, puede especificar un GROUP BY en CUST\_ID para mostrar un registro agregado para cada cliente y ordenar por ORDER\_AMT en la cláusula ORDER BY. Para obtener un TOP 10 verdadero, debe especificar una ordenación descendente en ORDER\_AMT de forma que los registros con las cantidades de pedidos más altas aparezcan en primer lugar en el resultado. Si usa un orden ascendente, los registros de resultados se ordenan desde la menor cantidad de pedidos a la mayor. Los registros incluidos que seleccione del conjunto de resultados tendrían en realidad los valores más bajos.

```
SELECT TOP 10 * ;
FROM testdata!customer INNER JOIN testdata!orders ;
ON Customer.cust_id = Orders.cust_id;
GROUP BY Customer.cust_id;
ORDER BY Orders.order_amt DESC
```

## Especificar destinos para resultados de consultas

Al utilizar cláusulas de la instrucción [SELECT - SQL](#), puede especificar varios destinos para almacenar el resultado de sus consultas.

Para enviar los resultados a este destino	Utilice esta cláusula
Otra tabla	INTO TABLE mitabla
Matriz	INTO ARRAY aMiMatriz
Tabla temporal	INTO CURSOR micursor
Ventana activa	TO SCREEN
Ventana Examinar	Predeterminado si no se especifica otro destino.

Una vez que los resultados están almacenados, puede utilizar comandos para controlar cómo se integran los resultados almacenados para su presentación o impresión.

### **Almacenar los resultados en una tabla, matriz o cursor**

Puede almacenar los resultados de sus consultas en una tabla, una matriz o un cursor para otros usos, tales como completar formularios e imprimir informes y etiquetas. Si desea almacenar los resultados sólo temporalmente, envíelos a una matriz o a un cursor. Si lo que desea es almacenar los resultados definitivamente, envíelos a una tabla.

#### **Para especificar una tabla como destino**

- Utilice la cláusula INTO de la instrucción [SELECT - SQL](#) para especificar un destino.

Este ejemplo muestra una cláusula INTO para una tabla:

```
SELECT * ;  
FROM tastrade!customer ;  
WHERE customer.country = "Canadá" ;  
INTO TABLE mitabla
```

#### **Para especificar una matriz como destino**

- Utilice la cláusula INTO de la instrucción [SELECT - SQL](#) para especificar un destino.

El ejemplo siguiente muestra una cláusula INTO para una matriz:

```
SELECT * ;  
FROM tastrade!customer ;  
WHERE customer.country = "Canadá" ;  
INTO ARRAY aMiMatriz
```

#### **Para especificar un cursor como destino**

- Utilice la cláusula INTO de la instrucción [SELECT - SQL](#) para especificar un destino.

Este ejemplo muestra una cláusula INTO para un cursor llamado micursor.

```
SELECT * ;  
FROM tastrade!customer ;  
WHERE customer.country = "Canadá" ;  
INTO CURSOR micursor
```

Si crea una tabla o una matriz, puede utilizarla como cualquier otra tabla o matriz en Visual FoxPro. Si crea un cursor, puede examinar o anexas su contenido. El cursor se abre en el menor área de trabajo disponible. Puede tener acceso al mismo con el nombre que le ha dado en la instrucción SELECT - SQL.

Los dos procedimientos siguientes describen dos formas comunes para incluir en una aplicación los resultados de consultas almacenados en tablas y cursores.

## Rellenar un control de formulario

Si desea mostrar los resultados de sus consultas en un formulario, puede utilizar una tabla, una matriz o un cursor para colocarlos en una cuadrícula, un cuadro de lista o un cuadro combinado.

### Para rellenar un control cuadro de lista o cuadro combinado con una tabla o cursor

1. En el [Diseñador de formularios](#), modifique el formulario que tiene el control que desea rellenar.
2. Establezca la propiedad [RowSourceType](#) a **3 - SQL Statement**.
3. En la propiedad [RowSource](#) del control, escriba una instrucción [SELECT - SQL](#) que incluya una cláusula INTO TABLE o INTO CURSOR.

### Para llenar un control cuadrícula con una tabla o un cursor

1. En el [Diseñador de formularios](#), modifique el formulario que tiene el control que desea llenar.
2. En el evento [Load](#) del formulario, escriba una instrucción [SELECT - SQL](#) que incluya una cláusula INTO TABLE o INTO CURSOR.
3. Establezca la propiedad [RecordSource](#) de la cuadrícula como el nombre de la tabla o cursor que haya creado en el paso 2.
4. Establezca la propiedad [RecordSourceType](#) de la cuadrícula a **0 – Table** (para una tabla) o **1 – Alias** (para un cursor).

## Imprimir el resultado en un informe o una etiqueta

Si su informe o etiqueta incluye grupos, o si necesita ordenar los datos de alguna otra manera, puede utilizar las distintas cláusulas de la instrucción SELECT - SQL para obtener el resultado exacto que necesita.

### Para enviar los resultados a un informe o una etiqueta existente

- Utilice la instrucción [SELECT - SQL](#) con un comando REPORT o LABEL.

El ejemplo siguiente utiliza las cláusulas GROUP BY y ORDER BY, así como el comando [REPORT FORM](#):

```
SELECT * ;
FROM tastrade!customer ;
WHERE customer.country = "Canadá" ;
GROUP BY customer.region ;
ORDER BY customer.postal_code, customer.company_name ;
INTO CURSOR MiCursor
REPORT FORM MYREPORT.FRX
```

El ejemplo siguiente utiliza un comando [LABEL FORM](#):

```
SELECT * ;
FROM tastrade!customer ;
WHERE customer.country = "Canadá" ;
GROUP BY customer.region ;
ORDER BY customer.postal_code, customer.company_name ;
INTO CURSOR micursor
LABEL FORM MYLABEL.LBX
```

Aunque la instrucción SELECT - SQL es el método más flexible para rellenar sus informes o etiquetas, no es el único método. Para obtener más información acerca de cómo establecer orígenes de datos para informes, consulte la sección [Controlar los orígenes de datos de un informe](#), más adelante en este mismo capítulo. Para ver más información acerca de la integración de los destinos de informes en sus aplicaciones, consulte la sección [Integrar consultas e informes](#), más adelante en este mismo capítulo.

### Mostrar el resultado en una ventana

Si desea mostrar el resultado de su instrucción SELECT - SQL, puede enviarlo a una ventana. La ventana Examinar es el destino predeterminado para los resultados de las consultas y no necesita incluir una cláusula de destino. Puede enviar también el resultado a la ventana principal de Visual FoxPro o a otra ventana activa.

### Para mostrar resultados en la ventana principal de Visual FoxPro

- Utilice la cláusula TO SCREEN de una instrucción [SELECT - SQL](#).

### Para mostrar resultados en otra ventana activa

- Defina una ventana, muéstrela para activarla y, a continuación, ejecute una consulta SQL u otro comando que muestre resultados en una ventana.

Este programa de ejemplo muestra la definición para una ventana temporal titulada Principales clientes, que muestra los nombres de las compañías con una facturación anual superior a los \$50.000.

### Mostrar los resultados de la consulta en una ventana

Código	Comentario
<pre>frmMyForm=createobj( "form" ) frmMyForm.Left = 1 frmMyForm.Top = 1 frmMyForm.Width = 130 frmMyForm.Height = 25 frmMyForm.Caption = "Principales clientes" frmMyForm.Show</pre>	Crea e inicia un objeto ventana temporal.
<pre>SELECT customer.company_name,        SUM(orders.freight) ; FROM tastrade!customer,</pre>	Introduce una instrucción SELECT - SQL.

```

tastrade!orders ;
WHERE customer.customer_id =
orders.customer_id ;
GROUP BY customer.company_name ;
HAVING SUM(orders.freight) > 5000 ;
ORDER BY 2 DESC

```

## Agregar informes y etiquetas

Después de recoger y organizar sus datos, puede agregar informes o etiquetas a su aplicación para imprimir los datos o mostrarlos en pantalla. Puede controlar los datos de su informe mediante los orígenes de datos que elija o manipular y combinar datos sin manipular con *variables de informe*. Las variables de informe almacenan valores que se calculan y utilizan en un informe.

### Controlar los orígenes de datos

Para controlar los orígenes de datos de un informe puede definir un entorno de datos que se almacenan con el informe o bien puede, mediante código, activar orígenes de datos específicos cada vez que ejecute un informe. Para ver más información acerca del uso del Diseñador de entornos de datos, consulte el capítulo 9, [Crear formularios](#).

Para	Agregue
Utilizar siempre los mismos orígenes de datos.	Tablas o útiles al entorno de datos del informe.
	<a href="#">DO consulta</a> o <a href="#">SELECT - SQL</a> al código de evento <a href="#">Init</a> del entorno de datos del informe.
Utilizar conjuntos diferentes de orígenes de datos.	<a href="#">USE tabla</a> , <a href="#">USE vista</a> , <a href="#">DO consulta</a> o <a href="#">SELECT - SQL</a> para el evento <a href="#">Click</a> u otro código que preceda a un comando <a href="#">REPORT</a> o <a href="#">LABEL</a> .

Si utiliza una tabla como origen de datos, los registros se procesarán e imprimirán en el orden en que aparecen en la tabla. Utilice alias sólo si no piensa usar el informe con ningún otro origen de datos que no sea la propia tabla. Si utiliza una vista o una consulta como origen de datos y los alias están incluidos en los controles del informe, éste podría mostrar de forma repetida el mismo registro en la página.

### Controlar el orden de registros

Puede usar los orígenes de datos usados por el informe para controlar el orden en que se imprimen los registros en su informe. Los registros se imprimen en el orden en que aparecen en la tabla, vista o consulta. Para ordenar los registros de una tabla, puede establecer un índice en el código o como parte del entorno de datos de un informe. Para una consulta, vista o código [SELECT - SQL](#), puede usar la cláusula [ORDER BY](#). Si no ordena los registros mediante los orígenes de datos, la única forma de usar únicamente el informe para ordenar los registros es a través de la propiedad [ORDER](#) de un cursor del entorno de datos.



## Controlar la selección de registros

Además del orden en que los registros aparecen en el informe, puede seleccionar qué registros se imprimen con el origen de datos, las opciones de impresión de informes o una combinación de ambas cosas.

Para usar	Agregue
Vista o consulta	Condiciones a la ficha <a href="#">Filtro</a>
<a href="#">SELECT - SQL</a>	Cláusula WHERE o HAVING
<a href="#">Diseñador de informes</a>	Configuración en el cuadro de diálogo <a href="#">Opciones de impresión</a>
Comando <a href="#">REPORT</a>	Expresiones Alcance, FOR o WHILE
Tabla	Índice filtro

## Proteger una sesión de datos de un informe

Para evitar que la sesión de datos de su informe se vea afectada por la sesión de datos global como resultado de las modificaciones realizadas por otros diseñadores, puede establecer la sesión de datos del informe como privada.

### Para establecer una sesión privada de datos

- En el menú **Informe**, elija **Sesión privada de datos**.

Para obtener más información sobre el uso del [Diseñador de entornos de datos](#), consulte el capítulo 9, [Crear formularios](#). Para obtener más información acerca de las sesiones de datos, consulte el capítulo 17, [Programar para acceso compartido](#).

Si desea mostrar el resultado de una consulta en un gráfico, puede usar el [Asistente para gráficos](#), el [Diseñador de consultas](#) o un comando [SELECT - SQL](#). Para usar el Diseñador de consultas o un comando SELECT - SQL, siga los pasos que se indican a continuación. Debe incluir por lo menos un campo numérico en el conjunto de resultados para crear un gráfico. Después de terminar la consulta, puede elegir entre seis tipos de gráficos, cada uno con dos variantes.

### Para modificar el gráfico

1. Examine la tabla que contiene el gráfico.
2. Haga doble clic en el campo general para mostrar el gráfico.
3. Haga doble clic en el gráfico para abrir Microsoft Graph y mostrar la barra de herramientas de Microsoft Graph.
4. Modifique el gráfico en Microsoft Graph.

## Refinar el diseño de página

Puede refinar el diseño de las páginas de su informe; para ello, defina múltiples columnas y cambie el área de página reservada para una banda cambiando el alto de cada banda.

### Definir múltiples columnas en una página

Para crear listas de teléfonos, etiquetas postales u otros tipos de listas, puede definir varias columnas por página.

### Para definir un informe de múltiples columnas

1. En el menú **Archivo**, elija **Configurar página**.

#### Cuadro de diálogo Configurar página con columnas definidas



2. En el área **Columnas**, escriba el número de columnas para la página. Es el mismo que el número de registros que desea imprimir en la página.
3. En el cuadro **Ancho**, escriba un valor para ancho de columna.
4. En el cuadro **Espacio**, escriba un valor para el espacio que desea que aparezca entre cada columna.

**Sugerencia** Si imprime grupos que empiecen en una nueva página, no use la opción Orden al imprimir.

5. Elija **Aceptar**.

El **Diseñador de informes** refleja sus modificaciones.

Si el diseño ya contiene controles de informe en la banda Detalle, es posible que tenga que moverlos o cambiar su tamaño para que se ajusten a los límites de la nueva columna.

### **Establecer el alto de bandas de informe**

Al diseñar el informe, puede cambiar el alto de una banda de informe. El alto de una banda de informe determina la cantidad de espacio que cada banda de informe usa en la página dentro de los márgenes de página. Por ejemplo, si la banda Título se establece a media pulgada (1,27 cm), el Título aparecerá en la primera media pulgada de espacio después del margen superior. La banda de detalle muestra la cantidad de espacio asignada para cada registro impreso. La siguiente información se aplica a todas las bandas del informe. Puede establecer parámetros adicionales para las bandas Encabezado de grupo y Pie. Para obtener información sobre las Bandas de grupo, consulte la sección "Agrupar datos en el diseño" en el capítulo 7, [Diseño de informes y etiquetas](#), más adelante en este capítulo.

### **Para establecer un alto de banda exacta**

1. Haga doble clic en la barra para la banda apropiada.

Aparece un cuadro de diálogo para la banda.

2. En el cuadro **Alto**, escriba un valor para el alto.
3. Elija **Aceptar**.

### **Usar expresiones y funciones en controles de campo**

Puede incluir controles de campo en su informe o etiqueta para mostrar valores de varias expresiones, incluyendo campos de tablas y vistas, variables y cálculos. Las siguientes secciones describen algunas expresiones y funciones usadas normalmente como campos múltiples, fechas o números de página.

### **Agregar controles de campo**

Puede agregar controles de campo de varias maneras.

### **Para agregar campos de tabla del entorno de datos**

1. Abra el entorno de datos del informe.
2. Seleccione una tabla o vista.
3. Arrastre campos al diseño.

### **Para agregar campos de la barra de herramientas**

1. En la barra de herramientas **Controles de informes**, inserte un control **Campo**.

2. En el cuadro de diálogo [Expresión de informe](#), elija el botón del cuadro de diálogo situado después del cuadro **Expresión**.
3. En el cuadro **Campos**, haga doble clic en el nombre del campo que desee.

El nombre de tabla y el nombre de campo aparecen en el cuadro **Expresión para campo del informe**.

**Nota** Si el cuadro **Campos** está vacío, agregue una tabla o vista al entorno de datos.

No tiene que guardar el alias del nombre de tabla en la expresión. Puede eliminarlo o borrar las opciones del [Generador de expresiones](#).

4. Elija **Aceptar**.
5. En el cuadro de diálogo **Expresión de informe**, elija **Aceptar**.

Después de escribir la expresión, puede cambiar el formato o establecer opciones de impresión, posición o ampliación. Para obtener más información, consulte "Agregar un comentario a un control" en el capítulo 7, [Diseñar informes y etiquetas](#), del *Manual del usuario* y [Establecer las opciones de impresión de un control](#) más adelante en este mismo capítulo.

### Insertar controles de campo concatenados

Después de agregar los campos de tabla, se dará cuenta de que no se imprimen de la forma deseada en la página. Por ejemplo, al imprimir los controles de campo para Ciudad, Región y Código postal de forma independiente aparecen espacios no deseados entre cada valor. Puede recortar o concatenar los campos de tabla en una expresión de campo. El espacio requerido por cada valor para este control variará. Puede establecer que el control se ajuste para cada valor.

### Para combinar varios campos de tabla en una expresión

1. En la barra de herramientas **Controles de informes**, inserte un control **Campo**.  
  
**Sugerencia** Ajuste el tamaño del campo a la menor cantidad de espacio requerido por la expresión. Si se necesita más espacio, puede establecer el control para que se alargue para valores grandes, pero no puede reducirlo si necesita más espacio.
2. En el cuadro de diálogo [Expresión de informe](#), seleccione el botón de diálogo situado junto al cuadro **Expresión**.
3. En el cuadro de diálogo [Generador de expresiones](#), seleccione **ALLTRIM(expC)** en el cuadro **Cadena**.

La función de cadena aparece en el cuadro **Expression** con **expC** seleccionado.

4. Haga doble clic en el nombre del primer campo que desea que aparezca en el control.

El nombre de campo reemplaza `expC`.

5. Escriba un signo más después del nombre de campo, seleccione un signo más después del nombre de campo o seleccione + en el cuadro de funciones **Cadena**.
6. Escriba o seleccione **Texto** en la lista Funciones de cadena y, a continuación, escriba una coma.
7. Repita los pasos 3 y 4 para agregar campos adicionales que completen la expresión y escriba **Aceptar**.
8. En el cuadro de diálogo **Expresión de informe**, seleccione **Ajustar al contenido del campo**.

Cuando el control esté lleno, el espacio asignado al control se ajusta hacia abajo para alojar el valor de la expresión. Para obtener más información sobre **Ajustar al contenido del texto**, consulte [Imprimir controles con valores de longitud variable](#), más adelante en este mismo capítulo.

Para combinar varios campos en una expresión, coloque una función [ALLTRIM\(\)](#) antes de cada nombre de campo, coloque la puntuación entre comillas y coloque un signo menos entre cada elemento de la expresión. Si las longitudes de valores de campo no varían, como códigos postales o abreviaturas, puede insertar sólo el nombre de campo, como en este ejemplo:

```
ALLTRIM(ciudad)+" , "+región+" "+código_postal
```

Observe que se utilizan espacios entre comillas, en lugar de una coma, para separar la región y el código postal.

Para ver más ejemplos, vea el informe Invoice.frx en el directorio ...\\Samples\\Vfp98\\Solution\\Reports de Visual Studio.

### Recortar y concatenar expresiones de caracteres

Para recortar y concatenar rápidamente expresiones de caracteres en el Generador de expresiones, puede colocar comas entre expresiones de caracteres. El valor de la expresión que precede a la coma se recorta. También puede usar punto y coma para colocar la expresión en una nueva línea, si el valor recortado tiene una longitud mayor que cero. El siguiente ejemplo muestra expresiones de caracteres para campos de una lista de distribución:

```
nombre_contacto; dirección; ciudad, región, código_postal
```

**Nota** Úselos cuando no quiera incluir puntuación en el valor.

Si usa estos métodos, asegúrese de que el campo tiene establecido Ajustar al contenido del texto. Para obtener más información, consulte [Imprimir controles con valores de longitud variable](#) más adelante en este mismo capítulo.

### Insertar la fecha actual

Puede insertar un control de campo que imprime la fecha actual.

### Para insertar la fecha actual

1. En la barra de herramientas **Controles de informes**, inserte un control **Campo**.
2. En el cuadro de diálogo [Expresión de informe](#), seleccione el botón de diálogo situado junto al cuadro **Expresión**.
3. En el [Generador de expresiones](#), seleccione **DATE( )** en la lista **Fecha**.
4. Elija **Aceptar**.
5. En el cuadro de diálogo **Expresión de informe**, elija **Aceptar**.

### Insertar un número de página

Las bandas Encabezado de página o Pie de página contienen normalmente un número de página. Si usa un asistente o Informe rápido, se inserta un número de página en la banda Pie de página.

### Para insertar un número de página



1. En la barra de herramientas **Controles de informes**, inserte un control **Campo**.
2. En el cuadro [Expresión de informe](#), seleccione el botón de diálogo situado junto al cuadro **Expresión**.
3. En el [Generador de expresiones](#), seleccione **\_pageno** en la lista **Variables**.
4. Elija **Aceptar**.
5. En el cuadro de diálogo **Expresión de informe**, elija **Aceptar**.

**Sugerencia** Puede usar este procedimiento para insertar cualquiera de las variables de sistema de la lista Variables en el informe.

### Definir variables de informe

Para manipular datos y mostrar el resultado de los cálculos en un informe, puede utilizar variables de informe. Puede calcular valores con variables de informe y usarlos después para calcular valores posteriores.

### Para definir una variable de informe

1. Abre o crea un informe.

2. En el menú **Informe**, elija **Variables**.
3. En el cuadro de diálogo [Variables del informe](#), seleccione el cuadro **Variables** y escriba un nombre para la variable.
4. En el cuadro **Almacenar valor**, escriba un nombre de campo o alguna otra expresión.
5. Si es necesario, seleccione una opción de cálculo.
6. Si es necesario, en el cuadro **Valor inicial**, escriba una expresión que establezca el valor inicial.
7. Elija **Aceptar**.

Puede utilizar la variable en cualquier expresión que introduzca en el informe.

Para contar todas las entradas Canadá en la tabla Company, utilice esta expresión y seleccione "Recuento" como opción de cálculo.

```
IIF(country="Canadá",1,0)
```

El siguiente ejemplo muestra tres variables para una hoja de tiempos sencilla:

Para almacenar este valor	Cree esta variable	Mediante esta expresión
Hora de llegada del empleado	tArrive	hour_in + (min_in / 60)
Hora de salida del empleado	tLeave	hour_out + (min_out / 60)
Tiempo total que el empleado estuvo presente	tDayTotal	tLeave - tArrive

Puede utilizar la variable tDayTotal en una gran variedad de cálculos, tales como el número de horas trabajadas en una semana, un mes, o un año; el promedio de número de horas trabajadas cada día, etc.

Para ver ejemplos de variables de informe, vea los informes Percent.frx e Invoice.frx en el directorio ...\\Samples\\Vfp98\\Solution\\Reports de Visual Studio.

### Reordenar variables de informe

Las variables de informe se evalúan en el orden en que aparecen en la lista y pueden afectar a los valores de las expresiones que las utilizan. Por ejemplo, si la variable 1 se utiliza para definir el valor de la variable 2, debe aparecer antes que la variable 2. En el ejemplo anterior de la hoja de tiempos, tArrive y tLeave deben preceder a tDayTotal.

### Para cambiar el orden de las variables de informe

1. En el menú **Informe**, elija **Variables**.
2. En el cuadro **Variable**, arrastre el botón situado a la izquierda de la variable para cambiar el orden.
3. Elija **Aceptar**.

### **Establecer el valor inicial de una variable**

Si utiliza una variable en cálculos, asegúrese de que inicializa dicha variable con un valor diferente de cero para no producir un error de división por cero. Si no especifica un valor, Visual FoxPro asignará un valor predeterminado.

### **Para establecer el valor inicial de una variable**

1. En el menú **Informe**, elija **Variables**.
2. En el cuadro **Variable**, seleccione la variable que desea inicializar.
3. En el cuadro **Valor inicial**, introduzca el valor.
4. Elija **Aceptar**.

Si reordena los grupos en su informe, sus variables de informe podrían no restablecerse en el campo correcto. Por ejemplo, si su informe contiene dos grupos, el primero agrupado por país y el segundo agrupado por fecha, y cambia el orden de los grupos, las variables continuarán ajustadas de acuerdo a las posiciones originales de los grupos.

Puede cambiar el valor de un cálculo si especifica cuándo se restablece la variable. De forma predeterminada, Visual FoxPro restablece las variables de informe al final del informe.

### **Para restablecer una variable al final de un informe, una página o una columna**

1. En el menú **Informe**, elija **Variables**.
2. En el cuadro **Restablecer**, elija una opción.
3. Elija **Aceptar**.

### **Para restablecer una variable al entrar o salir de alguna banda**

1. En el [Diseñador de informes](#), abra el informe.
2. Haga doble clic en la barra de la banda del informe que desee.
3. En el área **Ejecutar expresión** del cuadro de diálogo de la banda, elija uno de los botones al final del cuadro **Al entrar** o **Al salir**.



4. Escriba una expresión para restablecer la variable cada vez que se entre o se salga de esa banda.

## Formato de controles de campo

Después de insertar un control de campo, puede cambiar el tipo de datos del control y el formato de impresión. Los tipos de datos pueden ser Character, Numeric o Date. Cada uno de estos tipos de datos tiene sus propias opciones de formato, incluyendo la opción de crear su propia plantilla de formato. El formato determina cómo se muestra el campo cuando se imprime el informe o la etiqueta.

Puede escribir funciones de formato directamente en el cuadro Expresiones del cuadro de diálogo [Expresión de informe](#) o seleccionar opciones en el cuadro de diálogo Formato.

Normalmente, puede convertir toda la salida alfabética a mayúsculas, insertar comas o comas decimales en salidas numéricas, mostrar salidas numéricas en formato de moneda o convertir un formato de fecha en otro.

## Opciones de formato para controles de informe

Para controles de campo, puede establecer varias opciones de formato para cada tipo de datos.

### Para dar formato a un control de campo

1. Elija el control **Campo**.
2. En el cuadro de diálogo [Expresión de informe](#), elija el botón de diálogo situado junto al cuadro **Formato**.
3. En el cuadro de diálogo **Formato**, seleccione el tipo de datos para el campo: **Carácter**, **Número**, o **Fecha**.

El área **Opciones de edición** muestra las opciones de formato disponibles para ese tipo de datos.

**Nota** Este tipo de datos sólo se aplica al control del informe. Refleja el tipo de datos de la expresión y no cambia el tipo de datos del campo en la tabla.

4. Seleccione las opciones de alineación y formato que desee.

El cuadro de diálogo Formato muestra opciones diferentes en función del tipo de datos que elija. También puede crear una plantilla de formato si escribe caracteres en el cuadro Formato.

## Alinear texto en un campo

Puede alinear el contenido de campos en controles de dos formas. Esta configuración no cambia la posición del control del informe, sólo el contenido en el espacio del control.

### Para alinear el texto en un control de campo

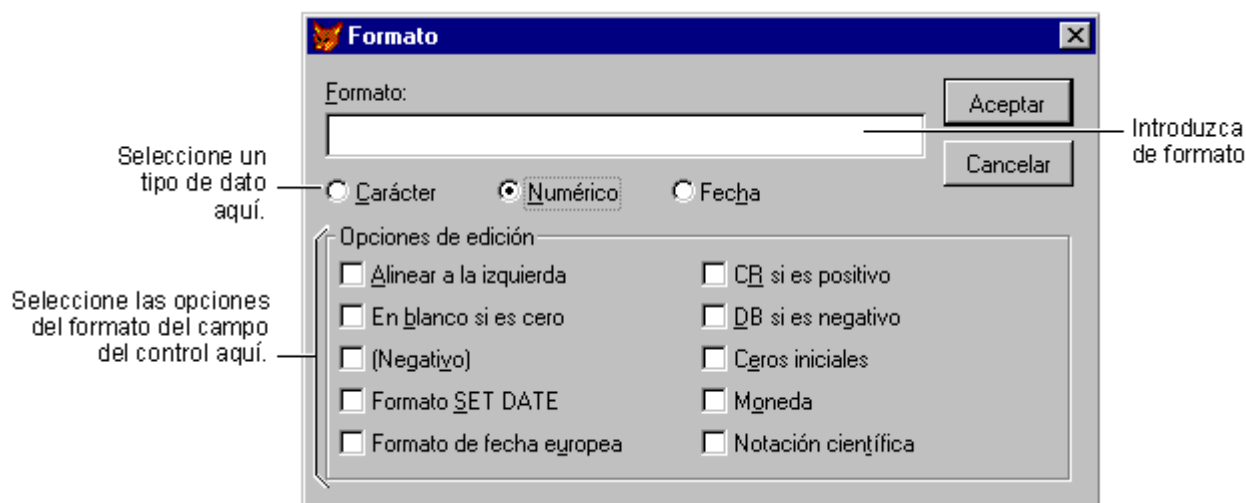
1. Seleccione los controles que desea cambiar.

2. En el menú **Formato**, elija **Alineación de texto**.
3. En el submenú, elija el comando apropiado.

### Para alinear texto en un campo

1. Elija el control **Campo**.
2. En el cuadro de diálogo [Expresión de informe](#), elija el botón de diálogo situado junto al cuadro **Formato**.

### Cuadro de diálogo Formato para una expresión de tipo Numérico



3. En el cuadro de diálogo **Formato**, seleccione el tipo de datos para el campo: **Carácter**, **Numérico** o **Fecha**.
4. Seleccione la alineación y las opciones de formato que quiera.

### Definir plantillas de formato de campo

Una plantilla de formato le permite personalizar el formato del campo. Si escribe una combinación de caracteres y códigos en el cuadro Formato del cuadro de diálogo [Expresión de informe](#) o el cuadro de diálogo Formato, puede crear una amplia variedad de formatos de impresión. Los caracteres que escriba aparecen como texto junto con el valor del campo. Los códigos que escriba determinan la apariencia de la salida de campos. Por ejemplo, si usa la siguiente plantilla de formato para un campo numérico de 10 dígitos, los caracteres (paréntesis, espacios y guiones) se imprimen junto con los datos numéricos.

Plantilla de formato	Salida impresa
(999) 999-9999	(123) 456-7890

### Cambio de fuentes

Puede cambiar el tamaño de fuente y el tamaño del texto para cada control campo o etiqueta, o puede cambiar la fuente predeterminada para el informe.

### Para cambiar las fuentes y el tamaño en un informe

1. Seleccione el control.
2. En el menú **Formato**, seleccione **Fuente**.

Aparece el cuadro de diálogo **Fuente**.

3. Seleccione la fuente apropiada y el tamaño en puntos y, a continuación, elija **Aceptar**.

### Para cambiar la fuente predeterminada

1. En el menú **Informe**, elija **Fuente predeterminada**.
2. En el cuadro de diálogo **Fuente**, seleccione la fuente apropiada y el tamaño en puntos que desee como valores predeterminados y, a continuación, elija **Aceptar**.

Sólo los controles insertados después de haber cambiado la fuente predeterminada reflejarán la nueva configuración de fuente. Para cambiar objetos existentes, selecciónelos a todos y cambie la fuente con la opción Fuente en el menú Formato.

### Cortar una imagen u objeto OLE

Es posible que la imagen o el objeto OLE que haya insertado no se ajuste al marco dibujado al crear el control. De forma predeterminada, la imagen o el objeto conservan su tamaño original. Puede recortarlo o reducirlo para que se ajuste a su marco.

Si la imagen u objeto OLE es mayor que el marco creado en el Diseñador de informes, sólo aparecerá en el marco una porción de la imagen o el objeto. La imagen o el objeto están acoplados a la parte superior izquierda del marco. No puede ver la parte inferior derecha que se extiende más allá del marco.

### Para ajustar una imagen al marco

1. En el [Diseñador de informes](#), cree un [Control Imagen/Control OLE dependiente](#).
2. En el cuadro de diálogo [Imagen para informe](#), seleccione **Cambiar la escala de la imagen, conservar la forma**.

Aparece la imagen completa, que llenará la mayor parte del marco y conservará las posiciones relativas. Esto protege a la imagen de la distorsión vertical u horizontal.

### Para rellenar el marco con la imagen

1. En el **Diseñador de informes**, cree un control **Imagen/Control OLE dependiente**.

2. En el cuadro de diálogo **Imagen para informe**, seleccione **Cambiar la escala de la imagen, rellenar el marco**.

La imagen cambia para llenar el marco que ha cambiado de tamaño. Si es necesario, la imagen se amplía verticalmente u horizontalmente para ajustarse al marco.

Para ver un ejemplo de un informe con imágenes, vea el informe Wrapping.frx en el directorio ... \Samples\Vfp98\Solution de Visual Studio.

### Centrar un objeto OLE

Los objetos OLE incluidos en un campo General pueden variar en forma y tamaño. Si el objeto de un campo General es más pequeño que el marco, aparece en la esquina superior izquierda del marco. Puede centrarlo para asegurar que todos los objetos más pequeños que el marco estén centrados en el marco del informe o de la etiqueta. Las imágenes de archivo no están centradas porque no varían.

### Para centrar objetos OLE de campo General

1. En el [Diseñador de informes](#), cree un control [Imagen/Control OLE dependiente](#).
2. En el cuadro de diálogo [Imagen para informe](#), seleccione **Centrar imagen**.

Los objetos OLE impresos se centran en el área en una vista previa del informe o al imprimirlo.

### Cambiar colores de controles de informe

Puede cambiar el color de un campo, etiqueta, línea o rectángulo.

### Para cambiar los colores

1. Seleccione los controles que desea modificar.
2. En la barra de herramientas **Paleta de colores**, elija **Color de primer plano** o **Color de fondo**.
3. Seleccione el color que desee.

### Guardar un informe como HTML

Puede usar la opción **Guardar como HTML** del menú **Archivo** cuando cree un informe para guardar el contenido de un formulario como un archivo HTML (Lenguaje de marcado de hipertexto).

### Para guardar un informe como HTML

1. Abra el informe.
2. Elija **Guardar como HTML** en el menú **Archivo**. (Se le pedirá que guarde el informe si lo ha modificado).

3. Escriba el nombre del archivo HTML que va a crear y elija **Guardar**.

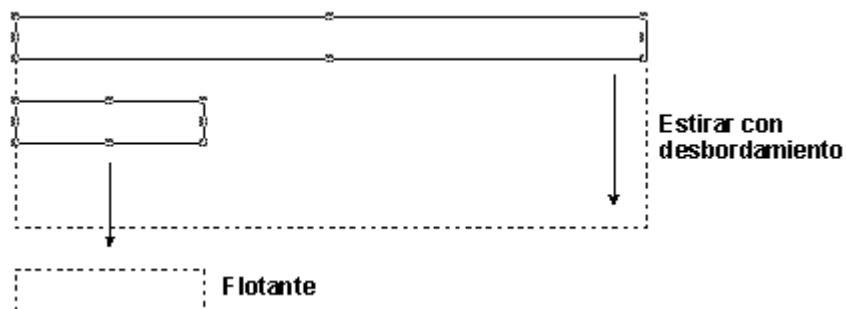
## Opciones de impresión para controles

El diseño general y la posición de bandas de los controles determinan el momento y el lugar en que se imprimen. También puede establecer opciones de impresión específicas para cada control. Cada control puede tener un tamaño predeterminado en base a su valor (campos y tablas) o el tamaño creado (líneas, rectángulos e imágenes). La longitud del control en el diseño define el ancho de presentación del control. Como el valor de algunos valores varía de registro a registro, puede establecer el alto del control para alargar hacia abajo y mostrar así el valor completo. Si no establece que se amplíe, el valor se truncará en el ancho de presentación. No puede cambiar el tamaño de controles Etiqueta, pero puede cambiar el tamaño de los demás controles.

### Imprimir controles con valores de longitud variable

Si quiere que un control sólo use el espacio necesario, puede establecer que se alargue. Por ejemplo, los valores de una expresión pueden variar de registro a registro. En lugar de asignar una cantidad fija de espacio en el informe que aloje el valor más largo, puede establecer que el control se alargue hacia abajo para mostrar todo el valor. Puede establecer que los controles situados por debajo del control que se amplía floten por debajo de la página con respecto al control que se ajusta.

### Ejemplos de controles que se amplían y controles que flotan



La opción de alargar está disponible para campos, líneas verticales, rectángulos y rectángulos redondeados.

Para ver un ejemplo de controles que se amplían y flotan, vea el informe Wrapping.frx en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.

### Para establecer que un campo se amplíe con su valor

1. Haga doble clic en el control campo para mostrar su cuadro de diálogo.
2. Seleccione **Ajustar al contenido del texto**.

Hay que establecer que los controles que se colocan con respecto a controles que se amplían floten, o se sobrescribirán.

**Para establecer que un control flote**

1. Haga doble clic en el control para mostrar su cuadro de diálogo.
2. En el cuadro de diálogo del control, seleccione **Flotante**.

**Precaución** Algunos de los datos no se han podido sobrescribir durante la impresión si: (1) coloca un campo respecto al fondo de la banda e incluye *por debajo* de este campo otro campo que está colocado respecto a la parte superior de la banda y se puede alargar; o (2) coloca un campo respecto a la parte superior de la banda e incluye *por encima* de este campo otro campo colocado respecto a la parte superior de la banda y que se puede alargar.

También puede establecer que se alarguen líneas, rectángulos y rectángulos redondeados. Se pueden alargar con respecto a la banda o, si forman parte de un grupo de controles, se pueden alargar con respecto al control más grande del grupo.

**Para establecer que se alargue una línea o un rectángulo**

1. Haga doble clic en el control para mostrar su cuadro de diálogo.
2. En el área **Alargar hacia abajo**, seleccione una opción.

**Para imprimir un borde en torno a un control que se puede alargar**

1. Dibuje un rectángulo alrededor de los controles que se pueden alargar.
2. Haga doble clic en el rectángulo para mostrar el cuadro de diálogo [Rectángulo/Línea](#).
3. En el área **Alargar hacia abajo**, seleccione **Alargar con relación al objeto más alto del grupo**.
4. Elija **Aceptar**.
5. Arrastre un cuadro de selección en torno al rectángulo.
6. En el menú **Formato**, elija **Agrupar**.

Los controladores de selección aparecen en las esquinas del rectángulo. A partir de este momento puede tratar todos los controles como uno sólo. El rectángulo se alargará con el campo de tamaño ajustable. Independientemente de cuánto se alargue el valor del campo, el rectángulo mantendrá su borde alrededor del campo. Puede colocar dos de estos grupos lado a lado y uno no se verá afectado por el alargamiento del otro.

**Para imprimir un control de tamaño ajustable por debajo de otro**

1. Inserte los dos controles en el diseño, uno debajo del otro.
2. Haga doble clic en la parte superior del control para mostrar el cuadro de diálogo del control.

3. En el área **Posición del campo**, seleccione **Borde superior de la banda** y, a continuación, elija **Aceptar**.
4. Haga doble clic en la parte inferior del control para mostrar el cuadro de diálogo del control.
5. En el área **Posición del campo**, seleccione **Flotante** y, a continuación, elija **Aceptar**.

Los dos valores de registro se imprimirán completamente y no se sobrescribirán.

### Establecer las opciones de impresión de un control

Puede controlar cuándo y cómo se imprime cada control de informe en el informe. Para obtener más información sobre las opciones de impresión, consulte [cuadro de diálogo Imprimir-Condiciones](#).

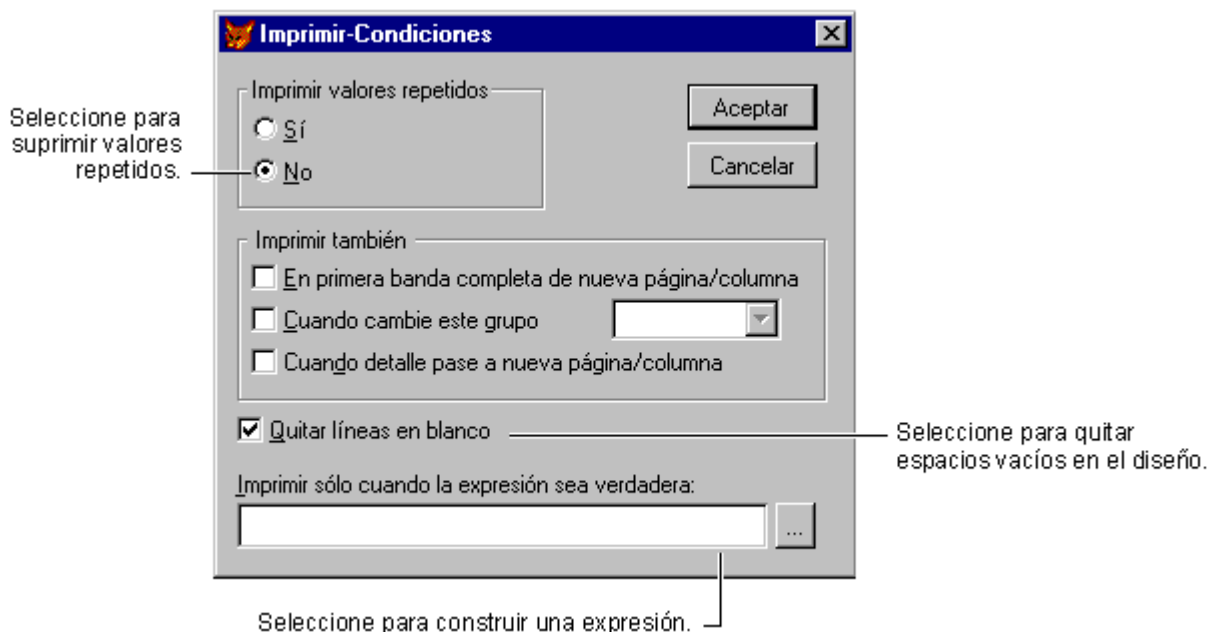
### Suprimir valores repetidos

Para controles de campo, puede suprimir los valores repetidos para registros consecutivos de forma que el valor se imprima una vez para el primer registro pero no aparezca en los siguientes registros hasta que cambie. Por ejemplo, si va a imprimir una factura y uno de los campos contiene la fecha de la transacción, la fecha sólo se imprime una vez para transacciones que tuvieron lugar en esa fecha.

### Para suprimir valores repetidos

1. Haga doble clic en el control para mostrar el cuadro de diálogo del control.
2. Elija **Imprimir-Condiciones** para mostrar el cuadro de diálogo **Imprimir-Condiciones**.

### Cuadro de diálogo Imprimir-Condiciones



3. En el área **Imprimir valores repetidos**, seleccione **No** y, a continuación, elija **Aceptar**.

**Para repetir en sólo una nueva página o columna**

1. Haga doble clic en el control.
2. Elija **Imprimir-Condicion**.
3. En el área **Imprimir valores repetidos**, seleccione **No**.
4. En el área **Imprimir también**, seleccione **En primera banda completa de nueva página/columna** y, a continuación, elija **Aceptar**.

**Para repetir cuando se desborda una banda de detalle a una nueva página o columna**

1. Haga doble clic en el control.
2. Elija **Imprimir-Condicion**.
3. En el área **También imprimir**, seleccione **Cuando detalle pase a nueva página/columna** y, a continuación, elija **Aceptar**.

**Generar expresiones de impresión**

Puede agregar expresiones a un control: se evalúan antes de imprimir el campo. Si la expresión se evalúa a falso (.F.), el campo no se imprimirá. Si agrega una expresión, se desactivan todas las demás opciones del cuadro de diálogo Imprimir-Condicion excepto Quitar líneas en blanco.

Para ver ejemplos de condiciones Imprimir-Condicion, vea los informes Colors.frx y Ledger.frx del ejemplo Soluciones.

**Para agregar una expresión de impresión**

1. Haga doble clic en el control.
2. Elija **Imprimir-Condicion**.
3. En el cuadro **Imprimir sólo cuando la expresión sea verdadera**, escriba una expresión.

–O bien–

Haga clic en el botón del diálogo para crear una expresión con el [Generador de expresiones](#).

**Suprimir líneas en blanco**

El informe puede contener registros que no contengan valores para cada control de campo. De forma predeterminada, Visual FoxPro deja en blanco el área para ese campo. Puede quitar estas áreas en blanco para crear una presentación de la información más agradable y continua.

**Para suprimir líneas en blanco**



1. Haga doble clic en el control que produce las líneas en blanco en el informe.
2. Elija **Imprimir-Condiciones**.
3. Seleccione **Quitar líneas en blanco**.

Visual FoxPro quitará la línea del informe si se evalúa como línea en blanco. Si el campo no se imprime o si el campo de tabla está vacío, Visual FoxPro comprueba si hay otros controles en la línea. Si no se encuentra ninguno, se quita la línea. Si no selecciona esta opción y no hay otros controles en la misma línea, se imprime una línea en blanco.

## Establecer opciones de impresión para grupos

Puede controlar cómo se imprimen grupos en el informe. A veces querrá que cada grupo empiece en una página distinta, o controlar cuándo se imprime el encabezado de grupo.

## Establecer saltos de página de grupos

Además de seleccionar el campo o expresión que hay que agrupar, el cuadro de diálogo Agrupar datos le permite especificar opciones de salto de página para grupos.

## Elegir una opción de encabezado de grupo

Puede que quiera que los grupos aparezcan en la siguiente columna para informes con varias columnas, en una nueva página para formularios o con una nueva página numerada como 1. El cuadro de diálogo [Agrupar datos](#) ofrece cuatro opciones para llevar a cabo estas tareas. Puede:

- Iniciar un grupo en una nueva columna.
- Iniciar cada grupo en una nueva página.
- Restablecer el número de página a 1 para cada grupo.
- Volver a imprimir el encabezado de grupo en cada página.

Después de escribir una expresión, puede seleccionar estas opciones en el área Propiedades de grupo.

## Evitar encabezados de grupo huérfanos

A veces un grupo puede imprimirse parcialmente en una página y terminar en la siguiente. Para evitar que un encabezado de grupo se imprima cerca del final de la página con la mayor parte de los registros en la página siguiente, puede establecer la distancia mínima de la parte inferior a la que se va a imprimir un encabezado de grupo. Si el encabezado se coloca más cerca de la parte inferior de la página que el número de pulgadas o centímetros que haya escrito, Visual FoxPro imprime el encabezado en una nueva página.

## Para evitar encabezados de grupo huérfanos

1. En el menú **Informe**, elija **Agrupar datos**.
2. En el cuadro de diálogo [Agrupar datos](#), elija o escriba un valor en el cuadro **Comenzar cada**

**grupo en una nueva página.**

**Sugerencia** Para determinar un valor para un control huérfano, agregue el alto del Encabezado de grupo de una a tres veces al alto de Detalle.

**Imprimir valores suprimidos cuando el grupo cambia**

Si se suprimen valores repetidos, es posible que desee imprimir cuando cambie un grupo concreto.

**Para imprimir valores repetidos cuando el grupo cambia**

1. Haga doble clic en el control para mostrar el cuadro de diálogo del control.
2. Elija el botón **Imprimir-Condiciones** para imprimir el cuadro de diálogo **Imprimir-Condiciones**.
3. Seleccione **Cuando cambie este grupo**.

Los grupos definidos para el informe aparecen en el cuadro.

4. Seleccione un grupo del cuadro y, a continuación, elija **Aceptar**.

**Repetir encabezados de grupo**

Cuando un grupo continúa en la página siguiente, es posible que quiera que el encabezado de grupo se repita en la parte superior del grupo para ver información continuada. Si tiene varios grupos de datos en el informe, el encabezado de las páginas subsiguientes será el del último grupo de la lista de grupos. Coloque todos los controles que desee imprimir para el encabezado de grupo en la banda de encabezado del último grupo de la lista.

**Para repetir el encabezado de grupo en la página siguiente**

- En el cuadro de diálogo [Agrupar datos](#), seleccione el grupo que desea repetir y, a continuación, elija **Volver a imprimir el encabezado de grupo en cada página**.

Si no desea repetir el encabezado de grupo, desactive esta casilla de verificación.

**Controlar el resultado de informes y etiquetas**

Puede controlar dónde se envía el resultado de un informe o una etiqueta si utiliza alguna de las siguientes palabras clave con el comando [REPORT](#) o [LABEL](#):

- PRINT
- PREVIEW
- FILE

Si no utiliza una de estas palabra clave, el informe se enviará a la pantalla o a la ventana activa.

## Seleccionar registros para imprimir

Cuando imprima un informe, es posible que desee limitar el número de registros que aparecen en el informe mediante criterios de selección. Puede:

- Elegir el alcance de registros si especifica una cantidad o intervalo.
- Generar una expresión FOR que seleccione registros que cumplan una condición.
- Generar una expresión WHILE que seleccione registros hasta que se encuentre uno que no cumpla una condición.

Puede usar cualquier combinación de estas opciones. La expresión WHILE pasa por alto los otros criterios.

## Imprimir una cantidad o un intervalo de registros

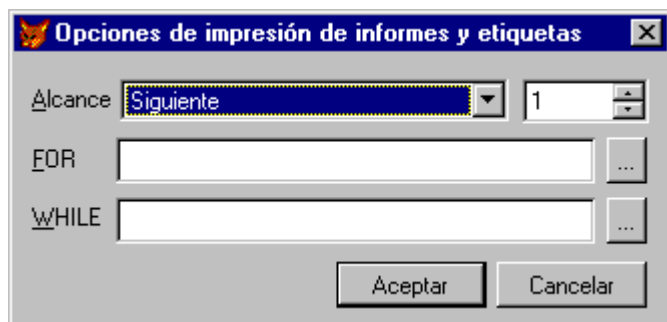
Puede limitar el número de registros si especifica una cantidad o un intervalo de registros. Con la opción Alcance, puede seleccionar un único registro o un grupo de registros colocados secuencialmente en el archivo.

**Nota** El índice activo y el puntero de registro activo afecta a los resultados de las opciones de alcance Siguiente y Resto. Por ejemplo, el siguiente registro de una tabla indexada por el último nombre es probablemente diferente que el de una tabla indexada por estado. Esto no afecta a la opción Registro porque el número para un registro no cambia cuando se indexa la tabla.

## Para seleccionar un número limitado de registros

1. En el menú **Archivo**, elija **Imprimir**.
2. En el cuadro de diálogo **Imprimir**, elija **Opciones**.
3. En el cuadro de diálogo **Opciones de impresión**, elija **Opciones**.

## Cuadro de diálogo Opciones de impresión de informes y etiquetas



4. En el cuadro de diálogo **Opciones de impresión de informes y etiquetas**, elija **Alcance**.
5. Seleccione la opción de [alcance](#) apropiada.

Para imprimir	Elija esta opción de alcance
Cada registro del archivo de origen	TODO
Un intervalo de registro que empieza por 1	SIGUIENTE
Un registro específico por número	REGISTRO
El registro actual más todos los que le siguen hasta el final del archivo	RESTO

Visual FoxPro imprime el informe con datos de los registros que tengan el alcance que haya seleccionado.

### Imprimir registros que cumplen una condición

Si los registros que desea seleccionar no están ordenados secuencialmente en la tabla, puede generar una expresión lógica que especifique criterios de selección que debe cumplir un registro para que se imprima. Por ejemplo, puede elegir imprimir todos los registros con un valor concreto en un campo.

### Para introducir criterios para seleccionar registros

1. En el menú **Archivo**, elija **Imprimir**.
2. En el cuadro de diálogo **Imprimir**, elija **Opciones**.
3. En el cuadro de diálogo **Opciones de impresión**, elija **Opciones**.
4. En el cuadro de diálogo **Opciones de impresión de informes y etiquetas**, elija **Alcance**.
5. En el cuadro **FOR**, escriba una expresión FOR.

–O bien–

Asegúrese de que los orígenes de registros usados por el informe están abiertos y, a continuación, elija el botón **FOR** para usar el [Generador de expresiones](#).

**Nota** No tiene que incluir el comando FOR en la expresión. Por ejemplo, escriba **country = "Canadá"** para ver únicamente los datos canadienses.

Visual FoxPro evalúa todos los registros e imprime el informe con los registros que cumplen la condición de la expresión.

### Controlar la selección de registros para imprimir

Al imprimir, puede especificar una condición que se tiene que cumplir para seguir la evaluación y selección de registros. Esta condición se escribe en una expresión WHILE. Mientras la expresión WHILE sea verdadera, Visual FoxPro procesa el origen de datos. Después de buscar un registro que no cumpla la condición, Visual FoxPro termina el proceso de evaluación e imprime los registros

seleccionados. Esta opción le permite seleccionar registros en base a información externa a los valores contenidos en los campos.

**Sugerencia** Si usa una expresión WHILE en un archivo que no se ha indexado, el proceso de selección puede terminar antes de evaluar todos los registros apropiados. Antes de imprimir el informe, asegúrese de que la tabla de origen tiene el índice activo apropiado para la expresión WHILE que desea usar.

### Para escribir criterios para terminar la selección de registros

1. En el menú **Archivo**, elija **Imprimir**.
2. En el cuadro de diálogo **Imprimir**, elija **Opciones**.
3. En el cuadro de diálogo **Opciones de impresión**, elija **Opciones**.
4. En el cuadro de diálogo **Opciones de impresión de informes y etiquetas**, elija **Alcance**.
5. En el cuadro de diálogo **WHILE**, escriba una expresión WHILE.

–O bien–

Elija el botón **WHILE** para usar el [Generador de expresiones](#).

**Nota** No tiene que incluir el comando WHILE en la instrucción. Por ejemplo, escriba **sales > 1000** para ver sólo las ventas por encima de 1000 pesetas.

Visual FoxPro imprime el informe con los registros que evalúa mientras la expresión sea verdadera.

### Imprimir informes y etiquetas

Si desea enviar el informe a la impresora, puede enviarlo directamente o mostrar el cuadro de diálogo Especificar impresora.

### Para enviar un informe a la impresora

- Utilice la palabra clave TO PRINTER del comando [REPORT](#) o [LABEL](#).

Por ejemplo, el código siguiente envía el informe MiInforme a la impresora predeterminada y hace que no se muestre en la pantalla:

```
REPORT FORM MIINFORME.FRX TO PRINTER NOCONSOLE
```

### Para mostrar el cuadro de diálogo Especificar impresora antes de imprimir el informe

- Utilice las palabras clave TO PRINTER PROMPT del comando [REPORT](#) o [LABEL](#).

Por ejemplo, el código siguiente muestra el cuadro de diálogo Especificar impresora, luego envía el

informe `MiInforme` a la impresora predeterminada y detiene la impresión del informe en la ventana activa:

```
REPORT FORM MIINFORME.FRX TO PRINTER PROMPT NOCONSOLE
```

## Vista preliminar de informes y etiquetas

Si desea mostrar una vista preliminar del informe, puede enviarlo a la ventana Vista preliminar en el Diseñador de informes.

### Para ver una vista preliminar de un informe

- Utilice la palabra clave `PREVIEW` del comando [REPORT](#).

Por ejemplo, el código siguiente muestra el informe en una ventana modal:

```
REPORT FORM MIINFORME.FRX PREVIEW
```

De forma predeterminada, la ventana Vista preliminar es modal, pero permite tener acceso a la barra de herramientas. Si desea realizar la vista preliminar de forma no modal, puede agregar la palabra clave `NOWAIT` al comando `REPORT`.

Por ejemplo, el código siguiente muestra el informe en una ventana no modal:

```
REPORT FORM MIINFORME.FRX PREVIEW NOWAIT
```

Si desea ver los resultados en una ventana específica, puede incluir la cláusula `WINDOW` para especificar una ventana creada con `DEFINE WINDOW`.

```
REPORT FORM MIINFORME.FRX PREVIEW WINDOW MYWINDOW
```

## Imprimir informes en un archivo

Si desea crear una versión electrónica del informe, puede enviarlo a un archivo con formato para su impresora o a un archivo ASCII. Si envía informes a archivos puede imprimirlos más tarde en un proceso por lotes.

Si desea crear un archivo ASCII, puede crear un archivo que incluya sólo el texto, guiones y signos más para representar líneas y formas. Los colores y las fuentes elegidas no están incluidos. También puede especificar el número de caracteres por línea y el número de líneas por página.

### Para imprimir un informe en un archivo ASCII

- Use las palabras clave `FILE` y `ASCII` del comando [REPORT](#).

El ejemplo siguiente define las variables para la página ASCII e imprime un informe llamado `Miinforme.frx` en un archivo ASCII llamado `Miarch.txt`.

### Imprimir en un archivo ASCII

Programa	Comentario
<code>_ascirows = nLines</code>	Define el número de líneas por página.
<code>_asciicols = nChars</code>	Define el número de caracteres por línea.
<code>REPORT FORM MIINFORME.FRX TO FILEMIARCH.TXT ASCII</code>	Ejecuta el informe.

### Guardar un informe como HTML

Puede usar la opción **Guardar como HTML** del menú **Archivo** cuando cree o modifique un informe para guardar su contenido como un archivo HTML (Lenguaje de marcado de hipertexto).

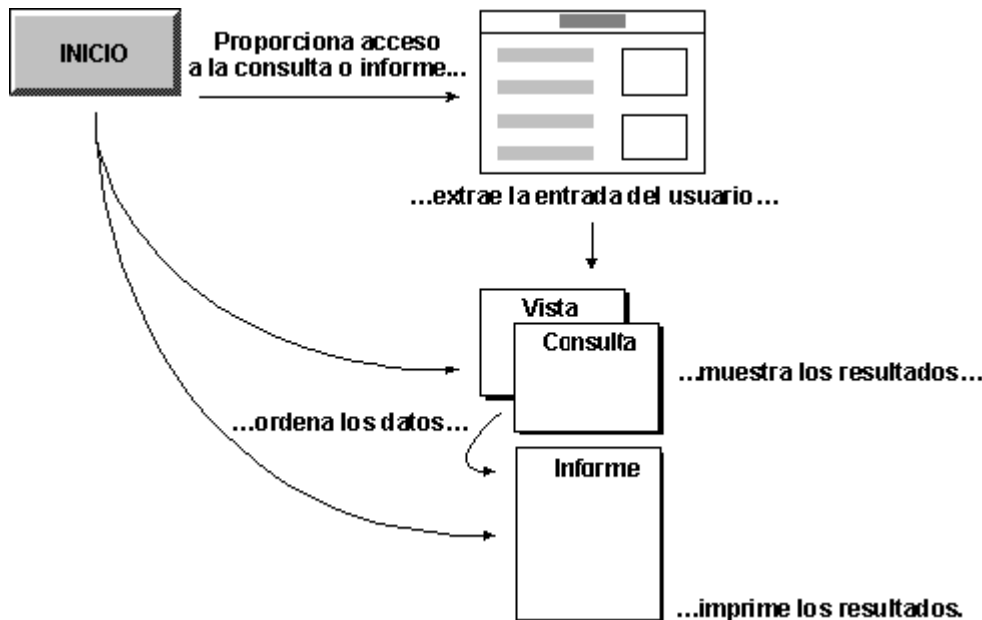
#### Para guardar un informe como HTML

1. Abra el informe.
2. Elija **Guardar como HTML** en el menú **Archivo**. Esta opción sólo está disponible cuando se ha guardado el informe en disco.
3. Escriba un nombre para el archivo HTML que desea crear y elija **Guardar**.

### Integrar consultas e informes

Cuando haya creado los componentes de su aplicación, puede integrarlos. Esta ilustración muestra algunas formas en las que puede agregar consultas e informes a su aplicación.

#### Algunas formas de integrar consultas e informes



Puede agregar código que ejecute una consulta o un informe a los siguientes objetos de su aplicación.

- Un botón de un formulario. Para ver más información acerca de formularios y botones, consulte el capítulo 9, [Crear formularios](#).
- Un elemento de un menú. Para ver más información acerca de cómo agregar elementos a un menú, consulte el capítulo 12, [Diseñar menús y barras de herramientas](#).

### Para agregar una consulta, vista o programa

- Agregue un comando [DO](#) o [USE](#) al código subyacente de un botón de comando de un formulario, un botón de una barra de herramientas o un elemento de menú.

Por ejemplo, agregue código similar al de las siguientes líneas:

```
DO MICON.S.QPR
DO MIPROG.PRG
USE mivista
```

Dispone de varias opciones para integrar informes en su aplicación.

- Si desea que el usuario simplemente inicie el informe y lo recoja impreso, puede hacer que el usuario inicie el informe si agrega el comando REPORT a un control de un formulario, un comando en un menú o un botón de una barra de herramientas.
- Si desea permitir al usuario escribir algunas variables utilizadas en el informe, puede recoger los valores introducidos, de la misma manera que hizo antes para las consultas. Por ejemplo, un usuario podría introducir un intervalo de fechas que incluirá el informe. Para ver más información al respecto, consulte [Recoger entradas de usuario con consultas](#) más adelante en este mismo capítulo.
- Si desea que el usuario cree informes personalizados, puede ofrecerle la posibilidad de crear nuevos informes o modificar los ya existentes con el Diseñador de informes.



## Para ejecutar informes y etiquetas

- Utilice los comandos [REPORT](#) o [LABEL](#).

Por ejemplo, podría utilizar código como el de las siguientes líneas:

```
REPORT FORM MIINF.FRX
LABEL FORM MIETIQL.LBX
```

## Para modificar informes y etiquetas

- Utilice los comandos [MODIFY REPORT](#) o [MODIFY LABEL](#).

Por ejemplo, agregue código similar al de las siguientes líneas:

```
MODIFY REPORT MIINF.FRX
MODIFY LABEL MIETIQ.LBX
```

## Para crear informes y etiquetas

- Utilice los comandos [CREATE REPORT](#) o [CREATE LABEL](#).

Por ejemplo, podría utilizar código similar al de las siguientes líneas:

```
CREATE REPORT MIINF.FRX
CREATE LABEL MIETIQ.LBX
```

## Recoger entradas de usuario con consultas

Si quiere recoger valores desde un formulario, puede utilizar variables en una instrucción SELECT - SQL y luego utilizarlos en la instrucción, o ejecutar la instrucción más tarde.

Para recoger valores para uso inmediato, puede utilizar explícitamente el nombre del formulario o una referencia abreviada para el formulario en su instrucción SELECT - SQL. En este ejemplo, la referencia abreviada está en la cláusula WHERE.

## Recogida de valores mediante referencias abreviadas en una instrucción SELECT - SQL

Código	Comentario
<pre>SELECT * ;   FROM tastrade!customer ;   WHERE customer.country = ;         THISFORM.ControlName1.Value ;   AND customer.region = THISFORM.ControlName2.Value ;   GROUP BY customer.postal_code ;   ORDER BY customer.postal_code, customer.company_name</pre>	<p>Utilice THISFORM como referencia abreviada para el formulario actualmente activo y sustituya los nombres de los controles por ControlName1 y ControlName2.</p>

Si no quiere utilizar referencias para los controles, puede definir variables en el código. Use variables en el código si desea almacenar los valores desde un formulario pero no utilizarlos necesariamente mientras el formulario esté activo.

### Recogida de valores para uso posterior

Código	Comentario
<code>cValue = THISFORM.ControlName.Value</code>	Defina la variable.
<pre>SELECT * ; FROM tastrade!customer ; WHERE customer.country = cValue ; GROUP BY customer.postal_code ; ORDER BY customer.postal_code, ; customer.company_name</pre>	Utilice la variable que ha definido en la instrucción SELECT - SQL.

Si no define la variable antes de ejecutar la consulta, aparecerá un mensaje de error que indique que la variable no se ha podido encontrar. Si la variable no está definida en el código, Visual FoxPro supondrá que la variable está pre-inicializada.

## Capítulo 13: Compilar una aplicación

Puede crear fácilmente aplicaciones orientadas a objetos controladas por eventos, de una en una. Este enfoque modular le permite comprobar la funcionalidad de cada componente a medida que lo crea. Cuando haya creado todos los componentes funcionales, podrá compilarlos en una única aplicación, que ensamblará los componentes ejecutables del proyecto (formularios, informes, menús, programas, etc.) en un único archivo que podrá distribuir a los usuarios junto con los datos.

Para crear rápidamente un proyecto completo con el [Marco de aplicaciones](#), puede usar el [Asistente para aplicaciones](#). Una vez creado el proyecto, el [Generador de aplicaciones](#) abrirá el nuevo proyecto de modo que pueda agregar una base de datos, tablas, informes y formularios.

En este capítulo se describe la forma de generar una aplicación típica de Visual FoxPro. Para obtener más información sobre el proceso de programación de aplicaciones de Visual FoxPro, consulte el capítulo 2, [Programar una aplicación](#), y el capítulo 14, [Probar y depurar aplicaciones](#). Si desea distribuir la aplicación, vea la parte 8, [Distribuir aplicaciones](#).

El proceso de generación de aplicaciones requiere lo siguiente:

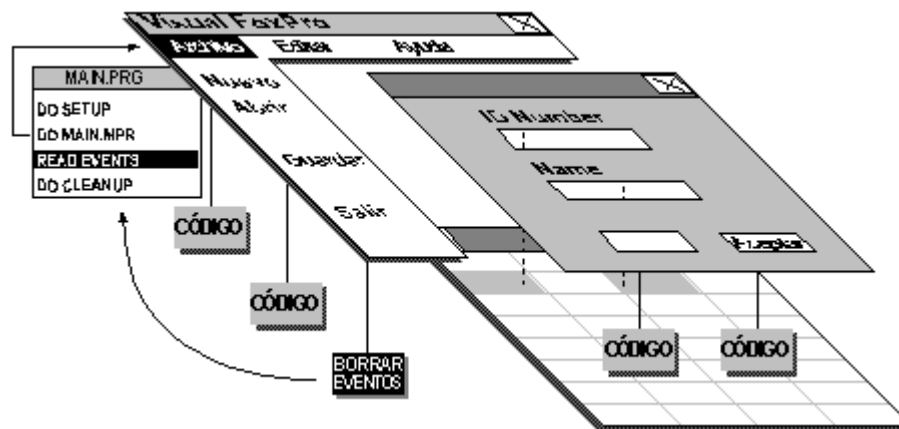
- [Estructurar una aplicación](#)
- [Agregar archivos a un proyecto](#)
- [Generar una aplicación a partir de un proyecto](#)

### Estructurar una aplicación

Una aplicación típica de base de datos consta de estructuras de datos, una interfaz de usuario, opciones de consulta y funciones de generación de informes. Para diseñar la estructura de su aplicación, considere detenidamente la función que ofrece cada componente y su relación con los demás componentes.

Una aplicación ensamblada de Visual FoxPro suele presentar al usuario un menú y uno o más formularios para introducir o mostrar datos. Para ofrecer determinada funcionalidad y mantener la integridad y seguridad de los datos, adjunte código a determinados eventos. Las consultas y los informes permiten que los usuarios extraigan información de la base de datos.

### Estructura de una aplicación típica de Visual FoxPro



A la hora de estructurar la aplicación, es necesario tener en cuenta las tareas siguientes:

- Establecer el punto de partida.
- Inicializar el entorno.
- Presentar la interfaz.
- Controlar el bucle de eventos.
- Restaurar el entorno original al terminar la aplicación.

En las secciones siguientes se proporcionan detalles sobre todas estas tareas. Normalmente podría crear un objeto Application para realizarlas; consulte la aplicación de ejemplo [Importadores Tasmanian](#) ubicada en el directorio ...\\Samples\\Vfp98\\Tastrade de Visual Studio para ver un ejemplo de esta técnica. Asimismo, si usa el [Asistente para aplicaciones](#) para compilar la aplicación, se creará un objeto Application. También puede, si lo desea, usar un programa como archivo principal que controle estas tareas. Para obtener más información, vea "[Estructurar un programa principal](#)".

### Establecer el punto de partida

Debe vincular todos los componentes y establecer un punto inicial para la aplicación con un [archivo principal](#). El archivo principal sirve como punto de partida para la ejecución de su aplicación y puede constar de un programa o formulario. Cuando los usuarios ejecuten la aplicación, Visual FoxPro inicia el archivo principal de la aplicación que, a su vez, ejecuta todos los demás componentes a medida que se vayan necesitando. Todas las aplicaciones deben tener un archivo principal. La mejor elección suele ser crear un programa principal en la aplicación. Sin embargo, puede combinar las

funciones del programa principal y la interfaz inicial de usuario si usa un formulario como programa principal.

Si usa un Asistente para aplicaciones para crear la aplicación, puede permitir que el asistente cree un programa de archivo principal de forma automática. No es necesario que especifique un archivo principal a menos que desee cambiarlo una vez finalizadas las acciones del asistente.

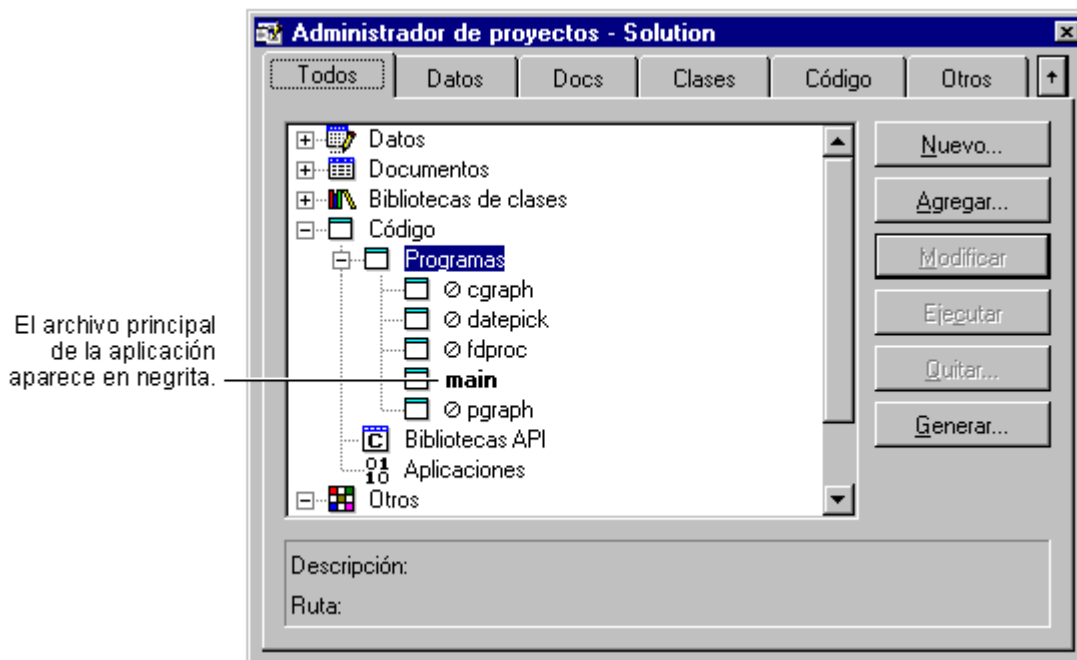
### Para establecer el punto de partida para una aplicación

1. En el [Administrador de proyectos](#), seleccione el archivo.
2. En el menú **Proyecto**, elija **Establecer principal**.

**Nota** El archivo que establezca como archivo principal de la aplicación se marca automáticamente como incluido, por lo que se considera como de sólo lectura después de compilar la aplicación.

Sólo se puede establecer como archivo principal un archivo del proyecto. El archivo principal se muestra en negrita, como se muestra en la ilustración siguiente.

### Establecer un archivo principal en el Administrador de proyectos



### Inicializar el entorno

Cuando haya creado un archivo principal, utilícelo para configurar el entorno de su aplicación. Puede que el entorno de desarrollo predeterminado de Visual FoxPro no sea el entorno más adecuado para la aplicación. El entorno predeterminado establece determinados valores para los comandos SET y las variables de sistema en el momento de abrir Visual FoxPro.

**Sugerencia** Para ver los valores predeterminados del entorno de desarrollo de Visual FoxPro, inicie

Visual FoxPro sin ningún archivo de configuración; para ello, escriba **VFP6 -C** y, a continuación, ejecute el comando [DISPLAY STATUS](#).

Siempre es conveniente guardar los valores iniciales del entorno y configurar un entorno específico para la aplicación en el preprograma.

### Para capturar comandos para el entorno actual

1. En el menú **Herramientas**, elija **Opciones**.
2. Presione **Mayúsculas** y elija **Aceptar** para mostrar los comandos SET del entorno en la ventana **Comandos**.
3. En la ventana **Comandos**, copie y pegue en el programa.

En un entorno específico de la aplicación, podría incluir código para:

- Inicializar variables.
- Establecer una ruta predeterminada.
- Abrir las bases de datos, las tablas libres y los índices necesarios. Si la aplicación requiere acceso a datos remotos, también se puede solicitar al usuario la información de inicio de sesión necesaria en la rutina de inicialización.
- Hacer referencia a archivos externos de biblioteca y procedimientos.

Por ejemplo, si desea comprobar el valor predeterminado del comando [SET TALK](#), almacenar este valor y establecer SET TALK como OFF para la aplicación, podría incluir el siguiente código en el procedimiento de configuración:

```
IF SET( 'TALK' ) = "ON"
    SET TALK OFF
    cTalkVal = "ON"
ELSE
    cTalkVal = "OFF"
ENDIF
```

Suele ser útil guardar los valores predeterminados en variables públicas, en una clase personalizada o como propiedades de un objeto Application para que pueda restaurar estos valores al salir de la aplicación.

```
SET TALK &cTalkVal
```

### Mostrar la interfaz inicial

La interfaz inicial de usuario puede ser un menú, un formulario o cualquier otro componente de usuario. Normalmente una aplicación mostrará una pantalla de inicio de sesión o un cuadro de diálogo de inicio antes de mostrar el menú o formulario inicial.

Para iniciar la interfaz de usuario en el programa principal puede usar un comando [DO](#) para ejecutar un menú o un comando [DO FORM](#) para ejecutar un formulario.

## Controlar el bucle de eventos

Cuando el entorno esté configurado y se muestre la interfaz de usuario inicial, podrá establecer un bucle de eventos para esperar la interacción del usuario.

### Para controlar el bucle de eventos

- Ejecute un comando [READ EVENTS](#), que hace que Visual FoxPro comience a procesar los eventos de usuario, como los clics del *mouse* y las pulsaciones de teclas.

Es importante situar correctamente el comando READ EVENTS en el archivo principal, porque todo el proceso de este archivo se suspende desde el momento en que se ejecuta el comando READ EVENTS hasta que se ejecuta un comando CLEAR EVENTS. Por ejemplo, podría ejecutar un comando READ EVENTS como el último comando de un procedimiento de inicialización, que se ejecutaría después de inicializar el entorno y mostrar la interfaz de usuario. Si no incluye el comando READ EVENTS, la aplicación volverá al sistema operativo después de la ejecución.

Una vez iniciado el bucle de eventos, la aplicación está bajo el control del último elemento mostrado de la interfaz de usuario. Por ejemplo, si se ejecutan los dos comandos siguientes en el archivo principal, la aplicación muestra el formulario Startup.scx:

```
DO FORM STARTUP.SCX  
READ EVENTS
```

Si no incluye en el archivo principal un comando READ EVENTS o su equivalente, la aplicación se ejecutará correctamente desde la ventana Comandos dentro del entorno de programación. Sin embargo, cuando se ejecuta desde el menú o la pantalla principal, la aplicación aparecerá durante unos instantes y después finalizará.

La aplicación también debe proporcionar una forma de terminar el bucle de eventos.

### Para terminar el bucle de eventos

- Ejecute un comando [CLEAR EVENTS](#).

Normalmente el comando CLEAR EVENTS se ejecuta desde un menú o botón de un formulario. El comando CLEAR EVENTS suspende el proceso del evento en Visual FoxPro y devuelve el control al programa que ejecutó el comando READ EVENTS e inició el bucle de eventos.

Para ver un sencillo ejemplo de programa, vea [Estructurar un programa principal](#) más adelante en este mismo capítulo.

**Precaución** Necesita establecer una forma de salir del bucle de eventos antes de iniciarlo. Asegúrese de que la interfaz tiene un mecanismo (como un botón "Salir" o un comando de menú Salir) para ejecutar el comando CLEAR EVENTS.

## Restaurar el entorno original

Para restaurar el valor original de las variables guardadas, puede sustituirlas mediante una macro en los comandos SET originales. Por ejemplo, si guardó la configuración de SET TALK en `cTalkVal`, ejecute el comando siguiente:

```
SET TALK &cTalkval
```

**Nota** Los nombres de variables utilizadas en la sustitución de macros no deben contener el prefijo "m." porque el punto presupone una concatenación de variables y producirá un error sintáctico.

Si inicializó el entorno en un programa que no sea aquél en el que se está realizando la restauración (por ejemplo, si inicializa llamando a un procedimiento, pero restaura llamando a otro) asegúrese de que puede tener acceso a los valores almacenados. Por ejemplo, almacene los valores para restaurarlos en variables públicas, clases personalizadas o como propiedades de un objeto Application.

## Estructurar un programa principal

Si usa un archivo de programa (.prg) como archivo principal en la aplicación, debe comprobar que incluya los comandos que gestionarán las tareas asociadas con las principales tareas de la aplicación. El archivo principal no tiene que ejecutar comandos directamente para realizar todas las tareas. Por ejemplo, es frecuente llamar a procedimientos o funciones para gestionar tareas como inicialización del entorno y limpieza.

**Nota** Si usó el [Asistente para aplicaciones](#) y le permitió crear el programa Main.prg, puede modificar el programa creado por el asistente en lugar de crear otro nuevo. Los asistentes utilizan una clase especial para definir un objeto para la aplicación. El programa principal incluye secciones para crear instancias y configurar el objeto.

### Para generar un único programa principal

1. Inicialice el entorno; para ello, abra bases de datos, declare variables, etc.
2. Establezca la interfaz inicial de usuario; para ello, llame a un menú o formulario.
3. Establezca el bucle de eventos; para ello, ejecute el comando [READ EVENTS](#).
4. Ejecute el comando [CLEAR EVENTS](#) desde un menú (como un comando Salir) o un botón (como un botón del comando Salir). El programa principal no debería ejecutar este comando.
5. Restablezca el entorno cuando salga de la aplicación.

Por ejemplo, su programa principal podría ser similar al siguiente:

Código	Comentarios
DO SETUP.PRG	Llama al programa para configurar el entorno (almacena los valores en variables públicas)
DO MAINMENU.MPR	Muestra la interfaz inicial de usuario
READ EVENTS	Establece el bucle de eventos. Un programa diferente (como Mainmenu.mpr) debe ejecutar el comando CLEAR EVENTS)
DO CLEANUP.PRG	Restaura el entorno antes de salir

## Agregar archivos a un proyecto

Un proyecto de Visual FoxPro consta de componentes independientes que se almacenan como archivos individuales. Por ejemplo, un proyecto sencillo puede constar de formularios (archivos .scx), informes (archivos .frx) y programas (archivos .prg y .fxp). Además un proyecto suele tener una o varias bases de datos (archivos .dbc), tablas (almacenadas en archivos .dbf y .fpt) e índices (archivos .cdx e .idx). Para incluirse en una aplicación, el archivo ha de agregarse al proyecto. De esa manera, al compilar la aplicación, Visual FoxPro puede incluir los archivos de ese componente en el producto terminado.

Puede agregar fácilmente archivos a un proyecto de varias maneras:

- Para crear un proyecto y agregar archivos existentes, use el Asistente para aplicaciones.
- Para agregar automáticamente archivos nuevos a un proyecto, abra un proyecto y, a continuación, cree los archivos nuevos en el Administrador de proyectos.
- Para agregar archivos existentes a un proyecto, abra un proyecto y agréguelos con el Administrador de proyectos.

Si ha utilizado el [Asistente para aplicaciones](#) o el Administrador de proyectos para crear los archivos, normalmente no necesitará hacer nada más ya que el archivo se incluye automáticamente en el proyecto. Sin embargo, existe una excepción a lo anterior si la aplicación incluye un archivo que el usuario vaya a modificar. Dado que los archivos incluidos son de sólo lectura, deberá marcar el archivo como excluido. Para obtener más detalles, vea el apartado [Referencias a archivos modificables](#) más adelante en este capítulo.

**Sugerencia** Para obtener una lista de los tipos de archivos y extensiones utilizados en Visual FoxPro, vea [Extensiones y tipos de archivos](#).

Si un archivo existente ya no forma parte del proyecto, puede agregarlo manualmente.

### Para agregar un archivo a un proyecto manualmente

1. En el [Administrador de proyectos](#), elija el tipo de componente que desea agregar; para ello,



selecciónelo en la jerarquía y, a continuación, haga clic en **Agregar**.

2. En el cuadro de diálogo **Abrir**, seleccione el archivo que desea agregar.

Visual FoxPro también agrega archivos al proyecto si se ha hecho referencia a los mismos en un programa o formulario. Por ejemplo, si un programa del proyecto incluye la línea siguiente, Visual FoxPro agrega el archivo Orders.scx al proyecto:

```
DO FORM ORDERS.SCX
```

Si se ha hecho referencia a un archivo de esta manera, no se incluye inmediatamente en un proyecto. Posteriormente, cuando compile el proyecto, Visual FoxPro resuelve las referencias a todos los archivos e incluye automáticamente los archivos implícitos en el proyecto. Además, si se ha hecho referencia a cualquier otro archivo mediante código definido por el usuario en el archivo nuevo, al compilar el proyecto también se resolverá esa referencia y se incluirá el archivo. Los archivos a los que se ha hecho referencia aparecerán en el Administrador de programas la próxima vez que vea el proyecto.

**Importante** Visual FoxPro podría no ser capaz de resolver referencias a archivos de imagen (.bmp y .msk), dependiendo de la forma en que se utilicen en el código. Por tanto, agregue las imágenes a los archivos manualmente. Además, Visual FoxPro no puede incluir automáticamente archivos a los que se haya hecho referencia mediante la sustitución de macros, porque el nombre del archivo no se conoce hasta que se ejecuta la aplicación. Si la aplicación establece referencias a archivos mediante la sustitución de macros, incluya estos archivos manualmente.

## Referencias a archivos modificables

Cuando compile un proyecto en una aplicación, los archivos incluidos en el proyecto se ensamblan en un único archivo de aplicación. Una vez compilado el proyecto, los archivos del proyecto que estén marcados como "incluidos" pasarán a ser de sólo lectura.

Los archivos que forman parte del proyecto, como las tablas, suelen estar diseñados para que los usuarios puedan modificarlos. En esos casos, debería agregar los archivos al proyecto pero marcándolos como excluidos. Los archivos excluidos siguen formando parte de la aplicación por lo que Visual FoxPro los registra como parte del proyecto, pero no se compilan en el archivo de aplicación y, de este modo, los usuarios podrán actualizarlos.

**Nota** Como opción predeterminada las tablas se marcan como excluidas porque Visual FoxPro supone que las tablas serán modificables en una aplicación.

Como regla general, los archivos que contengan programas ejecutables (formularios, informes, consultas, menús y programas) deberán incluirse en el archivo de aplicación y los archivos de datos deberán excluirse. Sin embargo, ha de determinar si va a incluir o excluir archivos según los requisitos de la aplicación. Por ejemplo, una tabla que contenga información sobre el sistema o datos utilizados únicamente para consulta puede incluirse en el archivo de aplicación para proteger dicha información ante cambios que se realicen sin previo aviso. A la inversa, podría excluir un archivo de informe (.frx) si la aplicación permite a los usuarios modificarlo.

Si excluye un archivo, deberá asegurarse de que Visual FoxPro pueda encontrar el archivo excluido

cuando se ejecute la aplicación. Por ejemplo, cuando en un formulario se hace referencia a una biblioteca de clases visuales, el formulario almacena una ruta de acceso relativa a esa biblioteca. Si incluye la biblioteca en el proyecto, entrará a formar parte del archivo de aplicación y el formulario siempre podrá localizarla. Sin embargo, si excluye la biblioteca, el formulario debe buscarla usando la ruta de acceso relativa o la ruta de acceso de búsqueda de Visual FoxPro (definida mediante el comando [SET PATH](#)). Si la biblioteca no se encuentra en las ubicaciones previstas (por ejemplo, si se ha cambiado de ubicación desde que se creó el formulario) Visual FoxPro muestra un cuadro de diálogo en el que se solicita al usuario que localice la biblioteca. Tal vez desee que los usuarios no vean este cuadro de diálogo. Para estar seguro, incluya todos los archivos que no necesiten actualización por parte de los usuarios.

**Nota** No puede incluir archivos de aplicación (.app) y debe optar por excluir los archivos de biblioteca (.ocx, .flt y .dll).

### Para excluir archivos modificables

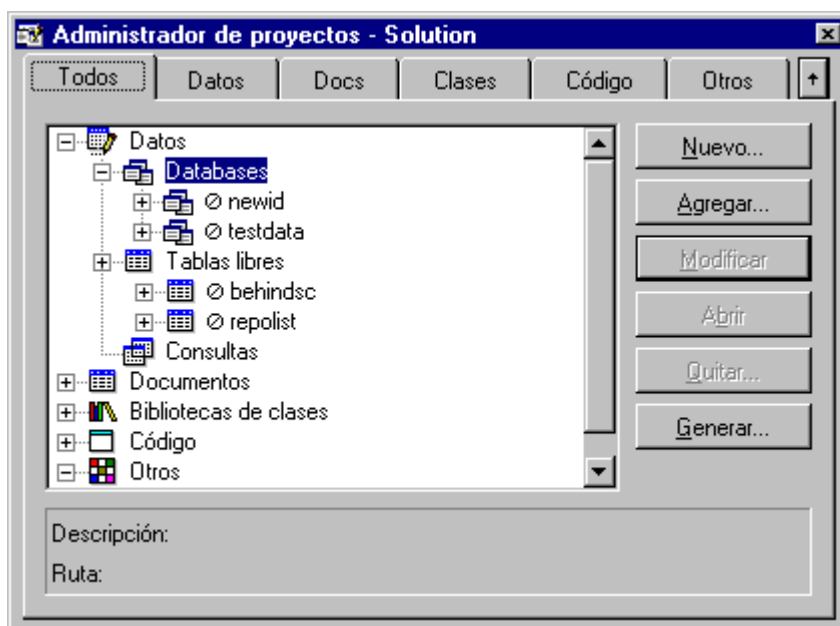
1. Seleccione el archivo modificable en el [Administrador de proyectos](#).
2. En el menú **Proyecto**, elija **Excluir**.

Si el archivo ya está excluido, el comando **Excluir** no está disponible; aparecerá el comando **Incluir** en su lugar.

Los archivos excluidos tienen el símbolo  $\emptyset$  a la izquierda de sus nombres.

**Nota** Los archivos marcados como archivos principales no se pueden marcar como excluidos. Para obtener detalles sobre los archivos principales, vea el apartado [Establecer el punto de partida](#) de este capítulo.

### Tablas marcadas como excluidas en un proyecto



**Sugerencia** Para ver simultáneamente todos los archivos de proyecto, elija **Información de proyecto** en el menú **Proyecto** y seleccione la ficha **Archivos**.

## Generar una aplicación a partir de un proyecto

Si puede generar el proyecto sin errores, podrá generar la aplicación. El resultado final de este proceso es un único archivo que incluye todos los archivos a los que se ha hecho referencia en el proyecto en un único archivo de aplicación (salvo los marcados como excluidos). Puede distribuir el archivo de aplicación, junto con los archivos de datos y con todos los demás archivos excluidos del proyecto a los usuarios y éstos pueden iniciar el archivo para ejecutar la aplicación.

Los pasos necesarios para crear una aplicación desde el proyecto son:

- Probar el proyecto.
- Generar un archivo de aplicación a partir del proyecto.

### Probar el proyecto

Para comprobar las referencias y que todos los componentes están disponibles, puede probar el proyecto. Para ello, deberá volver a generar el proyecto. Visual FoxPro tendrá que resolver las referencias a archivos y volver a compilar los archivos no actualizados.

### Para probar un proyecto

1. En el [Administrador de proyectos](#), elija **Generar**.
2. En el cuadro de diálogo [Opciones para generar](#) elija **Volver a generar el proyecto**.
3. Seleccione cualquier otra opción que necesite y elija **Aceptar**.

–O bien–

- Utilice el comando [BUILD PROJECT](#).

Por ejemplo, para generar una aplicación denominada Miproj.pjx, escriba:

```
BUILD PROJECT miproj
```

Si se producen errores durante el proceso de generación, se incluyen en un archivo que recibe el nombre del proyecto y la extensión .err y se localiza en el directorio actual. El recuento de errores de compilación se muestra en la barra de estado. También puede ver el archivo de errores.

### Para mostrar el archivo de errores

- Seleccione la casilla de verificación **Mostrar errores**.

Una vez generado el proyecto correctamente, deberá intentar ejecutarlo antes de crear una aplicación.

### Para ejecutar una aplicación

- En el **Administrador de proyectos**, resalte el programa principal y, a continuación, elija **Ejecutar**.

–O bien–

- En la ventana **Comandos**, ejecute un comando DO con el nombre del programa principal:

DO MAINPROG.PRG

Si el programa se ejecuta correctamente, estará preparado para generar un archivo de aplicación que contenga todos los archivos incluidos en el proyecto.

Debería repetir los pasos de regeneración y ejecución del proyecto siempre que agregue componentes al proyecto. A menos que elija Volver a compilar todos los archivos en el cuadro de diálogo Opciones para generar, sólo se volverán a compilar los archivos que se hayan modificado desde la última generación.

### Generar una aplicación desde un proyecto

Para generar un archivo terminado desde la aplicación, génerele en un archivo de aplicación. Los archivos de aplicación tienen la extensión .app. Para ejecutar la aplicación, los usuarios deben iniciar Visual FoxPro primero y después cargar el archivo .app.

Puede optar por generar un archivo de aplicación (.app) o un archivo ejecutable (.exe) desde el proyecto. Los usuarios pueden ejecutar un archivo .app si ya tienen una copia de Visual FoxPro. Además, puede crear un archivo .exe, que funciona con dos bibliotecas de vínculos dinámicos de Visual FoxPro (Vfp6r.dll y Vfp6enu.dll), las cuales se distribuyen con la aplicación para ofrecer un entorno de ejecución completo para Visual FoxPro. El segundo archivo es específico del país de destino de la aplicación. Para obtener más información, vea la parte 8, [Distribuir aplicaciones](#).

### Para generar una aplicación

1. En el **Administrador de proyectos**, elija **Generar**.
2. En el cuadro de diálogo [Opciones para generar](#), elija **Generar aplicación** si desea generar un archivo .app o **Generar ejecutable** si desea generar un archivo .exe.
3. Seleccione cualquier otra opción que necesite y elija **Aceptar**.

–O bien–

- Utilice los comandos [BUILD APP](#) o [BUILD EXE](#).

Por ejemplo, para generar una aplicación denominada Miapli.app a partir de un proyecto denominado Miproj.pjx, escriba:

```
BUILD APP miapli FROM miproy
```

Para crear una aplicación denominada Miapli.exe a partir de un proyecto llamado Miproj.pjx, escriba:

```
BUILD EXE miapli FROM miproy
```

**Nota** También puede usar el cuadro de diálogo Opciones de generación para crear un servidor de Automatización personalizado desde la aplicación de Visual FoxPro. Para obtener más información, vea [Crear servidores de Automatización](#) en el capítulo 16, "Agregar OLE".

Una vez creado y terminado un archivo de aplicación para el proyecto, todos los usuarios podrán ejecutarlo.

### Para ejecutar una aplicación como un archivo .app

- En Visual FoxPro, elija **Ejecutar** en el menú **Programa** y seleccione el archivo de aplicación.

–O bien–

- En la ventana **Comandos**, escriba [DO](#) y el nombre del archivo de aplicación.

Por ejemplo, para ejecutar una aplicación denominada MIAPLI, escriba:

```
DO miapli.app
```

Si ha creado un archivo .exe desde la aplicación, los usuarios podrán ejecutarlo de diversas maneras.

### Para ejecutar una aplicación como un archivo .exe

- En Visual FoxPro, elija **Ejecutar** en el menú **Programa** y seleccione el archivo de aplicación o, en la ventana **Comandos**, escriba [DO](#) y el nombre del archivo de aplicación.

Por ejemplo, para ejecutar un archivo .exe denominada Miapli.exe, escriba:

```
DO miapli.exe
```

–O bien–

- En Windows, haga doble clic en el icono del archivo .exe.

**Nota** Puede usar el [Asistente para instalación](#) para crear una rutina de instalación que instale los archivos apropiados.

## Capítulo 14: Probar y depurar aplicaciones

El proceso de prueba implica la búsqueda de problemas en el código, mientras que la depuración

consiste en aislar y resolver los problemas. La prueba y la depuración son partes inevitables del ciclo de desarrollo, que es mejor incorporar en una etapa inicial. La prueba y depuración de componentes individuales simplifica de forma significativa el proceso análogo de aplicaciones integradas.

Para obtener más información sobre la creación de una aplicación, consulte el capítulo 2, [Programar una aplicación](#) y el capítulo 13, [Compilar una aplicación](#).

Este capítulo trata los temas siguientes:

- [Preparar la prueba y depuración](#)
- [Depurar antes de que se produzcan errores](#)
- [Aislar los problemas](#)
- [Mostrar los resultados](#)
- [Registrar el trayecto del código](#)
- [Controlar errores de tiempo de ejecución](#)

## Preparar la prueba y depuración

Los programadores suelen buscar diferentes niveles de solidez cuando comprueban y depuran sus aplicaciones:

1. Ejecución sin bloqueos ni generación de mensajes de error.
2. Acciones apropiadas en situaciones frecuentes.
3. Acciones razonables o mensajes de error en diversas situaciones.
4. Recuperación inmediata ante interacciones imprevistas del usuario.

Visual FoxPro ofrece un variado conjunto de herramientas para facilitar la tarea de aislar e identificar los problemas del código para que puedan resolverse de forma eficaz. Sin embargo, una de las mejores maneras de crear aplicaciones robustas es buscar los posibles problemas antes de que se produzcan.

## Depurar antes de que se produzcan errores

La teoría ha demostrado que las buenas costumbres de escritura de código (uso de espacios en blanco, inclusión de comentarios, adhesión a las convenciones de nombres, etc.) tienden a reducir automáticamente el número de errores del código. Además, hay algunos pasos que se pueden realizar en el proceso de programación para que las operaciones de prueba y depuración resulten más fáciles posteriormente; entre ellos se incluyen:

- Crear un entorno de prueba
- Establecer aserciones
- Mostrar las secuencias de eventos

### Crear un entorno de prueba

El entorno de sistema en el que piensa ejecutar una aplicación tiene la misma importancia que el entorno de datos que ha instalado para la propia aplicación. Para garantizar la portabilidad y crear un contexto apropiado para las pruebas y la depuración, debe tener en cuenta lo siguiente:

- El hardware y el software
- Las rutas del sistema y las propiedades de archivo
- La estructura de directorios y las ubicaciones de los archivos

### **Hardware y software**

Para obtener la máxima portabilidad, debe programar las aplicaciones en la plataforma común de menor nivel en la que va a ejecutarlas. Para establecer una plataforma de base:

- Programe las aplicaciones utilizando el modo de vídeo común más bajo.
- Determine los requisitos básicos para la RAM y el espacio de almacenamiento de medios, incluyendo los controladores necesarios o el software que se ejecute de forma simultánea.
- Tenga en cuenta casos especiales de memoria, archivo y bloqueo de registros para las versiones en red y autónoma de las aplicaciones.

### **Rutas del sistema y propiedades de archivos**

Con el fin de garantizar que todos los archivos de programa necesarios son fácilmente accesibles en todos los equipos en los que se ejecuta la aplicación, también puede necesitar una configuración de archivo de base. Para definir una línea de base para la configuración, responda a las siguientes preguntas:

- ¿Necesita su aplicación rutas comunes de sistema?
- ¿Ha establecido propiedades adecuadas de acceso a archivos?
- ¿Se han establecido correctamente permisos de red para cada usuario?

### **Estructura de directorios y ubicaciones de archivos**

Si el código fuente hace referencia a rutas o a nombres de archivos absolutos, es obligatorio que tales rutas y archivos existan en el momento de instalar la aplicación en cualquier otro PC. Para evitar este caso puede seguir uno de estos procedimientos:

- Para obtener información adicional sobre el uso de los archivos de configuración, consulte el capítulo 3, [Configurar Visual FoxPro](#), en la *Guía de instalación*.
- Cree un directorio o una estructura de directorios independiente para mantener los archivos de origen apartados de los archivos de aplicación generados. De esta forma, puede comprobar las referencias de la aplicación terminada y saber exactamente los archivos que necesita distribuir.
- Use rutas de acceso relativas.

### **Establecer aserciones**

Puede incluir aserciones en el código para comprobar las hipótesis que tiene sobre el entorno de ejecución del código.

### Para establecer una aserción

- Use el comando [ASSERT](#) para identificar las hipótesis del programa.

Cuando la condición estipulada en el comando ASSERT se evalúa como falsa (.F.), aparece un cuadro de mensajes de aserciones y se repite en la ventana **Resultados del depurador**.

Por ejemplo, podría escribir una función que espera un valor del parámetro distinto de cero. La línea de código siguiente en la función le avisa si el valor del parámetro es 0:

```
ASSERT nParm != 0 MESSAGE "Recibido el parámetro 0"
```

Puede especificar si los mensajes de aserciones se van a mostrar mediante el comando [SET ASSERTS](#). Como valor predeterminado, los mensajes de aserciones no se muestran.

### Mostrar las secuencias de eventos

Cuando vea que se producen eventos en relación con otros eventos, puede determinar la ubicación más eficaz para incluir el código.

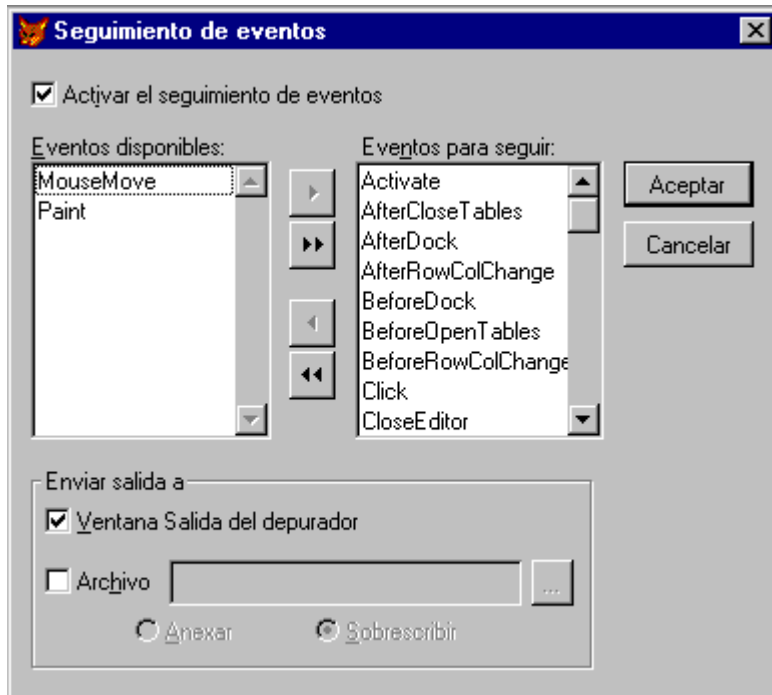
### Para hacer un seguimiento de eventos

- En el menú **Herramientas** de la ventana [Depurador](#), elija **Seguimiento de eventos**.  
–O bien–
- Use el comando [SET EVENTTRACKING](#).

El cuadro de diálogo **Seguimiento de eventos** permite seleccionar los eventos que desee ver.

### Cuadro de diálogo Seguimiento de eventos





**Nota** En este ejemplo, los eventos MouseMove y Paint se han eliminado de la lista Eventos para seguir, porque estos eventos se producen con tanta frecuencia que dificultan la visión de las secuencias de los demás eventos.

Cuando el seguimiento de eventos está activado, cada vez que se produzca un evento de sistema en la lista Eventos para seguir, el nombre del evento aparecerá en la ventana [Resultados del depurador](#) o se escribirá en un archivo. Si opta por mostrar los eventos en la ventana **Resultados del depurador**, también podrá guardarlos en un archivo según se describe en [Mostrar los resultados](#) más adelante en este mismo capítulo.

**Nota** Si la ventana Resultados del depurador no está abierta, los eventos no se presentarán en la lista aun cuando esté activado el cuadro Ventana Resultados del depurador.

## Aislar los problemas

Una vez identificados los problemas mediante la prueba, puede usar el entorno de depuración de Visual FoxPro para aislarlos con los pasos siguientes:

- Iniciar una sesión de depuración
- Seguimiento del código
- Suspensión de la ejecución del programa
- Mostrar los valores almacenados
- Mostrar los resultados

### Iniciar una sesión de depuración

Para iniciar una sesión de depuración, abra el entorno de depuración.

## Para abrir el depurador

- En el menú **Herramientas**, elija **Depurador**.

**Nota** Si va a realizar la depuración en el entorno de Visual FoxPro, elija la herramienta de depuración que desee abrir en el menú Herramientas.

También puede abrir el depurador con cualquiera de los comandos siguientes:

[DEBUG](#)

[SET STEP ON](#)

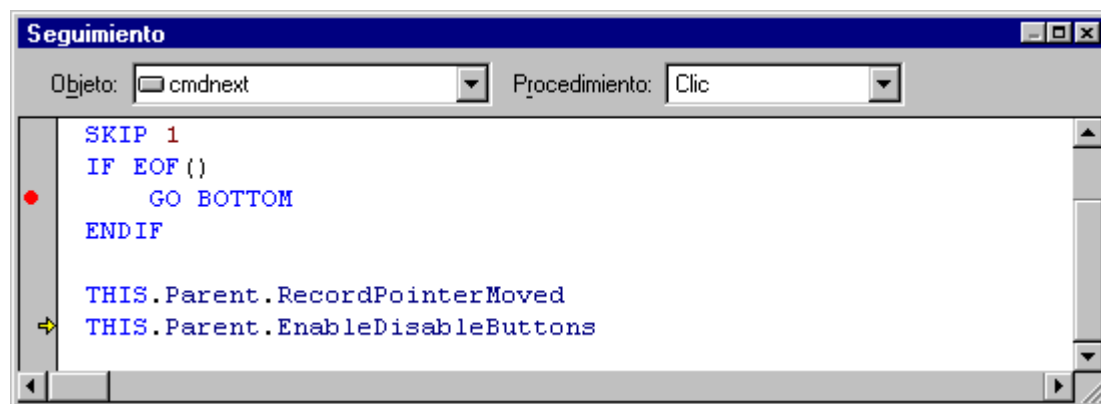
[SET ECHO ON](#)

El depurador se abre automáticamente siempre que se encuentra una condición de punto de interrupción.

## Seguimiento del código

Una de las estrategias de depuración más útiles que tiene a su disposición es la posibilidad de realizar un seguimiento del código, ver cada línea de código según se ejecute y comprobar los valores de las variables, propiedades y la configuración del entorno.

### Código en la ventana Seguimiento



### Para hacer el seguimiento del código

1. Inicie una sesión de depuración.
2. Si no hay ningún programa abierto en la ventana [Seguimiento](#), elija **Ejecutar** en el menú **Depurar**.
3. Elija **Paso a paso por instrucciones** en el menú **Depurar** o haga clic en el botón **Paso a paso** de la barra de herramientas.

Una flecha en el área gris situada a la izquierda del código indica la siguiente línea que se va a ejecutar.

**Sugerencia** Puede aplicar las sugerencias siguientes:

- Establezca puntos de interrupción para reducir el intervalo de código por el que necesita pasar.
- Puede ignorar una línea de código que sepa que va a generar un error colocando el cursor sobre la línea de código después de la línea problemática y eligiendo **Configurar siguiente instrucción** en el menú **Depurar**.
- Si tiene mucho código asociado con eventos Timer, puede evitar el seguimiento de este código desactivando **Mostrar eventos Timer** en la ficha **Depurar** del cuadro de diálogo **Opciones**.

Si aísla un problema cuando esté depurando código de un programa u objeto, podrá repararlo inmediatamente.

### Para reparar los problemas encontrados durante el seguimiento del código

- En el menú **Depurar**, elija **Reparar**.

Cuando elija **Reparar** en el menú **Depurar**, la ejecución del programa se cancela y el editor de código se abre donde esté situado el cursor en la ventana **Seguimiento**.

## Suspensión de la ejecución del programa

Los puntos de interrupción permiten suspender la ejecución del programa. Una vez suspendida, puede comprobar los valores de las variables y propiedades, ver la configuración del entorno y examinar secciones de código línea a línea sin tener que recorrer todo el código.

**Sugerencia** También puede suspender la ejecución de un programa que esté en ejecución si presiona la tecla ESC en la ventana **Seguimiento**.

### Suspender la ejecución en una línea de código

Puede establecer puntos de interrupción en el código para suspender la ejecución del programa de diferentes maneras. Si conoce el punto exacto en el que desea suspender la ejecución del programa, puede establecer un punto de interrupción directamente en esa línea de código.

### Para definir un punto de interrupción en una línea de código determinada

En la ventana [Seguimiento](#), busque la línea de código en la que desee establecer el punto de interrupción y realice una de las operaciones siguientes:

1. Sitúe el cursor en la línea de código.
2. Presione F9 o haga clic en el botón **Alternar punto de interrupción** de la barra de

herramientas **Depurador**.

–O bien–

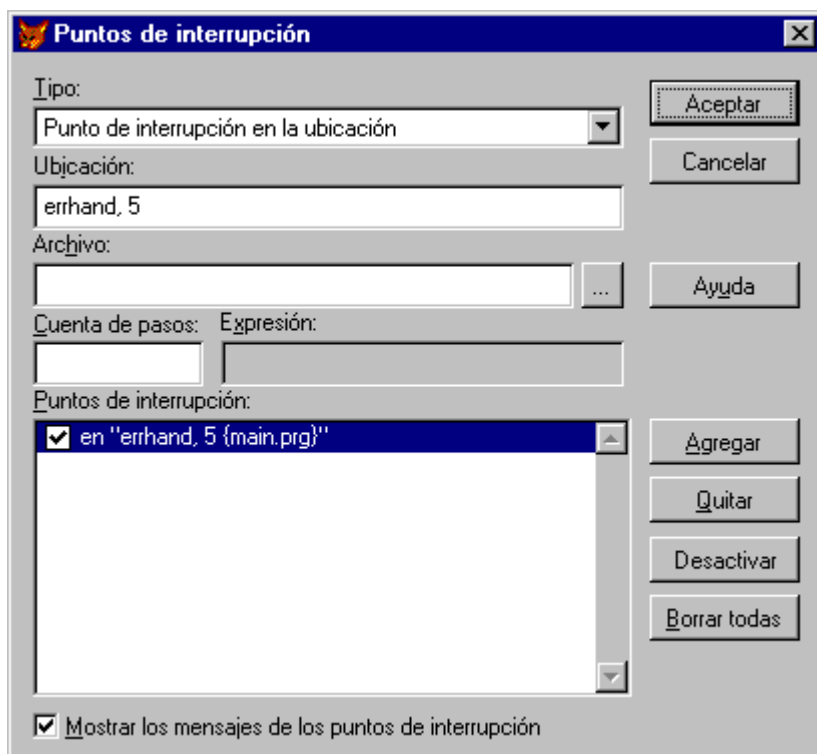
- Haga doble clic en el área gris situada a la izquierda de la línea de código.

Aparecerá un punto sólido en el área gris de la izquierda de la línea de código para indicar que se ha establecido un punto de interrupción en esa línea.

**Sugerencia** Si va a depurar objetos, puede localizar líneas de código determinadas en la ventana **Seguimiento**; para ello, elija el objeto en la lista Objeto y el método o evento en la lista Procedimiento.

También puede establecer puntos de interrupción si especifica ubicaciones y archivos en el cuadro de diálogo **Puntos de interrupción**.

### Punto de interrupción en una ubicación



### Ejemplos de ubicaciones y archivos de puntos de interrupción

Ubicación	Archivo	Dónde se suspende la ejecución
ErrHandler	C:\Myapp\Main.prg	La primera línea ejecutable de un procedimiento denominado ErrHandler en Main.prg.
Main,10	C:\Myapp\Main.prg	La décima línea del programa

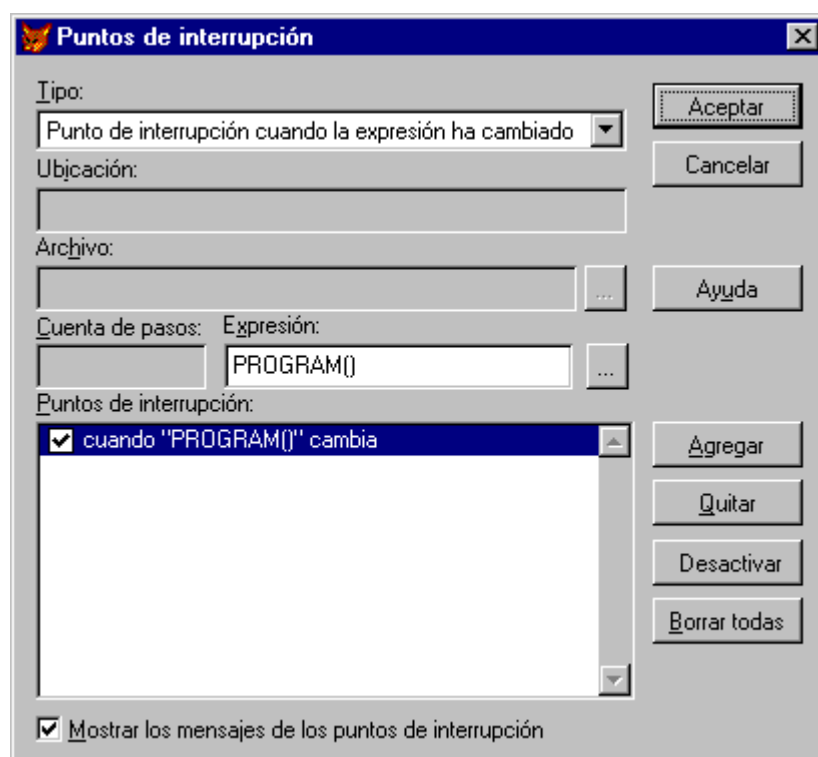
denominado `Main`.

Click	C:\Myapp\Form.scx	La primera línea ejecutable de cualquier procedimiento, función, método o evento denominado <code>Click</code> en <code>Form.scx</code> .
<code>cmdNext.Click</code>	C:\Myapp\Form.scx	La primera línea ejecutable asociada al evento <code>Click</code> de <code>cmdNext</code> en <code>Form.scx</code> .
<code>cmdNext::Click</code>		La primera línea ejecutable del evento <code>Click</code> de cualquier control cuya <code>ParentClass</code> sea <code>cmdNext</code> en cualquier archivo.

## Suspender la ejecución cuando los valores cambian

Si desea saber cuándo cambia el valor de una variable o una propiedad, o cuándo cambia una condición del tiempo de ejecución, puede establecer un punto de interrupción en una expresión.

### Punto de interrupción cuando la expresión ha cambiado



### Para suspender la ejecución del programa cuando ha cambiado el valor de una expresión

1. En el menú **Herramientas** de la ventana [Depurador](#), elija **Puntos de interrupción** para abrir el cuadro de diálogo [Puntos de interrupción](#).
2. En la lista **Tipo**, elija **Punto de interrupción cuando la expresión ha cambiado**.

3. Escriba la expresión en el cuadro **Expresión**.

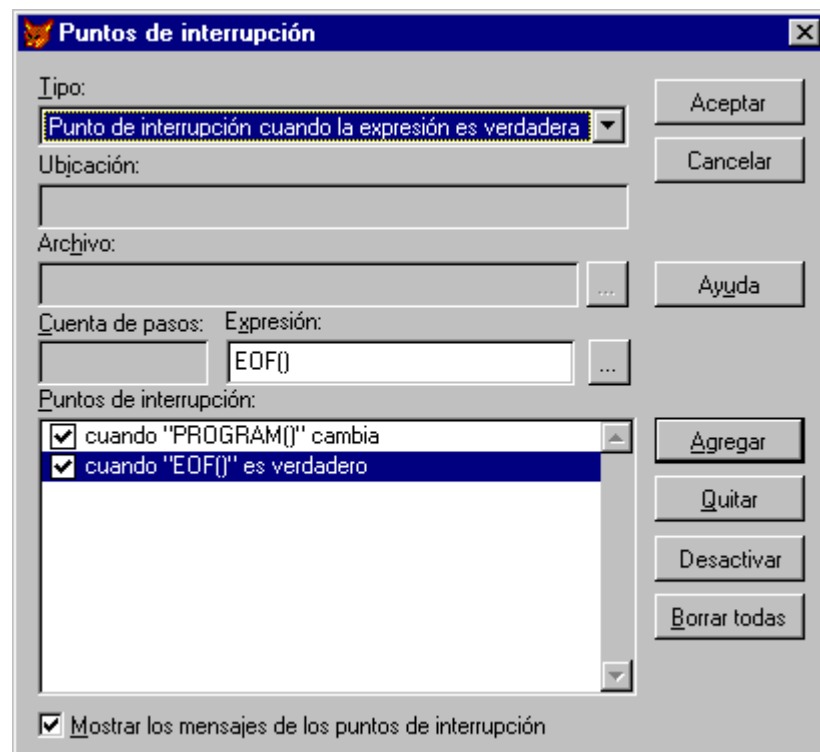
### Ejemplos de expresiones de puntos de interrupción

Expresión	Uso
RECNO( )	Suspender la ejecución cuando el puntero de registro se mueve en la tabla.
PROGRAM( )	Suspender la ejecución en la primera línea de cualquier programa, procedimiento, método o evento nuevo.
myform.Text1.Value	Suspender la ejecución siempre que se cambie el valor de esta propiedad de forma interactiva o mediante programación.

### Suspender de forma condicional la ejecución

A menudo deseará suspender la ejecución de un programa no en una línea determinada, sino cuando una determinada condición sea verdadera.

### Punto de interrupción en una expresión



**Para suspender la ejecución del programa cuando una expresión se evalúe como verdadera**

1. En el menú **Herramientas** de la ventana [Depurador](#), elija **Puntos de interrupción** para abrir el

cuadro de diálogo [Puntos de interrupción](#).

2. En la lista **Tipo**, elija **Punto de interrupción cuando la expresión es verdadera**.
3. Escriba la expresión en el cuadro **Expresión**.
4. Elija **Agregar** para agregar el punto de interrupción a la lista **Puntos de interrupción**.


### Ejemplos de expresiones de puntos de interrupción

Expresión	Uso
EOF( )	Suspender la ejecución cuando el puntero de registro se ha desplazado detrás del último registro de una tabla.
'CLICK' \$PROGRAM( )	Suspender la ejecución de la primera línea de código asociado a un evento Click o DblClick.
nReturnValue = 6	Si el valor devuelto de un cuadro de mensajes se almacena en nReturnValue, se suspende la ejecución cuando un usuario elige <b>SÍ</b> en el cuadro de mensajes.

### Suspender de forma condicional la ejecución en una línea de código

Puede especificar que la ejecución del programa se suspenda en una línea determinada sólo cuando una condición concreta sea verdadera.

### Punto de interrupción cuando la expresión es verdadera



**Puntos de interrupción**

Tipo:

Ubicación:

Archivo:

Cuenta de pasos:  Expresión:

Puntos de interrupción:

- ☒ en "setpath, 4 {main.prg}" cuando "oldebugmode=.T."

☒ Mostrar los mensajes de los puntos de interrupción

Botones: Aceptar, Cancelar, Ayuda, Agregar, Quitar, Desactivar, Borrar todas

**Para suspender la ejecución del programa en una línea determinada cuando la expresión se evalúa como verdadera**

1. En el menú **Herramientas** de la ventana [Depurador](#), elija **Puntos de interrupción** para abrir el cuadro de diálogo [Puntos de interrupción](#).
2. En la lista **Tipo**, elija **Punto de interrupción cuando la expresión es verdadera**.
3. Escriba la ubicación en el cuadro **Ubicación**.
4. Escriba la expresión en el cuadro **Expresión**.
5. Elija **Agregar** para agregar el punto de interrupción a la lista **Puntos de interrupción**.
6. Elija **Aceptar**.

**Sugerencia** Suele ser más fácil buscar la línea de código en la ventana Seguimiento, establecer un punto de interrupción y modificarlo en el cuadro de diálogo Punto de interrupción. Para ello, cambie el **Tipo** de **Punto de interrupción en la ubicación** a **Punto de interrupción cuando la expresión es verdadera** y, a continuación, agregue la expresión.

**Quitar puntos de interrupción**

Puede desactivar los puntos de interrupción sin quitarlos en el cuadro de diálogo **Puntos de interrupción**. Puede eliminar puntos de interrupción del tipo "punto de interrupción en la ubicación" en la ventana **Seguimiento**.

**Para quitar un punto de interrupción de una línea de código**

En la ventana [Seguimiento](#), busque el punto de interrupción y realice una de las operaciones siguientes:

- Sitúe el cursor en la línea de código y elija **Alternar punto de interrupción** en la barra de herramientas **Depurador**.

–O bien–

- Haga doble clic en el área gris situada a la izquierda de la línea de código.

**Mostrar los valores almacenados**

En la ventana Depurador, puede ver fácilmente los valores de tiempo de ejecución de las variables, elementos de matriz, propiedades y expresiones en las ventanas siguientes:

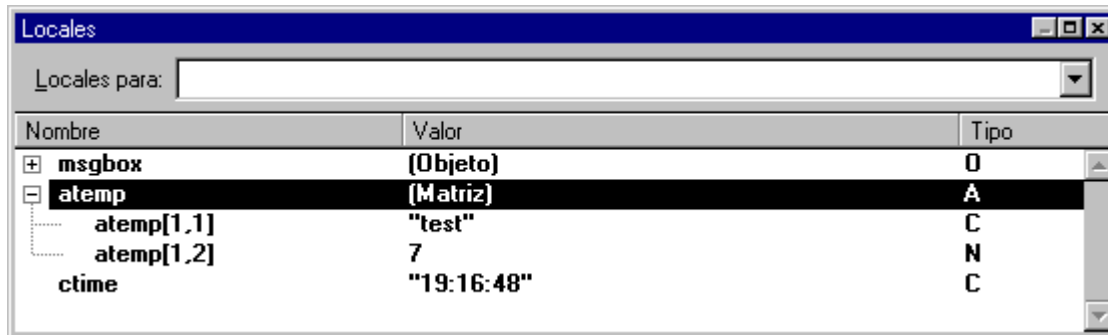
- Ventana Locales
- Ventana Inspección
- Ventana Seguimiento



## Mostrar los valores almacenados en la ventana Locales

La ventana Locales muestra todas las variables, matrices, objetos y miembros de objetos que están visibles en cualquier programa, procedimiento o método de la pila de llamadas. Como opción predeterminada, los valores del programa que se está ejecutando actualmente se muestran en la ventana Locales. Puede ver estos valores para otros programas o procedimientos de la pila de llamadas si elige los programas o procedimientos en la lista Locales para.

### Ventana Locales



Nombre	Valor	Tipo
msgbox	[Objeto]	O
atemp	[Matriz]	A
atemp[1,1]	"test"	C
atemp[1,2]	7	N
ctime	"19:16:48"	C

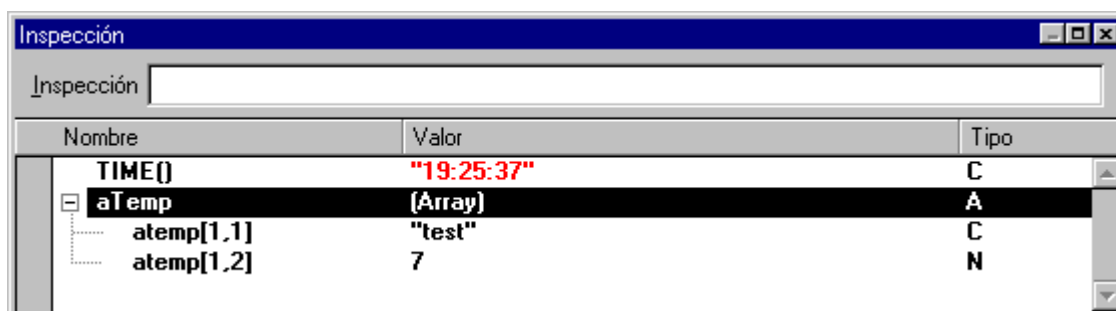
Puede efectuar un análisis descendente de matrices u objetos si hace clic en el signo más (+) situado junto al nombre de la matriz u objeto en las ventanas Locales e Inspección. Cuando efectúe el análisis descendente, podrá ver los valores de todos los elementos de las matrices y las configuraciones de propiedades de los objetos.

Puede incluso cambiar los valores de las variables, elementos de matrices y propiedades en las ventanas Locales e Inspección seleccionando la variable, elemento de matriz o propiedad si hace clic en la columna Valor y escribe un valor nuevo.

## Mostrar los valores almacenados en la ventana Inspección

En el cuadro Inspección de la ventana Inspección, escriba cualquier expresión válida de Visual FoxPro y presione ENTRAR. El valor y el tipo de expresión aparecen en la lista de la ventana Inspección.

### Ventana Inspección



Nombre	Valor	Tipo
TIME()	"19:25:37"	C
aTemp	[Array]	A
atemp[1,1]	"test"	C
atemp[1,2]	7	N

**Nota** En la ventana Inspección no se pueden introducir expresiones que creen objetos.

También puede seleccionar variables o expresiones en la ventana Seguimiento o en otras ventanas del Depurador y arrastrarlas hasta la ventana Inspección.

Los valores que se hayan modificado aparecen en rojo en la ventana Inspección.

### Para quitar un elemento de la lista de la ventana Inspección

Seleccione el elemento y elija una de las opciones siguientes:

- Presione SUPR.
- O bien–
- En el menú contextual, elija **Eliminar inspección**.

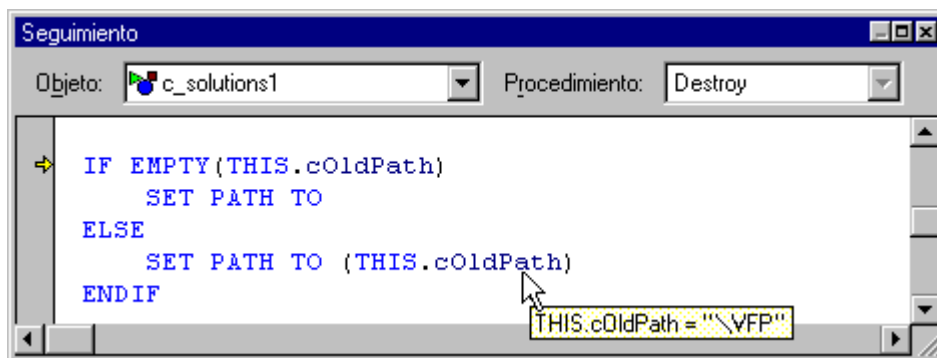
### Para modificar una inspección

- Haga doble clic en la inspección en la ventana **Inspección** y modifíquela.

### Mostrar los resultados en la ventana Seguimiento

Sitúe el cursor sobre cualquier variable, elemento de matriz o propiedad de la ventana Seguimiento para mostrar su valor actual en una sugerencia del valor.

### Sugerencia del valor en la ventana Seguimiento



## Mostrar los resultados

El comando [DEBUGOUT](#) permite escribir valores de la ventana Resultados del depurador en un registro de archivo de texto. Además puede usar el comando [SET DEBUGOUT TO](#) o la ficha [Depurar](#) del cuadro de diálogo Opciones.

Si no está escribiendo comandos DEBUGOUT en un archivo de texto, la ventana [Resultados del depurador](#) debe estar abierta para que pueda escribir los valores de DEBUGOUT. La siguiente línea de código se imprime en la ventana Resultados del depurador en el momento en que se ejecuta la línea de código:

```
DEBUGOUT DATETIME( )
```

Además, puede activar el seguimiento de eventos, que se describe en apartados anteriores de este capítulo y optar por hacer que el nombre y los parámetros de cada evento que se produzca se muestren en la ventana Resultados del depurador.

## Registrar el trayecto del código

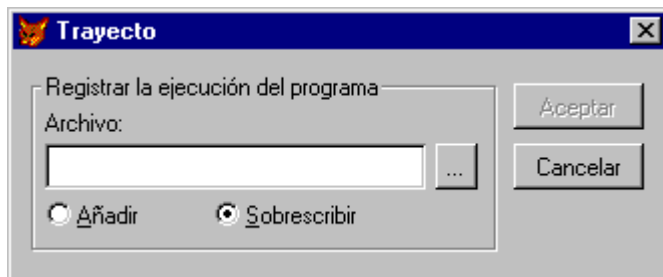
En un proceso de programación posterior, podría perfeccionar el código para obtener más rendimiento y garantizar que lo ha probado correctamente mediante un registro de la información de trayecto del código.

El trayecto del código ofrece información sobre las líneas de código que se han ejecutado y el tiempo empleado para ejecutarlas. Esta información puede ayudarle a identificar las áreas de código que no se están ejecutando y, por tanto, tampoco se están comprobando, así como las que podría modificar para obtener mayor rendimiento.



Puede activar y desactivar el trayecto del código si hace clic en el botón Trayecto de código de la barra de herramientas de la ventana [Depurador](#). Si activa el trayecto de código, se abre el cuadro de diálogo [Trayecto](#) donde puede especificar un archivo en el que se guardará esta información.

### Cuadro de diálogo Trayecto



También puede activar o desactivar el registro de trayecto mediante programación mediante el comando [SET COVERAGE TO](#). Por ejemplo, podría incluir el comando siguiente en la aplicación inmediatamente antes de la parte de código que desea investigar:

```
SET COVERAGE TO miregistro.log
```

Después de la sección de código cuyo trayecto desee registrar, podría incluir el comando siguiente para desactivar el trayecto de código:

```
SET COVERAGE TO
```

Una vez especificado un archivo para la información de trayecto, cambie a la ventana principal de Visual FoxPro y ejecute el programa, formulario o aplicación. En el archivo de registro se escribe la información siguiente para cada línea de código que se ejecute:

- Tiempo, en segundos, empleado en la ejecución de la línea.
- La clase a la que pertenece el código.
- El método o procedimiento en el que se encuentra la línea.
- El número de la línea de código.
- El archivo en el que se encuentra el código.

La forma más sencilla de extraer información del archivo de registro es convertirla en una tabla de modo que se puedan establecer filtros, ejecutar consultas e informes, ejecutar comandos y manipular la tabla de otras maneras.

La [aplicación Analizador de trayecto](#) crea un cursor a partir de los datos generados en el registro de trayecto y lo utiliza en una ventana para facilitar el análisis.

El programa siguiente convierte en una tabla el archivo de texto creado por el registro de trayecto:

```
cFileName = GETFILE('DBF')
IF EMPTY(cFileName)
    RETURN
ENDIF

CREATE TABLE (cFileName) ;
    (duration n(7,3), ;
    class c(30), ;
    procedure c(60), ;
    line i, ;
    file c(100))

APPEND FROM GETFILE('log') TYPE DELIMITED
```

## Controlar errores de tiempo de ejecución

Los errores de tiempo de ejecución se producen después de comenzar la ejecución de la aplicación. Entre las acciones que podrían generar errores de tiempo de ejecución se encuentran la escritura en un archivo que no existe, el intento de abrir una tabla que ya está abierta, el intento de seleccionar una tabla que se ha cerrado, la presencia de un conflicto de datos, la división de un valor entre cero, etc.

Los comandos y funciones siguientes resultan útiles para anticiparse y gestionar los errores de tiempo de ejecución.

Para	Use
Llenar una matriz con información de error	<a href="#">AERROR()</a>
Abrir la ventana Depurador o Seguimiento	<a href="#">DEBUG</a> o <a href="#">SET STEP ON</a>
Generar un error específico para probar el control de errores	<a href="#">ERROR</a>
Devolver un número de error	<a href="#">ERROR()</a>
Devolver una línea de programa en ejecución	<a href="#">LINENO()</a>

Devolver una cadena de mensaje de error	<a href="#">MESSAGE()</a>
Ejecutar un comando cuando se produce un error	<a href="#">ON ERROR</a>
Devolver comandos asignados a los comandos de control de errores	<a href="#">ON()</a>
Devolver el nombre del programa actualmente en ejecución	<a href="#">PROGRAM()</a> o <a href="#">SYS(16)</a>
Volver a ejecutar el comando anterior	<a href="#">RETRY</a>
Devolver cualquier parámetro de mensaje de error actual	<a href="#">SYS(2018)</a>

## Previsión de errores

La primera línea de defensa contra errores de tiempo de ejecución es anticiparse a que ocurran y codificarlos. Por ejemplo, esta línea de código mueve el puntero al siguiente registro de la tabla:

```
SKIP
```

Este código funciona a menos que el puntero de registro ya esté situado detrás del último registro de la tabla y es en este punto donde se producirá el error.

Las líneas de código siguiente se anticipan a este error y lo evitan:

```
IF !EOF ( )
  SKIP
  IF EOF ( )
    GO BOTTOM
  ENDIF
ENDIF
```

Otro ejemplo es esta línea de código que muestra el cuadro de diálogo **Abrir** para permitir al usuario abrir una tabla en un área de trabajo nueva:

```
USE GETFILE( 'DBF' ) IN 0
```

El problema es que el usuario podría elegir **Cancelar** en el cuadro de diálogo **Abrir** o escribir el nombre de un archivo que no existe. El código siguiente se anticipa a esta situación comprobando que el archivo existe antes de que el usuario intente utilizarlo:

```
IF FILE(cNewTable)
  USE (cNewTable) IN 0
ENDIF
```

El usuario final también podría escribir el nombre de un archivo que no sea una tabla de Visual FoxPro. Para solucionar este problema, podría abrir el archivo con las funciones de E/S de archivo a bajo nivel, analizar el encabezado binario y comprobar que el archivo es una tabla válida. Sin embargo, esto implica un pequeño trabajo y podría ralentizar perceptiblemente la aplicación. Lo mejor sería controlar la situación durante el tiempo de ejecución mostrando un mensaje como "Abra

otro archivo. Este archivo no es una tabla". cuando se produzca el error 15, "No es una tabla".

No puede, y probablemente tampoco lo desee, anticipar todos los errores posibles, por lo que tendrá que interrumpir algunos mediante código que se ejecutará en caso de que se produzca un error de tiempo de ejecución.

## Controlar errores de procedimientos

Cuando se produzca un error en el código de procedimientos, Visual FoxPro comprobará el código de control de errores asociado a una rutina [ON ERROR](#). Si no existe ninguna rutina ON ERROR, Visual FoxPro muestra el mensaje de error predeterminado de Visual FoxPro. Para obtener una lista completa de los mensajes de error de Visual FoxPro y los números de error, vea la Ayuda.

### Crear una rutina ON ERROR

Puede incluir cualquier comando o expresión válida de FoxPro después de ON ERROR, pero normalmente se llama a un procedimiento o programa de control de errores.

Para ver el funcionamiento de ON ERROR, puede escribir en la ventana Comandos un comando irreconocible, como:

```
qxy
```

Aparecerá un cuadro de diálogo de mensaje de error estándar de Visual FoxPro que indica "No se reconoce el verbo de comando". No obstante, si ejecuta las líneas de código siguientes, aparecerá el error número 16 impreso en la ventana de salida activa en lugar de mostrarse el mensaje de error estándar en un cuadro de diálogo:

```
ON ERROR ?ERROR()  
qxy
```

Si se ejecuta ON ERROR sin escribir nada más detrás, se restablecen los mensajes de error incorporados de Visual FoxPro:

```
ON ERROR
```

De forma esquemática, el código siguiente ilustra un controlador de errores ON ERROR:

```
LOCAL lcOldOnError  
  
* Guardar el controlador de errores original  
lcOldOnError = ON("ERROR")  
  
* Ejecutar ON ERROR con el nombre de un procedimiento  
ON ERROR DO errhandler WITH ERROR(), MESSAGE()  
  
* código al que se aplica la rutina de control de errores  
  
* Restablecer el controlador de errores original  
ON ERROR &lcOldOnError  
  
PROCEDURE errhandler  
LOCAL aErrInfo[1]
```

```

AERROR(aErrInfo)
DO CASE
    CASE aErrInfo[1] = 1 && El archivo no existe
        * mostrar el mensaje apropiado
        * y emprender alguna acción para solucionar el problema.
    OTHERWISE
        * mostrar un mensaje genérico, y quizá enviar
        * un mensaje electrónico de alta prioridad al administrador
ENDPROC

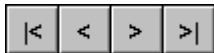
```

## Controlar errores en clases y objetos

Cuando se produce un error en el código del método, Visual FoxPro comprueba el código de gestión de errores asociado al evento Error del objeto. Si no se ha escrito código a nivel del objeto para el evento Error, se ejecuta el código del evento Error heredado de la clase principal o de otra clase superior de la jerarquía de clases. Si no se ha escrito código para el evento Error en ninguna clase de la jerarquía, Visual FoxPro comprueba la existencia de una rutina ON ERROR. Si no existiera, Visual FoxPro mostrará el mensaje de error predeterminado de Visual FoxPro.

En las clases puede encapsularse todo lo que necesita un control, incluido el control de errores para que el control pueda utilizarse en diversos entornos. Si posteriormente descubre otro error que podría encontrar el control, puede agregar la gestión de ese error a la clase; de este modo, todos los objetos basados en la clase heredarán automáticamente la función de control del nuevo error.

Por ejemplo, la clase `vcr` de la biblioteca de clases Buttons.vcx, ubicada en el directorio ... \Samples\Vfp98\Classes de Visual Studio se basa en la clase de contenedor de Visual FoxPro.



Cuatro botones de comandos del contenedor controlan el desplazamiento por la tabla moviendo el puntero de registro en una tabla con los comandos siguientes:

```
GO TOPSKIP - 1SKIP 1GO BOTTOM.
```

Podría producirse un error si un usuario elige uno de los botones y no hay ninguna tabla abierta. Visual FoxPro intenta escribir en una tabla valores almacenados en el búfer cuando se mueve el puntero de registro. Por tanto, podría producirse un error si se activa el almacenamiento optimista de filas en búfer y otro usuario ha cambiado un valor en el registro almacenado en el búfer.

Estos errores podrían producirse cuando el usuario elige cualquiera de los botones; por tanto, no tiene sentido tener cuatro métodos diferentes de control de errores. El código siguiente asociado al evento Error de cada uno de los botones de comando transfiere la información del error a una sola rutina de gestión de errores de la clase:

```

LPARAMETERS nError, cMethod, nLine
THIS.Parent.Error(nError, cMethod, nLine)

```

El código siguiente está asociado al evento Error de la clase `vcr`. El código real es diferente debido a los requisitos de codificación para la localización.

```

Parameters nError, cMethod, nLine
DO CASE

```

```

CASE nError = 13 && No se ha encontrado el alias
    cNewTable = GETFILE('DBF')
    IF FILE(cNewTable)
        SELECT 0
        USE (cNewTable)
        This.SkipTable = ALIAS()
    ELSE
        This.SkipTable = ""
    ENDIFCASE nError = 1585 && Conflicto de datos
* Actualizar conflicto controlado por una clase de comprobación de datos
    nConflictStatus = ;
    THIS.DataChecker1.CheckConflicts()
    IF nConflictStatus = 2
        MESSAGEBOX "No se puede resolver un conflicto de datos."
    ENDIFOTHERWISE
* Mostrar información sobre otros errores.

    cMsg="Error:" + ALLTRIM(STR(nError)) + CHR(13) ;
    + MESSAGE()+CHR(13)+"Programa:"+PROGRAM()

    nAnswer = MESSAGEBOX(cMsg, 2+48+512, "Error")
DO CASE
    CASE nAnswer = 3    &&Anular
        CANCEL
    CASE nAnswer = 4    &&Reintentar
        RETRY
    OTHERWISE          && Ignorar
        RETURN
ENDCASEENDCASE

```

Es posible que desee estar seguro de que suministra información para un error que nunca ha controlado. De lo contrario, el código del evento de error se ejecutará pero no realizará ninguna acción y ya no se mostrará el mensaje de error predeterminado de Visual FoxPro. El resultado es que ni el programador ni el usuario sabrán lo que ha sucedido.

Dependiendo de los usuarios a los que esté destinada la aplicación, podría suministrar más información en el caso de un error no controlado, como el nombre y número de teléfono de la persona a la que se puede solicitar ayuda.

## Retorno desde el código de control de errores

Después de ejecutarse el código de control de errores, se ejecuta la línea de código que sigue a la que ha producido el error. Si desea volver a ejecutar la línea de código que ha producido el error cuando haya cambiado la situación que lo ha producido, use el comando [RETRY](#).

**Nota** Puede llamar al evento Error cuando el error encontrado no esté asociado a una línea del código. Por ejemplo, si llama al método CloseTables de un entorno de datos cuando AutoCloseTables está establecido como verdadero (.T.) y se libera el formulario, se generará un error interno cuando Visual FoxPro intente volver a cerrar las tablas. Puede interrumpir en este error pero no existe ninguna línea de código para ejecutar RETRY.

# Capítulo 15: Optimizar aplicaciones

Cuando utilice Visual FoxPro para diseñar y ejecutar aplicaciones, querrá obtener el máximo rendimiento tanto de su sistema operativo como de Visual FoxPro y de su aplicación.



Para obtener información sobre la forma de obtener el máximo rendimiento del equipo y del sistema operativo, consulte el capítulo 4, [Optimizar el sistema](#), de la *Guía de instalación e Índice principal*.

En este capítulo se describe:

- [Optimizar tablas e índices](#)
- [Usar Rushmore para agilizar el acceso a datos](#)
- [Optimizar formularios y controles](#)
- [Optimizar programas](#)
- [Optimizar controles ActiveX](#)
- [Optimizar aplicaciones en entornos multiusuario](#)
- [Optimizar el acceso a datos remotos](#)
- [Optimizar aplicaciones internacionales](#)

## Optimizar tablas e índices

Puede agilizar el acceso a datos de tablas mediante índices y almacenamiento en búfer de forma eficaz. Además, puede usar la tecnología Rushmore para optimizar las consultas.

### Usar índices

El uso de índices permite agilizar el acceso a los datos de una tabla. Si se agrega un índice a una tabla se agilizan las búsquedas, especialmente si se puede utilizar la tecnología Rushmore para optimizar la búsqueda. El uso de índices también permite trabajar con los datos en un orden determinado, como ver una tabla de clientes ordenada por apellido.

Si los registros de una tabla tienen claves exclusivas, cree un índice principal o candidato en el campo. Estos tipos de índice permiten que Visual FoxPro valide la tecla a un nivel inferior con lo que se consigue el máximo rendimiento.

Además de usar índices en los campos utilizados para buscar y ordenar, también debería indexar todos los campos relacionados con una combinación. Si combina dos tablas en campos que no están indexados, la operación de combinación puede tardar mucho más tiempo en realizarse.

Una característica importante de Visual FoxPro es que puede crear un índice en cualquier expresión (en algunas bases de datos sólo puede indexar los campos). Esta capacidad permite usar índices para optimizar las operaciones de búsqueda, ordenación o combinación en conjuntos de campos o en expresiones derivadas de campos. Por ejemplo, puede crear un índice de un campo de nombres basado en una expresión que use la función [SOUNDEX\(\)](#). De esa manera, la aplicación puede proporcionar un acceso muy rápido a los nombres que guarden un cierto parecido.

Al agregar índices a las tablas, debe ver si la mejora en los tiempos de recuperación compensa la pérdida de rendimiento al actualizar la tabla. A medida que agregue más índices a la tabla, las actualizaciones e inserciones en la tabla se ralentizarán porque Visual FoxPro necesita actualizar cada índice.

Por último, evite el uso de índices en campos que contengan sólo unos cuantos valores discretos,

como los campos lógicos. En estos casos, el índice sólo contiene un pequeño número de entradas y el trabajo de mantener el índice probablemente sea mayor que la ventaja que se consigue a la hora de realizar búsquedas.

Para obtener detalles sobre la forma de crear índices de forma eficaz al usar la tecnología Rushmore, vea [Usar la tecnología Rushmore para agilizar el acceso a los datos](#), más adelante en este capítulo.

## Optimizar combinaciones

Cuando vaya a crear combinaciones mediante [SELECT - SQL](#), las situaciones siguientes pueden reducir el rendimiento y producir resultados imprevistos:

- Combinación de tablas en datos que no sean una clave principal o exclusiva en una de las tablas.
- Combinación de tablas que contengan campos vacíos.

Para evitar estas situaciones, cree combinaciones que se basen en la relación entre las claves principales de una tabla y las claves externas de la otra. Si crea una combinación basada en datos que no sean exclusivos, el resultado final puede ser el producto de las dos tablas. Por ejemplo, la instrucción SELECT - SQL siguiente crea una combinación que puede producir un resultado muy grande:

```
SELECT *;  
FROM  tastrade!customer INNER JOIN tastrade!orders ;  
      ON  Customer.postal_code = Orders.postal_code
```

En el ejemplo, el código postal identifica de forma exclusiva una ubicación dentro de una ciudad pero tiene poco valor si lo que pretende es establecer una correspondencia entre las filas de clientes y las filas de sus pedidos. El código postal no identifica necesariamente un cliente o un pedido de forma exclusiva. En su lugar, cree una combinación con una instrucción como la siguiente:

```
SELECT *;  
FROM  tastrade!customer INNER JOIN tastrade!orders ;  
      ON  Customer.customer_id = Orders.customer_id
```

En este ejemplo, el campo `customer_id` identifica de forma exclusiva un cliente determinado y los pedidos que pertenecen a ese cliente y, por tanto, crea un conjunto de resultados que combina la fila de clientes con cada fila de pedidos.

Además, tenga cuidado al combinar tablas que tengan campos vacíos porque Visual FoxPro establecerá una correspondencia con los campos vacíos. Sin embargo, no establece una correspondencia con los campos que contengan el valor nulo. Al crear una combinación, califique las expresiones del campo en la condición de combinación probando con una cadena vacía.

Por ejemplo, si piensa que el campo de identificación del cliente de la tabla `Orders` podría estar vacío, use una instrucción como la siguiente para filtrar los registros de pedidos que no tengan número de cliente:

```
SELECT *;  
FROM  tastrade!customer INNER JOIN tastrade!orders ;  
      ON  Customer.customer_id = Orders.customer_id; (error w/o ;)
```

```
WHERE tastrade!orders <> ""
```

**Sugerencia** También puede probar una cadena vacía con la función [EMPTY\(\)](#), pero la inclusión de una llamada a la función dentro de la expresión de filtro no resulta tan rápida como efectuar una comparación con un valor constante.

## Usar el Administrador de proyectos

Cuando use el Administrador de proyectos, puede combinar un número ilimitado de programas y procedimientos en un solo archivo .app o .exe. Con esto se aumenta considerablemente la velocidad de ejecución del programa por varias razones.

En primer lugar, Visual FoxPro abre un archivo de programa y lo deja abierto. Después, cuando se ejecuta un comando [DO](#) en un programa contenido en el archivo, Visual FoxPro no necesita abrir un archivo adicional.

En segundo lugar, una aplicación de uno o dos archivos reduce el número de archivos necesarios en el directorio de trabajo. La velocidad de todas las operaciones con el archivo aumentará a medida que el sistema operativo tenga menos entradas de directorios que examinar al abrir, cambiar el nombre o eliminar archivos.

Para obtener más información sobre el uso del Administrador de proyectos para crear aplicaciones, consulte el capítulo 13, [Compilar una aplicación](#).

## Sugerencias generales para la optimización de tablas e índices

Para crear tablas e índices de la forma más rápida posible, siga las recomendaciones que se muestran a continuación.

- Si no está activado el almacenamiento de registros o tablas en búfer, use [INSERT - SQL](#) en lugar de APPEND BLANK seguido por REPLACE, especialmente con una tabla indexada en un entorno multiusuario, porque los índices deben actualizarse una sola vez.
- Si necesita anexar un gran número de registros a una tabla indexada, sería más rápido quitar el índice, anexar los registros y después volver a crear el índice.
- En las instrucciones SQL, evite las llamadas a funciones en la medida de lo posible, especialmente en instrucciones que devuelvan más de un registro, porque la instrucción debe volverse a evaluar (lo que implica una nueva llamada a las funciones) para cada registro. Si crea una instrucción SQL con datos variables, use las expresiones de nombre o la sustitución de macros en lugar de la función EVALUATE( ). Una estrategia aún mejor es crear toda la instrucción de forma dinámica, no en cláusulas individuales. Para obtener más información, vea [Usar macros](#) y [Crear expresiones de nombre](#), en la Ayuda.
- Si suele usar un orden de índice determinado, puede mejorar el rendimiento si ordena periódicamente la tabla en este orden.
- Use archivos .CDX en lugar de .IDX en entornos multiusuario porque podrá actualizar un solo archivo .CDX con más rapidez que si actualiza múltiples archivos .IDX.

## Usar Rushmore para agilizar el acceso a los datos

Para ayudarle a optimizar el rendimiento de las aplicaciones, Visual FoxPro incluye la tecnología

Rushmore para acceso a los datos. Con Rushmore puede ejecutar determinadas operaciones complejas con tablas muchísimo más rápido que sin esta tecnología.

## Explicar la tecnología Rushmore

La tecnología Rushmore es una técnica de acceso a datos que usa los índices estándar de Visual FoxPro para optimizar el acceso a datos. Puede utilizar Rushmore con cualquier índice de Visual FoxPro, incluyendo los índices de FoxPro 1.x (.idx), los índices compactos (.idx) o los índices compuestos (.cdx).

Tanto los índices .cdx como los índices compactos .idx utilizan una técnica de compresión que genera índices de hasta 1/16 del tamaño de los índices de formato tradicional. Visual FoxPro puede procesar índices compactos con mayor rapidez porque son físicamente más pequeños. Esto significa que Visual FoxPro requiere menos acceso a disco para procesar un índice y puede almacenar localmente una parte mayor del índice. Aunque Rushmore, como otras técnicas de acceso a archivos, se beneficia del menor tamaño de los índices compactos, también funciona correctamente con los índices de formatos antiguos.

Cuando Visual FoxPro procesa tablas muy grandes en equipos con poca memoria, es posible que Rushmore no disponga de suficiente memoria para operar. En este caso, aparecerá un mensaje de advertencia ("No hay suficiente memoria para la optimización"). Su programa funcionará correctamente sin perder ningún dato; sin embargo, la consulta no se beneficiará de la tecnología Rushmore.

En su forma más sencilla, Rushmore incrementa la velocidad de los comandos en operaciones con una sola tabla mediante cláusulas FOR que especifican los conjuntos de registros en función de los índices existentes. Además, Rushmore puede aumentar la velocidad de funcionamiento de algunos comandos, como [LOCATE](#) e [INDEX](#). Para ver una lista completa de los comandos optimizables, consulte la sección "Usar Rushmore con tablas".

Los comandos SQL de Visual FoxPro utilizan la tecnología Rushmore como herramienta básica para la optimización de consultas de múltiples tablas, utilizando los índices existentes e incluso creando nuevos índices con fines específicos para aumentar la velocidad de dichas consultas.

## Usar Rushmore con tablas

Utilice Rushmore para optimizar el acceso a datos según el número de tablas que intervengan. Cuando tenga acceso a una única tabla, podrá beneficiarse de Rushmore siempre que aparezca una cláusula FOR. Cuando tenga acceso a múltiples tablas, las consultas SELECT - SQL prevalecen sobre todas las optimizaciones Rushmore. En un comando SQL, Visual FoxPro decide lo que se necesita para optimizar una consulta y realiza las operaciones automáticamente. No es necesario que abra tablas o índices. Si SQL decide que necesita índices, crea índices temporales para su propio uso.

## Para utilizar Rushmore

Elija una de las siguientes opciones:

- Para tener acceso a datos que estén contenidos en una única tabla, utilice una cláusula FOR en un comando como [AVERAGE](#), [BROWSE](#) o [LOCATE](#) o use comandos SQL para actualizar

tablas. Para obtener una lista completa de comandos que utilizan la cláusula FOR, consulte la tabla siguiente.

–O bien–

- Para tener acceso a datos contenidos en más de una tabla, utilice un comando [SELECT - SQL](#), [DELETE - SQL](#) y [UPDATE - SQL](#).

La tabla siguiente muestra una relación de los comandos que utilizan cláusulas FOR. Rushmore está diseñado para que su velocidad sea proporcional al número de registros obtenidos en la operación.

### Comandos que se pueden optimizar con cláusulas FOR

<a href="#">AVERAGE</a>	<a href="#">BLANK</a>
<a href="#">BROWSE</a>	<a href="#">CALCULATE</a>
<a href="#">CHANGE</a>	<a href="#">COPY TO</a>
<a href="#">COPY TO ARRAY</a>	<a href="#">COUNT</a>
<a href="#">DELETE</a>	<a href="#">DISPLAY</a>
<a href="#">EDIT</a>	<a href="#">EXPORT TO</a>
<a href="#">INDEX</a>	<a href="#">JOIN WITH</a>
<a href="#">LABEL</a>	<a href="#">LIST</a>
<a href="#">LOCATE</a>	<a href="#">RECALL</a>
<a href="#">REPLACE</a>	<a href="#">REPLACE FROM ARRAY</a>
<a href="#">REPORT</a>	<a href="#">SCAN</a>
<a href="#">SET DELETED</a>	<a href="#">SET FILTER</a>
<a href="#">SORT TO</a>	<a href="#">SUM</a>
<a href="#">TOTAL TO</a>	

Si utiliza una cláusula de alcance, además de una expresión de cláusula FOR optimizable, el alcance debe ser ALL o REST para poder aprovechar la tecnología Rushmore. Las cláusulas de alcance NEXT o RECORD desactivan Rushmore. Puesto que el alcance predeterminado es ALL, Rushmore funcionará cuando omita la cláusula de alcance.

Rushmore puede utilizar cualquiera de los índices abiertos salvo los índices filtrados y UNIQUE.

**Nota** Para obtener un rendimiento óptimo, no establezca el orden de la tabla.

Si crea índices o etiquetas, el orden se establecerá automáticamente. Si desea sacar el máximo rendimiento de Rushmore a la hora de manejar un conjunto de datos masivo con un orden

determinado, ejecute [SET ORDER TO](#) para desactivar el control de los índices y luego utilice el comando [SORT](#).

### Indexado eficaz para Rushmore

Rushmore no puede aplicarse a todos los índices. Si usa una cláusula FOR en el comando [INDEX](#), Rushmore no puede utilizar el índice para la optimización. Por ejemplo, debido a que contiene una cláusula FOR, la instrucción siguiente no puede optimizarse:

```
INDEX ON ORDNUM FOR DISCOUNT > 10 TAG ORDDISC
```

De forma análoga, Rushmore no puede usar un índice creado con una *condición* NOT. Por ejemplo, la instrucción siguiente puede optimizarse:

```
INDEX ON DELETED( ) TAG DEL
```

Sin embargo, ésta no puede optimizarse:

```
INDEX ON NOT DELETED( ) TAG NOTDEL
```

En el caso especial de que desee excluir los registros eliminados de una consulta, use un índice como el del primer ejemplo para agilizar las operaciones cuando haya establecido SET DELETED como ON.

### Funcionamiento sin Rushmore

Las operaciones de obtención de datos se ejecutan sin la optimización Rushmore en las siguientes situaciones:

- Cuando Rushmore no puede optimizar la expresión de la cláusula FOR en un comando posiblemente optimizable. Para más información sobre la creación de expresiones optimizables FOR, consulte la sección [Combinar expresiones básicas optimizables](#).
- Cuando un comando que puede beneficiarse de Rushmore contiene una cláusula WHILE.
- Cuando se disponga de poca memoria. La recuperación de datos sigue su curso, pero no se optimiza.

### Desactivar Rushmore

Aunque no deseará hacerlo a menudo, es posible desactivar Rushmore. Cuando ejecute un comando que utilice Rushmore, Visual FoxPro determinará inmediatamente qué registros coinciden con la expresión de la cláusula FOR. A continuación, el comando manipula estos registros.

Si un comando que se puede optimizar modifica la clave de índice en la cláusula FOR, el conjunto de registros sobre el que opera Rushmore puede quedarse desfasado. En este caso, puede desactivar Rushmore para asegurarse de que disponga de la información más actualizada de la tabla.

#### Para desactivar Rushmore para un comando individual

- Utilice la cláusula NOOPTIMIZE.

Por ejemplo, este comando [LOCATE](#) no está optimizado:

```
LOCATE FOR DueDate < {^1998-01-01} NOOPTIMIZE
```

Puede desactivar o activar globalmente Rushmore para todos los comandos que se benefician de Rushmore, con el comando [SET OPTIMIZE](#).

### Para desactivar Rushmore globalmente

- Utilice el código siguiente:

```
SET OPTIMIZE OFF
```

### Para desactivar Rushmore globalmente

- Utilice el código siguiente:

```
SET OPTIMIZE ON
```

El valor predeterminado de la optimización Rushmore está definido como ON.

## Optimizar expresiones Rushmore

La tecnología Rushmore depende de la presencia de una *expresión básica optimizable* en una cláusula FOR. Una expresión básica optimizable puede formar una expresión completa o puede aparecer como parte de una expresión. También puede combinar expresiones básicas para formar una expresión optimizable compleja.

### Crear expresiones básicas optimizables

Una expresión básica optimizable toma una de las dos formas siguientes:

*eIndex relOp eExp*

–O bien–

*eExpr relOp eIndex*

Una expresión básica optimizable tiene las siguientes características:

- *eIndex* coincide exactamente con la expresión sobre la cual está construido un índice.
- *eExp* es cualquier expresión y puede incluir variables y campos de otras tablas no relacionadas.
- *OpRel* es uno de los siguientes operadores: <, >, =, <=, >=, <>, #, ==, o !=. También puede utilizar las funciones [ISNULL\(\)](#), [BETWEEN\(\)](#) o [INLIST\(\)](#) (o sus equivalentes SQL, como IS NULL, etc.).

Puede utilizar BETWEEN( ) o INLIST( ) de las dos formas siguientes:

*eIndex* BETWEEN(*eIndex*, *eExpr*, *eExpr*)

–O bien–

*eExpr* INLIST(*eIndex*, *eExpr*)

**Nota** ISBLANK( ) y EMPTY( ) no son optimizables con la tecnología Rushmore.

Si crea los índices *firstname*, *custno*, UPPER(*lastname*) y *hiredate*, cada una de las siguientes expresiones es optimizable:

```
firstname = "Carlos"
custno >= 1000
UPPER(lastname) = "Martín"
hiredate < {^1997-12-30}
```

Una expresión optimizable puede contener variables y funciones que se evalúan como un valor concreto. Por ejemplo, al usar el índice *addr*, si ejecuta el comando "WASHINGTON AVENUE" TO *cVar*, las siguientes instrucciones también serían expresiones básicas optimizables:

```
ADDR = cVar
ADDR = SUBSTR(cVar, 8, 3)
```

### Cuándo se optimizan las consultas

Es importante entender cuándo se optimizarán las consultas y cuándo no. Visual FoxPro optimiza las condiciones de búsqueda; para ello, busca una coincidencia exacta entre el lado izquierdo de una expresión de filtro y una expresión de clave de índice. Por tanto, Rushmore puede optimizar una expresión sólo si busca la expresión exacta usada en un índice.

Por ejemplo, imagine que acaba de crear una tabla y va a agregar el primer índice mediante un comando como el siguiente:

```
USE CUSTOMERS
INDEX ON UPPER(cu_name) TAG name
```

El comando siguiente no es optimizable porque la condición de búsqueda se basa únicamente en el campo *cu\_name* y no en una expresión que esté indexada:

```
SELECT * FROM customers WHERE cu_name = "ACME"
```

En su lugar, debería crear una expresión optimizable con un comando como el siguiente, en la que la expresión que se esté buscando coincida exactamente con la expresión indexada:

```
SELECT * FROM customers WHERE UPPER(cu_name) = "ACME"
```

**Sugerencia** Para determinar el nivel de optimización Rushmore utilizado, llame a [SYS\(3054\)](#).

### Combinar expresiones básicas optimizables



Puede combinar expresiones simples o complejas basadas en las cláusulas FOR o WHERE para incrementar la velocidad de recuperación de datos. Esto es posible si las expresiones FOR tienen las características de las expresiones básicas optimizables.

Las expresiones básicas pueden ser optimizables. Puede combinar expresiones básicas con los operadores lógicos AND, OR y NOT para formar una expresión de cláusula FOR compleja que también se puede optimizar. Una expresión creada con una combinación de expresiones básicas optimizables es totalmente optimizable. Si una o más de las expresiones básicas no son optimizables, la expresión compleja se podría optimizar parcialmente o bien no ser optimizable en absoluto.

Un conjunto de reglas determina si una expresión formada por expresiones básicas optimizables o no optimizables se puede optimizar totalmente, parcialmente o no se puede optimizar. La tabla siguiente resume las reglas de optimización de consultas de Rushmore.

### Combinar expresiones básicas

Expresión básica	Operador	Expresión básica	Resultados de la consulta
Optimizable	AND	Optimizable	Totalmente optimizable
Optimizable	OR	Optimizable	Totalmente optimizable
Optimizable	AND	No optimizable	Parcialmente optimizable
Optimizable	OR	No optimizable	No optimizable
No optimizable	AND	No optimizable	No optimizable
No optimizable	OR	No optimizable	No optimizable
—	NOT	Optimizable	Totalmente optimizable
—	NOT	No optimizable	No optimizable

Puede utilizar el operador AND para combinar dos expresiones optimizables en una expresión totalmente optimizable:

```
FIRSTNAME = "CARLOS" AND HIREDATE < {^1997-12-30}      && Optimizable
```

En el siguiente ejemplo, el operador OR combina una expresión básica optimizable con una expresión no optimizable para crear una expresión que no es optimizable:

```
FIRSTNAME = "CARLOS" OR "S" $ LASTNAME                && No optimizable
```

Puede crear una expresión totalmente optimizable si utiliza el operador NOT con una expresión optimizable:

```
NOT FIRSTNAME = "CARLOS"                               && Totalmente optimizable
```

También puede utilizar paréntesis para agrupar combinaciones de expresiones básicas.

### Combinar expresiones complejas

Del mismo modo que puede combinar expresiones básicas, puede combinar expresiones complejas para crear una expresión aún más compleja totalmente optimizable, parcialmente optimizable o no optimizable. A su vez, puede combinar estas expresiones más complejas para crear expresiones que se pueden optimizar total o parcialmente o que no se pueden optimizar. La tabla siguiente describe los resultados de combinar estas expresiones complejas. Estas reglas también se aplican a expresiones agrupadas con paréntesis.

### Combinación de expresiones complejas

Expresión	Operador	Expresión	Resultado
Totalmente optimizable	AND	Totalmente optimizable	Totalmente optimizable
Totalmente optimizable	OR	Totalmente optimizable	Totalmente optimizable
Totalmente optimizable	AND	Parcialmente optimizable	Parcialmente optimizable
Totalmente optimizable	OR	Parcialmente optimizable	Parcialmente optimizable
Totalmente optimizable	AND	No optimizable	Parcialmente optimizable
Totalmente optimizable	OR	No optimizable	No optimizable
—	NOT	Totalmente optimizable	Totalmente optimizable
Parcialmente optimizable	AND	Parcialmente optimizable	Parcialmente optimizable
Parcialmente optimizable	OR	Parcialmente optimizable	Parcialmente optimizable
Parcialmente optimizable	AND	No optimizable	Parcialmente optimizable
Parcialmente optimizable	OR	No optimizable	No optimizable
—	NOT	Parcialmente optimizable	No optimizable
No optimizable	AND	No optimizable	No optimizable
No optimizable	OR	No optimizable	No optimizable

—	NOT	No optimizable	No optimizable
---	-----	----------------	----------------

Puede combinar expresiones totalmente optimizables con el operador OR para crear una expresión que también sea totalmente optimizable:

```
* Expresión totalmente optimizable
(FIRSTNAME = "CARLOS" AND HIREDATE < {^1997-12-30}) ;
  OR (LASTNAME = "" AND HIREDATE > {^1996-12-30})
```

Para crear expresiones parcialmente optimizables, combine una expresión totalmente optimizable con una expresión que no sea optimizable. En el siguiente ejemplo, el operador AND se utiliza para combinar estas expresiones:

```
* Expresión parcialmente optimizable
(FIRSTNAME = "FRED" AND HIREDATE < {^1997-12-30}) ;
  AND "S" $ LASTNAME
```

Las expresiones parcialmente optimizables se pueden combinar para crear una expresión que también sea parcialmente optimizable:

```
* Expresión parcialmente optimizable
(FIRSTNAME = "CARLOS" AND "S" $ LASTNAME) ;
  OR (FIRSTNAME = "PEDRO" AND "T" $ LASTNAME)
```

La combinación de expresiones que no sean optimizables crea una expresión que tampoco se puede optimizar:

```
* Expresión que no es optimizable
("CARLOS" $ FIRSTNAME OR "S" $ LASTNAME) ;
  OR ("MAIN" $ STREET OR "AVE" $ STREET)
```

## Optimizar formularios y controles

También puede mejorar de forma significativa los formularios y controles de la aplicación.

**Sugerencia** Si desea más información sobre el establecimiento y obtención de propiedades de forma eficaz, vea [Referencias a propiedades de objetos de forma eficaz](#), más adelante en este capítulo.

### Usar el entorno de datos

Si utiliza el entorno de datos del Diseñador de formularios o el Diseñador de informes, el rendimiento de la apertura de la tabla es mucho más rápido que si se ejecutaran los comandos [USE](#), [SET ORDER](#) y [SET RELATION](#) en el evento Load del formulario. Cuando use el entorno de datos, Visual FoxPro utiliza las llamadas al motor de nivel inferior para abrir las tablas y configurar los índices y las relaciones.

### Limitar el número de formularios en un conjunto de formularios

Utilice conjuntos de formularios solamente cuando sea necesario que un grupo de formularios comparta una sesión de datos privada. Cuando utilice un conjunto de formularios, Visual FoxPro crea

instancias de todos los formularios y de todos los controles de todos los formularios del conjunto, aun cuando sólo se muestre en pantalla el primer formulario del conjunto. Esto puede resultar lento e innecesario si los formularios no tienen que compartir una sesión de datos privada. En su lugar, debería ejecutar [DO FORM](#) para los otros formularios cuando sean necesarios.

Sin embargo, si utiliza un conjunto de formularios, obtendrá una pequeña mejora del rendimiento cuando tenga acceso a los formularios del conjunto porque los formularios ya estarán cargados aunque no visibles.

## Carga dinámica de controles de página en un marco de páginas

Los marcos de página, al igual que los conjuntos de formularios, cargan todos los controles de cada página en el momento de cargar el marco de páginas lo que da lugar a una demora perceptible cuando se carga este último. En su lugar, podría cargar los controles de página dinámicamente a medida que se fueran necesitando, creando una clase fuera de los controles de cada página y cargándolos cuando se activara la página.

### Para cargar dinámicamente los controles de página

1. Diseñe el formulario de la forma habitual, incluyendo todos los controles en todas las páginas.
2. Una vez terminado el diseño, vaya a la segunda página del marco de página y guarde los controles allí existentes como una clase.
3. Abra la clase que acaba de crear y compruebe que los controles sigan dispuestos de la forma correcta.
4. Repita los pasos 2 y 3 para la tercera y las páginas siguientes del marco de página.
5. En el evento [Activate](#) de la segunda página y de las siguientes del marco de página, agregue objetos y déjelos visibles.

Por ejemplo, si la clase de controles se llama `cnrpage1`, debe agregar el código siguiente:

```
IF THIS.ControlCount = 0  
THIS.AddObject("cnrpage1","cnrpage1")  
THIS.cnrpage1.Visible = .T.  
ENDIF
```

## Vinculación dinámica de controles a datos

Puede agilizar el tiempo de carga de un formulario que contenga numerosos controles vinculados a datos si retrasa la vinculación de esos controles hasta el momento en que sean necesarios.

### Para vincular dinámicamente controles a datos

1. Sitúe las tablas y vistas del formulario en el entorno de datos de modo que se abran cuando se cargue el formulario.
2. Para cada control dependiente, agregue código al código de evento [GotFocus](#) que vincula el

control al valor de los datos. Por ejemplo, el código siguiente vincula un control ComboBox al campo `customer.company`:

```
* Comprobar si ya se ha vinculado el control.
IF THIS.RecordSource = ""
* Establecer el valor del origen de registros
* y establecer el tipo de origen de registros como "fields"
THIS.RecordSource = "customer.company"
THIS.RecordSourceType = 6
THIS.Refresh
ENDIF
```

## Retardo de la actualización de pantalla

Si realiza varios cambios en la pantalla, por ejemplo, cambia los valores de varios controles a la vez, puede reducir el tiempo global necesario para actualizar la pantalla si retarda la actualización de pantalla hasta que se realicen todos los cambios. Por ejemplo, si hace que los controles queden visibles o invisibles, cambia los colores de los controles o mueve los registros de controles dependientes, resulta mucho más eficaz retardar el relleno de color de estos controles hasta que se hayan completado todos los cambios:

### Para retardar la actualización de la pantalla

1. Establezca la propiedad [LockScreen](#) del formulario como verdadera.
2. Actualice los controles cuando sea necesario.
3. Llame al método [Refresh](#) del formulario.
4. Establezca la propiedad `LockScreen` del formulario como falsa.

El ejemplo siguiente cambia varias propiedades de la pantalla a la vez, se desplaza a un registro nuevo y sólo entonces actualiza la pantalla con información nueva. Si `LockScreen` no se hubiera establecido como verdadero, en cada una de estas operaciones se volverían a dibujar los controles afectados individualmente y el rendimiento global de la actualización parecería retardado.

```
THISFORM.LockScreen = .T.
THISFORM.MyButton.Caption = "Guardar"
THISFORM.MyGrid.BackColor = RGB (255, 0, 0)    && Rojo
SKIP IN customers
SKIP IN orders
THISFORM.Refresh
THISFORM.LockScreen = .F.
```

**Sugerencia** Esta técnica no proporciona ninguna ventaja si está actualizando un único control.

## Reducir código en métodos usados con frecuencia

Puesto que el método [Refresh](#) y el evento [Paint](#) suelen utilizarse con frecuencia, puede aumentar el rendimiento de formularios si reduce la cantidad de código de estos métodos. De forma análoga, para acelerar el tiempo de carga de un formulario, podría mover código desde el evento `Init` a los métodos usados con menos frecuencia como [Activate](#), [Click](#) y [GotFocus](#). Entonces, use una propiedad del

control (como [Tag](#) o una propiedad personalizada) para hacer un seguimiento de si el control ya ha ejecutado el código que sólo necesita ejecutarse una vez.

## Optimizar programas

Escribiendo código cuidadosamente, puede escribir programas más rápidos. Son varias las maneras de aumentar el rendimiento de los programas en Visual FoxPro:

- Seguir las sugerencias generales sobre el rendimiento de programación que aparecen a continuación.
- Usar expresiones de nombres en lugar de la sustitución mediante macros.
- Hacer referencia a propiedades de objetos de forma eficaz.

### Sugerencias generales sobre el rendimiento

Para escribir programas más rápidos, siga las recomendaciones que se indican a continuación.

- Elija el tipo de datos correcto. En particular, use el tipo de datos Integer para la información numérica cuando sea posible, puesto que se procesa de forma más eficaz. Siempre que sea posible, use el tipo de datos Integer para los valores de claves externas, lo que dará lugar a archivos de datos e índices más pequeños (y, por tanto, más rápidos) y a combinaciones más rápidas.

**Nota** Para ver un ejemplo que muestre la forma de crear un índice más pequeño y, por tanto, más rápido, ejecute Solution.app en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio. Elija **Ver los ejemplos mediante una lista filtrada**, seleccione **Índice** en la lista desplegable y después elija **Crear índices pequeños usando BINTOC( )** en la lista que aparece.

- Evite volver a abrir archivos, puesto que esto ralentiza el rendimiento. En su lugar, asigne archivos a áreas de trabajo cuando los abra y después use el comando [SELECT](#) para elegir un área de trabajo específica cuando se precise.
- Los bucles [FOR ... ENDFOR](#) son más rápidos que los bucles [DO WHILE ... ENDDO](#).
- Cuando copie datos de varios campos, [SCATTER TO ARRAY](#) es más rápido que [SCATTER MEMVAR](#).
- Para optimizar el uso de la memoria, evite la creación de objetos antes de necesitarlos y borre los objetos cuando deje de trabajar con ellos para liberar memoria.

**Sugerencia** Puede comprobar cuánta memoria consume cada objeto si llama a la función [SYS\(1016\)](#).

- Envíe la salida a la ventana superior siempre que sea posible; la actualización de ventanas situadas detrás de la superior es bastante más lenta. No se debe, en ningún caso, hacer que la salida se desplace detrás de una ventana.
- Desactive la presentación del estado con el comando [SET TALK OFF](#), que elimina el trabajo de la actualización de la pantalla.
- Establezca el comando [SET DOHISTORY](#) como OFF para evitar la actualización de la ventana de comandos cada vez que se ejecute un programa.

## Usar expresiones de nombre en lugar de sustitución de macros

Si utiliza expresiones de nombre en lugar de la sustitución de macros, el rendimiento del programa aumentará notablemente. Por ejemplo, si asigna un valor a la variable `cFile`, una expresión de nombre creada con `cFile` será más rápida que la sustitución de macros.

```
cFile = "CUST"
use &cFile      && Sustitución de macros, lenta
use (cFile)    && Expresión de nombre: rápida, preferida
```

## Referencias a propiedades de objetos de forma eficaz

Comprendiendo la forma en que funciona Visual FoxPro con las propiedades y objetos, puede optimizar la ejecución de las aplicaciones.

### Optimizar referencias a una propiedad repetidas

Cuando haga referencia a una propiedad de objeto con la sintaxis *objeto.propiedad*, Visual FoxPro debe buscar el objeto antes de poder tener acceso a la propiedad. Si debe tener acceso a la propiedad repetidas veces, esta estrategia de búsqueda puede ralentizar el rendimiento.

Para evitar las referencia al mismo procedimiento varias veces (como en un bucle), lea el valor de la propiedad en una variable, realice los cambios y establezca la propiedad una sola vez cuando termine. Por ejemplo, el código siguiente rellena una matriz de propiedades; para ello, crea primero una matriz en la memoria, la rellena y después establece la propiedad una sola vez al final:

```
* Copiar cadena a una variable localvariable
lcChar = THISFORM.cCharString
LOCAL laCharArray[256]   && Crear matriz local
FOR nCounter = 1 to 256
    laCharArray[x] = SUBSTR(laChar,x,1)
ENDFOR
* Copiar la matriz local a la matriz de propiedades
ACOPY(laCharArray,THISFORM.aCharArray)
```

### Hacer referencia a múltiples propiedades de forma eficaz

Si actualiza más de una propiedad para el objeto, Visual FoxPro debe buscar el objeto varias veces, lo que puede afectar al rendimiento. En el ejemplo siguiente, el código hace que Visual FoxPro busque en cuatro objetos (como `THISFORM`, `pgfCstInfo`, `pgCstName` y `txtName`) para encontrar la propiedad que se va a establecer. Dado que el código establece dos propiedades, la búsqueda a cuatro niveles se realiza dos veces:

```
THISFORM.pgfcstInfo.pgCstName.txtName.Value = ;
    "Carlos Martín"
THISFORM.pgfcstInfo.pgCstName.txtName.BackColor = ;
    RGB (0,0,0) & Rojo oscuro
```

Para evitar este volumen de trabajo, use el comando [WITH ... ENDWITH](#). Este método hace que Visual FoxPro busque el objeto una sola vez. Por ejemplo, el ejemplo siguiente realiza la misma tarea que el anterior, pero de forma más rápida:

```
WITH THISFORM.pgfcstInfo.pgCstName.txtName  
    .Value = "Carlos Martín"  
    .BackColor = RGB (0,0,0) & Rojo oscuro  
ENDWITH
```

También puede almacenar una referencia a un objeto en una variable y después incluir la variable en lugar de la referencia al objeto:

```
oControl = THISFORM.pgfcstInfo.pgCstName.txtName  
oControl.Value = "Carlos Martín"  
oControl.BackColor = RGB (0,0,0) & Rojo oscuro
```

## Optimizar controles ActiveX

Si utiliza Automatización o controles ActiveX en la aplicación, puede ajustar la aplicación para obtener el máximo rendimiento de los controles ActiveX y de Automatización.

### Usar controles ActiveX de forma eficaz

Para obtener el máximo rendimiento a la hora de utilizar controles ActiveX en los formularios, siga estas recomendaciones:

- Inicie los servidores de Automatización en primer lugar. Los controles enlazados a campos generales suelen funcionar mejor cuando los servidores de estos tipos de datos (como Microsoft Excel o Word) ya se están ejecutando en el equipo cliente.
- Inserte objetos "como un icono". Cuando inserte un control ActiveX en un campo, hágalo como un icono o un marcador de posición en lugar de insertar el objeto completo. Esto reduce la cantidad de espacio de almacenamiento necesario porque Visual FoxPro almacena una imagen de presentación con el objeto, lo que consume gran cantidad de espacio de almacenamiento. Si se inserta un objeto como un icono también se aumenta el rendimiento para dibujar el objeto.
- Use controles de imágenes. Si desea mostrar un mapa de bits (como, el logotipo de una empresa), los controles de imágenes son mucho más rápidos que los controles OLEBound.
- Use vínculos manuales siempre que sea posible. Los vínculos manuales a los objetos son más rápidos porque evitan el tiempo de notificación requerido para los vínculos automáticos y porque el servidor no necesita iniciarse para dibujar el objeto. Si no necesita actualizar un objeto con frecuencia, use vínculos manuales.

## Optimizar el rendimiento de Automatización

Si la aplicación interactúa con otras aplicaciones, puede conseguir el máximo rendimiento mediante las técnicas siguientes.

### Evitar múltiples instancias del servidor

En algunos casos, los servidores de Automatización (como Microsoft Excel) iniciarán una nueva instancia, aun cuando ya haya una ejecutándose. Para evitar esto y aumentar el rendimiento, use la función [GetObject\(\)](#) en lugar de [CreateObject\(\)](#). Por ejemplo, la llamada siguiente siempre utilizará una instancia existente:



```
x = GetObject(,"excel.Application")
```

De lo contrario, la llamada siguiente creará una instancia nueva:

```
x = CreateObject("excel.Application")
```

Si llama a `GetObject( )` pero el servidor no está funcionando, aparecerá el error 1426. En ese caso, puede interrumpir el error y llamar a `CreateObject( )`:

```
ON ERROR DO oleErr WITH ERROR()  
x = GetObject(,"excel.application")  
ON ERROR && restablecer el controlador de errores del sistema  
  
PROCEDURE oleErr  
PARAMETER mError  
IF mError = 1426 then  
    x = CreateObject("excel.application")  
ENDIF
```

### Hacer referencia a objetos de forma eficaz

La ejecución de expresiones que usen objetos en un servidor de Automatización puede resultar muy costosa, especialmente cuando se evalúan varias veces. Es mucho más económico almacenar las referencias de objetos en variables para tenerlas como referencia. Para obtener más información, vea [Optimizar referencias repetidas a una propiedad](#), en este mismo capítulo.

## Optimizar aplicaciones en entornos multiusuario

Si va a escribir aplicaciones para un entorno multiusuario, el rendimiento es un factor especialmente importante, porque las operaciones ineficaces se multiplican. Además, si varios usuarios tienen acceso a datos, la aplicación debe gestionar problemas de simultaneidad y acceso a la red.

Para administrar estos problemas, puede:

- Ajustar el intervalo de reintentos de bloqueo.
- Usar el proceso de transacciones de forma eficaz.

Asimismo, las sugerencias para trabajar con datos almacenados en servidores remotos pueden resultarle especialmente útiles. Para obtener más información, vea [Optimizar el acceso a datos remotos](#), más adelante en este capítulo.

### Ajustar el intervalo de reintentos de bloqueo

Si la aplicación intenta bloquear un registro o una tabla sin éxito, puede hacer que Visual FoxPro reintente el bloqueo automáticamente después de un breve intervalo. Sin embargo, cada intento de bloqueo da lugar a un tráfico de red más intenso. Si ya fuera intenso, el envío de repetidas solicitudes de bloqueo agregará una sobrecarga de trabajo a la red y provocará una ralentización global para todos los usuarios.

Para controlar esta situación, puede ajustar el intervalo entre los intentos de bloqueo. Si usa un

intervalo más grande (lo que provocaría menos reintentos por segundo) se reducirá el tráfico de la red y se aumentará el rendimiento.

### Para ajustar el intervalo de reintentos de bloqueo

- Llame a la función [SYS\(3051\)](#); para ello, indique el número de milisegundos que debe esperar entre cada intento de bloqueo.

### Usar el procesamiento de transacciones de forma eficaz

Cuando utiliza el procesamiento de transacciones, debe diseñar las transacciones de forma que se reduzca al mínimo el efecto que puedan tener en otros usuarios. Mientras esté abierta una transacción, todos los bloqueos establecidos durante la transacción permanecerán bloqueados hasta que se confirmen o se deshagan. Aun cuando ejecute un comando [UNLOCK](#) explícito, los bloqueos se mantendrán hasta que ejecute el comando [END TRANSACTION](#) o [ROLLBACK](#).

Además si anexa registros a una tabla, es imprescindible que Visual FoxPro bloquee el encabezado de la tabla. El encabezado permanece bloqueado durante el proceso de la transacción, lo que impide que los demás usuarios también puedan anexar registros.

Para reducir al mínimo el impacto de las transacciones, diseñelas de modo que comiencen y terminen lo más cerca posible de la actualización real de los datos; la transacción ideal sólo contiene instrucciones de actualización de datos.

Si va a agregar el proceso de transacciones a actualizaciones de datos realizadas en un formulario, no abra una transacción, ejecute el formulario y después confirme la transacción cuando el formulario esté cerrado. En su lugar, ponga las instrucciones de proceso de transacciones en el código de eventos para el botón Guardar, por ejemplo:

```
* Guardar método del botón de comando cmdSave
BEGIN TRANSACTION
UPDATE PRODUCTS SET reorder_amt = 0 WHERE discontinued = .T.
END TRANSACTION
```

## Optimizar el acceso a datos remotos

La recuperación de datos de cualquier base de datos remota resulta cara. Con el fin de obtener datos de una base de datos del servidor, hay que seguir estos pasos:

1. El cliente ejecuta la consulta en la base de datos remota.
2. El servidor analiza y compila la consulta.
3. El servidor genera un conjunto de resultados.
4. El servidor indica al cliente que se ha completado el resultado.
5. El cliente recoge los datos del servidor a través de la red. Este paso puede realizarse en una sola operación o el cliente puede solicitar que los resultados se envíen en partes a medida que se

vayan necesitando.

Puede usar varias técnicas para aumentar la velocidad de la recuperación o actualización de los datos. En la sección siguiente se tratan estas estrategias:

- Recuperar sólo los datos necesarios
- Actualizar tablas remotas de forma eficaz
- Enviar instrucciones en un lote
- Establecer el tamaño del paquete
- Retrasar la recuperación de datos memo y binarios
- Almacenar localmente datos de consulta
- Crear reglas locales

## Obtener sólo los datos necesarios

La mayor parte de las aplicaciones que usan datos remotos, formularios e informes no necesitan tener acceso a todos los datos de una tabla a la vez. Por tanto, puede aumentar el rendimiento si crea vistas remotas que busquen o actualicen únicamente los campos y registros que desee con lo que se reduce al mínimo la cantidad de datos que deben transmitirse a través de la red.

Para crear consultas que minimicen el trabajo de recuperación de datos de orígenes remotos, siga estas sugerencias:

- Especifique sólo los campos que necesite. No use la instrucción `SELECT * FROM customers` a menos que necesite todos los campos de la tabla.
- Incluya una cláusula `WHERE` para limitar el número de registros transferidos. Cuanto más específica sea la cláusula `WHERE`, menos registros se transmitirán al equipo y con más rapidez se terminará la consulta.
- Si no puede predecir durante el diseño los valores que se van a utilizar en una cláusula `WHERE`, puede utilizar parámetros en la cláusula. Cuando se ejecute la consulta, Visual FoxPro usará el valor de una variable de parámetro o solicitará al usuario el valor de búsqueda. Por ejemplo, esta consulta permite a la aplicación o al usuario rellenar la región en el tiempo de ejecución:

```
SELECT cust_id, company, contact, address ;  
FROM customers ;  
WHERE region = ?pcRegion
```

- Establezca la propiedad [NoDataOnLoad](#) del objeto Cursor en el entorno de datos correspondiente. Esta técnica se suele utilizar con vistas parametrizadas en las que los datos del parámetro proceden del valor de un control de un formulario.

## Actualizar tablas remotas de forma eficaz

Cuando se usa una vista para actualizar una tabla en un origen de datos remoto, Visual FoxPro debe comprobar si los registros que se están actualizando han sufrido alguna modificación. Para ello, Visual FoxPro debe examinar los datos en el servidor y compararlos con los datos existentes en su equipo. En algunos casos, esta operación puede resultar lenta.

Para optimizar el proceso de actualización de datos en orígenes de datos remotos, puede especificar la forma en que Visual FoxPro debe comprobar los registros modificados. Para ello, tiene que indicar la cláusula WHERE que Visual FoxPro debe generar para realizar la actualización.

Imagine, por ejemplo, que está usando una vista basada en una tabla de clientes en un origen de datos remoto. Ha creado la vista mediante una instrucción [SELECT - SQL](#) como la siguiente:

```
SELECT cust_id, company, address, contact ;
      FROM customers ;
      WHERE region = ?vpRegion
```

Quiere actualizar los cuatro campos que ha especificado en la vista, salvo el campo clave (*cust\_id*). En la tabla siguiente se presenta la cláusula WHERE que Visual FoxPro generará para cada una de las opciones disponibles en la cláusula SQL WHERE.

**Nota** La función [OLDVAL\(\)](#) devuelve la versión preactualizada de los campos que se han modificado y la función [CURVAL\(\)](#) devuelve el valor actual almacenado en el origen de datos remoto. Si compara estos valores, Visual FoxPro puede determinar si el registro ha cambiado en el origen de datos remoto desde que se transfirió al equipo.

Valor	Cláusula WHERE resultante
Sólo campos clave	WHERE OLDVAL( <i>cust_id</i> ) = CURVAL( <i>cust_id</i> )
Campos clave y actualizables (predeterminado)	WHERE OLDVAL( <i>cust_id</i> ) = CURVAL( <i>cust_id</i> ) AND OLDVAL(<mod_fld1>) = CURVAL(<mod_fld2>) AND OLDVAL(<mod_fld2>) = CURVAL(<mod_fld2>) AND ...
Campos clave y modificados	WHERE OLDVAL( <i>cust_id</i> ) = CURVAL( <i>cust_id</i> ) AND OLDVAL( <i>company</i> ) = CURVAL( <i>company</i> ) AND OLDVAL( <i>contact</i> ) = CURVAL( <i>contact</i> ) AND OLDVAL( <i>address</i> ) = CURVAL( <i>address</i> )
Clave y marca de hora	WHERE OLDVAL( <i>cust_id</i> ) = CURVAL( <i>cust_id</i> ) AND OLDVAL( <i>timestamp</i> ) = CURVAL( <i>timestamp</i> )

En general, debería elegir una opción para la cláusula SQL WHERE en este orden de preferencia:

1. **Clave y marca de hora**, si la base de datos remota admite los campos de marca de hora, que es la forma más rápida de indicar si se ha modificado algún registro.
2. **Campos clave y modificados**, porque los campos que se actualizan en el servidor son casi siempre un subconjunto del número total de campos que se podría actualizar.
3. **Campos clave y actualizables**.
4. **Sólo campos clave**. El uso de esta configuración implica que el servidor remoto insertará un

registro totalmente nuevo que use la clave modificada y eliminará el registro anterior.

## Enviar instrucciones en un lote

Algunos servidores (como Microsoft SQL Server) permiten enviar un lote de instrucciones SQL en un solo paquete. Esto aumenta el rendimiento porque se reduce el tráfico de la red y porque el servidor puede compilar múltiples instrucciones a la vez.

Por ejemplo, si especifica un tamaño de lote de cuatro y actualiza 10 registros en una base de datos, Visual FoxPro envía en un solo lote cuatro instrucciones como la siguiente a la base de datos del servidor:

```
UPDATE customer SET contact = "Agustina Rivera" ;  
  WHERE cust_id = 1;  
UPDATE customer SET contact = "Cristina Martínez" ;  
  WHERE cust_id = 2;  
UPDATE customer SET company = "Enrique Ballina" ;  
  WHERE cust_id = 3;  
UPDATE customer SET contact = "Ernesto Méndez" ;  
  WHERE cust_id = 4
```

## Para enviar instrucciones en un lote

- En el cuadro de diálogo **Opciones**, elija la ficha **Datos remotos** y en **Registros para actualizar por lotes** especifique el número de registros que van a incluirse en el lote.

—O bien—

- Llame a las funciones [DBSETPROP\(\)](#) o [CURSORSETPROP\(\)](#) para establecer estas propiedades:
  - Establezca Transaction a 2.
  - Establezca BatchUpdateCount al número de instrucciones que se van a enviar en un lote.

—O bien—

1. En el **Diseñador de vistas**, elija **Opciones avanzadas** del menú **Consulta** para abrir el cuadro de diálogo **Opciones avanzadas**.
2. En el área **Rendimiento**, situada junto a **Número de registros para actualizar por lotes**, especifique el número de instrucciones que se van a enviar en un lote.

**Nota** Debería probar con diferentes valores para esta propiedad y la propiedad PacketSize para optimizar las actualizaciones.

## Establecer el tamaño del paquete

Puede optimizar el acceso a servidores remotos; para ello, ajuste el tamaño del paquete de red que se envía y se obtiene de la base de datos remota. Por ejemplo, si la red admite tamaños grandes (más de 4096 bytes), puede aumentar el tamaño del paquete en Visual FoxPro con el fin de enviar más datos

cada vez que lea o escriba en la red.

### Para establecer el tamaño del paquete

- Llame a las funciones [DBSETPROP\(\)](#) o [CURSORSETPROP\(\)](#) y establezca la propiedad PacketSize a un valor entero positivo. El valor predeterminado es 4096.

**Nota** Es posible que distintos proveedores de red administren esta propiedad de forma distinta por lo que deberá consultar la documentación de su servicio de red. Novell® NetWare®, por ejemplo, tiene un tamaño de paquete máximo de 512 bytes por lo que si se establece la propiedad PacketSize a un valor superior a éste no se obtendrá ninguna ventaja adicional.

### Retardo de la recuperación de datos memo y binarios

Si está almacenando datos memo o binarios en un servidor remoto, puede aumentar el rendimiento; para ello, retarde la transferencia de estos datos hasta que la aplicación la requiera realmente.

#### Para retardar la recuperación de los datos memo y binarios

- En el cuadro de diálogo **Opciones**, elija la ficha **Datos remotos** y en **Opciones predeterminadas de vista remota**, establezca **Buscar memo**.

–O bien–

- Llame a las funciones [DBSETPROP\(\)](#) o [CURSORSETPROP\(\)](#) para establecer la propiedad FetchMemo.

### Almacenamiento local de datos de consulta

Muchas aplicaciones incluyen datos de consulta estáticos, como abreviaturas de estados, códigos postales y cargos de empleados. Si la aplicación contiene este tipo de datos y si la tabla no es demasiado grande, podría aumentar la velocidad de la aplicación; para ello, mantenga copias de esta información en el equipo de cada usuario, ya que las consultas no generan tráfico de red.

Esta técnica es especialmente útil para los datos que nunca cambian o lo hacen muy esporádicamente. Si los datos cambian en alguna ocasión, debe diseñar una estrategia para transferir una copia nueva de la tabla de consultas al equipo de cada usuario.

### Crear reglas locales

Puede aumentar el rendimiento de la aplicación si crea reglas locales a nivel de campo y de registro en Visual FoxPro, en lugar de utilizar las reglas definidas en el servidor. Estas reglas pueden impedir que los datos que no son compatibles con las reglas de datos o de la empresa se introduzcan en la base de datos.

Mediante la definición de reglas en Visual FoxPro, se detectan los datos no válidos antes de enviarse a través de la red, lo que es más rápido y ofrece un mejor control para administrar las condiciones de error. Sin embargo, el uso de reglas locales también implica su coordinación con las reglas del

servidor remoto. Por ejemplo, si se modifican las reglas del servidor, tendría que cambiar las reglas locales para que coincidieran.

Para obtener información sobre la creación de reglas locales, vea la sección [Actualizar datos en una vista](#) del capítulo 8, "Crear vistas".

## Optimizar aplicaciones internacionales

Si va a programar aplicaciones internacionales, podría necesitar administrar la secuencia de ordenación de los datos para obtener el máximo rendimiento. Esta sección trata sobre lo siguiente:

- Usar secuencias de ordenación de forma eficaz.
- Usar [SELECT - SQL](#) con múltiples secuencias de ordenación.

### Usar secuencias de ordenación de forma eficaz

Si los datos no incluyen [marcas diacríticas](#), como acentos (á) o diéresis (ü), puede aumentar el rendimiento mediante la secuencia de ordenación del equipo porque:

- Las claves de índices que no son del equipo son dos veces más grandes puesto que contienen la información diacrítica.
- La ordenación que no es del equipo usa muchas reglas especiales para indexar caracteres con el fin de que devuelvan los resultados adecuados.

Dado que la secuencia de ordenación del equipo es más rápida, se suele preferir para combinaciones y búsquedas mientras que otras secuencias de ordenación resultan perfectas para ordenar registros.

Cuando cree un índice, Visual FoxPro usa el valor actual de [SET COLLATE](#). Por tanto, si desea crear dos índices con dos secuencias de ordenación, puede usar una secuencia de comandos como la siguiente:

```
SET COLLATE TO "MACHINE"
INDEX ON lastname TAG _lastname      && combinar/buscar índice
SET COLLATE TO "GENERAL"
INDEX ON lastname TAG lastname      && índice de ordenación
```

Cuando desee buscar, seleccionar o combinar el campo lastname, ejecute el comando SET COLLATE TO "MACHINE" antes de realizar la operación. Rushmore utilizará el índice creado en la secuencia de ordenación del equipo y la operación de búsqueda será muy rápida.

### Usar SQL SELECT con múltiples secuencias de ordenación

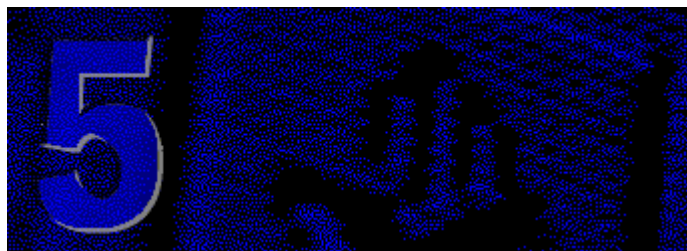
Cuando ejecute un comando [SELECT - SQL](#), Visual FoxPro usará la secuencia de ordenación actual para realizar la búsqueda y para las cláusulas ORDER BY y GROUP BY. Si desea buscar y ordenar mediante diferentes secuencias de ordenación, puede dividir los comandos SQL en dos pasos, como se indica a continuación:

```
* Seleccionar registros usando una secuencia de ordenación
SET COLLATE TO "MACHINE"
SELECT * FROM table INTO CURSOR temp1 ;
```

```
WHERE lname = "Alzaga"  
* Ordenar registros usando una secuencia de ordenación diferente  
SET COLLATE TO "GENERAL"  
SELECT * FROM temp1 INTO TABLE output ORDER BY lastname
```



# Manual del programador, Parte 5: Ampliar aplicaciones



Para ampliar una aplicación básica de Visual FoxPro, puede activarla para que funcione con múltiples usuarios, aprovechar las ventajas de los controles ActiveX y las aplicaciones compatibles con la Automatización, y agregar características internacionales.

## Capítulo 16 [Agregar OLE](#)

Use controles ActiveX y vinculación e incrustación de objetos en su aplicación para vincular datos y aprovechar las posibilidades de otras aplicaciones.

## Capítulo 17 [Programar para acceso compartido](#)

Si su aplicación se ejecuta en red o contiene formularios que tienen acceso a los mismos datos, la aplicación necesitará compartir dichos datos de manera eficaz para ofrecer la máxima productividad.

## Capítulo 18 [Programar aplicaciones internacionales](#)

Aprenda a lanzar sus aplicaciones hacia el mercado mundial diseñándolas para su uso internacional.

# Capítulo 16: Agregar OLE

Puede ampliar la potencia de las aplicaciones de Visual FoxPro con las capacidades de otras aplicaciones de Automatización o de [controles ActiveX](#). En los formularios o campos de tipo General de sus aplicaciones, puede incluir datos como texto, sonido, imágenes y vídeo procedentes de otras aplicaciones. Puede ver o manipular estos datos visualmente con la aplicación que los creó, o puede manipular la información invisible y controlar automáticamente la aplicación mediante programación.

Otras aplicaciones pueden ampliar la eficacia de Visual FoxPro mediante Automatización. Incluso puede crear servidores de Automatización (componentes ActiveX) en Visual FoxPro a los que pueden tener acceso todas las aplicaciones en modo local o remoto.

Este capítulo trata sobre lo siguiente:

- [Diseñar una aplicación OLE](#)

- [Agregar objetos OLE a sus aplicaciones](#)
- [Usar controles ActiveX](#)
- [Manipular objetos con Automatización](#)
- [Crear subclases de objetos](#)
- [Controlar Visual FoxPro desde otras aplicaciones](#)
- [Crear servidores de Automatización](#)
- [Usar Automatización remota](#)

## Diseñar una aplicación OLE

Las aplicaciones compatibles con Automatización y los componentes COM pueden actuar como [servidores de Automatización](#), clientes o como ambos. Los componentes que actúan como servidores pueden proporcionar objetos a otra aplicación mientras que los que actúan como clientes pueden crearlos.

Puede incorporar fácilmente la eficacia y la flexibilidad de aplicaciones como *Microsoft Excel* y *Microsoft Word* en las aplicaciones de Visual FoxPro. Dado que Visual FoxPro también actúa como servidor, puede proporcionar funciones que se pueden integrar en paquetes de soluciones basados en Microsoft Office o en otros componentes COM.

Los objetos OLE insertables provienen de aplicaciones compatibles con OLE tales como Excel o Word. Entre estos objetos se incluyen documentos de Word y hojas de cálculo de Excel. En los formularios puede vincular o incrustar estos objetos con el control contenedor OLE y puede almacenar los objetos en los campos de tipo General de una tabla, mostrándolos en sus formularios con el control OLE dependiente.

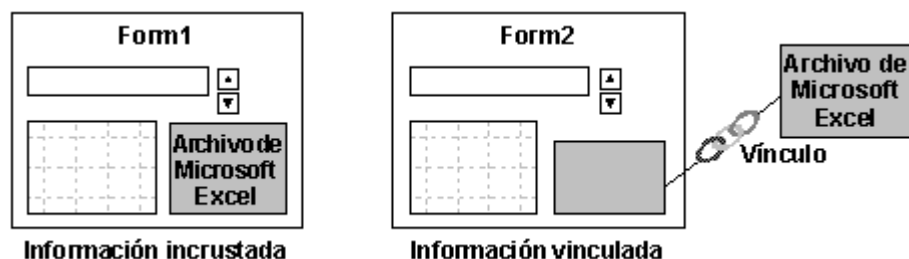
En una aplicación de Visual FoxPro, puede usar la tecnología OLE y ActiveX de diversas maneras. Antes de crear una aplicación, considere las distintas formas en que puede crearla.

### Vincular o incrustar objetos OLE

Puede incrustar o vincular archivos de otras aplicaciones para Windows en sus tablas y formularios. Por ejemplo, puede incrustar o vincular un documento de Word en un campo de tipo General de una tabla y puede incrustar o vincular una hoja de cálculo de Excel en un formulario.

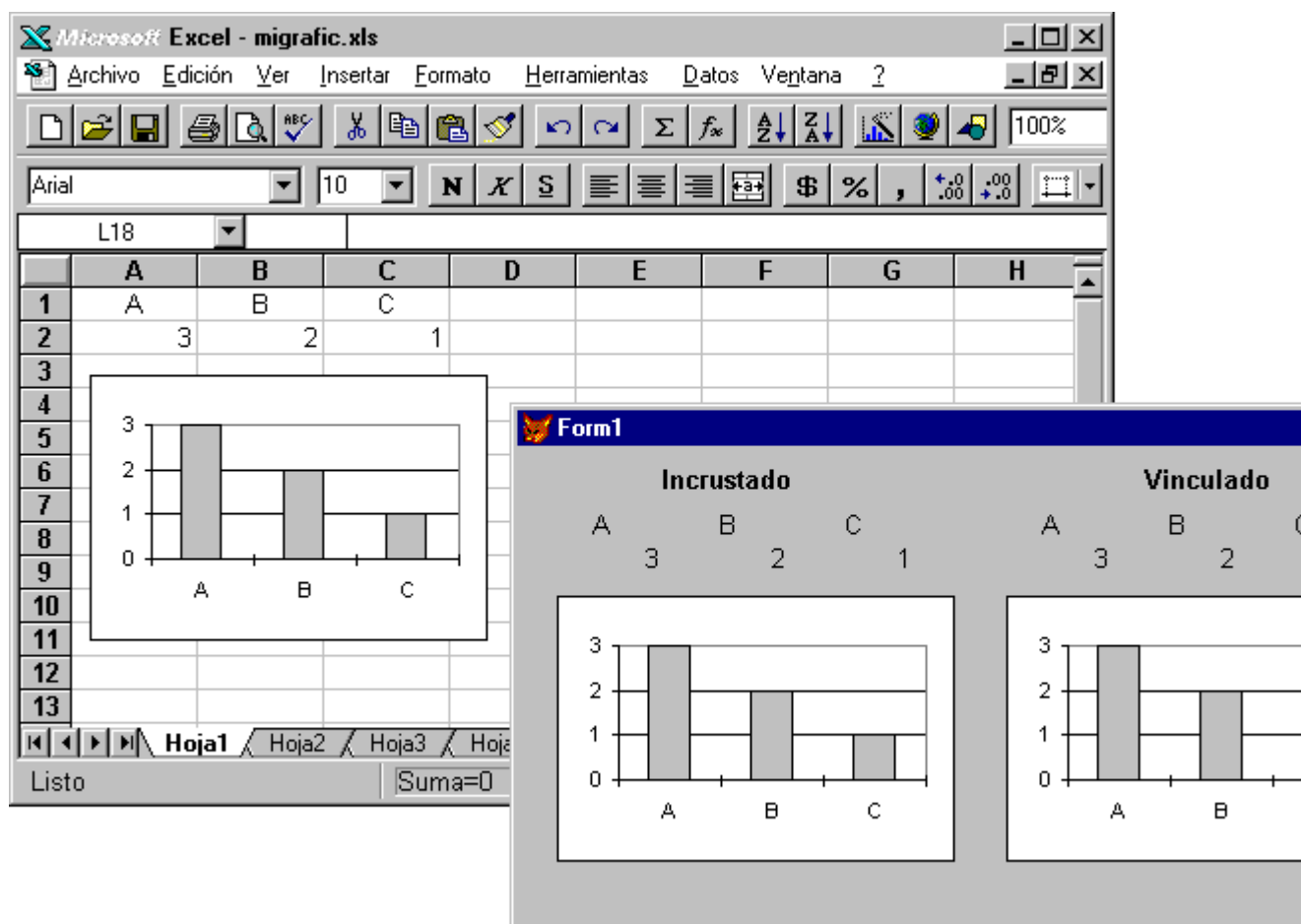
La diferencia entre incrustación y vinculación está en la forma en que se almacenan los datos. Mediante la incrustación, los datos se almacenan en una tabla o en un formulario; esto no ocurre mediante la vinculación. Por ejemplo, cuando incrusta una hoja de cálculo de Excel en un formulario, el formulario contiene una copia de la hoja de cálculo y otras aplicaciones no pueden modificar esta copia. Sin embargo, cuando la vincula, el formulario contiene tan sólo una referencia a la hoja de cálculo y no la hoja de cálculo propiamente dicha.

### Incrustar y vincular información



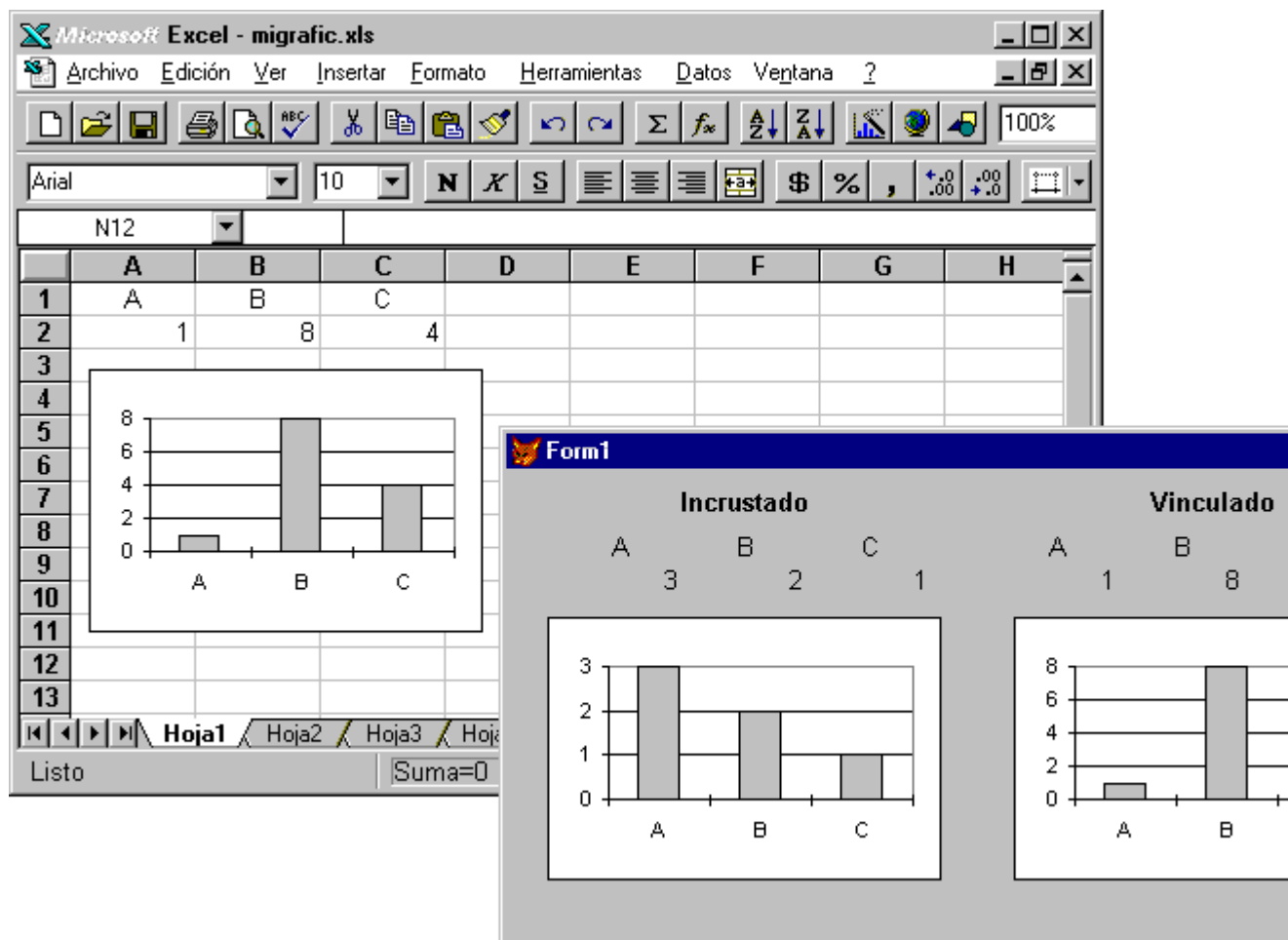
Tanto los datos incrustados como los vinculados están relacionados con el contenido original del archivo del servidor, como se muestra en la ilustración:

### Una hoja de cálculo incrustada y vinculada en un formulario



Pero cuando se modifica el archivo original, los datos vinculados se actualizan automáticamente para reflejar el cambio; sin embargo, los datos incrustados no lo hacen:

### Datos vinculados que se han actualizado en un formulario



Los datos incrustados no son necesariamente estáticos. Tanto los datos incrustados como los vinculados pueden mostrarse, modificarse y manipularse de forma interactiva y mediante programación en Visual FoxPro.

## Agregar objetos OLE dependientes o no dependientes



En un formulario o un informe, puede crear objetos que dependan de campos de tablas de tipo General. Estos objetos se llaman *objetos OLE dependientes* y puede usarlos para mostrar el contenido de campos de tipo General. Puede crear objetos OLE dependientes con el control [OLE dependiente](#) situado en la barra de herramientas Controles de formularios. De forma alternativa puede crear *objetos OLE independientes* con el control [contenedor OLE](#). Un objeto OLE independiente no está conectado a campos de tipo General de una tabla.

## Agregar objetos OLE a sus aplicaciones

Puede agregar objetos OLE a tablas y formularios de forma interactiva o mediante programación.

### Agregar objetos OLE a tablas

Cuando diseñe las tablas para la aplicación, considere si necesita objetos OLE en las tablas. Por ejemplo, suponga que tiene una tabla de empleados y quiere incluir documentos de Word que contengan las evaluaciones de los empleados. Para incluir las evaluaciones, tiene que definir un campo General en la tabla. Luego, agregue los documentos a la tabla mediante vinculación o incrustación de dichos documentos en el campo General.

### Para agregar un objeto OLE a una tabla

1. Use el [Diseñador de tablas](#) para crear una tabla con el campo General.
2. Abra la ventana correspondiente al campo General; para ello, examine la tabla y haga doble clic en el campo General o utilice el comando [MODIFY GENERAL](#).
3. En el menú **Edición**, elija **Insertar objeto**.

–O bien–

- Use el comando [APPEND GENERAL](#).

Para obtener más información sobre cómo agregar objetos OLE con el Diseñador de tablas, consulte el capítulo 10, [Uso compartido de información con otras aplicaciones](#), del *Manual del usuario*.

### Agregar objetos OLE a tablas

Puede agregar objetos OLE a tablas mediante programación con el comando [APPEND GENERAL](#). Con este comando puede importar un objeto OLE de un archivo y colocarlo en un campo General. Si el campo ya contiene un objeto, el nuevo objeto lo reemplazará.

**Nota** A diferencia de APPEND y APPEND BLANK, APPEND GENERAL no agrega ningún registro nuevo a la tabla.

Puede utilizar APPEND GENERAL para incrustar objetos OLE o vincular objetos OLE creados por aplicaciones tales como Excel o Word. Estas aplicaciones son compatibles tanto para vincular como para incrustar. Sin embargo, algunas aplicaciones, como por ejemplo Microsoft Graph, tan sólo permiten incrustar.

Suponga que tiene archivos de Word para Windows que desea almacenar en una tabla de Visual FoxPro. Si la tabla tiene un campo General llamado WordDoc, puede incrustar los documentos mediante el código siguiente:

```
CREATE TABLE oletable (name c(24), worddoc g)
CD GETDIR()

nArchivos = ADIR(aArchivosWord, "*.doc")
IF nArchivos > 0
    FOR i = 1 to nArchivos
        APPEND BLANK
        REPLACE oletable.Name WITH aArchivosWord(i,1)
        APPEND GENERAL WordDoc FROM aArchivosWord(i,1)
    ENDFOR
END
```

```
ELSE
  MESSAGEBOX("No hay archivos de Word".)
ENDIF
```

**Nota** Este código sólo busca los archivos que terminen en .doc, la extensión estándar empleada por los archivos de Word. Puesto que Word para Windows y OLE se dan cuenta de este hecho, los archivos se asocian automáticamente con el servidor de Word para Windows cuando usted utiliza APPEND GENERAL.

Si emplea una extensión diferente de la esperada por el servidor, deberá declarar la clase de servidor mediante el empleo de la cláusula CLASS. Por ejemplo, si agrega la clase de Word al ejemplo anterior, el código se convierte en:

```
APPEND GENERAL WordDoc FROM ArchivosWord(i,1) CLASS "Word.Document.6"
```

Si tiene archivos con extensiones comunes (por ejemplo, .bmp) que otros servidores pueden utilizar, puede emplear la cláusula CLASS para especificar el servidor que desea usar para esos archivos. Alternativamente, si prefiere vincular objetos en lugar de incrustarlos, emplee la palabra clave LINK.

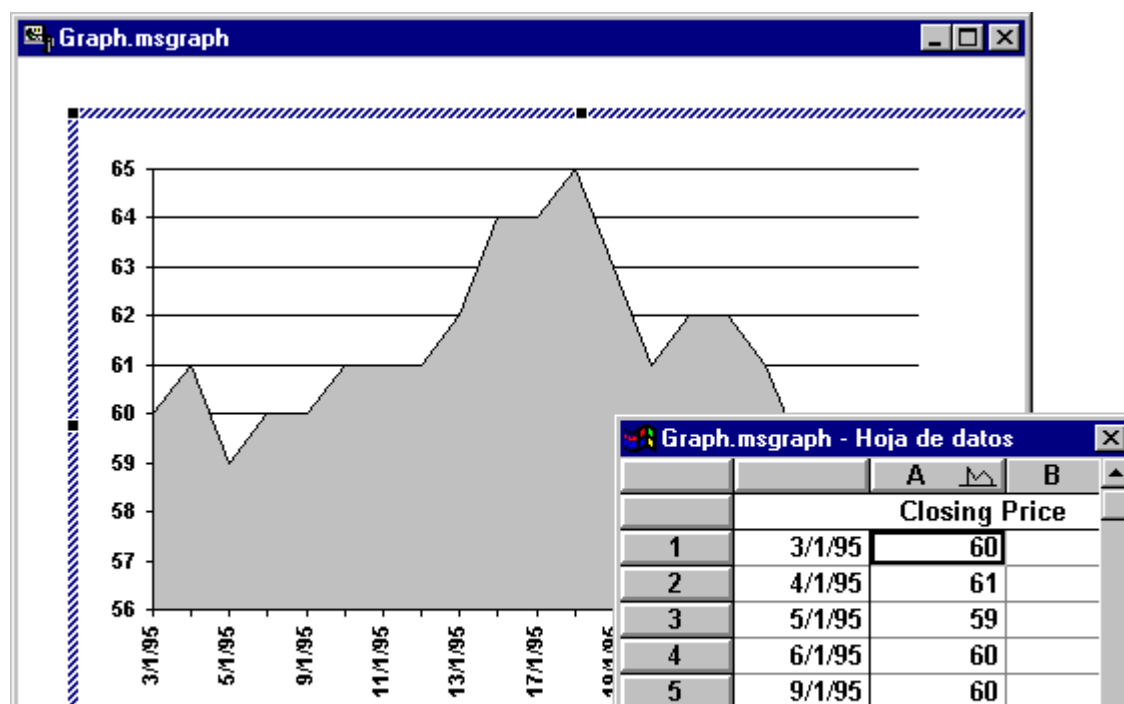
```
APPEND GENERAL WordDoc FROM ArchivosWord(i,1) LINK CLASS "Word.Document.6"
```

Además, puede reemplazar datos de un objeto con la palabra clave DATA de [APPEND GENERAL](#), como ilustra el ejemplo siguiente de Microsoft Graph.

### Actualización de Microsoft Graph

Microsoft Graph es una aplicación incrustada. Los valores de un gráfico de Microsoft Graph se basan en los valores de una hoja de datos.

### Objeto de Microsoft Graph en un campo General



6	10/1/95	61
7	11/1/95	61

Para cambiar los datos de un gráfico de Microsoft Graph mediante programación, debe crear una cadena que contenga los datos nuevos, incluyendo las tabulaciones, los retornos de carro y los avances de línea, y pasarla a un objeto de Microsoft Graph con la cláusula DATA del comando [APPEND GENERAL](#).

En el ejemplo siguiente se supone que tiene una tabla, denominada `stock`, con los campos `date` y `close` (entre otros) correspondientes a la fecha y al precio de cierre del stock. El objeto de Microsoft Graph se almacena en el campo General `msgraph` de una tabla denominada `graph`. El ejemplo actualiza un gráfico con los precios de cierre de stock desde los 30 últimos días.

Código	Comentarios
<pre>#DEFINE CRLF CHR(13)+CHR(10) #DEFINE TAB CHR(9) LOCAL lcData</pre>	Define los caracteres de retorno de carro y tabulación.
<pre>SELECT date, close;   FROM Stock WHERE BETWEEN(date, ;     DATE(),DATE() - 30) ;   ORDER BY date INTO CURSOR wtemp</pre>	Selecciona los valores con los que se desea actualizar el gráfico; en este caso, los valores de fecha y cierre de los stocks correspondientes a los 30 últimos días.
<pre>SELECT wtemp lcData = " " + ;   TAB + "Precio de cierre" + CRLF SCAN   lcData = lcData + DTOC(date)   lcData = lcData + TAB   lcData = lcData + ;     ALLTRIM(STR(close)) + CRLF ENDSCAN</pre>	<p>Genera una cadena de caracteres (<code>lcData</code>) de datos desde la posición del cursor para actualizar el gráfico.</p> <p>"Precio de cierre", como encabezado de la columna, es el texto predeterminado que se mostrará en la leyenda del gráfico.</p>
<pre>SELECT graph APPEND GENERAL msgraph DATA lcData</pre>	Envíe los nuevos valores al gráfico en la cláusula DATA del comando APPEND GENERAL.
<pre>USE IN wtemp</pre>	Cierre el cursor.

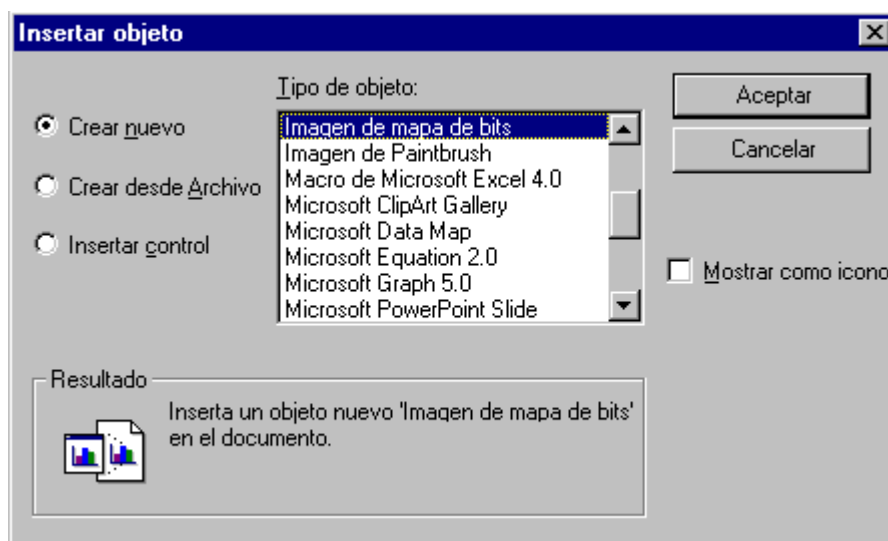
**Nota** También puede mostrar objetos OLE desde campos de tipo General en los informes. Para obtener detalles sobre la presentación de objetos OLE en informes, consulte "Agregar un campo de tipo general" en el capítulo 7, [Diseñar informes y etiquetas](#), del *Manual del usuario*.

## Agregar objetos OLE a formularios

Con el Diseñador de formularios, puede agregar objetos OLE insertables a formularios mediante el control [contenedor OLE](#). Además, puede mostrar objetos OLE de los campos de tipo General con el control [OLE dependiente](#).

**Para agregar un objeto insertable a un formulario**

1. En el [Diseñador de formularios](#), agregue un control [contenedor OLE](#) a su formulario. Aparecerá el cuadro de diálogo **Insertar objeto**.
2. En el cuadro de diálogo [Insertar objeto](#), seleccione **Crear nuevo** o **Crear desde archivo**.

**Cuadro de diálogo Insertar objeto**

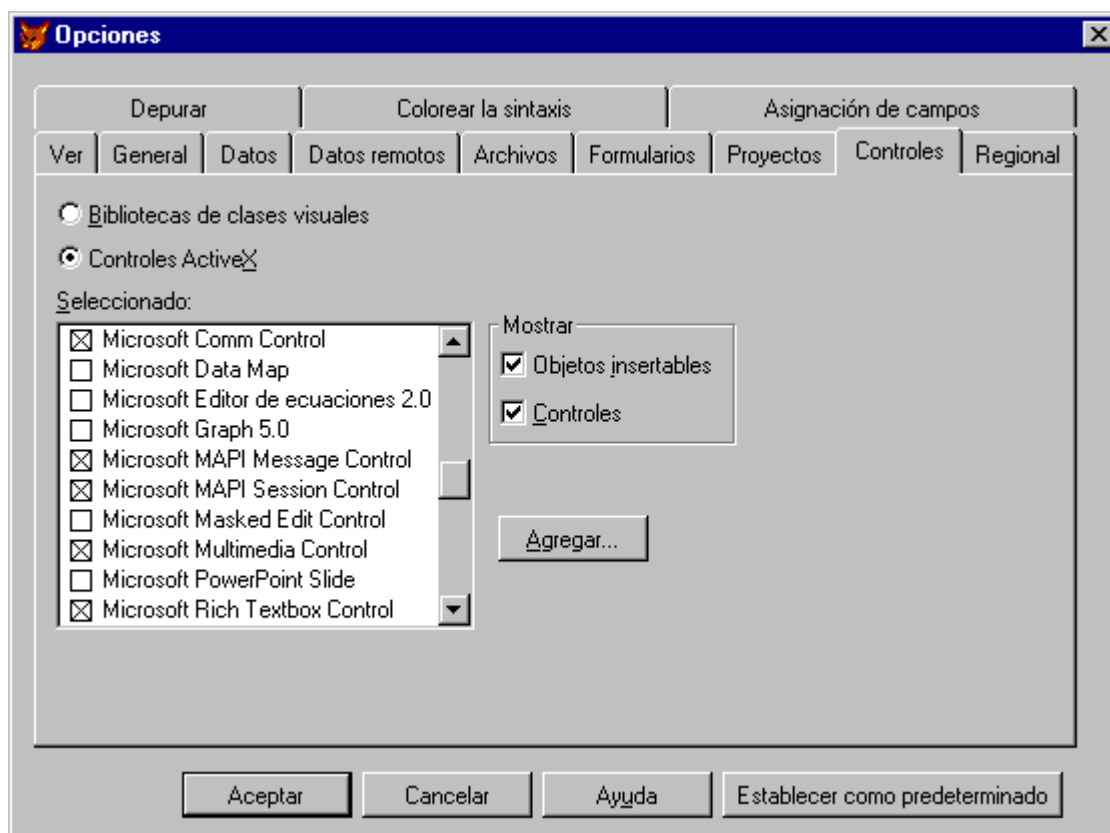
3. Elija el objeto OLE apropiado en la lista **Tipo de objeto**.

Puede personalizar la barra de herramientas Controles de formularios para agregar directamente objetos OLE específicos.

**Para agregar objetos OLE a la barra de herramientas Controles de formularios**

1. En el menú **Herramientas**, elija **Opciones**.
2. En la ficha [Controles](#) del cuadro de diálogo **Opciones**, elija **Controles ActiveX**.



**Ficha Controles del cuadro de diálogo Opciones**

3. En la lista **Seleccionado**, elija los objetos OLE y los controles ActiveX que desee dejar disponibles en la barra de herramientas **Controles de formularios**.

4. Elija **Establecer como predeterminado** y, a continuación, **Aceptar**.
5. En la barra de herramientas **Controles de formularios**, elija **Ver clases** y, a continuación, **Controles ActiveX**.

### Para mostrar un objeto OLE de un campo General

1. En el [Diseñador de formularios](#), agregue un control [OLE dependiente](#) a su formulario.
2. Especifique el campo General que contiene los datos; para ello, establezca la propiedad [ControlSource](#) del objeto.

Por ejemplo, si el nombre de la tabla es `Inventory` y el nombre del campo General es `Current`, establezca la propiedad `Control Source` como `Inventory.Current`.

También puede mostrar un objeto OLE de un campo General mediante programación:

Código	Comentarios
<code>frm1 = CREATEOBJECT("form")</code>	Crea un formulario
<code>frm1.ADDOBJECT("olb1", "oleboundcontrol")</code>	Agrega un control
<code>frm1.olb1.ControlSource = "Inventory.Current"</code>	Vincula los datos al control
<code>frm1.olb1.Visible = .T.</code> <code>frm1.Visible = .T.</code>	Deja visibles el control y el formulario

### Funcionamiento interactivo con objetos OLE

Si agrega un objeto OLE a un formulario o campo General, puede modificar los datos y mostrar las características del objeto en tiempo de ejecución o durante el diseño.

**Nota** No puede modificar los datos de un objeto OLE en un control OLE dependiente durante el diseño.

Algunos objetos OLE admiten la modificación directa, por lo que puede modificar el objeto en la ventana utilizada por la aplicación. Por ejemplo, si hace doble clic en un objeto de hoja de cálculo de Microsoft Excel en un campo General, en lugar de iniciar una copia de Microsoft Excel en otra ventana, los nombres de los menús cambian para reflejar la estructura de menús de Microsoft Excel y se muestran las barras de herramientas predeterminadas de Microsoft Excel. El programador o el

se muestran las barras de herramientas predeterminadas de Microsoft Excel. El programador o el usuario de la aplicación podrá entonces modificar el objeto Excel sin abandonar la aplicación.

**Nota** Sólo puede modificar directamente los objetos incrustados, no los vinculados.

También puede abrir el servidor de Automatización en otra ventana, modificar los datos o ver las características en dicha ventana y hacer que los valores nuevos se reflejen en la aplicación cuando vuelva a ella.

### Para modificar un objeto OLE directamente en una ventana de campo General

- En el menú **Edición**, seleccione el tipo de objeto específico y, en el submenú, elija **Modificar**.

Por ejemplo, si el objeto es un documento de Word, seleccione el elemento de menú **Objeto documento**; si el objeto es un gráfico de Microsoft Graph, seleccione el elemento de menú **Objeto gráfico**.

—O bien—

- Haga doble clic en el objeto.

### Para abrir la aplicación para un objeto OLE en una ventana de un campo General

- En el menú **Edición**, seleccione el tipo de objeto específico y, en el submenú, elija **Abrir**.

Cuando agregue un objeto OLE a un formulario en el control [contenedor OLE](#) o el control [OLE dependiente](#), tendrá mayor control sobre la apertura y modificación del objeto.

Si establece la propiedad AutoActivate de un control contenedor OLE u OLE dependiente, puede determinar si el objeto OLE se abrirá o modificará cuando el control aparezca con el foco o cuando el usuario haga doble clic sobre el control. La propiedad AutoVerbMenu especifica si el menú contextual del control ActiveX permite al usuario abrir o modificar el objeto OLE.

Para controlar el acceso de modo que el objeto OLE sólo pueda abrirse o modificarse mediante programación con el método [DoVerb](#), establezca [AutoActivate](#) como 0 - Manual y [AutoVerbMenu](#) como falso (.F.) .

### Controlar los menús

Cuando un usuario modifica directamente un objeto OLE, aparece la barra de menús del objeto OLE y no los menús de la aplicación. Si crea un título de menú y desea que se muestre mientras el usuario modifica objeto OLE, seleccione **Negociar** en el cuadro de diálogo **Opciones de la acción** del Diseñador de menús. Para obtener más información, consulte el capítulo 11, [Diseñar menús y barras de herramientas](#) o la cláusula NEGOTIATE en el tema [DEFINE PAD](#) de la Ayuda.

## Usar controles ActiveX

Los controles ActiveX son objetos con funcionalidad encapsulada que ofrecen propiedades, eventos y métodos. Proporcionan una amplia gama de funciones que se puede utilizar fácilmente. Algunos de

métodos. Proporcionan una amplia gama de funciones que se puede utilizar fácilmente. Algunos de los controles ActiveX incluidos en Visual FoxPro son:

- Los controles de Windows 95, como RichText y los controles TreeView.
- Los controles del sistema, como los de comunicaciones y MAPI.

Los controles ActiveX son versátiles porque se pueden crear subclases para crear otros controles y se pueden controlar mediante los eventos, métodos y propiedades que tengan asociados. No puede crear controles ActiveX con Visual FoxPro; sin embargo, puede crearlos con Microsoft OLE Custom Control Developer's Kit suministrado con Microsoft Visual C++<sup>®</sup> 4.0 y con Microsoft Visual Basic<sup>®</sup> Control Creation Edition versión 5.0.

Para obtener más información sobre el acceso a controles ActiveX, consulte el capítulo 27, [Extender Visual FoxPro con bibliotecas externas](#). Para obtener más información sobre la creación de controles ActiveX específicos de Visual FoxPro, consulte el capítulo 28, [Acceso a la API de Visual FoxPro](#).

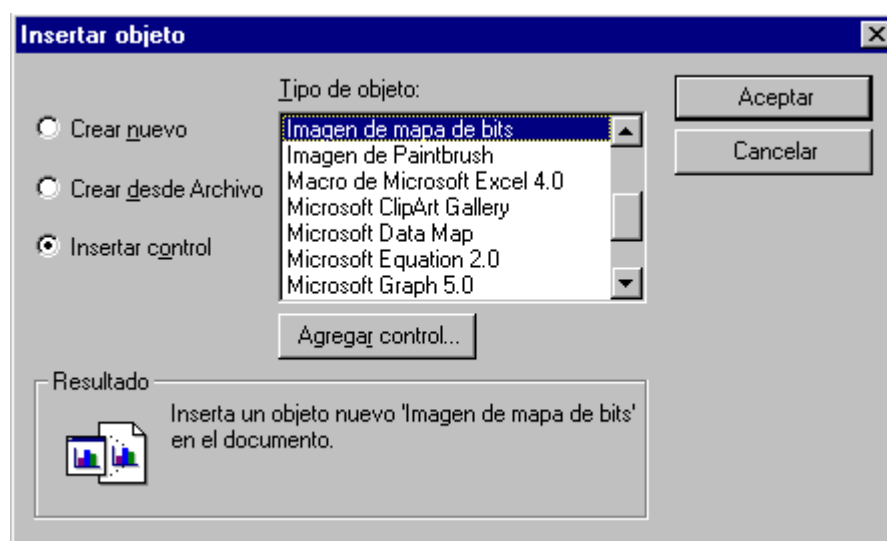
## Agregar controles ActiveX a un formulario

Los controles ActiveX de Visual FoxPro deben estar incluidos en un control contenedor OLE (la clase de base es OLEControl). Cuando se agrega un control Contenedor OLE a un formulario, puede elegir el control ActiveX que desea agregar al formulario.

### Para agregar un control ActiveX a un formulario

1. En la barra de herramientas **Controles de formularios**, elija [Control Contenedor OLE](#) y arrástrelo para incluirlo en el formulario.
2. En el cuadro de diálogo [Insertar objeto](#), elija **Insertar control**.

### Cuadro de diálogo Insertar objeto



3. En la lista **Tipo de control**, seleccione el control ActiveX que desee.

#### 4. Elija **Aceptar**.

### Administrar controles ActiveX dependientes

Si un control ActiveX admite la vinculación sencilla de datos, Visual FoxPro ofrecerá una propiedad ControlSource para el control. Todo lo que tiene que hacer es establecer la propiedad ControlSource en un campo de la tabla y el valor mostrado en el control ActiveX reflejará el valor del campo subyacente. Los cambios realizados en el valor del control se guardan en el campo.

Para obtener ejemplos sobre el uso de controles ActiveX, ejecute el archivo Solution.app ubicado en el directorio ...\\Samples\\Vfp98\\Solution de Visual Studio.

**Nota** Para asegurar que se procesan todos los eventos de controles ActiveX, establezca la propiedad [AutoYield](#) del objeto Application de Visual FoxPro como falsa (.F.).

### Manipular objetos con Automatización

Los objetos OLE de los formularios o programas, o los controles ActiveX dentro de los controles Contenedor OLE, pueden manipularse con código de la misma manera en que se programan los objetos de Visual FoxPro nativos.

### Manipular propiedades extrínsecas de objetos

En el código, puede manipular un objeto mediante sus propiedades. La manera en que se hace referencia a una propiedad depende de si el objeto es autónomo o forma parte de un contenedor, como el control [Contenedor OLE](#) u [OLE dependiente](#).

**Nota** Los controles ActiveX siempre forman parte de un objeto Contenedor OLE.

Un objeto de un contenedor consta de dos partes: el propio objeto y el contenedor alrededor del mismo. Tanto el objeto como el contenedor tienen propiedades y, a veces, tienen los mismos nombres de propiedades. Para asegurarse de que hace referencia a las propiedades del objeto, anexe siempre la propiedad [Object](#) del contenedor al nombre del objeto. Por ejemplo, el código siguiente hace referencia a la propiedad Left del objeto.

```
frm1.olecontrol1.Object.Left = 25  && Propiedad Left del objeto
```

Si se omite la propiedad Object, se hace referencia a la propiedad Left del contenedor.

```
frm1.olecontrol1.Left= 25  && Propiedad Left del contenedor
```

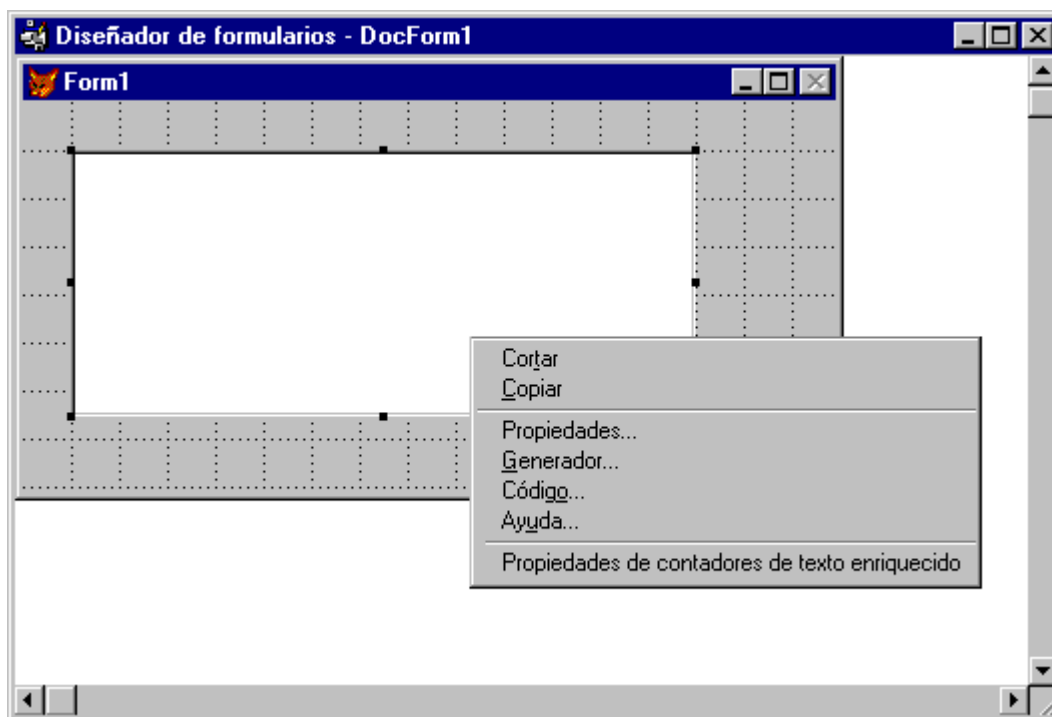
Por ejemplo, suponga que tiene una aplicación que envía correo cuando el usuario hace clic en un botón de comando Enviar. Si ha agregado un control de mensajes de Microsoft MAPI a un formulario como olecontrol1, el código asociado al evento Click del botón de comando podría ser:

```
THISFORM.olecontrol1.Object.Enviar  
THISFORM.olecontrol1.Object.Send(.T.)
```

Además de usar la propiedad `Object` para hacer referencia a las propiedades del objeto contenido, puede emplear otras propiedades del control contenedor. Por ejemplo, puede hacer referencia a la propiedad [OLEClass](#) de sólo lectura para identificar el tipo de objeto del contenedor y la propiedad [Sizable](#) para impedir que los usuarios puedan cambiar el tamaño de un objeto. Para obtener detalles sobre las propiedades del control contenedor, consulte [Control Contenedor OLE](#).

En los diseñadores de formularios y clases, las propiedades de los controles ActiveX se muestran en la ventana [Propiedades](#) de Visual FoxPro, pero la mayoría de los controles ActiveX también tienen su propia interfaz para establecer propiedades comunes. Puede ver esta interfaz de las propiedades si selecciona la opción `Propiedades específica del objeto` en el menú contextual del control ActiveX. Por ejemplo, para abrir el cuadro de diálogo `Propiedades` correspondiente a un control `RichText`, elija `Propiedades del control Microsoft RichText` en el menú contextual.

### Apertura del cuadro de diálogo `Propiedades` del control `RichText`



### Usar métodos extrínsecos de objetos

Además de establecer y recuperar las propiedades de los objetos, puede manipular un objeto usando los métodos que admite. Por ejemplo, puede usar el método `Add` de un objeto de colección de Microsoft Excel para crear un libro de trabajo de Excel.

El siguiente ejemplo de automatización OLE emplea el método Add para crear un libro de trabajo de Excel, el método Save para guardar el libro de trabajo y el método Quit para salir de Excel:

Código	Comentarios
<code>oleApp = CREATEOBJECT("Excel.Application")</code>	Inicia Excel.
<code>oleApp.Visible=.T.</code>	Muestra Excel.
<code>OleApp.Workbooks.Add</code>	Crea un libro de trabajo.
<code>OleApp.Cells(1,1).Value=7</code>	Establece el valor de la celda.
<code>OleApp.ActiveWorkbook.SaveAs("C:\TEMP.XLS")</code>	Guarda el libro.
<code>OleApp.Quit</code>	Sale de Excel.

Si crea un objeto con el control contenedor OLE o el control OLE dependiente, puede usar el método [DoVerb](#) del control para ejecutar un verbo en el objeto. Por ejemplo, use DoVerb(0) para ejecutar el verbo predeterminado, DoVerb( - 1) para activar el objeto para edición visual, y DoVerb( - 2) para abrir el objeto en una ventana distinta.

**Nota** Consulte la documentación de la aplicación para determinar qué comandos de Automatización se admiten. Por ejemplo, los complementos de Microsoft Excel no están disponibles para Automatización.

## Establecer tiempos de espera

Cuando se transfiere una solicitud a un objeto OLE, el servidor OLE la procesa. No tendrá mucho control sobre el proceso del servidor, pero podrá especificar el tiempo que puede esperar para que finalice el proceso si establece las propiedades [OLERequestPendingTimeout](#) y [OLEServerBusyTimeout](#). También podrá determinar lo que ocurrirá cuando haya transcurrido ese tiempo si establece la propiedad [OLEServerBusyRaiseError](#).

## Acceso a colecciones de objetos

Un tipo de objeto puede representar un único objeto o una colección de objetos relacionados. Por ejemplo, un objeto Workbook de Excel representa un único libro, mientras que el objeto Workbooks representa todos los libros actualmente cargados. Puesto que el objeto Workbooks representa una colección de objetos, se llama objeto de colección.

En código, una colección es una lista no ordenada en la que la posición de un objeto puede cambiar siempre que se agreguen o quiten objetos de la colección. Para tener acceso a un objeto de una colección debe iterar dentro de la colección mediante la propiedad Count de la misma. La propiedad Count devuelve el número de elementos de la colección. Además, puede usar el método Item para

devolver un elemento de la colección.

Por ejemplo, para mostrar los nombres de las hojas de cálculo en un libro de Excel, use el siguiente código:

```
oleApp = CREATEOBJECT("Excel.Application")
oleApp.Workbooks.Add
FOR EACH x IN oleApp.Workbooks
    ? x.Name
ENDFOR
```

También puede tener acceso a una colección dentro de otra colección. Por ejemplo, puede tener acceso a una colección de celdas dentro de un intervalo con el código siguiente:

```
oleApp = CREATEOBJECT("Excel.sheet")
oleApp.Workbooks.Add
oleApp.Range(oleApp.Cells(1,1),oleApp.Cells(10,10)).Value=100
oleApp.Visible=.T.
```

## Usar matrices de objetos

Puede transferir matrices a métodos y recibir matrices de vuelta. Sin embargo, debe transferir las matrices por referencia prefijando el nombre de la matriz con el signo @.

Por ejemplo, para transferir una matriz de Visual FoxPro a Microsoft Excel, considere el siguiente código. Crea una matriz en Visual FoxPro, asigna a la matriz algunos valores, inicia Microsoft Excel, crea un libro, establece un valor como la primera celda de un libro y entonces copia el valor a las otras hojas de la matriz:

```
DIMENSION aV(3)
aV(1) = "Hoja1"
aV(2) = "Hoja2"
aV(3) = "Hoja3"
oleApp=CREATEOBJECT("Excel.Application")
oleApp.Workbooks.Add
oleI=oleApp.Workbooks.Item(1)
oleI.Sheets.Item(1).Cells(1,1).Value = 83
oleI.Sheets(@aV).;
    FillAcrossSheets(oleI.Worksheets("Hoja1").Cells(1,1))

oleApp.Visible = .T.
```

Además, el siguiente ejemplo devuelve una matriz a Visual FoxPro y muestra el contenido de la matriz:

```
oleApp = CREATEOBJECT("Excel.Application")
aOleArray = oleApp.GetCustomListContents(3)
FOR nIndex = 1 to ALLEN(aOleArray)
    ? aOleArray(nIndex)
ENDFOR
```

**Nota** Con Visual FoxPro no puede transferir matrices de más de dos dimensiones a objetos OLE. Para obtener más información acerca de cómo trabajar con matrices en Visual FoxPro, consulte el capítulo 3, [Programación orientada a objetos](#) y [Descripción general del lenguaje](#).



## Liberar objetos extrínsecos

Un servidor de Automatización se libera automáticamente si no está visible y si no hay variables en el ámbito que hagan referencia al objeto. Puede usar el comando [RELEASE](#) para liberar la variable asociada a un objeto. Si el servidor está visible, use el método [Quit](#) para liberarlo.

## Crear subclases de objetos

Puede crear objetos personalizados creando subclases de las clases de base incluidas en Visual FoxPro. Por ejemplo, el código siguiente crea una subclase del control Outline suministrado con Visual FoxPro:

### Creación de una subclase del control Outline

Código	Comentarios
<pre>PUBLIC frmMyForm, cFilename SET SAFETY OFF</pre>	Declara variables y las inicializa.
<pre>frmMyForm = CREATEOBJECT("form") frmMyForm.Width = 100 frmMyForm.ADDOBJECT("oleOut1","myoutline") DIMENSION aSection(3) aSection(1) = "Table" aSection(2) = "Field" aSection(3) = "Index"</pre>	Crea un formulario, agrega el control Outline personalizado al formulario y después crea una matriz para los elementos que presenta el control.
<pre>cFilename = GETFILE("dbc","Seleccione un archivo DBC") USE (cFilename) INDEX ON objecttype FOR (objecttype = "Table" ;     OR objecttype = "Field" ;     OR objecttype = "Index" ) ; TAG fname</pre>	Solicita una base de datos que contiene la información que va a presentar el control.
<pre>FOR nIndex = 1 TO 3 STEP 1     frmMyForm.oleOut1.AddItem(aSection(nIndex))     frmMyForm.oleOut1.Indent;     ((frmMyForm.oleOut1.ListCount-1)) = 1     SCAN         IF objecttype = aSection(nIndex)             frmMyForm.oleOut1.Additem(objectname)             frmMyForm.oleOut1.Indent;             ((frmMyForm.oleOut1.ListCount-1)) = 2         ENDIF     ENDSCAN GO TOP ENDFOR</pre>	Recopila información de la base de datos y la agrega al control.
<pre>frmMyForm.oleOut1.Visible = .T. frmMyForm.Show</pre>	Deja el control visible y muestra el formulario.

```

DEFINE CLASS myoutline AS olecontrol
    OleClass = "msoutl.outline"
    Top = 5
    Left = 5
    Height = 10
    Width = 60
ENDDEFINE

```

Define una subclase del control Contenedor OLE y agrega el control Outline estableciendo la propiedad OleClass del contenedor y definiendo los otros valores personalizados.

Si desea distribuir las aplicaciones, hay algunas otras consideraciones. Para obtener más información, consulte el capítulo 25, "[Generar una aplicación para su distribución](#)".

## Controlar Visual FoxPro desde otras aplicaciones

Dado que Visual FoxPro actúa como servidor (con conformidad de nivel 2) y como cliente, las aplicaciones que admiten Automatización pueden crear instancias de Visual FoxPro, ejecutar comandos de Visual FoxPro y tener acceso a sus objetos.

Puede incluso manipular Visual FoxPro desde aplicaciones que no admitan Automatización mediante [Fpole.dll](#)

Puede controlar Visual FoxPro desde otras aplicaciones mediante el objeto Application de Visual FoxPro. Se crea automáticamente un objeto Application siempre que se inicia Visual FoxPro directamente, mediante DDE o mediante Automatización.

Por ejemplo, las líneas de código siguiente en Visual Basic®, o en un módulo de Microsoft Excel, crean una referencia a un objeto Application de Visual FoxPro:

```

Dim oFox as Object
Set oFox = CreateObject("VisualFoxPro.Application")

```

Una vez establecida una referencia al objeto Application de Visual FoxPro, puede llamar a métodos asociados con el objeto y tener acceso a otros objetos mediante las propiedades de la colección del objeto Application.

### Métodos del objeto Application

[DataToClip](#)

[Help](#)

[DoCmd](#)

[Quit](#)

[Eval](#)

[RequestData](#)

El ejemplo siguiente usa Visual Basic para código de aplicaciones en un módulo Excel para crear un objeto Application de Visual FoxPro, abrir una tabla de Visual FoxPro y agregar los resultados de una consulta a la hoja de cálculo activa:

```

Sub FoxTest()
Dim oFox as Object
Set oFox = CreateObject("VisualFoxPro.Application")

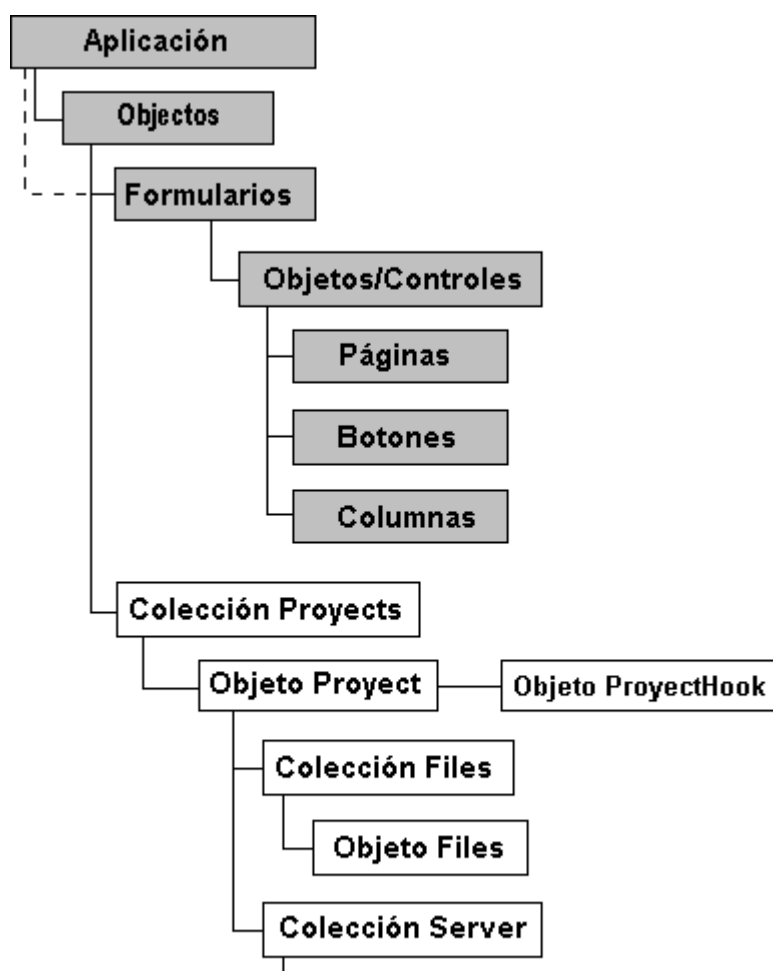
```

```
oFox.DoCmd "USE customer"
oFox.DoCmd "SELECT contact, phone FROM customer
    WHERE country = " + Chr$(39) + USA+ Chr$(39) + " INTO CURSOR cust"
oFox.DataToClip "cust",,3
Range("A1:B1").Select
ActiveSheet.Paste
End Sub
```

## Modelo de objeto Application de Visual FoxPro

Cada vez que inicia Visual FoxPro se crea un objeto Application automáticamente, ya sea directamente, a través de Automatización o de DDE. Este objeto Application proporciona acceso a todos los objetos creados en una sesión de Visual FoxPro a través de las propiedades de la colección.

### Modelo de objeto Application de Visual FoxPro



## Objeto Server

### Acceso a objetos a través de las propiedades Collection

El objeto Application de Visual FoxPro y todos los objetos contenedores de Visual FoxPro tienen asociada una propiedad Count y una propiedad Collection. La propiedad Collection es una matriz que hace referencia a cada objeto contenido. La propiedad Count es una propiedad numérica que indica el número de objetos contenidos.

En la tabla siguiente se muestran los objetos y la propiedades Collection y Count correspondientes.

Objeto	Propiedad Collection	Propiedad Count
<a href="#">Application</a>	<a href="#">Objects</a> <a href="#">Forms</a>	<a href="#">Count</a> <a href="#">FormCount</a>
<a href="#">FormSet</a>	<a href="#">Forms</a>	<a href="#">FormCount</a>
<a href="#">Form</a>	<a href="#">Objects</a> <a href="#">Controls</a>	<a href="#">Count</a> <a href="#">ControlCount</a>
<a href="#">PageFrame</a>	<a href="#">Pages</a>	<a href="#">PageCount</a>
<a href="#">Page</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Grid</a>	<a href="#">Columns</a>	<a href="#">ColumnCount</a>
<a href="#">CommandGroup</a>	<a href="#">Buttons</a>	<a href="#">ButtonCount</a>
<a href="#">OptionGroup</a>	<a href="#">Buttons</a>	<a href="#">ButtonCount</a>
<a href="#">Column</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">ToolBar</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Container</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>
<a href="#">Control</a>	<a href="#">Controls</a>	<a href="#">ControlCount</a>

Estas propiedades permiten el uso de un bucle para manipular mediante programación todos los objetos contenidos o sólo los especificados. Por ejemplo, las líneas de código siguientes establecen la propiedad Visible de todos los formularios como verdadera (.T.):

```
FOR EACH Form IN Application.Forms
    Form.Visible = .T.
ENDFOR
```

### Crear servidores de Automatización

Con Visual FoxPro puede crear servidores de Automatización (componentes COM) que empaquetan el código para realizar tareas comunes a muchas aplicaciones o que implementan reglas comerciales complejas. Estas tareas y reglas están disponibles para otros programadores de la empresa y para los

usuarios de herramientas que admitan la automatización.

Por ejemplo, podría crear una o varias clases para controlar las reglas comerciales de toda la compañía. Una aplicación cliente que utilice los objetos de regla comercial transferiría los parámetros de entrada en una llamada a un método y el servidor de Automatización podría estar demasiado ocupado transfiriendo datos de varios orígenes y realizando cálculos completos antes de devolver la respuesta.

Los ejemplos de servidores de Automatización están instalados en el directorio ...\\Samples\\Vfp98\\Servers de Visual Studio.

## Crear el servidor

Todo lo que necesita para crear un servidor de Automatización en Visual FoxPro es un proyecto que contenga clases definidas como OLEPUBLIC. Puede tener en el proyecto tantas clases OLEPUBLIC como desee y se pueden definir en archivos de programa (.prg) o en bibliotecas de clases (.vcx).

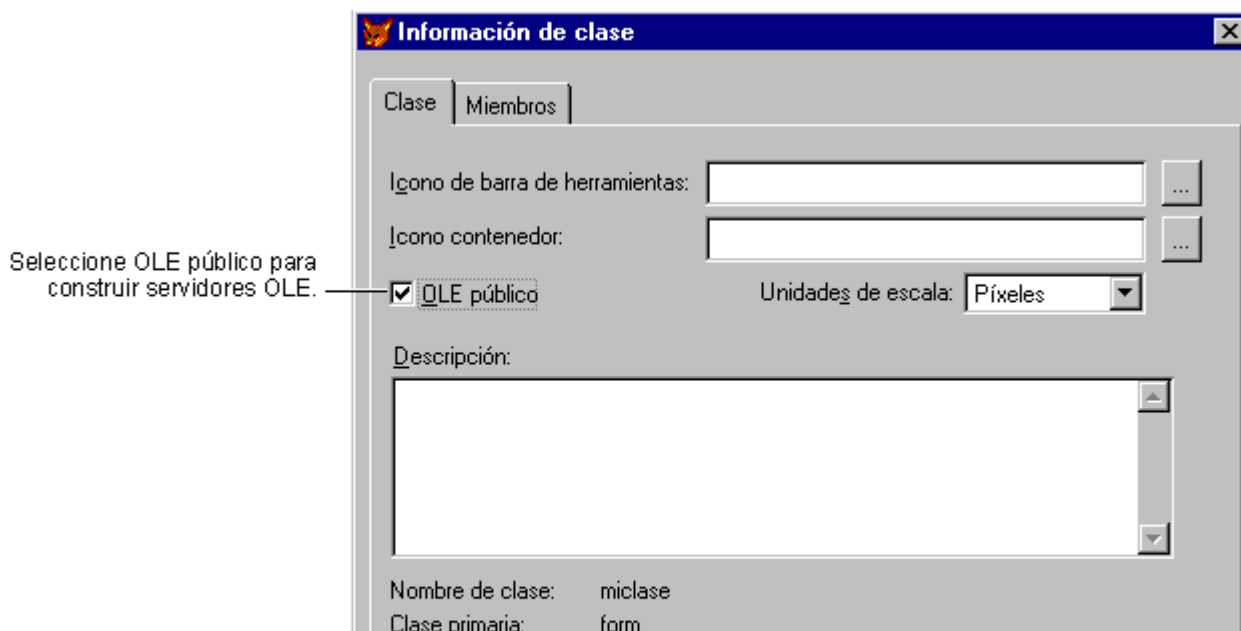
Por ejemplo, la definición de clase siguiente en un archivo de programa crea una clase pública OLE personalizada:

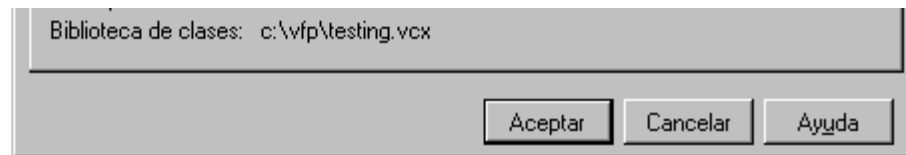
```
DEFINE class person AS CUSTOM OLEPUBLIC
    FirstName = SPACE(30)
    LastName = SPACE(45)

    PROCEDURE GetName
        RETURN THIS.FirstName + " " + THIS.LastName
    ENDPROC
ENDDDEFINE
```

Cuando diseñe una clase en el Diseñador de clases, seleccione **OLE público** en el cuadro de diálogo [Información de clase](#) para designar la clase como OLEPUBLIC.

### Cuadro de diálogo Información de clase





## Compilación del servidor

En Visual FoxPro puede crear un servidor de Automatización [fuera de proceso](#) o [en proceso](#). Un componente *fuera de proceso* es un ejecutable (archivo .exe) que ejecuta su propio proceso. La comunicación entre una aplicación cliente y un servidor fuera de proceso se denomina comunicación *entre procesos*. Un componente *en proceso* es una biblioteca de vínculos dinámicos (dll) que se ejecuta en el mismo espacio de direcciones de proceso que el cliente que la llama.

Cada componente tiene sus propias ventajas. Un servidor en proceso es más rápido porque no hay necesidad de comunicación entre procesos. Por otro lado, un servidor fuera de proceso puede utilizarse en modo remoto, función no disponible en el servidor en proceso. Además, dado que el servidor en proceso y el cliente comparten un espacio de direcciones de proceso, cualquier error grave en el archivo .dll interrumpirá el funcionamiento del cliente, mientras que un error en un .exe fuera de proceso sólo interrumpiría el servidor.

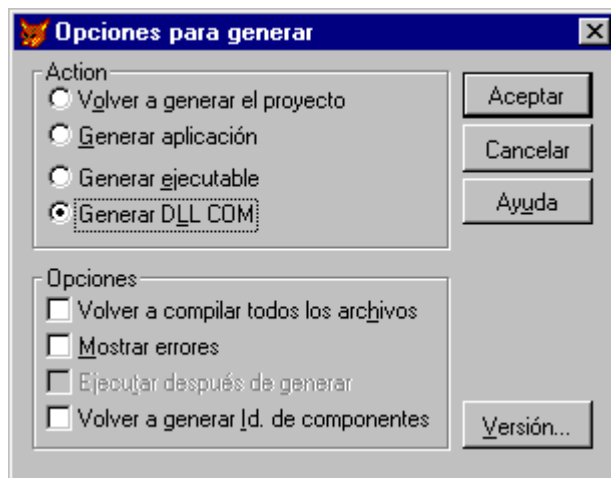
Cuando se crea un ejecutable con clases OLE públicas, no se pierde ninguna de las capacidades .exe habituales. Podrá hacer funcionar el ejecutable, proporcionar una interfaz de usuario y todas las funciones habituales que incluiría en una aplicación. Puede aumentar, por tanto, la capacidad de expansión de la aplicación permitiendo que otras aplicaciones exploten las funciones específicas que desea ofrecer.

**Nota** Si hay más de un usuario que intenta tener acceso al servidor de Automatización, pueden surgir conflictos. Si ha proporcionado acceso a Automatización además de una interfaz de usuario para la funcionalidad, proporcione un nivel adicional de coherencia comprobando la interfaz para garantizar que el entorno no se ha modificado.

## Para compilar un servidor de Automatización

1. En el **Administrador de proyectos**, elija **Generar**.
2. En el cuadro de diálogo [Opciones de generación](#), elija **Generar ejecutable** o **Generar DLL OLE**.

### Cuadro de diálogo Opciones de generación



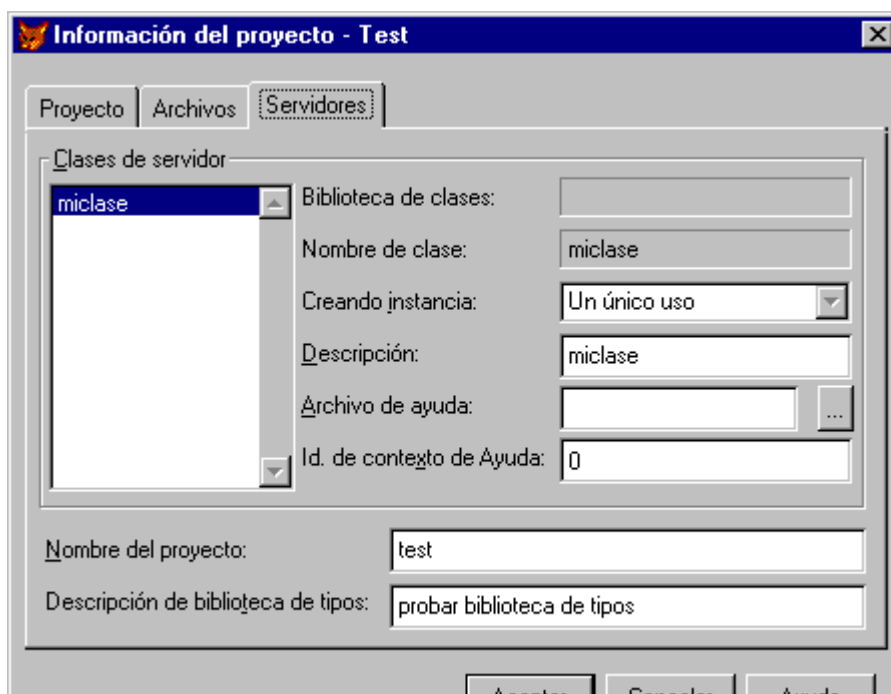
#### 3. Elija **Aceptar**.

–O bien–

- Use los comandos [BUILD DLL](#) o [BUILD EXE](#).

Una vez creado el proyecto, puede ver las clases de servidor mostradas en el cuadro de diálogo [Información del proyecto](#), donde también puede especificar un archivo de ayuda y un identificador del contexto en la Ayuda para cada clase. Este archivo de ayuda puede abrirse desde los examinadores de objetos más genéricos.

### Cuadro de diálogo Información del proyecto





Puede elegir valores de creación de instancias específicos de la clase en el cuadro de diálogo Información del proyecto. Las opciones de creación de instancias son las siguientes:

- **No se puede crear** Aún cuando la clase esté marcada como OLE pública, no estará disponible en otras aplicaciones. Por ejemplo, podría tener una biblioteca estándar de clases OLE públicas en múltiples aplicaciones y desactivar la automatización de una o varias clases para una sola aplicación.
- **Uso único** Todas las aplicaciones cliente que utilicen el servidor crean una instancia diferente de la clase de servidor. Cada instancia tiene un solo subproceso de ejecución. Aunque instancias independientes requieren más memoria, si se elige Uso único el sistema operativo podrá aplicar multitarea de asignación prioritaria.
- **Multiuso** Una vez creado el servidor, otras aplicaciones pueden utilizar la misma instancia.

**Nota** Si realiza cambios en la ficha Servidores del cuadro de diálogo Información del proyecto, será necesario volver a generar la .dll o el .exe para que entre en vigor la nueva configuración.

Cuando cree un proyecto con clases OLE públicas, se crean tres archivos:

- El archivo .dll o .exe
- Un archivo de biblioteca de tipos (.tlb)
- Un archivo de registro (.vbr).

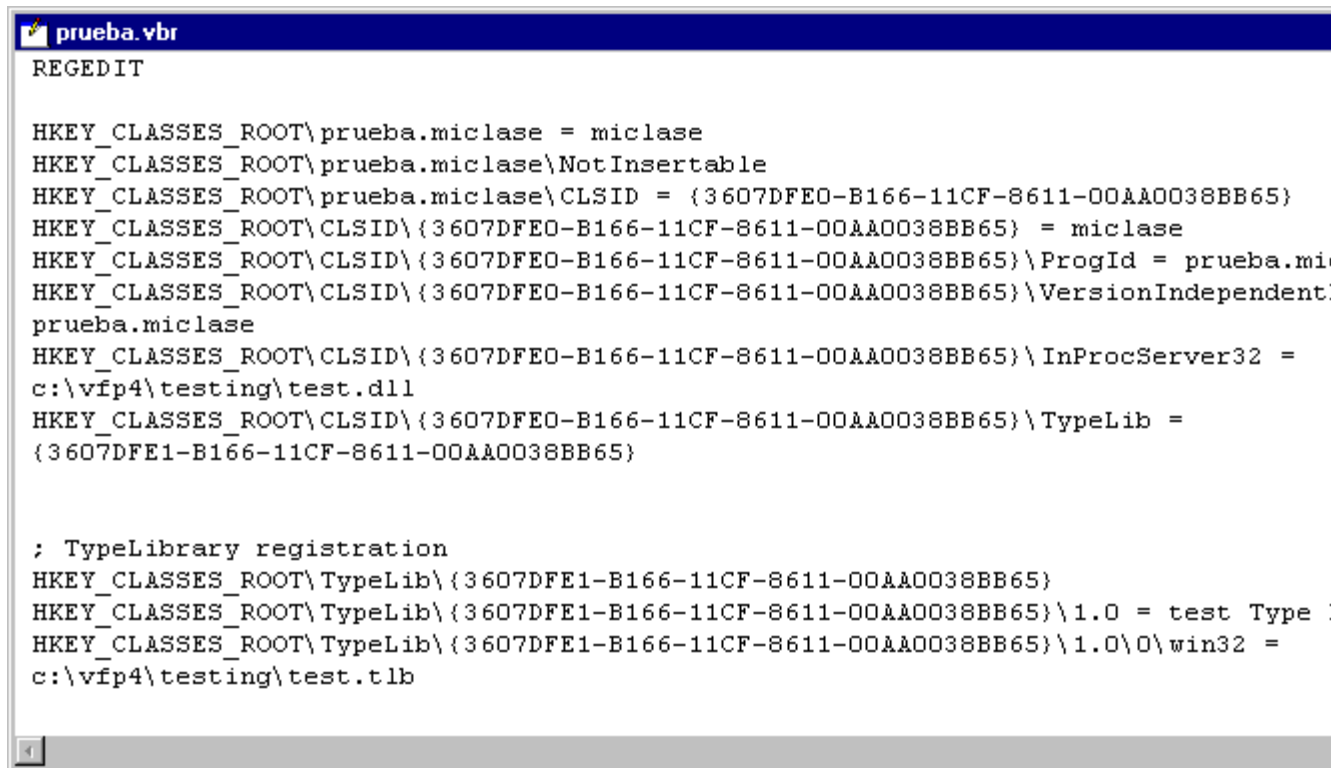
El archivo de biblioteca de tipos es un archivo binario que presenta una relación de todas las clases publicadas en el servidor de Automatización, junto con sus propiedades, métodos y eventos. Los examinadores de objetos OLE leen esta información y la presentan en una interfaz legible.

El archivo de registro muestra [identificadores únicos globales](#) (GUID) de las clases del servidor.

**Nota** Un archivo de registro .vbr es igual que un archivo .reg, salvo que el primero no incluye rutas de acceso en código.

### **Archivo .VBR con GUID para cada clase OLE pública de un proyecto**





```
prueba.vbr
REGEDIT

HKEY_CLASSES_ROOT\prueba.miclasa = miclasa
HKEY_CLASSES_ROOT\prueba.miclasa\NotInsertable
HKEY_CLASSES_ROOT\prueba.miclasa\CLSID = {3607DFE0-B166-11CF-8611-00AA0038BB65}
HKEY_CLASSES_ROOT\CLSID\{3607DFE0-B166-11CF-8611-00AA0038BB65} = miclasa
HKEY_CLASSES_ROOT\CLSID\{3607DFE0-B166-11CF-8611-00AA0038BB65}\ProgId = prueba.mi
HKEY_CLASSES_ROOT\CLSID\{3607DFE0-B166-11CF-8611-00AA0038BB65}\VersionIndependent:
prueba.miclasa
HKEY_CLASSES_ROOT\CLSID\{3607DFE0-B166-11CF-8611-00AA0038BB65}\InProcServer32 =
c:\vfp4\testing\test.dll
HKEY_CLASSES_ROOT\CLSID\{3607DFE0-B166-11CF-8611-00AA0038BB65}\TypeLib =
{3607DFE1-B166-11CF-8611-00AA0038BB65}

; TypeLibrary registration
HKEY_CLASSES_ROOT\TypeLib\{3607DFE1-B166-11CF-8611-00AA0038BB65}
HKEY_CLASSES_ROOT\TypeLib\{3607DFE1-B166-11CF-8611-00AA0038BB65}\1.0 = test Type
HKEY_CLASSES_ROOT\TypeLib\{3607DFE1-B166-11CF-8611-00AA0038BB65}\1.0\0\win32 =
c:\vfp4\testing\test.tlb
```

## Registrar un servidor de Automatización

Los servidores OLE quedarán disponibles para otras aplicaciones cuando se hayan agregado los servidores al Registro de Windows. Cuando genere un servidor de Automatización, se registrará automáticamente en la máquina donde se ha efectuado la generación. También puede registrar los servidores en otras máquinas.

Cuando utilice el [Asistente para instalación](#) de Visual FoxPro para crear discos de instalación, el programa de instalación registrará los servidores en los equipos de los clientes. También puede registrar servidores manualmente.

### Para registrar un componente .exe

- Ejecute el archivo .exe con el modificador **/regserver**.

Por ejemplo, para registrar Miservid.exe, ejecute el comando siguiente:

```
miservid /regserver
```

### Para quitar una entrada del registro de componentes .exe

- Ejecute el archivo .exe con el modificador **/unregserver**.

Por ejemplo, para eliminar Miservid.exe del registro, ejecute el comando siguiente:

```
miservid /unregserver
```

### Para registrar un componente .dll

- Ejecute REGSVR32 con el nombre del servidor.

Por ejemplo, para registrar Miservid.dll, ejecute el comando siguiente:

```
REGSVR32 miservid.dll
```

### Para quitar una entrada del registro de componentes .dll

- Ejecute REGSVR32 con el nombre del servidor y el parámetro **/u**.

Por ejemplo, para registrar Miservid.dll, ejecute el comando siguiente:

```
REGSVR32 /u miservid.dll
```

**Nota** El registro contiene el nombre completo de la ruta de acceso al archivo, por lo que si mueve el archivo tendrá que registrarlo de nuevo.

### Usar el servidor de Automatización

Una aplicación que puede crear objetos de Automatización puede crear objetos basados en el servidor de Automatización, establecer propiedades que no estén **HIDDEN** o **PROTECTED** y llamar a métodos. Por ejemplo, suponiendo que el nombre del servidor sea *foxole* y que contenga una clase denominada *persona* con un método *GetName*, el código siguiente podría ejecutarse en Visual FoxPro 3.0:

```
oTest = CREATEOBJECT("foxole.persona")  
cName = oTest.GetName()
```

Se podría crear código similar en Microsoft Excel o Visual Basic:

```
Set oTest = CreateObject("foxole.person")  
cName$ = oTest.GetName()
```

### Detener o devolver errores de los servidores de Automatización

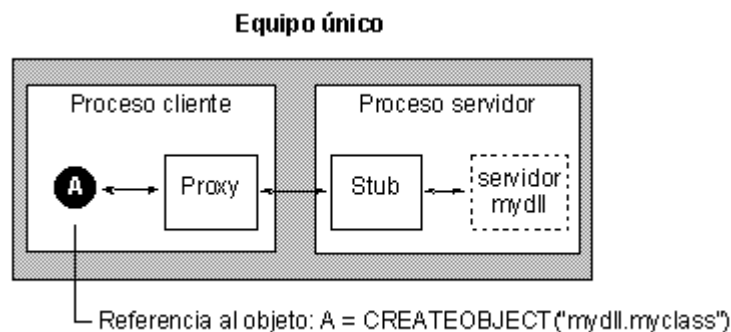
La única interacción con los objetos proporcionados por un servidor de Automatización (componente COM) se hace mediante los métodos y las propiedades de las clases expuestas. Cuando una aplicación cliente llama a un método de un objeto y se produce un error en el servidor de Automatización, el método devuelve un valor de error o surge un error en la aplicación cliente.

La aplicación cliente decide si debe avisar al usuario o continuar con otra ruta de acceso de ejecución. El propio servidor de Automatización nunca interactúa con el usuario. Esto permite que la aplicación del servidor de Automatización sea transparente para la aplicación cliente. El servidor de Automatización puede ser local (es decir, se ejecuta en el equipo del usuario) o se puede usar la característica de Automatización remota de Visual FoxPro para ejecutarlo en un servidor de red.

## Usar Automatización remota

En las situaciones habituales de Automatización, tanto el cliente como el servidor se encuentran en un solo equipo y comparten los mismos recursos, como la memoria y el procesador.

### Automatización en un único equipo

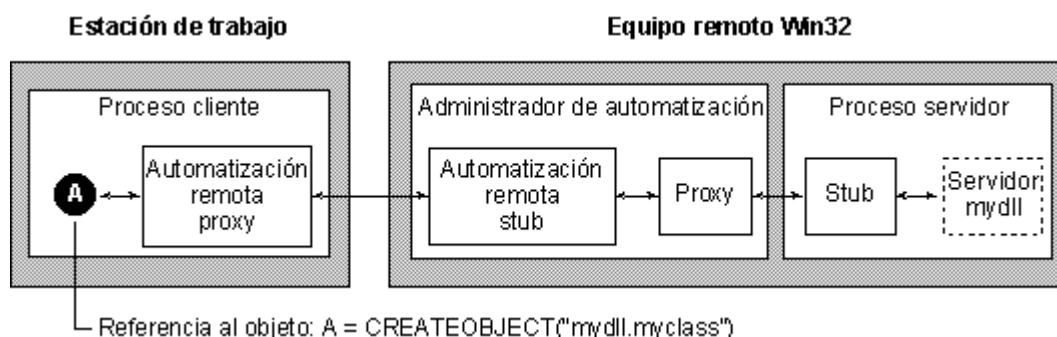


Cuando se crean servidores locales para Automatización, puede utilizarlos en modo remoto. La automatización remota permite la misma flexibilidad, capacidad de ampliación y eficacia que la automatización local, pero en una red. La Automatización remota permite que:

- Los servidores utilicen recursos diferentes.
- Muchos usuarios diferentes tengan acceso al mismo servidor.

Puede configurar un servidor y un equipo local para la automatización remota con el Administrador de conexiones de automatización remota, con lo que se guarda la configuración en el registro. Al ejecutarse el Administrador de Automatización en el equipo servidor, se administra la automatización de modo que el mismo código que manipula un objeto local pueda manipular automáticamente un objeto remoto.

### Automatización remota



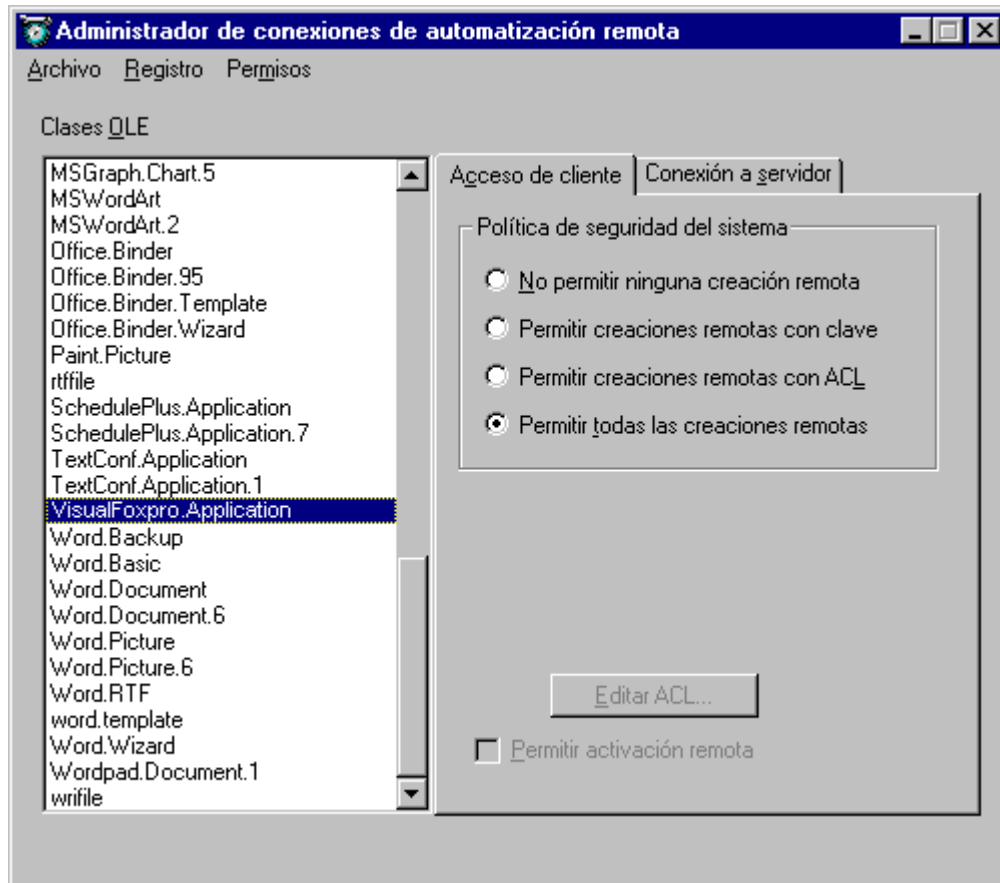
### Configurar el servidor

El primer paso en la activación de la automatización remota es configurar el equipo servidor para el acceso del cliente en el Administrador de conexiones de automatización remota.

Para configurar el servidor de automatización remota:

**Para configurar el servidor de automatización remota**

1. Copie el archivo ejecutable del servidor de Automatización (.EXE) en el servidor y ejecútelo una vez para registrarlo en el Registro de Windows.
2. En el equipo servidor, ejecute Racmgr32.exe, el Administrador para conexiones.



3. Seleccione la clase en la lista **Clases OLE**.
4. En la ficha **Acceso de cliente**, elija **Permitir creaciones remotas mediante clave**.
5. En la ficha **Acceso de cliente**, compruebe que la opción **Permitir activación remota** esté activada.

Una vez activado el acceso al cliente en el Administrador de conexiones de automatización remota, ejecute el Administrador de automatización, Autmgr32.exe, en el equipo servidor. Autmgr32.exe está instalado en la carpeta System de Windows 95 o en la carpeta System32 de Windows NT. Esto permitirá las conexiones de automatización remota desde otros equipos.

**Configurar el cliente**

Una vez configurado el equipo servidor, podrá configurar el equipo cliente.

**Para configurar el equipo local para la automatización remota**

1. Copie en la máquina cliente el archivo .vbr que se creó cuando se creó el servidor de Automatización.
2. Ejecute CLIREG32 con el nombre del archivo .vbr. Por ejemplo, si el archivo es MiServid.VBR, ejecute el comando siguiente en el símbolo del sistema:

```
CLIREG32 Miservid.vbr
```

3. En el cuadro de diálogo que se abre, introduzca la dirección de red de la máquina servidor y elija el protocolo de red (normalmente TCP/IP).

## Opciones de política de seguridad del sistema

En la tabla siguiente se describen las opciones del área Política de seguridad del sistema de la ficha Acceso de cliente del Administrador de conexiones de automatización remota.

Nombre	Valor	Descripción
No permitir ninguna creación remota	0	No permite la creación de ningún objeto.
Permitir creaciones remotas por índice	2	Sólo puede crearse un objeto si está activada la casilla de verificación Permitir activación remota. Esto modifica su CLSID en el Registro de Windows para incluir la configuración de subclaves siguiente: AllowRemoteActivation = Y
Permitir creaciones remotas con ACL	3	Un usuario puede crear un objeto sólo si el usuario está incluido en la lista de control de acceso del CLSID del Registro de Windows. Sólo para Windows NT.
Permitir todas las creaciones remotas	1	Permite la creación de cualquier objeto. No se recomienda fuera el entorno de programación.

<sup>1</sup>La columna Valor presenta la configuración preferente de RemoteActivationPolicy del Administrador de automatización en el Registro de Windows.

## Usar la autenticación en Automatización remota

Para los servidores de Automatización remotos que se ejecuten en cualquier sistema operativo Windows, el [procedimiento de llamada remota](#) (RPC) ofrece los niveles de autenticación siguientes.

Nombre	Valor	Descripción
Ninguno	0	Usar el valor predeterminado de la red.
Ninguno	1	Sin autenticación.
Conectar	2	Se autentica la conexión al servidor.

Llamar	3	Se autentica sólo al comienzo de cada llamada a procedimiento remota, cuando el servidor recibe la solicitud. No se aplica a las secuencias de protocolo basadas en la conexión (las que comienzan con el prefijo "ncacn").
Paquete	4	Comprueba que todos los datos recibidos proceden del cliente esperado.
Integridad del paquete	5	Comprueba que no se ha modificado ningún dato transferido entre el cliente y el servidor.
Privacidad del paquete	6	Comprueba todos los niveles anteriores y codifica los valores de argumentos de cada llamada a procedimiento remoto.

La necesidad de autenticación RPC debería evaluarse detenidamente, porque a medida que aumenta el nivel de autenticación RPC, disminuye el rendimiento. Puede especificar un nivel de autenticación para cada clase del servidor de Automatización, de modo que tanto los niveles costosos, como la codificación, no necesiten aplicarse a todo el componente.

Por ejemplo, un servicio de datos implementado como servidor de Automatización remoto podría tener una clase Logon utilizada para transmitir la información del usuario y la contraseña y esta clase podría requerir la autenticación Privacidad del paquete. Otras clases expuestas por el servidor podrían utilizar un nivel de autenticación más bajo.

## Solucionar problemas de Automatización remota

A continuación se presentan varias sugerencias en caso de que surjan dificultades.

Problema	Acción
Código de error OLE 0x800706d9: No hay más puntos finales disponibles en el administrador de puntos finales.	Compruebe que el Administrador de Automatización se está ejecutando en el equipo servidor y que el nombre de este equipo se haya introducido correctamente en el cuadro Dirección de la red del Administrador de conexiones de automatización remota.
Visual FoxPro: la aplicación no aparece en la lista de Clases OLE del Administrador de automatización remota.	<ol style="list-style-type: none"> <li>1. Ejecute Regedit.exe para abrir el registro.  vfp6.exe -r</li> <li>2. Elimine todas las referencias a Microsoft Visual FoxPro.</li> <li>3. Ejecute Visual FoxPro con la etiqueta -r en la línea de comandos:</li> </ol>

---

## Capítulo 17: Programar para acceso compartido

Si crea una aplicación que se va a ejecutar en varios equipos de un entorno de red o en la que varias instancias de un formulario tendrán acceso a los mismos datos, deberá programar para acceso compartido. Acceso compartido significa proporcionar modos eficaces de utilizar y compartir datos entre usuarios, así como de restringir el acceso cuando sea necesario.

Visual FoxPro proporciona soporte para acceso compartido o exclusivo a datos, opciones de bloqueo, sesiones de datos, almacenamiento de registros y tablas en búfer, y transacciones. Aunque estas características son especialmente útiles en entornos compartidos, también puede utilizarlas en entornos de un solo usuario.

En este capítulo se tratan los temas siguientes:

- [Controlar el acceso a datos](#)
- [Actualizar datos](#)
- [Administrar conflictos](#)

### Controlar el acceso a datos

Puesto que los datos se encuentran en los archivos, la administración eficaz de los datos comienza con el control del entorno de estos archivos. Debe elegir cómo tener acceso a los datos y cómo y cuándo limitar ese acceso.

#### Acceso a datos

En un entorno compartido, hay dos formas de tener acceso a datos: desde archivos exclusivos o desde archivos compartidos. Si abre un tabla para acceso compartido, otros usuarios también tendrán acceso al archivo. Si abre una tabla para acceso exclusivo, ningún otro usuario podrá leer o escribir en ese archivo. Puesto que el uso exclusivo elimina muchas de las ventajas que implica compartir datos en una red, deberá hacerse un uso comedido de esta característica.

#### Usar tablas con acceso exclusivo

La forma más restrictiva de abrir un archivo consiste en abrirlo de forma exclusiva. Cuando abra una tabla a través de la interfaz, se abrirá para uso exclusivo de forma predeterminada. También puede abrir explícitamente una tabla para uso exclusivo mediante comandos de Visual FoxPro.

#### Para abrir una tabla para uso exclusivo

- Escriba los comandos siguientes en la ventana **Comandos**:

```
SET EXCLUSIVE ON  
USE cMiTabla
```

–O bien–

- Escriba el comando siguiente en la ventana **Comandos**:

```
USE cMiTabla EXCLUSIVE
```

Los comandos siguientes requieren que abra una tabla para uso exclusivo:

- [ALTER TABLE](#)
- [INDEX](#) al crear, agregar o eliminar una etiqueta de índice compuesto.
- [INSERT \[BLANK\]](#)
- [MODIFY STRUCTURE](#): si utiliza este comando para cambiar la estructura de una tabla, deberá abrir la tabla de forma exclusiva. Sin embargo, podrá utilizar este comando en modo de sólo lectura cuando abra la tabla para uso compartido.
- [PACK](#)
- [REINDEX](#)
- [ZAP](#)

Visual FoxPro devolverá el error "Se requiere apertura exclusiva del archivo" si intenta ejecutar uno de estos comandos en una tabla compartida.

Puede restringir el acceso a una tabla mediante la función [FLOCK\(\)](#). Si utiliza [FLOCK\(\)](#) para bloquear la tabla, los demás usuarios no podrán escribir en ella, aunque sí podrán leerla.

### Usar tablas con acceso compartido

Cuando abra una tabla para uso compartido, varias estaciones de trabajo podrán utilizar la misma tabla a la vez. Cuando abra una tabla a través de la interfaz, podrá anular la configuración predeterminada [EXCLUSIVE ON](#). Puede abrir explícitamente una tabla para uso compartido mediante los comandos de Visual FoxPro.

### Para abrir una tabla para uso compartido

- Escriba los comandos siguientes en la ventana **Comandos**:

```
SET EXCLUSIVE OFF  
USE cMiTabla
```

–O bien–

- Escriba los comandos siguientes en la ventana **Comandos**:

```
USE cMiTabla SHARED
```

Cuando agregue o cambie datos en una tabla compartida, en primer lugar deberá bloquear el registro afectado o toda la tabla. Hay varias formas de bloquear un registro o una tabla abierta para uso compartido:



- Utilice un comando que realice un bloqueo automático de registro o tabla. Consulte la tabla de comandos que realizan bloqueo automático en la sección [Elegir bloqueos automáticos o manuales](#).
- Bloquee manualmente uno o varios registros, o una tabla completa, mediante las funciones de bloqueo de registro o de tabla.
- Inicie el almacenamiento en búfer mediante la función [CURSORSETPROP\(\)](#).

Los archivos memo y de índice asociados siempre se abren con el mismo estado compartido que su tabla.

Si la aplicación sólo utiliza una tabla con fines de consulta y todos los usuarios de la aplicación tienen acceso a la misma, podrá mejorar el rendimiento de la tabla si la marca como de sólo lectura.

## Bloquear datos

Si comparte el acceso a los archivos, también deberá administrar el acceso a los datos bloqueando tablas y registros. Los bloqueos, a diferencia de los permisos de acceso, pueden proporcionar un control de los datos a largo y a corto plazo. Visual FoxPro proporciona tanto bloqueo automático como manual.

### Elegir bloqueos de registro o de tabla

El bloqueo de registro, tanto si es automático como si es manual, impide que un usuario escriba en un registro en el que está escribiendo otro usuario. El bloqueo de tabla impide que otros usuarios escriban en la tabla, aunque pueden leerla. Puesto que el bloqueo de tabla prohíbe a otros usuarios actualizar los registros de una tabla, deberá hacer un uso comedido de esta característica.

### Elegir entre bloqueo automático y manual

Además del bloqueo de registro o de tabla, también puede elegir entre bloqueo automático y manual. Muchos comandos de Visual FoxPro intentan bloquear automáticamente un registro o una tabla antes de que se ejecute el comando. Si el registro o la tabla están bloqueados correctamente, el comando se ejecutará y se liberará el bloqueo.

### Comandos que bloquean automáticamente registros y tablas

Comando	Alcance del bloqueo
<a href="#">ALTER TABLE</a>	Toda la tabla
<a href="#">APPEND</a>	Encabezado de la tabla
<a href="#">APPEND BLANK</a>	Encabezado de la tabla
<a href="#">APPEND FROM</a>	Encabezado de la tabla
<a href="#">APPEND FROM ARRAY</a>	Encabezado de la tabla
<a href="#">APPEND MEMO</a>	Registro actual

<a href="#">APPEND MEMO</a>	Registro actual
<a href="#">BLANK</a>	Registro actual
<a href="#">BROWSE</a> , <a href="#">CHANGE</a> y <a href="#">EDIT</a>	Registro actual y todos los registros de campos con alias de tablas relacionadas cuando comienza la edición de un campo
<a href="#">CURSORSETPROP()</a>	Depende de los parámetros
<a href="#">DELETE</a>	Registro actual
<a href="#">DELETE NEXT 1</a>	Registro actual
<a href="#">DELETE RECORD n</a>	Registro <i>n</i>
<a href="#">DELETE</a> de más de un registro	Toda la tabla
<a href="#">DELETE - SQL</a>	Registro actual
<a href="#">GATHER</a>	Registro actual
<a href="#">INSERT</a>	Toda la tabla
<a href="#">INSERT - SQL</a>	Encabezado de la tabla
<a href="#">MODIFY MEMO</a>	Registro actual cuando comienza la edición
<a href="#">READ</a>	Registro actual y todos los registros de campos con alias
<a href="#">RECALL</a>	Registro actual
<a href="#">RECALL NEXT 1</a>	Registro actual
<a href="#">RECALL</a> registro <i>n</i>	Registro <i>n</i>
<a href="#">RECALL</a> de más de un registro	Toda la tabla
<a href="#">REPLACE</a>	Registro actual y todos los registros de campos con alias
<a href="#">REPLACE NEXT 1</a>	Registro actual y todos los registros de campos con alias
<a href="#">REPLACE</a> registro <i>n</i>	Registro <i>n</i> y todos los registros de campos con alias
<a href="#">REPLACE</a> de más de un registro	Toda la tabla y todos los archivos de campos con alias
<a href="#">SHOW GETS</a>	Registro actual y todos los registros de campos con alias
<a href="#">TABLEUPDATE()</a>	Depende del almacenamiento en búfer
<a href="#">UPDATE</a>	Toda la tabla
<a href="#">UPDATE - SQL</a>	Toda la tabla

## Características del bloqueo de registros

Los comandos que intentan bloquear registros son menos restrictivos que los comandos que bloquean tablas. Si bloquea un registro, los demás usuarios podrán seguir agregando o eliminando otros registros. Si otro usuario ya ha bloqueado un registro o una tabla o, si ha abierto la tabla de forma exclusiva, no se podrá bloquear dicha tabla o registro. Los comandos que intentan bloquear un registro y fallan devuelven el error "Registro utilizado por otra persona".

Los comandos [BROWSE](#), [CHANGE](#), [EDIT](#) y [MODIFY MEMO](#) no bloquean un registro hasta que usted lo modifique. Si está modificando campos procedentes de registros de tablas relacionadas, los registros relacionados se bloquearán si es posible. El intento de bloqueo fallará si el registro actual o alguno de los registros relacionados está bloqueado por otro usuario. Si se consigue realizar el bloqueo, podrá modificar el registro; el bloqueo se liberará cuando usted vaya a otro registro o active otra ventana.

## Características del bloqueo de tablas y encabezados

Algunos comandos de Visual FoxPro bloquean toda una tabla mientras que otros sólo bloquean su encabezado. Los comandos que bloquean la tabla completa son más restrictivos que los comandos que sólo bloquean el encabezado de la tabla. Cuando bloquee el encabezado de la tabla, otros usuarios no podrán agregar ni eliminar registros, aunque sí podrán cambiar los datos de los campos.

Los usuarios pueden compartir la tabla sin ocasionar conflictos cuando se ejecuta el comando [APPEND BLANK](#) pero puede producirse un error cuando otro usuario también anexa un registro BLANK a la tabla. Puede interceptar el error "Archivo utilizado por otra persona", que se devuelve cuando dos o más usuarios ejecutan APPEND BLANK simultáneamente. Los comandos que bloquean toda la tabla devuelven el error "Archivo utilizado por otra persona" si la tabla no puede bloquearse. Para cancelar el intento de bloqueo, presione ESC.

## Ejemplo: bloqueo automático

En el ejemplo siguiente, el usuario bloquea automáticamente el encabezado de la tabla; para ello, anexa registros de otra tabla, aunque `customer` se haya abierto como un archivo compartido.

```
SET EXCLUSIVE OFF
USE customer
APPEND FROM oldcust FOR status = "OPEN"
```

## Bloqueo manual

Puede bloquear manualmente un registro o una tabla mediante las funciones de bloqueo.

### Para bloquear manualmente un registro o una tabla

- Utilice uno de estos comandos:

```
RLOCK ( )
LOCK ( )
FLOCK ( )
```

[RLOCK\(\)](#) y [LOCK\(\)](#) son idénticos y bloquean uno o varios registros, mientras que [FLOCK\(\)](#) bloquea un archivo. Las funciones [LOCK\(\)](#) y [RLOCK\(\)](#) pueden aplicarse a un encabezado de tabla. Si especifica 0 como el registro para [LOCK\(\)](#) o [RLOCK\(\)](#) y la prueba indica que el encabezado está desbloqueado, la función bloqueará el encabezado y devolverá verdadero (.T.).

Una vez bloqueado un registro o una tabla, asegúrese de liberar el bloqueo mediante [UNLOCK](#) lo antes posible para proporcionar acceso a otros usuarios.

Estas funciones de bloqueo manual realizan las siguientes acciones:

- Comprueban el estado de bloqueo del registro o la tabla.
- Si la prueba indica que el registro está desbloqueado, bloquean el registro o la tabla y devuelven verdadero (.T.).
- Si no se puede bloquear el registro o la tabla, vuelven a intentarlo, dependiendo del valor actual de [SET REPROCESS](#).
- Devuelven verdadero (.T.) o falso (.F.), indicando si el intento de bloqueo ha tenido éxito o no.

**Sugerencia** Si desea comprobar el estado de bloqueo de un registro en la sesión sin bloquear el registro, use la función [ISRLOCKED\(\)](#) o [ISFLOCKED\(\)](#).

Si falla el intento de bloquear un registro o una tabla, el comando [SET REPROCESS](#) y la rutina de error actual determinarán si vuelve a intentarse el bloqueo. [SET REPROCESS](#) afecta al resultado de un intento de bloqueo sin éxito. Puede controlar el número de intentos de bloqueo o el período de tiempo durante el cual se intenta un bloqueo mediante [SET REPROCESS](#).

### Ejemplo: bloqueo manual

El ejemplo siguiente abre la tabla `customer` para acceso compartido y utiliza [FLOCK\(\)](#) para intentar bloquearla. Si la tabla se bloquea con éxito, [REPLACE ALL](#) actualizará todos sus registros. [UNLOCK](#) libera el bloqueo de archivo. Si el archivo no puede bloquearse porque otro usuario haya bloqueado el archivo o uno de sus registros, se mostrará un mensaje.

```
SET EXCLUSIVE OFF
SET REPROCESS TO 0
USE customer      && Abrir tabla compartida
IF FLOCK()
REPLACE ALL contact ;      && Reemplazar y desbloquear
WITH UPPER(contact)
UNLOCK
ELSE      && Mensaje de salida
WAIT "Otro usuario está utilizando el archivo." WINDOW NOWAIT
ENDIF
```

### Desbloquear datos

Después de establecer un bloqueo de registro o de archivo y completar una operación de datos en un entorno compartido, deberá liberar el bloqueo lo antes posible. Hay varios modos de liberar los bloqueos. En algunos casos, basta con desplazarse al registro siguiente para desbloquear los datos. En otros casos es preciso ejecutar comandos explícitos.

Para desbloquear un registro que se ha bloqueado automáticamente, sólo necesitará mover el puntero de registro, aunque haya establecido **MULTILOCKS ON**. No obstante, deberá eliminar explícitamente el bloqueo de un registro que haya bloqueado manualmente; mover el puntero de registro no es suficiente.

En la tabla siguiente se describen los efectos que producen diversos comandos sobre el bloqueo manual y automático de registros y tablas.

Comando	Efecto
<a href="#"><u>UNLOCK ALL</u></a>	Libera los bloqueos de registro y archivo en el área de trabajo actual.
<a href="#"><u>UNLOCK</u> ALL</a>	Libera todos los bloqueos de todas las áreas de trabajo de la sesión actual.
<a href="#"><u>SET MULTILOCKS OFF</u></a>	Activa la liberación automática del bloqueo actual al asegurar un bloqueo nuevo.
<a href="#"><u>FLOCK()</u></a>	Libera todos los bloqueos de registro de un archivo afectado antes de bloquear el archivo.
<a href="#"><u>CLEAR ALL</u></a> , <a href="#"><u>CLOSE ALL</u></a> , <a href="#"><u>USE</u></a> , <a href="#"><u>QUIT</u></a>	Libera todos los bloqueos de registro y archivo.
<a href="#"><u>END TRANSACTION</u></a>	Libera los bloqueos automáticos.
<a href="#"><u>TABLEUPDATE()</u></a>	Libera todos los bloqueos antes de actualizar la tabla.

**Precaución** Si se ha bloqueado automáticamente un registro en una FDU y el usuario desplaza el puntero fuera del registro y lo vuelve a colocar sobre él, el bloqueo se liberará. Utilice el almacenamiento de tablas en búfer para evitar este problema.

## Usar sesiones de datos

Para asegurarse de que todos los usuarios de un entorno compartido disponen de un duplicado exacto y seguro del entorno, y que múltiples instancias de un formulario pueden funcionar independientemente, Visual FoxPro proporciona sesiones de datos.

Una sesión de datos es una representación del entorno de trabajo dinámico actual. Podría considerar la sesión de datos como un entorno de datos en miniatura dentro de una sesión de Visual FoxPro abierta en un equipo. Cada sesión de datos contiene:

- Una copia de los elementos en el entorno de datos del formulario
- Cursores que representan las tablas abiertas, sus índices y relaciones.

El concepto de una sesión de datos se entiende fácilmente cuando se considera lo que ocurre al abrir el mismo formulario simultáneamente en dos estaciones de trabajo diferentes en una aplicación multiusuario. En este caso, cada estación de trabajo ejecuta una sesión de Visual FoxPro diferente y,

por tanto, tiene su propio conjunto de áreas de trabajo: cursores que representan las tablas base abiertas, los índices y las relaciones.

Sin embargo, si abre en un solo equipo múltiples instancias del mismo formulario en un solo proyecto, dentro de la misma sesión de Visual FoxPro, los formularios comparten la sesión de datos predeterminada, lo que representa un único entorno de trabajo dinámico. Cada instancia del formulario abierto en la misma sesión de Visual FoxPro usa el mismo conjunto de áreas de trabajo y las acciones en una única instancia de un formulario que mueven el puntero de registro en un área de trabajo afectan automáticamente a otras instancias del mismo formulario.

### Usar sesiones privadas de datos

Si desea tener más control sobre múltiples instancias de un formulario, puede implementar sesiones privadas de datos. Cuando el formulario utiliza sesiones privadas de datos, Visual FoxPro crea una nueva sesión de datos para cada instancia del control Form, FormSet o Toolbar que crea la aplicación. Cada sesión privada de datos contiene:

- Una copia diferente de cada tabla, índice y relación del entorno de datos del formulario.
- Un número ilimitado de áreas de trabajo.
- Punteros de registro para la copia de cada tabla, independientes de las tablas base del formulario.

El número de sesiones de datos disponibles está limitado sólo por la memoria y el espacio en disco disponible en el sistema.

Las sesiones privadas de datos se implementan al establecer la propiedad `DataSession` para el formulario. La propiedad `DataSession` tiene dos configuraciones:

- 1 – Sesión predeterminada de datos (la configuración predeterminada).
- 2 – Sesión privada de datos.

El valor predeterminado de la propiedad `DataSession` es 1.

### Para activar las sesiones privadas de datos

Elija una de las opciones siguientes:

- En el [Diseñador de formularios](#), establezca la propiedad `DataSession` del formulario como **2 – Sesión privada de datos**.

–O bien–

- En el código, establezca la propiedad `DataSession` a 2.

Por ejemplo, escriba:

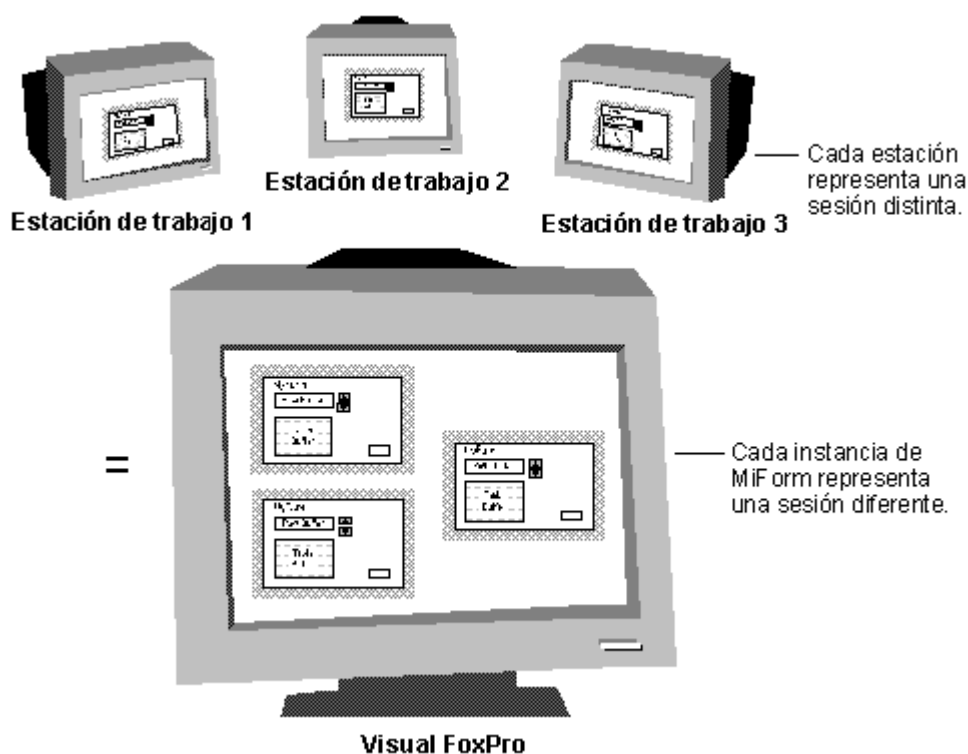
```
frmFormName.DataSession = 2
```

**Nota** Sólo puede establecer la propiedad `DataSession` durante el diseño v. en tiempo de

ejecución, esta propiedad es de sólo lectura.

Cuando un formulario utiliza sesiones privadas de datos, cada instancia de un formulario abierta en un solo equipo en una única sesión de Visual FoxPro usa su propio entorno de datos. El uso de sesiones privadas de datos es similar a la ejecución simultánea del mismo formulario desde diferentes estaciones de trabajo.

### Múltiples sesiones de datos equivalentes



### Identificar sesiones de datos

Cada sesión privada de datos se identifica por separado. Puede ver el contenido de cada sesión de datos en la ventana [Sesión de datos](#). También puede cambiar la descripción de la sesión de datos mediante comandos en el código del evento Load.

Puede ver el número de identificación de cada sesión de datos mediante la propiedad de tiempo de ejecución [DataSessionID](#). El ejemplo siguiente muestra la propiedad DataSessionID de un formulario denominado frmMiFormulario:

```
DO FORM frmMiFormulario
? frmMiFormulario.DataSessionID
```

Si activa el formulario mediante la cláusula NAME, puede usar el nombre del formulario para tener acceso a la propiedad DataSessionID, como en el código siguiente:

```
DO FORM MiFormulario NAME one
? one.DataSessionID
```

La propiedad `DataSessionID` sirve para identificar una sesión de datos determinada. Evite cambiar la propiedad `DataSessionID` de una instancia de un formulario porque los controles dependientes de datos pierden sus orígenes de datos cuando se modifica esta propiedad.

### **Actualizar datos con múltiples instancias de un formulario**

Mientras que las sesiones privadas de datos generan áreas de trabajo diferentes que contienen copias independientes de las tablas abiertas, índices y relaciones de un formulario, cada copia del formulario hace referencia a los mismos archivos de tablas base y de índices base subyacentes. Cuando un usuario actualiza un registro en una instancia de un formulario, se actualiza la tabla base a la que hace referencia el formulario. Los cambios realizados en otra instancia del formulario pueden verse al desplazarse por el registro modificado.

Los bloqueos realizados en registros o tablas en una sesión privada de datos son respetados por otras sesiones privadas de datos. Por ejemplo, si el usuario de la sesión de datos 1 pone un bloqueo en un registro, el usuario de la sesión de datos 2 no puede bloquear el registro. Si el usuario de la sesión 1 abre una tabla de forma exclusiva, el usuario de la sesión de datos 2 no puede abrir la tabla. Respetando los bloqueos realizados por otras sesiones de datos, Visual FoxPro protege la integridad de las actualizaciones en las tablas base subyacentes.

### **Personalizar el entorno de una sesión de datos**

Debido a que las sesiones de datos controlan el alcance de determinados comandos SET, puede usar sesiones privadas de datos para establecer configuraciones personalizadas de comandos SET dentro de una sola sesión de Visual FoxPro.

Por ejemplo, el comando [SET EXACT](#), que controla las reglas utilizadas al comparar cadenas de caracteres de diferentes longitudes, pertenece al ámbito de la sesión de datos actual. La configuración predeterminada de SET EXACT es Off, lo que especifica que, para ser equivalentes, las expresiones deben coincidir, carácter a carácter, hasta que se llega al final de las expresiones en el lado derecho. Tal vez desee activar búsquedas "borrosas" o equivalentes; para ello, establezca SET EXACT como OFF para la sesión de datos actual. Sin embargo, la aplicación podría contener un formulario determinado que requiriera coincidencias exactas. Podría establecer la propiedad `DataSession` como 2 para el formulario que requiere coincidencias exactas, para activar las sesiones privadas de datos y después establecer SET EXACT a ON para ese formulario. Si ejecuta un comando SET sólo para el formulario que utiliza sesiones privadas de datos, se mantiene intacta la configuración global de la sesión de Visual FoxPro a la vez que permite la configuración personalizada para un formulario determinado.

### **Ignorar la asignación automática de sesión de datos**

Cuando las sesiones privadas de datos para un formulario están en uso, los cambios realizados en los datos de un formulario no se representan automáticamente en otras instancias del mismo formulario. Si desea que todas las instancias de un formulario tengan acceso a los mismos datos y reflejen automáticamente los cambios realizados en datos comunes, puede ignorar la asignación automática de sesión de datos.



## Para ignorar la asignación automática de sesión de datos

- Utilice uno de estos comandos:

`SET DATASESSION TO 1`

–O bien–

`SET DATASESSION TO`

Ambos comandos permiten controlar la sesión de datos predeterminada mediante la ventana Comandos y el Administrador de proyectos.

## Almacenar en búfer del acceso a datos

Si desea proteger los datos durante las actualizaciones, utilice búferes. El almacenamiento en búfer de registros y tablas de Visual FoxPro ayuda a proteger las operaciones de mantenimiento y actualización de datos en registros individuales y múltiples registros de datos en entornos multiusuario. Los búferes pueden comprobar, bloquear y liberar automáticamente registros o tablas.

Mediante el almacenamiento en búfer es posible detectar y resolver fácilmente conflictos en operaciones de actualización de datos: el registro actual se copia a una ubicación de memoria o de disco administrada por Visual FoxPro. Los demás usuarios podrán seguir teniendo acceso al registro original simultáneamente. Cuando se desplace fuera del registro o intente actualizarlo mediante programación, Visual FoxPro intentará bloquear el registro, comprobará que otros usuarios no hayan realizado cambios y, a continuación, escribirá las modificaciones. Después de intentar actualizar los datos, deberá resolver los conflictos que impiden que las modificaciones se escriban en la tabla original.

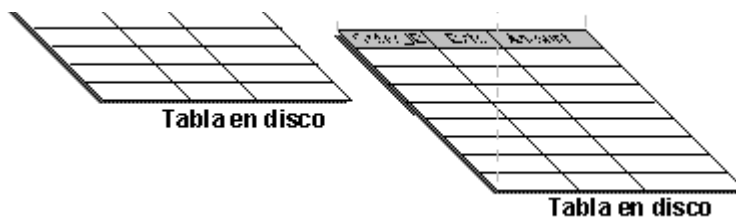
## Elegir un método de almacenamiento en búfer

Antes de activar el almacenamiento en búfer, evalúe el entorno de datos para elegir el método de almacenamiento en búfer y las opciones de bloqueo que mejor se ajusten a las necesidades de edición de la aplicación, los tipos y tamaños de registros y tablas, y cómo se utiliza y actualiza la información, entre otros factores. Una vez activado el almacenamiento en búfer, permanecerá en vigor hasta que lo desactive o hasta que cierre la tabla.

Visual FoxPro dispone de dos tipos de almacenamiento en búfer: registro y tabla.

## Almacenamiento en búfer de registros y tablas de Visual FoxPro





- Para tener acceso, modificar y escribir un registro individual cada vez, elija el almacenamiento de registros en búfer.

El almacenamiento de registros en búfer proporciona la validación del proceso adecuado con una repercusión mínima sobre las operaciones de actualización de datos que realizan otros usuarios en un entorno multiusuario.

- Para almacenar en búfer las actualizaciones a varios registros, elija el almacenamiento de tablas en búfer.

El almacenamiento de tablas en búfer es el modo más efectivo de administrar varios registros de una tabla o registros secundarios en una relación uno a varios.

- Para proporcionar la máxima protección para los datos existentes, utilice transacciones de Visual FoxPro.

Las transacciones pueden utilizarse por sí solas, pero obtendrá una mayor efectividad si las emplea como encapsuladores para comandos de almacenamiento en búfer de tablas o registros. Para obtener más información al respecto, consulte la sección, [Administrar actualizaciones mediante transacciones](#), más adelante en este mismo capítulo.

## Elegir un modo de bloqueo

Visual FoxPro proporciona almacenamiento en búfer en dos modos de bloqueo: pesimista y optimista. Estas opciones determinan cuándo se bloquean uno o más registros y cuándo y cómo se liberan.

### Almacenamiento pesimista en búfer

El almacenamiento pesimista en búfer impide que otros usuarios de un entorno multiusuario tengan acceso a un determinado registro o tabla mientras usted realiza cambios en el mismo. Un bloqueo pesimista proporciona el entorno más seguro para cambiar registros individuales, aunque puede hacer más lentas las operaciones del usuario. Este modo de almacenamiento en búfer es bastante similar al mecanismo de bloqueo estándar de versiones anteriores de FoxPro, con la ventaja adicional del almacenamiento de datos incorporados en búfer.

### Almacenamiento optimista en búfer

El almacenamiento optimista en búfer es un modo eficaz de actualizar registros ya que los bloqueos sólo se realizan en el momento en que se escribe el registro, minimizando de este modo el tiempo que

solo se realizan en el momento en que se escribe el registro, minimizando de este modo el tiempo que un único usuario monopoliza el sistema en un entorno multiusuario. Cuando utilice el almacenamiento en búfer de registros o tablas para vistas, Visual FoxPro impondrá el bloqueo optimista.

El valor de la propiedad Buffering, establecido mediante la función [CURSORSETPROP\(\)](#) determina los métodos de almacenamiento en búfer y de bloqueo.

En la tabla siguiente se resumen los valores válidos para la propiedad Buffering:

Para activar	Utilice este valor
Sin almacenamiento en búfer. El valor predeterminado.	1
Bloqueos pesimistas de registros que bloquean el registro ahora y actualizan cuando el puntero se mueve o se ejecuta <a href="#">TABLEUPDATE()</a> .	2
Bloqueos optimistas de registros que esperan hasta que el puntero se mueve y después bloquean y actualizan.	3
Bloqueos pesimistas de tablas que bloquean el registro ahora y actualizan posteriormente al ejecutarse <a href="#">TABLEUPDATE()</a> .	4
Bloqueos optimistas de tablas que esperan hasta <a href="#">TABLEUPDATE()</a> para bloquear y actualizar los registros modificados.	5

El valor predeterminado de Buffering es 1 para las tablas y 5 para las vistas. Si utiliza el almacenamiento en búfer para tener acceso a datos remotos, la propiedad Buffering tendrá un valor 3, almacenamiento optimista de filas en búfer o 5, almacenamiento optimista de tablas en búfer. Para obtener más información sobre el acceso a los datos de tablas remotas, consulte el capítulo 6, [Consultar y actualizar múltiples tablas](#), del *Manual del usuario*.

**Nota** Establezca MULTILOCKS como ON para todos los modos de almacenamiento en búfer por encima de 1.

### Activar el almacenamiento de registros en búfer

Active el almacenamiento de registros en búfer mediante la función [CURSORSETPROP\(\)](#).

### Para activar el bloqueo pesimista de registros en el área de trabajo actual

- Utilice esta función y este valor:

```
CURSORSETPROP("Buffering", 2)
```

Visual FoxPro intenta bloquear el registro en la ubicación del puntero. Si el bloqueo tiene éxito, Visual FoxPro situará el registro en un búfer y permitirá su modificación. Cuando mueva el puntero de registro o ejecute el comando [TABLEUPDATE\(\)](#), Visual FoxPro escribirá el registro almacenado en búfer en la tabla original.

### Para activar el bloqueo optimista de registros en el área de trabajo actual

- Utilice esta función y este valor:

```
CURSORSETPROP("Buffering", 3)
```

Visual FoxPro escribe en un búfer el registro situado en la ubicación del puntero y permite modificaciones. Cuando mueva el puntero de registro o ejecute el comando [TABLEUPDATE\(\)](#), Visual FoxPro intentará un bloqueo del registro. Si se produce el bloqueo, Visual FoxPro comparará el valor actual del registro en el disco con el valor original del búfer. Si ambos valores son iguales, las modificaciones se escribirán en la tabla original; si la comparación entre estos valores indica que son distintos, Visual FoxPro generará un error.

### Activar el almacenamiento de tablas en búfer

Active el almacenamiento de tablas en búfer mediante la función [CURSORSETPROP\(\)](#).

### Para activar el bloqueo pesimista de múltiples registros en el área de trabajo actual

- Utilice esta función y este valor:

```
CURSORSETPROP("Buffering", 4)
```

Visual FoxPro intenta bloquear el registro situado en la ubicación del puntero. Si se produce el bloqueo, Visual FoxPro situará el registro en un búfer y permitirá su modificación. Utilice el comando [TABLEUPDATE\(\)](#) para escribir en la tabla original los registros almacenados en búfer.

### Para activar el bloqueo optimista de múltiples registros en el área de trabajo actual

- Utilice esta función y este valor:

```
CURSORSETPROP("Buffering", 5)
```

Visual FoxPro escribe los registros en un búfer y permite modificarlos hasta que usted ejecute un comando [TABLEUPDATE\(\)](#). A continuación, Visual FoxPro llevará a cabo la siguiente secuencia en cada registro del búfer:

- Intenta un bloqueo en cada registro modificado.
- Cuando realiza el bloqueo, compara el valor actual de cada registro del disco con el valor original del búfer.
- Escribe las modificaciones en la tabla original si la comparación indica que ambos valores son iguales.
- Genera un error si los valores son distintos.

Cuando se activa el almacenamiento de tablas en búfer, Visual FoxPro sólo intentará realizar actualizaciones después de que se ejecute el comando [TABLEUPDATE\(\)](#).

### Agregar y eliminar registros de búferes de tablas

Puede anexas y eliminar registros mientras está activado el almacenamiento de tablas en búfer: los registros anexados se agregan al final del búfer. Para tener acceso a todos los registros del búfer, incluidos los registros anexados, utilice la función [RECNO\(\)](#). La función RECNO( ) devuelve números negativos secuenciales para los registros que usted anexe a un búfer de tabla. Por ejemplo, si inicia el almacenamiento de tablas en búfer, modifica los registros 7, 8 y 9, y a continuación anexa tres registros, el búfer contendrá valores RECNO( ) de 7, 8, 9, - 1, - 2 y - 3.

### Búfer después de modificar y anexas registros

BÚFER DE TABLAS	
7	Editar registro
8	Editar registro
9	Editar registro
-1	Añadir registro
-2	Añadir registro
-3	Añadir registro

Sólo podrá quitar del búfer registros anexados si utiliza el comando [TABLEREVERT\(\)](#). Para cualquier registro anexado, TABLEUPDATE( ) y TABLEREVERT( ) eliminan el valor negativo de RECNO( ) para ese registro y mantienen la secuencia.

### Búfer después de modificar, eliminar un registro anexado y anexas otro

BÚFER DE TABLAS	
7	Editar registro
8	Editar registro
9	Editar registro
-1	Añadir registro
-2	Eliminar registro
-3	Añadir registro
-4	Añadir registro

Mientras utiliza un búfer de tabla, puede usar el comando GO con el valor negativo de RECNO( ) para tener acceso a un determinado registro anexado. Por ejemplo, utilizando el ejemplo anterior, puede escribir:

```
GO 7      &&   se mueve al primer registro almacenado en búfer
GO -3     &&   se mueve al sexto registro almacenado en búfer(3º anexado)
```

### Agregar registros a un búfer de tabla

- Utilice el comando [APPEND](#) o [APPEND](#) BLANK después de activar el almacenamiento de tablas en búfer.

Los registros anexados tienen números de RECNO( ) negativos y secuenciales en sentido ascendente.

~~Para quitar un registro anexado de un búfer de tabla~~

**Para quitar un registro anexado de un búfer de tabla**

1. Utilice el comando [GO](#) con un valor negativo para situar el puntero de registro en el registro que desea eliminar.
2. Utilice el comando [DELETE](#) para marcar el registro para su eliminación.
3. Utilice el comando [TABLEREVERT\(\)](#) para quitar el registro del búfer.

**Nota** La función [TABLEREVERT\(\)](#) también afecta al estado de las filas eliminadas o modificadas.

**Para quitar todos los registros anexados de un búfer de tabla**

- Utilice [TABLEREVERT\(\)](#) con un valor de verdadero (.T.).

[TABLEREVERT\(\)](#) quita de un búfer de tabla los registros anexados sin escribir los registros en la tabla. [TABLEUPDATE\(\)](#) escribe en una tabla todos los registros que están almacenados en búfer, aunque se hayan marcado para su eliminación.

## Actualizar datos

Para actualizar datos, puede usar almacenamiento en búfer, transacciones o vistas.

**Realizar actualizaciones con almacenamiento en búfer**

Después de elegir el método de almacenamiento en búfer y el tipo de bloqueo, active el almacenamiento en búfer de registros o tablas.

**Para activar el almacenamiento en búfer**

Elija una de las opciones siguientes:

- En el **Diseñador de formularios**, establezca la propiedad **BufferModeOverride** del cursor en el entorno de datos del formulario.

–O bien–

- En el código, establezca la propiedad **Buffering**.

Por ejemplo, puede activar el almacenamiento pesimista de filas en búfer si coloca el código siguiente en el procedimiento **Init** de un formulario:

```
CURSORSETPROP('Buffering', 2)
```

Después escriba código para las operaciones de actualización en el código del método apropiado para los controles.

Para escribir las modificaciones en la tabla original, utilice [TABLEUPDATE\(\)](#). Para cancelar las

modificaciones después de una operación fallida de actualización en una tabla restringida por reglas, utilice [TABLEREVERT\(\)](#). `TABLEREVERT()` es válido aunque el almacenamiento explícito de tablas en búfer no esté activado.

El ejemplo siguiente demuestra cómo actualizar registros cuando está activado el almacenamiento pesimista de registros en búfer.

### Ejemplo de actualización mediante búferes de registros y tablas

Código	Comentarios
<pre>OPEN DATABASE testdata USE customers CURSORSETPROP('Buffering', 2)</pre>	En el código Init del formulario, abre la tabla y activa el almacenamiento pesimista de registros en búfer.
<pre>lModificado = .F. FOR nNúmCampo = 1 TO FCOUNT()     IF GETFLDSTATE(nNúmCampo) = 2         lModificado = .T.         EXIT     ENDIF ENDFOR</pre>	<p>Se desplaza por campos y comprueba los que se han modificado.</p> <p>Nota: este código podría estar en el evento Click de un botón de comando "Guardar" o "Actualizar".</p>
<pre>IF lModificado     nResultado = MESSAGEBOX(         ("El registro cambió. ¿Modificar? ;         4+32+256, "Cambiar datos</pre>	Busca el siguiente registro modificado.
<pre>    IF nResultado7         TABLEREVERT (.F.)     ENDIF ENDIF</pre>	Muestra el valor actual y da al usuario la posibilidad de invertir el cambio del campo actual.
<pre>SKIP IF EOF()     MESSAGEBOX( "ya está al final"     THISFORM.Refresh</pre>	SKIP garantiza que se escribe el último cambio.

### Administrar actualizaciones mediante transacciones

Incluso si utiliza el almacenamiento en búfer pueden surgir problemas. Si desea proteger las operaciones de actualización y recuperar una sección completa de código como una unidad, utilice transacciones.

La incorporación de transacciones a su aplicación proporciona una protección adicional al almacenamiento en búfer de registros y tablas de Visual FoxPro, situando una sección completa de

almacenamiento en búfer de registros y tablas de Visual FoxPro, situando una sección completa de código en una unidad protegida y recuperable. Puede anidar transacciones y emplearlas para proteger actualizaciones almacenadas en búfer. Las transacciones de Visual FoxPro sólo están disponibles con tablas pertenecientes a una base de datos.

### Encapsular segmentos de código

Una transacción actúa como un envoltorio que almacena en memoria caché o en disco las operaciones de actualización de datos, en lugar de aplicar directamente esas actualizaciones a la base de datos. La actualización real de la base de datos se realiza al final de la transacción. Si por alguna razón el sistema no puede realizar las operaciones de actualización en la base de datos, podrá anular la transacción completa para que no se lleve a cabo ninguna actualización.

**Nota** Las operaciones de actualización almacenadas en búfer que se haya realizado fuera de una transacción se ignoran en otra transacción de la misma sesión de datos.

### Comandos que controlan transacciones

Visual FoxPro proporciona tres comandos para controlar una transacción:

Para	Utilice
Iniciar una transacción	<a href="#">BEGIN TRANSACTION</a>
Determinar el nivel actual de la transacción	<a href="#">TXNLEVEL()</a>
Invertir todos los cambios realizados desde la instrucción BEGIN TRANSACTION más reciente	<a href="#">ROLLBACK</a>
Bloquear registros, grabar en disco todos los cambios realizados en tablas de la base de datos desde la instrucción BEGIN TRANSACTION más reciente y, a continuación, desbloquear los registros	<a href="#">END TRANSACTION</a>

Puede utilizar transacciones para encapsular modificaciones a tablas, archivos .cdx estructurales y archivos memo asociados a tablas de una base de datos. Las operaciones en las que intervienen variables de memoria y otros objetos no respetan las transacciones, por lo que no se pueden anular o grabar.

**Nota** Cuando se utilicen los datos almacenados en tablas remotas, el control de los comandos de la transacción sólo actualiza los datos de la copia local del cursor de la vista; las actualizaciones de tablas base remotas no se ven afectadas. Para activar las transacciones manuales en tablas remotas, utilice [SQLSETPROP\(\)](#) y, a continuación, controle la transacción mediante [SQLCOMMIT\(\)](#) y [SQLROLLBACK\(\)](#).

En general, deberá utilizar las transacciones con búferes de registro en lugar de con almacenamiento en búfer de tablas, salvo para envolver llamadas de [TABLEUPDATE\(\)](#). Si incluye un comando [TABLEUPDATE\(\)](#) en una transacción, podrá anular una actualización fallida, resolver la causa del fallo y, a continuación, volver a intentar [TABLEUPDATE\(\)](#) sin perder datos. De este modo se asegurará de que la actualización se produce como una operación de "todo o nada".



Aunque el procesamiento simple de transacciones proporciona operaciones seguras de actualización de datos en situaciones normales, no ofrece protección total contra fallos del sistema. Si se interrumpe el suministro eléctrico o el sistema queda bloqueado durante el procesamiento del comando [END TRANSACTION](#), la actualización de datos puede fallar.

Utilice el código siguiente como plantilla para transacciones:

```
BEGIN TRANSACTION
* Actualizar registros
IF lSuccess = .F. && se produce un error
    ROLLBACK
ELSE    && aplicar las modificaciones
    * Validar los datos
    IF    && ocurre un error
        ROLLBACK
    ELSE
        END TRANSACTION
    ENDIF
ENDIF
```

### Usar transacciones

Las reglas siguientes se aplican a las transacciones:

- Una transacción comienza con el comando [BEGIN TRANSACTION](#) y termina con el comando [END TRANSACTION](#) o [ROLLBACK](#). Una instrucción END TRANSACTION que no vaya precedida de una instrucción BEGIN TRANSACTION generará un error.
- Una instrucción ROLLBACK que no vaya precedida de una instrucción BEGIN TRANSACTION generará un error.
- Una vez comenzada una transacción, permanecerá en vigor hasta que comience el END TRANSACTION correspondiente (o hasta que se ejecute un comando ROLLBACK), incluso en programas y funciones, a menos que se termine la aplicación, lo que producirá una anulación.
- Visual FoxPro utiliza datos almacenados en caché en el búfer de transacción antes de utilizar datos del disco para consultas sobre los datos relacionados con transacciones. De este modo se asegura la utilización de los datos más actuales.
- Si la aplicación termina durante una transacción, todas las operaciones se anularán.
- Una transacción sólo funciona en un contenedor de base de datos.
- No es posible utilizar el comando [INDEX](#) si sobrescribe un archivo de índice existente o si hay abierto algún archivo de índice .cdx.
- Las transacciones pertenecen al ámbito de las sesiones de datos.

Las transacciones presentan los comportamientos de bloqueo siguientes:

- En una transacción, Visual FoxPro impone un bloqueo en el momento en que un comando lo solicita directa o indirectamente. Cualquier comando de desbloqueo directo o indirecto procedente del sistema o del usuario se almacena localmente hasta que se complete la transacción mediante los comandos ROLLBACK o END TRANSACTION.
- Si utiliza un comando de bloqueo como [FLOCK\(\)](#) o [RLOCK\(\)](#) en una transacción, la instrucción END TRANSACTION no liberará el bloqueo. En ese caso, deberá desbloquear explícitamente todos los bloqueos realizados en una transacción.

## Anidamiento de transacciones

Las transacciones anidadas proporcionan grupos lógicos de operaciones de actualización de tablas que están aislados de los procesos concurrentes. No es necesario que los pares BEGIN TRANSACTION...END TRANSACTION estén en la misma función o el mismo procedimiento. Las transacciones anidadas tienen estas reglas:

- Se pueden anidar hasta cinco pares [BEGIN TRANSACTION...END TRANSACTION](#).
- Las actualizaciones realizadas en una transacción anidada no se graban hasta que se llama a la instrucción END TRANSACTION más externa.
- En las transacciones anidadas, una instrucción END TRANSACTION sólo funciona sobre la transacción iniciada por la última instrucción BEGIN TRANSACTION ejecutada.
- En las transacciones anidadas, una instrucción [ROLLBACK](#) sólo funciona sobre la transacción iniciada por la última instrucción BEGIN TRANSACTION ejecutada.
- La actualización más interna de una serie de transacciones anidadas sobre los mismos datos tiene prioridad sobre todas las demás del mismo bloque de transacciones anidadas.

Observe en el ejemplo siguiente que, puesto que los cambios en una transacción anidada no se escriben en disco, sino en el búfer de transacciones, la transacción más interna sobrescribirá los cambios realizados en los mismos campos STATUS de la transacción anterior:

```
BEGIN TRANSACTION    &&  transacción 1
  UPDATE EMPLOYEE ;   &&  primera modificación
    SET STATUS = "Contrato" ;
    WHERE EMPID BETWEEN 9001 AND 10000
  BEGIN TRANSACTION    &&  transacción 2
    UPDATE EMPLOYEE ;
      SET STATUS = "Exento" ;
      WHERE HIREDATE > {1/1/93} &&  se sobrescribe
  END TRANSACTION &&  transacción 2
END TRANSACTION      &&  transacción 1
```

El siguiente ejemplo de transacción anidada elimina un registro de cliente y todas las facturas relacionadas. La transacción se anulará si se producen errores durante un comando [DELETE](#). Este ejemplo demuestra la agrupación de operaciones de actualización de tablas para impedir que las actualizaciones se completen parcialmente y para evitar conflictos de simultaneidad.

## Ejemplo de modificación de registros en transacciones anidadas

Código	Comentarios
DO WHILE TXNLEVEL( ) > 0 ROLLBACK ENDDO	Limpia el entorno de otras transacciones.
CLOSE ALL SET MULTILOCKS ON SET EXCLUSIVE OFF	Establece el entorno para el almacenamiento en búfer.

```

OPEN DATABASE test
USE mrgtest1
CURSORSETPROP('buffering',5)
GO TOP

```

Activa el almacenamiento optimizado de tablas en búfer.

```

REPLACE fld1 WITH "cambiado"
SKIP
REPLACE fld1 WITH "otro cambio"
MESSAGEBOX("modificar primer campo de ambos " + ;
           " registros en otro equipo ")

```

Cambia un registro.

Cambia otro registro.

```

BEGIN TRANSACTION
lSuccess = TABLEUPDATE(.T.,.F.)

```

Inicia la transacción 1 e intenta actualizar todos los registros modificados que no están vigentes.

```

IF lSuccess = .F.
    ROLLBACK
    AERROR(aErrors)
    DO CASE
    CASE aErrors[1,1] = 1539
    ...
    CASE aErrors[1,1] = 1581
    ...
    CASE aErrors[1,1] = 1582

```

Si falla la actualización, anula la transacción.

Obtiene un error de AERROR(). Determina la causa del fallo.

Si ha fallado un desencadenante, soluciona el problema.

Si un campo no acepta valores reales, soluciona el problema.

Si se ha infringido una regla de campo, soluciona el problema.

```

CASE aErrors[1,1] = 1585
    nNextModified = getnextmodified(0)
    DO WHILE nSigModificado <> 0
        GO nSigModificado
        RLOCK()
        FOR nField = 1 to FCOUNT()
            cField = FIELD(nCampo)

            if OLDVAL(cCampo) <> CURVAL(cCampo)

```

Si otro usuario ha modificado un registro, localiza el primer registro modificado.

Hace un bucle a través de todos los registros modificados, comienza con el primero.

Bloquea cada registro para asegurarse de que se puede actualizar.

Comprueba si hay cambios en el campo.

Compara el valor almacenado en memoria con el valor en disco; a continuación muestra un cuadro de diálogo al usuario.

```

nResult = MESSAGEBOX(
("Otro usuario ha " + ;
 "modificado los datos "+ ;
 "¿Desea mantener los cambios?", 4+48, ;
 "Registro modificado")

```

<pre>                 IF nResultado = 7                     TABLEREVERT(.F.)                     UNLOCK record nSigModificado                 ENDIF </pre>	Si el usuario responde 'No', invi registro uno y lo desbloquea.
<pre>                 EXIT             ENDIF         ENDFOR </pre>	Sale del bucle 'FOR nCampo'.
<pre>         ENDDO </pre>	Obtiene el siguiente registro modificado.
<pre>         BEGIN TRANSACTION         TABLEUPDATE(.T.,.T.)         END TRANSACTION         UNLOCK </pre>	<p>Inicia la transacción 2 y actualiz todos los registros no invertidos están en vigor.</p> <p>Finaliza la transacción 2.</p> <p>Libera el bloqueo.</p>
<pre>         CASE aErrors[1,1] = 109         ...         CASE aErrors[1,1] = 1583         ...         CASE aErrors[1,1] = 1884         ...         OTHERWISE             MESSAGEBOX( "Error desconocido "+                 " mensaje: " + STR(aErrors[1,1]))         ENDCASE </pre>	<p>Si el registro está en uso por otr usuario, soluciona el problema.</p> <p>Si se ha infringido una regla de soluciona el problema.</p> <p>Si hubo una infracción de índice único, soluciona el problema.</p> <p>De lo contrario, muestra un cua diálogo al usuario.</p>
<pre>     ELSE         END TRANSACTION     ENDIF </pre>	Termina la transacción 1.

### Proteger actualizaciones remotas

Las transacciones pueden protegerle frente a los errores generados por el sistema durante actualizaciones de datos en tablas remotas. El ejemplo siguiente utiliza una transacción para envolver operaciones de escritura de datos en una tabla remota.

### Ejemplo de una transacción de una tabla remota

Código	Comentarios
<pre> hConex = CURSORGETPROP('connecthandle') SQLSETPROP(hConnect, 'transmode', DB_TRANSMANUAL) </pre>	<p>Obtiene el controlador de conexión y activa transacciones manuales.</p>

BEGIN TRANSACTION	Comienza la transacción manual.
<pre> lExito = TABLEUPDATE(.T.,.F.) IF lExito = .F.     SQLROLLBACK (hConex)     ROLLBACK </pre>	Intenta actualizar todos los registros que no están en vigor. Si la actualización ha fallado, anula la transacción en la conexión del cursor.
<pre> AERROR(aErrors) DO CASE </pre>	Obtiene el error de AERROR().
<pre> CASE aErrors[1,1] = 1539 ... </pre>	Si ha fallado un desencadenante, soluciona el problema.
<pre> CASE aErrors[1,1] = 1581 ... </pre>	Si un campo no acepta valores nulos, soluciona el problema.
<pre> CASE aErrors[1,1] = 1582 ... </pre>	Si se ha infringido una regla de campo, soluciona el problema.
<pre> CASE aErrors[1,1] = 1585     nSigModificado = GETNEXTMODIFIED(0)     DO WHILE nSigModificado &lt;&gt; 0         GO nSigModificado </pre>	Si otro usuario ha cambiado un registro, soluciona el problema.
	Recorre todos los registros modificados, comenzando con el primero.
<pre> FOR nCampo = 1 to FCOUNT()     cCampo = FIELD(nCampo)     IF OLDVAL(cCampo) &lt;&gt; CURVAL(cCampo)         nResultado = MESSAGEBOX(             "Otro usuario ha ;             modificado los datos. ;             ¿Desea mantener los cambios?",4+48,;             "Registro modificado") </pre>	Comprueba si ha habido cambios en cada campo.
	Compara el valor almacenado en búfer con el valor del disco y, a continuación, muestra un cuadro de diálogo al usuario.
<pre> IF nResultado = 7     TABLEREVERT(.F.) ENDIF EXIT ENDIF ENDFOR nSigModificado = ; GETNEXTMODIFIED(nSigModificado) ENDDO </pre>	Si el usuario responde 'No', invierte el registro uno.
	Sale del bucle "FOR nCampo...".
	Obtiene el siguiente registro modificado.
<pre> TABLEUPDATE(.T.,.T.) SQLCOMMIT(hConex) </pre>	Actualiza todos los registros no invertidos que están vigentes y ejecuta una grabación.
<pre> CASE aErrors[1,1] = 100 </pre>	Si el usuario responde 'No', invierte el registro uno.

<pre> CASE aErrors[1,1] = 109     * Controlar el error </pre>	<p>El error 109 indica que otro usuario está utilizando el registro.</p>
<pre> CASE aErrors[1,1] = 1583     * Controlar el error </pre>	<p>El error 1583 indica que se ha infringido una regla de la fila.</p>
<pre> CASE aErrors[1,1] = 1884     * Controlar el error </pre>	<p>El error 1884 indica que se ha infringido la unicidad del índice.</p>
<pre> OTHERWISE     * Controlar errores genéricos. </pre>	
<pre>         MESSAGEBOX("Mensaje de error desconocido:" ;             + STR(aErrors[1,1]))     ENDCASE </pre>	<p>Muestra un cuadro de diálogo al usuario.</p>
	<p>Fin del control de errores.</p>
<pre> ELSE     SQLCOMMIT(hConnect)     END TRANSACTION ENDIF </pre>	<p>Si se controlaron todos los errores y la transacción completa se ha realizado con éxito, ejecuta una confirmación y termina la transacción.</p>

## Administrar el rendimiento

Cuando haya creado la aplicación multiusuario, podrá emplear las siguientes sugerencias para mejorar su rendimiento:

- Sitúe los archivos temporales en una unidad local.
- Elija entre ordenar e indexar archivos.
- Prepare el acceso exclusivo a los archivos.
- Sincronice el bloqueo de archivos.

### Colocar los archivos temporales en una unidad local

Visual FoxPro crea sus archivos temporales en la carpeta de directorios temporales predeterminada de Windows. Las sesiones de modificación de texto también pueden crear temporalmente una copia completa del archivo que se está modificando (un archivo .BAK).

Si las estaciones de trabajo locales disponen de disco duro con abundante espacio libre, podrá mejorar el rendimiento si sitúa estos archivos temporales en la unidad local o en una unidad RAM. Al redirigir estos archivos a una unidad local o RAM aumentará el rendimiento, ya que se reduce el acceso a la unidad de red.

Puede especificar una ubicación alternativa para esos archivos si incluye las instrucciones EDITWORK, SORTWORK, PROGWORK y TMPFILES en el archivo de configuración Config.fpw. Para obtener más información sobre las opciones de configuración, consulte el capítulo

4, [Optimizar el sistema](#), de la *Guía de instalación*.

### Elegir entre ordenar e indexar archivos

Cuando los datos que contiene una tabla son relativamente estáticos, el procesamiento de tablas ordenadas secuencialmente sin un orden establecido mejora el rendimiento. Esto no significa que las tablas ordenadas no puedan o no deban aprovechar los archivos de índice; el comando [SEEK](#) que requiere un índice, es incomparable para encontrar registros rápidamente. Sin embargo, cuando encuentre un registro con SEEK, podrá desactivar la ordenación.

### Preparar el acceso exclusivo a los archivos

Los comandos que se ejecutan cuando ningún otro usuario necesita tener acceso a los datos, como las actualizaciones nocturnas, pueden verse beneficiados si abre los archivos de datos para uso exclusivo. Cuando se abren archivos para uso exclusivo, el rendimiento mejora porque Visual FoxPro no necesita comprobar el estado de los bloqueos de registros o archivos.

### Sincronizar el bloqueo de archivos

Para disminuir la contención entre los usuarios para tener acceso de escritura a una tabla o un registro, reduzca el tiempo que un registro o una tabla permanecen bloqueados. Para ello, bloquee el registro únicamente después de modificarlo en lugar de hacerlo durante la modificación. El almacenamiento optimista de filas en búfer proporciona el menor tiempo de bloqueo.

Para obtener más información sobre la mejora del rendimiento, consulte el capítulo 4, [Optimizar el sistema](#), de la *Guía de instalación e Índice principal*. También encontrará información sobre la mejora del rendimiento en aplicaciones cliente-servidor en el capítulo 22, [Optimizar el rendimiento cliente-servidor](#), de este manual.

### Administrar actualizaciones con vistas

Puede usar la tecnología de administración de conflictos de actualización en las vistas de Visual FoxPro para administrar el acceso de múltiples usuarios a los datos. Las vistas controlan lo que se envía a las tablas base de la vista mediante la propiedad WhereType. Puede establecer esta propiedad para vistas locales y remotas. La propiedad WhereType proporciona cuatro configuraciones:

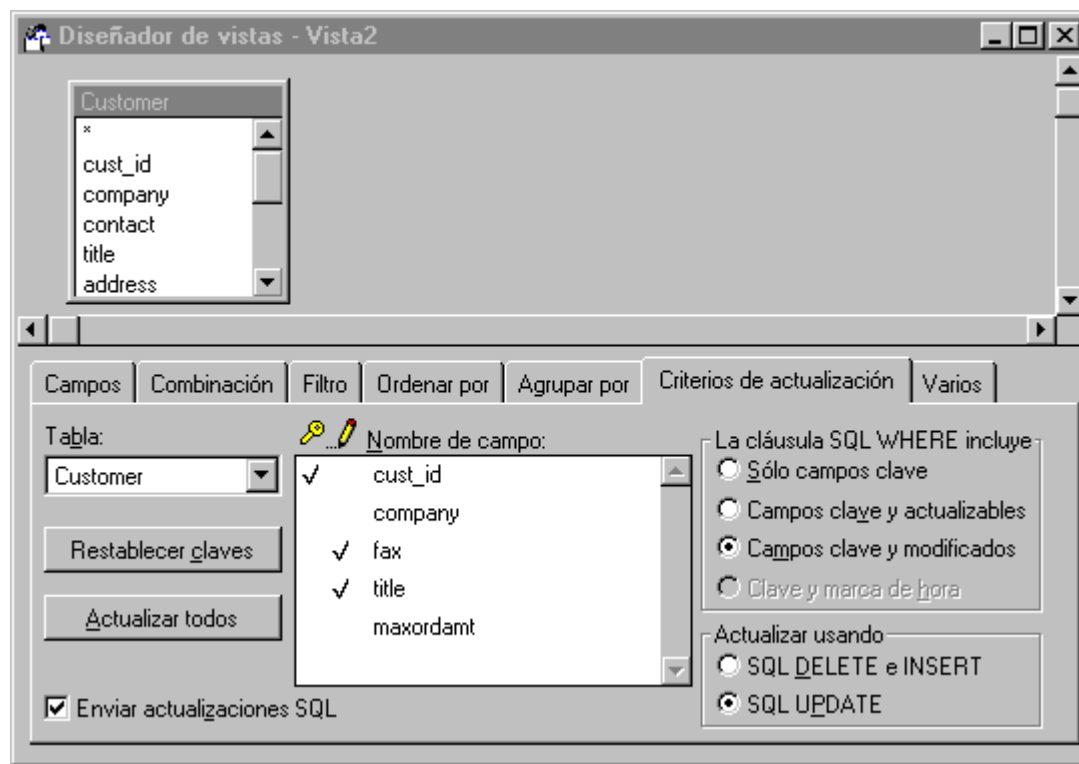
- DB\_KEY
- DB\_KEYANDUPDATABLE
- DB\_KEYANDMODIFIED (la predeterminada)
- DB\_KEYANDTIMESTAMP

Si elige una de estas cuatro configuraciones, puede controlar cómo genera Visual FoxPro la cláusula WHERE de la instrucción SQL Update enviada a las tablas base de la vista. Puede elegir la configuración que desee mediante la ficha [Criterios de actualización](#) del Diseñador de vistas o puede usar [DBSETPROP\(\)](#) para establecer WhereType para una definición de vista. Para cambiar la configuración de WhereType para un cursor en la vista activa, use [CURSORSETPROP\(\)](#).

Por ejemplo, suponga que tiene una vista remota sencilla basada en la tabla Customer que incluye siete campos: cust. id, company, phone, fax, contact, title y timestamp. La clave principal de la

Los campos `cust_id`, `company`, `phone`, `fax`, `contact`, `title` y `address` de la vista principal de la vista es `cust_id`.

### La ficha Criterios de actualización muestra los campos actualizables de la vista



Sólo ha establecido como actualizables dos campos: `contact_name` y `contact_title`. Desea que el usuario pueda cambiar el nombre de la persona de contacto de la empresa y su cargo desde la vista. Sin embargo, si cambian otros datos de la compañía, como su dirección, los cambios deben pasar a un coordinador que identificará el efecto de los cambios en la empresa, como si va a cambiar la región de ventas del cliente. Ahora que la vista se ha configurado para enviar actualizaciones, puede elegir WhereType según sus preferencias.

Ahora suponga que cambia el nombre del campo `contact` para un cliente, pero no cambia el valor en el otro campo actualizable, `title`. Según este ejemplo, la sección siguiente trata sobre cómo afectará la configuración de WhereType a la cláusula WHERE que Visual FoxPro genera para enviar el nombre del nuevo contacto a las tablas base.

### Comparar únicamente el campo clave

La actualización menos restrictiva usa la configuración `DB_KEY`. La cláusula WHERE utilizada para actualizar tablas remotas consta únicamente del campo clave principal con la propiedad `KeyField` o `KeyFieldList`. A menos que se haya modificado o eliminado el campo clave principal en la tabla base desde que se recuperó el registro, se inicia el proceso de actualización.

En el caso del ejemplo anterior, Visual FoxPro prepararía una instrucción de actualización con una cláusula WHERE que compara el valor del campo `cust_id` con el del campo `cust_id` en la fila de la tabla base:

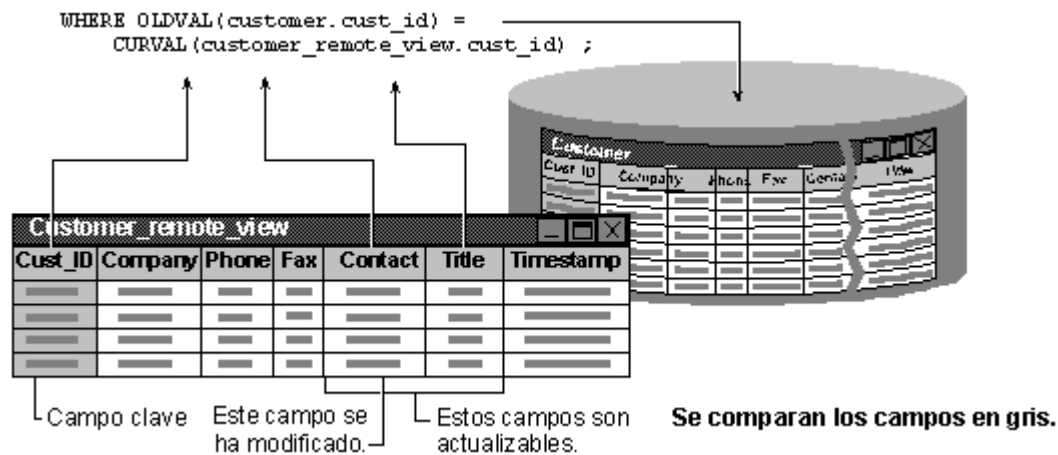
```
WHERE CURSOR (customer.cust_id) = CURSOR (customer_remote.cust_id)
```



```
WHERE OLDVAL(customer.cust_id) = CURVAL(customer_remote_view.cust_id)
```

Cuando la instrucción de actualización se envía a la tabla base, sólo se comprueba el campo clave.

### El campo clave de la vista se compara con el campo correspondiente de la tabla base

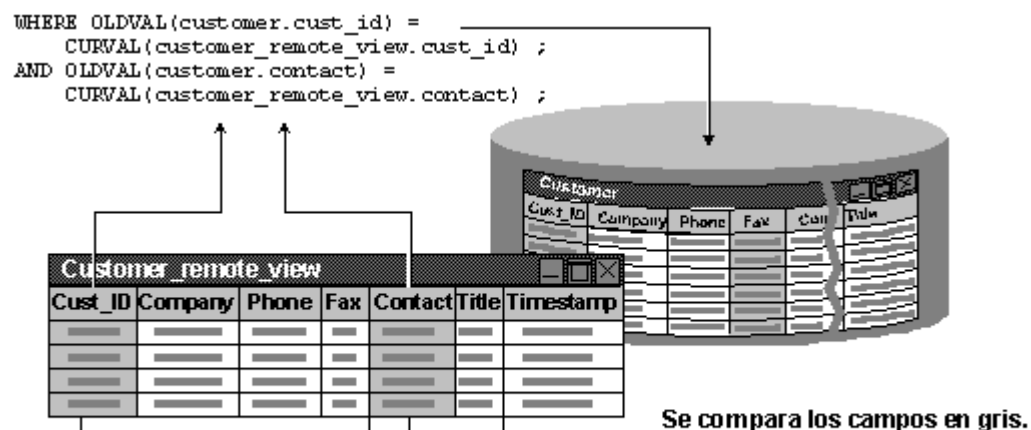


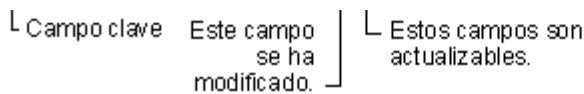
### Comparar los campos clave y los campos modificados en la vista

La configuración DB\_KEYANDMODIFIED, que es la predeterminada, es ligeramente más restrictiva que DB\_KEY. DB\_KEYANDMODIFIED compara tan sólo el campo clave y los campos actualizables que se han modificado en la vista con sus campos correspondientes en la tabla base. Si modifica un campo en la vista, pero no es actualizable, los campos no se comparan con los datos de la tabla base.

La cláusula WHERE utilizada para actualizar tablas base consta de los campos principales especificados con la propiedad KeyFieldList y de todos los campos que se hayan modificado en la vista. En el caso del ejemplo anterior, Visual FoxPro preparará una instrucción de actualización que compare los valores del campo cust\_id (porque es el campo principal) y el campo contact, porque se ha cambiado el nombre de la persona de contacto. Aun cuando sea actualizable, el campo title no se incluye en la instrucción de actualización porque no se ha modificado.

### Los campos clave y modificados de la vista se comparan con sus campos correspondientes de la tabla base



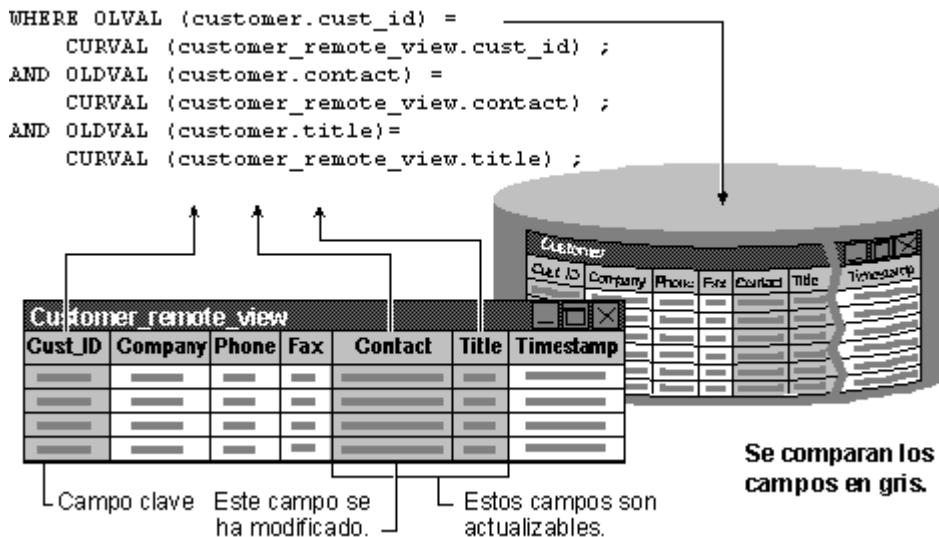


### Comparar el campo clave y todos los campos actualizables

La configuración DB\_KEYANDUPDATABLE compara el campo clave y todos los campos actualizables (independientemente de si se han modificado o no) de la vista con sus correspondientes en la tabla base. Si el campo es actualizable y no se ha modificado en la vista pero sí en la tabla base, se produce un fallo en la actualización.

La cláusula WHERE utilizada para actualizar tablas base consta de campos principales especificados con la propiedad KeyField o KeyFieldList y de todos los demás campos que son actualizables. En el caso del ejemplo, Visual FoxPro prepararía una instrucción de actualización que compara los valores de los campos cust\_id, contact y title con los mismos campos de la fila de la tabla base.

### Todos los campos actualizables de la vista se comparan con sus homólogos de la tabla base



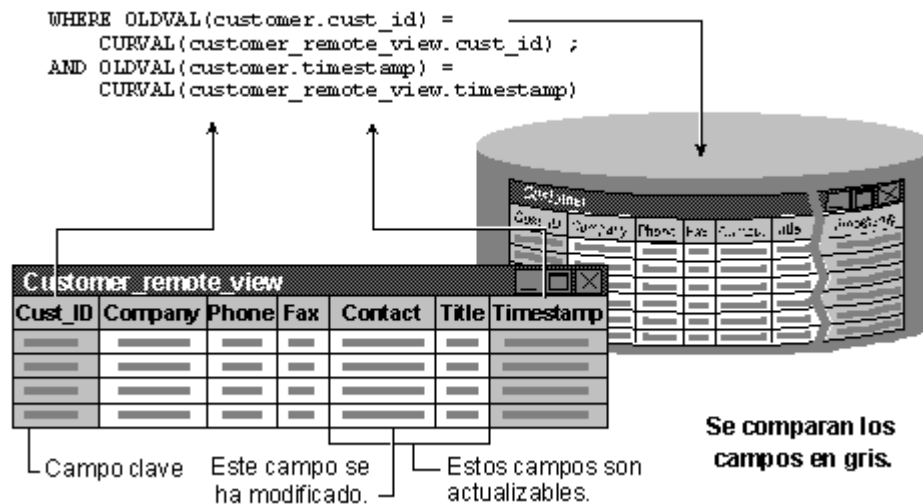
### Comparar la marca de hora de todos los campos de un registro de la tabla base

DB\_KEYANDTIMESTAMP es el tipo de actualización más restrictivo y sólo está disponible si la tabla base tiene una columna de marca de hora. Visual FoxPro compara la marca de hora actual del registro de la tabla base con la marca de la hora en que los datos se incorporaron a la vista. Si se modifica un campo del registro de la tabla base, aunque no sea el campo que se pretende cambiar, o incluso un campo de la vista, se produce un fallo en la actualización.

En el caso del ejemplo, Visual FoxPro prepara una instrucción de actualización que compara los valores del campo cust\_id y el valor del campo timestamp con los de los campos de la fila de la

... el valor del campo `cust_id` y el valor del campo `timestamp` con los de los campos de la misma en la tabla base.

**La marca de hora del registro de la vista se compara con la marca de hora del registro de la tabla base.**



Para actualizar correctamente los datos mediante la configuración `DB_KEYANDTIMESTAMP` con una vista en la que haya múltiples tablas, debe incluir el campo de marca de hora en la vista para todas las tablas que sean actualizables. Por ejemplo, si tiene tres tablas en una vista y desea actualizar sólo dos y elige la configuración `DB_KEYANDTIMESTAMP`, debe bajar los campos de marca de hora de las dos tablas al conjunto de resultados. También puede usar valores lógicos en la propiedad `CompareMemo` para determinar si los campos memo se incluyen en la detección de conflictos.

## Administrar conflictos

Cuando elija el almacenamiento en búfer, transacciones o vistas, debe administrar los conflictos que puedan surgir durante el proceso de actualización.

### Administrar conflictos en el almacenamiento en búfer

Puede hacer que las operaciones de actualización de datos sean más eficientes si elige cuidadosamente cómo y cuándo abrir, almacenar en búfer y bloquear datos en un entorno multiusuario. Deberá limitar el tiempo que un registro o una tabla están sometidos a conflictos de acceso. A pesar de todo, tendrá que anticiparse a los conflictos que se producen de forma inevitable y administrarlos. Un *conflicto* se produce cuando un usuario intenta bloquear un registro o una tabla que ya ha bloqueado otro usuario. Dos usuarios no pueden bloquear el mismo registro o la misma tabla a la vez.

La aplicación deberá contener una rutina para administrar estos conflictos. Si la aplicación no tiene una rutina de conflictos, el sistema puede quedar bloqueado. Se produce un *punto muerto* cuando un usuario ha bloqueado un registro o una tabla e intenta bloquear otro registro ya bloqueado por un segundo usuario que, a su vez, está intentando bloquear el registro bloqueado por el primer usuario. Aunque esto no suele ocurrir, cuanto más tiempo permanezca bloqueado un registro o una tabla, mayores posibilidades habrá de que se produzca un punto muerto.

## Detectar errores

Diseñar una aplicación multiusuario o agregar soporte de red a un sistema de un solo usuario exige interceptar colisiones y errores. El uso de los búferes de registro y tabla de Visual FoxPro simplifica parte de este trabajo.

Si intenta bloquear un registro o una tabla que ya están bloqueados por otro usuario, Visual FoxPro devolverá un mensaje de error. Puede utilizar [SET REPROCESS](#) para resolver automáticamente los intentos fallidos de bloqueo. Este comando, en combinación con una rutina [ON ERROR](#) y el comando [RETRY](#) permite continuar o cancelar los intentos de bloqueo.

El ejemplo siguiente demuestra el reprocesamiento automático de una operación fallida mediante SET REPROCESS.

### Ejemplo del uso de SET REPROCESS y ON ERROR para administrar colisiones de usuarios

Código	Comentarios
<pre>ON ERROR DO rep_err ERROR(),MESSAGE() SET EXCLUSIVE OFF SET REPROCESS TO AUTOMATIC USE customer IF !FILE('copi_cli.dbf')     COPY TO cus_copy ENDIF</pre>	<p>Esta rutina se ejecuta si se produce un error.</p> <p>Abre los archivos de forma no exclusiva.</p> <p>El reprocesamiento de bloqueos fallidos es automático.</p> <p>Abre la tabla.</p> <p>Si es necesario, crea la tabla APPEND FROM.</p>
<pre>DO anexar_blanco DO reem_sig DO reem_todo DO reem_act DO anexar_regs</pre>	<p>La rutina principal comienza aquí.</p> <p>Estos comandos son ejemplos de los códigos que podrían ejecutarse en el curso del programa.</p>
<pre>ON ERROR</pre>	<p>La rutina principal termina aquí.</p>
<pre>PROCEDURE anexar_blanco     APPEND BLANK RETURN ENDPROC</pre>	<p>Rutina que anexa un registro en blanco.</p>
<pre>PROCEDURE reem_sig     REPLACE NEXT 1 contact WITH ;         PROPER(contact) RETURN ENDPROC</pre>	<p>Rutina que reemplaza los datos del registro actual.</p>

```

PROCEDURE reem_todo
  REPLACE ALL contact WITH ;
  PROPER(contact)
  GO TOP
RETURN
ENDPROC

```

Rutina que reemplaza los datos de todos los registros.

```

PROCEDURE reem_act
  REPLACE contact WITH PROPER(contact)
RETURN
ENDPROC

```

Rutina que reemplaza los datos del registro actual.

```

PROCEDURE anexar_reg
  APPEND FROM copi_cli
RETURN
ENDPROC

```

Rutina que anexa registros de otro archivo.

El ejemplo siguiente muestra un procedimiento de error que se inicia cuando el usuario presiona ESC.

### Ejemplo de tratamiento de errores mediante la tecla ESC

Código	Comentario
<pre> PROCEDURE rep_err   PARAMETERS errnum, msg </pre>	Este programa se activa cuando surge un error y el usuario presiona la tecla ESC durante el proceso de espera.
<pre> DO CASE </pre>	Averigua de qué tipo de error se trata. ¿Es "Archivo utilizado por otra persona"?
<pre>   CASE errnum = 108     línea1 = "El archivo no se puede bloquear."     línea2 = " Inténtelo más tarde..." </pre>	
<pre>   CASE errnum = 109 .OR. errnum = 130     línea1 = "El registro no se puede bloquear."     línea2 = "Inténtelo más tarde." </pre>	¿Es "Registro utilizado por otra persona"?
<pre>   OTHERWISE     línea1 = msg + " "     línea2 = ;     "Póngase en contacto con el administrador del sistema."   ENDCASE </pre>	¿O es desconocido?
<pre> =MESSAGEBOX( Línea1 + línea2, 48, "Error" RETURN </pre>	Muestra el mensaje de error en un cuadro de diálogo con el signo de exclamación y el botón "Aceptar".

---

## Detectar y resolver conflictos

Durante las operaciones de actualización de datos, sobre todo en entornos compartidos, puede resultar conveniente determinar qué campos han cambiado o cuáles son los valores originales o actuales de los campos modificados. El almacenamiento en búfer de Visual FoxPro y las funciones [GETFLDSTATE\(\)](#), [GETNEXTMODIFIED\(\)](#), [OLDVAL\(\)](#) y [CURVAL\(\)](#) permiten determinar qué campo ha cambiado, buscar los datos modificados y comparar los valores actuales, originales y modificados para decidir cómo solucionar un error o un conflicto.

### Para detectar un cambio en un campo

- Después de una operación de actualización, utilice la función [GETFLDSTATE\(\)](#).

[GETFLDSTATE\(\)](#) funciona sobre datos no almacenados en búfer. Sin embargo, esta función es aún más efectiva cuando se ha activado el almacenamiento de registros en búfer. Por ejemplo, utilice [GETFLDSTATE\(\)](#) en el código de un botón "Ignorar" de un formulario. Cuando mueva el puntero de registro, Visual FoxPro comprobará el estado de todos los campos del registro como en el ejemplo siguiente:

```
lModificado = .F.
FOR nNúmCampo = 1 TO FCOUNT( )    && Comprobar todos los campos
    if GETFLDSTATE(nNúmCampo) = 2    && modificado
        lModificado = .T.
        EXIT    && Insertar aquí rutina para actualizar/guardar
    ENDIF    && Vea el ejemplo siguiente
ENDFOR
```

### Para detectar y localizar un registro modificado en datos almacenados en búfer

- Utilice la función [GETNEXTMODIFIED\(\)](#).

[GETNEXTMODIFIED\(\)](#), con cero como parámetro, busca el primer registro modificado. Si otro usuario realiza cambios en la tabla almacenada en búfer, los cambios que encuentre el comando [TABLEUPDATE\(\)](#) en el búfer ocasionarán conflictos. Puede evaluar los valores conflictivos y resolverlos mediante las funciones [CURVAL\(\)](#), [OLDVAL\(\)](#) y [MESSAGEBOX\(\)](#). [CURVAL\(\)](#) devuelve el valor actual de registro en el disco, mientras que [OLDVAL\(\)](#) devuelve el valor del registro en el momento en que se almacenó en el búfer.

### Para determinar el valor original de un campo almacenado en búfer

- Utilice la función [OLDVAL\(\)](#).

[OLDVAL\(\)](#) devuelve el valor de un campo almacenado en búfer.

### Para determinar el valor actual de un campo almacenado en búfer en el disco

- Utilice la función [CURVAL\(\)](#).

[CURVAL\(\)](#) devuelve el valor actual en el disco de un campo almacenado en búfer antes de realizar

modificaciones.

Puede crear un procedimiento de control de errores que compare los valores actual y original, lo que le permitirá determinar si desea grabar el cambio actual como definitivo o aceptar un cambio anterior en los datos de un entorno compartido.

El ejemplo siguiente utiliza GETNEXTMODIFIED( ), CURVAL( ) y OLDVAL( ) para proporcionar al usuario una opción informada en una operación de actualización. Este ejemplo continúa a partir de la detección del primer registro modificado y puede estar contenido en un botón "Actualizar" o "Guardar" de un formulario.

### Código del evento Click para un botón Actualizar o Guardar

Código	Comentarios
DO WHILE GETNEXTMODIFIED(nCurRec) <> 0 GO nCurRec RLOCK( )	Hace un bucle a través de  Bloquea el registro modif
FOR nField = 1 TO FCOUNT(cAlias) cField = FIELD(nField) IF OLDVAL(cField) <> CURVAL(cField) nResult = MESSAGEBOX("Otro usuario ha; modificado los datos. ; ¿Desea conservar las modificaciones?", 4+48+0, ; "Registro modificado")	Busca el conflicto.  Compara el valor original valor actual del disco y, a continuación, pregunta al qué debe hacer en relació conflicto.
IF nResult = 7 TABLEREVERT(.F.) UNLOCK RECORD nCurRec ENDIF ENDIF ENDFOR nCurRec = GETNEXTMODIFIED(nCurRec) ENDDO	Si el usuario selecciona 'N invierte este registro y, a continuación, elimina el t
	Busca el siguiente registro modificado.
TABLEUPDATE(.T., .T.)	Fuerza la actualización de los registros.

### Detectar conflictos con campos memo

Puede usar la propiedad CompareMemo para controlar cuándo deben usarse campos memo para detectar conflictos en la actualización. Esta propiedad de la vista y del cursor determina si los campos memo (de los tipos M o G) están incluidos en la cláusula WHERE de actualización. La configuración predeterminada, Verdadero (.T.), significa que los campos memo están incluidos en la cláusula WHERE. Si establece esta propiedad como Falso (.F), los campos memo no participan en la cláusula WHERE de actualización, independientemente de la configuración de UpdateType.

La detección optimista de conflictos de los campos memo se desactiva cuando CompareMemo se establece como Falso. Para activarla, establezca CompareMemo como Verdadero (.T.).

## Reglas para administrar conflictos

La administración de los conflictos encontrados en entornos compartidos pueden requerir código extenso y repetitivo. Una rutina completa de administración de conflictos hace lo siguiente:

- Detecta un conflicto
- Identifica la naturaleza y la ubicación del conflicto
- Proporciona suficiente información para que el usuario pueda resolver el conflicto correctamente

Para obtener un ejemplo de una rutina de administración de conflictos, vea la clase del comprobador de datos en Samples.vcx, ubicado en el directorio ...\\Samples\\Vfp98\\Classes de Visual Studio. Simplemente agregue la clase a un formulario y llame al método CheckConflicts antes de realizar cualquier operación que escriba datos almacenados en búfer en la tabla, por ejemplo, mover el puntero de registro si está utilizando el almacenamiento de filas en búfer, cerrar una tabla o ejecutar TABLEUPDATE().

# Capítulo 18: Programar aplicaciones internacionales

Para entrar en el mercado mundial, deberá diseñar las aplicaciones de Visual FoxPro de modo que sean tan efectivas a nivel internacional como en su propio país. En este capítulo se describe el uso de las características internacionales de Visual FoxPro para generar aplicaciones para [configuraciones regionales](#) seleccionadas.

En este capítulo se tratan los temas siguientes:

- [Diseñar una aplicación internacional](#)
- [Diseñar la interfaz](#)
- [Escribir datos internacionales](#)
- [Trabajar con páginas de códigos](#)
- [Ordenar datos en aplicaciones internacionales](#)
- [Trabajar con juegos de caracteres codificados en dos bytes](#)
- [Crear o modificar programas](#)
- [Administrar archivos en una aplicación internacional](#)

## Diseñar una aplicación internacional

Para diseñar una aplicación internacional suelen darse estos tres pasos: crear datos, escribir código y diseñar una interfaz de usuario. Sin embargo, para dar estos pasos es necesario considerar las siguientes preguntas:



- ¿Qué datos son aceptables?
- ¿Cómo se escribe código para una aplicación internacional?
- ¿Qué debe tenerse en cuenta a la hora de diseñar una interfaz de usuario?

En las secciones siguientes se da respuesta a estas preguntas y se plantean otras que debe tener en cuenta antes de diseñar la aplicación.

**Sugerencia** Puede reducir el coste de programación de una aplicación internacional y ponerla en el mercado más rápidamente si la diseña como aplicación internacional desde el principio, en lugar de modificarla posteriormente para su uso internacional.

## Diseñar datos internacionales

Para crear datos internacionales para una aplicación, deberá escribirlos manualmente, importarlos de otras aplicaciones o anexarlos a archivos y campos memo existentes. Para obtener más información sobre cómo importar y anexar datos, consulte el capítulo 9, [Importar y exportar datos](#), del *Manual del usuario*.

### Qué datos son aceptables

Para decidir qué datos son aceptables, determine en primer lugar en qué país se va a utilizar la aplicación. La configuración propia de cada país determina el contenido cultural de los datos, así como los idiomas en los que éstos se diseñan.

Asimismo, los idiomas afectarán a la [página de códigos](#) con la que se diseñan los datos. Una página de códigos es un juego de caracteres que utiliza el PC para mostrar correctamente los datos, a menudo para tratar los *caracteres internacionales*. Los caracteres internacionales incluyen caracteres que tienen [marcas diacríticas](#). Estas marcas se colocan encima, debajo o entre letras para indicar cambios de sonido con respecto a la forma sin marca. Las marcas diacríticas más frecuentes son el acento grave (` como en `a), el acento agudo (´ como en ´a), el acento circunflejo (^ como en ^a), la tilde (~ como en ã), la diéresis (¨ como en ¨a), el anillo (° como en °a) y la barra inclinada (/ como en /a), siempre usadas en las vocales.

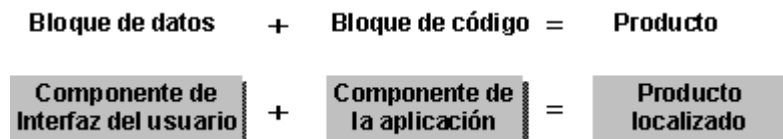
Normalmente los datos se marcan automáticamente con la página de códigos apropiada cuando se trabaja con ellos. Sin embargo, si asigna manualmente una página de códigos a una tabla o produce un cambio en la misma, los usuarios podrían no reconocer algunos o todos los datos mostrados. Para obtener detalles sobre las páginas de códigos, consulte [Trabajar con páginas de códigos](#) más adelante en este capítulo.

Algunos idiomas, como el chino, el coreano o el japonés, usan *DBCS* (juegos de caracteres codificados en dos bytes) para representar sus datos. Si la aplicación pudiera ejecutarse en estos entornos, podría necesitar funciones de tratamiento de cadenas especiales y secuencias de ordenación para que la aplicación funcionase correctamente. Para obtener más detalles sobre cómo trabajar en entornos DBCS, consulte [Trabajar con juegos de caracteres codificados en dos bytes](#) más adelante en este capítulo.

### Cómo escribir código

Una aplicación consta de un componente de interfaz de usuario y un componente de aplicación. El componente de interfaz de usuario contiene gráficos, cadenas de texto y opciones relacionadas con la configuración propia de cada país, como fechas, monedas, valores numéricos y separadores. El componente de aplicación contiene el código que se ejecuta para todas las configuraciones nacionales, incluido el código que procesa las cadenas y los gráficos empleados en la interfaz de usuario.

### Los componentes de una aplicación



Cuando diseñe una aplicación, mantenga separados los componentes de aplicación y de interfaz de usuario, ya que disponer de componentes independientes facilita la adaptación y el mantenimiento de la aplicación. Por ejemplo, con componentes separados no es necesario examinar el código fuente para adaptar elementos de la interfaz. Para obtener más información sobre cómo escribir código, consulte [Crear o modificar programas](#), más adelante en este mismo capítulo.

### Diseñar una interfaz de usuario

Los menús, cuadros de diálogo, mensajes, iconos y mapas de bits empleados en la interfaz de usuario deben ser acordes a la configuración propia del país para el que diseña la aplicación. Por ejemplo, si diseña la aplicación para usuarios de Alemania o Francia, los cuadros de diálogo deberán ser lo bastante grandes para mostrar correctamente las instrucciones, si éstas se traducen al alemán o al francés. Asimismo, las imágenes utilizadas en iconos y mapas de bits deben ser culturalmente correctas para que sean comprensibles en los países de destino. Para obtener más información sobre el diseño de interfaces de usuario, consulte [Diseñar la interfaz](#), más adelante en este mismo capítulo.

### Probar la aplicación

Para comprobar una aplicación internacional, debe comprobar las dependencias de país e idioma de la configuración regional para la que está diseñada la aplicación. La comprobación implica asegurar que los datos y la interfaz de usuario de la aplicación presentan conformidad con los estándares de la configuración regional en cuanto a la fecha y la hora, los valores numéricos, los separadores de lista y las medidas.

## Diseñar la interfaz

Puesto que el texto tiende a aumentar cuando se adapta una aplicación a otro país, tome las debidas precauciones al diseñar los siguientes componentes de la interfaz de usuario:

- Mensajes de la aplicación
- Menús y formularios
- Iconos y mapas de bits

### Crear mensajes de la aplicación

## Crear mensajes de la aplicación

Cuando cree mensajes en su aplicación, tenga en cuenta que la longitud de las cadenas de texto varía de un idioma a otro. La tabla siguiente muestra el crecimiento medio de las cadenas, a partir de su longitud inicial.

Longitud en inglés (en caracteres)	Crecimiento para cadenas traducidas
1 a 4	100%
5 a 10	80%
11 a 20	60%
21 a 30	40%
31 a 50	20%
más de 50	10%

## Diseño de menús y formularios

Al igual que los mensajes, los menús y los cuadros de diálogo pueden aumentar de tamaño cuando se traduce la aplicación. Por ejemplo, observe los formularios siguientes, que forman parte de una aplicación de ejemplo para cajeros automáticos. La primera figura muestra el formulario en inglés y la segunda su equivalente en español. Puede ver el espacio adicional que se ha reservado para el aumento del texto en el formulario.

**Sugerencia** Si deja espacio para el aumento del texto en una interfaz, los traductores tendrán que dedicar menos tiempo a reajustar el tamaño de los controles y a volver a diseñar la interfaz.

**El texto ocupa más espacio cuando se traduce.**

Welcome

Please enter your pin number:

\*\*\*\*

Please choose an account:

☐ Savings account

☐ Checking account

OK

Bienvenido!!

Por favor, introduzca su numero de indentificacion:

\*\*\*\*

Por favor, elija una cuenta:

☐ Cuenta de ahorros

☐ Cuenta corriente

Aceptar

En los menús y formularios, evite barras de estado con mucho texto. Asimismo, evite las abreviaturas, ya que es posible que no existan en otros idiomas.

## Usar iconos y mapas de bits

Si se utilizan correctamente, los iconos y los mapas de bits pueden formar una parte importante de la interfaz de usuario. Sin embargo, el significado de los iconos y los mapas de bits puede ser más ambiguo que el del texto. Por tanto, siga estas directrices cuando utilice iconos y mapas de bits:

- Utilice imágenes universalmente reconocidas. Por ejemplo, utilice un sobre para representar correo, pero no utilice un buzón porque no es un símbolo universal.
- Utilice imágenes respetuosas con las culturas. Por ejemplo, evite utilizar imágenes de símbolos religiosos y animales.
- Evite el uso de texto en mapas de bits, ya que el aumento del texto puede representar un problema, al igual que en otras partes de la interfaz.
- Evite la jerga, el lenguaje vulgar, el humor, el lenguaje extravagante y los estereotipos étnicos.
- Use Información sobre herramientas para explicar los iconos, que tienen la ventaja adicional de poder aumentar de tamaño automáticamente para adaptarse al texto que muestran.
- Si utiliza imágenes de hombres y mujeres, asegúrese de que su sexo y el cargo que desempeñan son adecuados, y que los gestos y las imágenes del cuerpo humano son adecuados en la cultura de destino.
- Utilice el color correctamente. Por ejemplo, evite utilizar combinaciones de colores asociadas a banderas nacionales o movimiento políticos.

Si no sabe con seguridad si un icono o un mapa de bits resulta apropiado, consulte con un ciudadano del país para el que está diseñando la aplicación.

## Escribir datos internacionales

Un aspecto importante de la programación de aplicaciones internacionales es saber cómo escribir datos en la aplicación. Los datos se incorporan a la aplicación de dos maneras:

- El usuario escribe los datos.
- Usted o los usuarios importan los datos de archivos existentes.

En las secciones siguientes se tratan estos dos métodos.

## Escribir caracteres internacionales

Puede escribir caracteres internacionales en Visual FoxPro mediante el teclado. El método exacto que debe utilizar depende del idioma en el que esté trabajando. En entornos de caracteres de un único byte, puede escribir los caracteres directamente o presionar una combinación de teclas del teclado. Por otro lado, los entornos DBCS suelen proporcionar un Editor de métodos de entrada (Input Method Editor, IME), que es una aplicación que puede utilizar para escribir caracteres.

### Escribir caracteres mediante el teclado

Con un teclado internacional, puede mostrar caracteres internacionales presionando simplemente las teclas dedicadas a esos caracteres. Si su teclado no dispone de teclas para caracteres internacionales, podrá escribirlos empleando el mapa de caracteres de Windows o presionando la tecla alt en combinación con las teclas del teclado numérico auxiliar.

El modo más sencillo de escribir un carácter internacional consiste en copiarlo desde el mapa de caracteres. En Windows 95, el mapa de caracteres está disponible en el menú Accesorios.

También puede escribir un carácter internacional si presiona alt en combinación con un número de cuatro dígitos que empieza por cero y se introduce desde el teclado numérico.

**Nota** No puede escribir caracteres internacionales en FoxFont. Por ejemplo, si abre la ventana Comandos, cambia a FoxFont y presiona una tecla dedicada, el resultado no es el carácter impreso en la tecla. Para obtener los mejores resultados, evite el uso de FoxFont en aplicaciones internacionales.

### Para crear un carácter internacional

- Copie el carácter desde el mapa de caracteres y péguelo en el documento.  
—O bien—
- Mantenga presionada la tecla alt y escriba un cero seguido del código ASCII de tres dígitos.

**Sugerencia** La barra de estado del mapa de caracteres muestra la combinación de teclas correspondiente a cada carácter seleccionado en el mapa.

Por ejemplo, para escribir ö (código ASCII 246), presione bloq num en el teclado numérico y, a continuación, alt+0246. Asegúrese de que utiliza una fuente estándar de Windows, no FoxFont o FoxPrint.

**Solución de problemas** Si los caracteres no se transportan correctamente, compruebe si está utilizando FoxFont. Por ejemplo, FoxFont es la opción predeterminada para las ventanas definidas por el usuario que se crean mediante [DEFINE WINDOW](#) (si se omite la cláusula FONT). Utilice la cláusula FONT para especificar una fuente distinta de la estándar de Windows al crear ventanas definidas por el usuario, de modo que los caracteres internacionales se muestren correctamente.

**Escribir caracteres mediante un IME**

### **Escribir caracteres mediante un IME**

Si trabaja en un entorno IME, puede usar un Editor de métodos de entrada para escribir caracteres en Visual FoxPro. El IME es una aplicación suministrada con el entorno que permite escribir caracteres del teclado para mostrar una selección de caracteres internacionales y elegir el carácter que desea. Por ejemplo, un IME para chino podría permitirle escribir una representación Pinyin de una palabra china y después mostrar una lista de caracteres que coinciden con la representación. Cuando seleccione el carácter que desee, el IME lo pega en Visual FoxPro.

Puede controlar cuándo muestra Visual FoxPro un IME si establece la propiedad [IMEMode](#) o llama a la función [IMESTATUS\(\)](#). Si activa la ventana del IME, Visual FoxPro muestra automáticamente el IME cuando realice operaciones de modificación en una ventana del sistema, como las ventanas Examinar y Modificar. Si desactiva la ventana del IME, puede invocar al editor presionando las teclas apropiadas del teclado.

### **Agregar y copiar datos internacionales**

Si importa o copia datos de archivos delimitados mediante los comandos [APPEND FROM](#) o [COPY TO](#), puede especificar qué carácter se está utilizando en el archivo para separar los campos. Por ejemplo, es frecuente en muchos países europeos usar un punto y coma (;) como delimitador de campos, mientras que los delimitadores más habituales en Estados Unidos son la coma (,), el tabulador o el espacio.

Para importar o copiar archivos y especificar un delimitador, agregue la cláusula DELIMITED WITH CHARACTER a los comandos APPEND FROM o COPY TO:

```
COPY TO mitxt.txt DELIMITED WITH _ WITH CHARACTER ";"
```

## **Trabajar con páginas de códigos**

Los datos almacenados en Visual FoxPro suelen estar etiquetados con una *página de códigos*, que es una tabla de caracteres y los números correspondientes en memoria que utiliza Windows para mostrar datos correctamente. Por ejemplo, si introduce la letra C en un archivo .dbf, la letra se almacena en el disco duro como el número 67. Cuando abra el archivo, Visual FoxPro determina su página de códigos, la inspecciona para buscar el carácter correspondiente al número 67 y muestra el carácter (C) en el monitor.

Las páginas de códigos corresponden aproximadamente a cuatro alfabetos diferentes. Por ejemplo, Windows suministra páginas de códigos para los idiomas inglés, alemán, escandinavos, etc. Usando una página de códigos diferente, las aplicaciones pueden mostrar correctamente caracteres de estos diferentes alfabetos.

### **Páginas de códigos en Visual FoxPro**

Visual FoxPro muestra datos mediante una única página de códigos que, como opción predeterminada, es la página de códigos utilizada por Windows. Sin embargo, puede anular la página de códigos de Windows si especifica otra página alternativa en el archivo de configuración (debe especificar una página de códigos válida).

Las tablas de Visual FoxPro se etiquetan con la página de códigos que estaba en uso cuando se creó la tabla. Cuando use la tabla, Visual FoxPro comprueba la página de códigos de la tabla con la página de códigos actual. Si coinciden, Visual FoxPro muestra los datos tal como son. Si no existe ninguna página de códigos para la tabla (por ejemplo, la tabla se creó en una versión anterior de FoxPro), Visual FoxPro solicita una página de códigos y marca el archivo con esa página.

Si la página de códigos de la tabla no coincide con la del sistema, Visual FoxPro intenta traducir los caracteres de la primera a la página de códigos actual. Por ejemplo, si está utilizando Visual FoxPro y la página de códigos actual del sistema es la inglesa, el carácter ù se representa por el valor ANSI 252. Si la página de códigos de la tabla representa el carácter ù como el valor ANSI 219, Visual FoxPro traduce todas las instancias del valor ANSI 219 a ANSI 252 para que se muestren correctamente.

Las traducciones de las páginas de códigos no funcionan siempre correctamente porque las páginas suelen contener caracteres que no se representan uno a uno en otras páginas de códigos. Por ejemplo, no puede asignar datos que contienen los caracteres de dibujo de línea de MS-DOS® en Windows, porque las páginas de códigos de Windows no contienen estos caracteres. Tampoco puede traducir datos creados en la página de códigos del ruso en una página de códigos del inglés, porque no existe una correspondencia mutua entre los alfabetos de estos idiomas. Por último, Visual FoxPro no podría contener un mapa de traducción de caracteres para una página de códigos determinada. En ese caso, los datos se muestran sin traducción de página de códigos. Visual FoxPro no muestra ningún error que indique que no se está produciendo traducción de páginas de códigos. Cualquiera de las situaciones anteriores pueden hacer que algunos caracteres se muestren de forma indebida.

Si desea crear una aplicación para una configuración regional determinada, puede evitar los problemas de traducción de las páginas de códigos si crea los componentes de la aplicación con la página de códigos diseñada para esa configuración regional y ese entorno. Por ejemplo, para crear una aplicación destinada a Rusia, deberá utilizar la página de códigos 1251 o 866 para los usuarios de entornos Windows o MS-DOS, respectivamente. Para obtener una lista completa, consulte [Páginas de códigos compatibles con Visual FoxPro](#).

Si necesita escribir algunos caracteres no representados por teclas del teclado, puede hacerlo usando ALT junto con una combinación de teclas del teclado numérico. Sin embargo, no olvide que la misma combinación de teclas usada en diferentes entornos suele presentar resultados distintos. Por ejemplo, si introduce ALT+0182 con la página de códigos 1252 en Visual FoxPro, verá el símbolo del párrafo. En contraste, si introduce ALT+0182 con la página de códigos 437 en FoxPro para MS-DOS, verá un carácter gráfico con una doble línea vertical que se junta con una línea horizontal sencilla.

Aunque Visual FoxPro admite muchas páginas de códigos, sólo se suelen utilizar unas pocas. Con Visual FoxPro para Windows, por ejemplo, los usuarios de habla inglesa utilizan normalmente la página de códigos 1252. Sin embargo, en FoxPro para MS-DOS, suelen utilizar la página de códigos 437.

Al trabajar con páginas de códigos, no olvide comprobar que la interfaz de usuario y los datos se muestran correctamente usando la página de códigos diseñada para una configuración regional determinada. Si ve caracteres inesperados en la pantalla, compruebe la página de códigos asociada.

## Especificar la página de códigos para archivos .dbf

Al crear archivos .DBF, Visual FoxPro les asigna automáticamente marcas de página de códigos para distinguir qué página de códigos utilizan. Sin embargo, si utiliza archivos .dbf procedentes de versiones anteriores de FoxPro, es posible que no tengan marcas de página de códigos.

Puede determinar si un archivo .dbf tiene una marca de página de códigos mediante la función [CPDBF\(\)](#) después de abrir el archivo o haciendo que Visual FoxPro lo compruebe cuando abra el archivo.

### Para comprobar las marcas de páginas de códigos automáticamente

1. En el menú **Herramientas**, elija **Opciones**.
2. Seleccione la ficha [Datos](#).
3. Active la casilla de verificación **Pedir página de códigos**, si no lo ha hecho todavía.

Si desea guardar esta configuración para futuras sesiones de Visual FoxPro, elija **Establecer como predeterminado**.

**Sugerencia** En lugar de activar la casilla de verificación **Pedir página de códigos** puede utilizar el comando [SET CPDIALOG](#) para comprobar las páginas de códigos.

Si un archivo no tiene ninguna marca de página de códigos, deberá agregarla tal como se describe en la sección siguiente.

### Agregar marcas de página de códigos

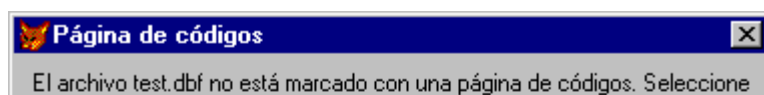
Si utiliza un archivo .dbf de una versión anterior de FoxPro, posiblemente no tenga marca de página de códigos. Sin dicha marca, es posible que el archivo no se muestre correctamente. Si está activada la comprobación automática de páginas de códigos, cuando abra el archivo puede determinar si tiene o no marca de página de códigos. Si no la tiene, puede agregarla.

### Para agregar una marca de página de códigos manualmente a un archivo .dbf

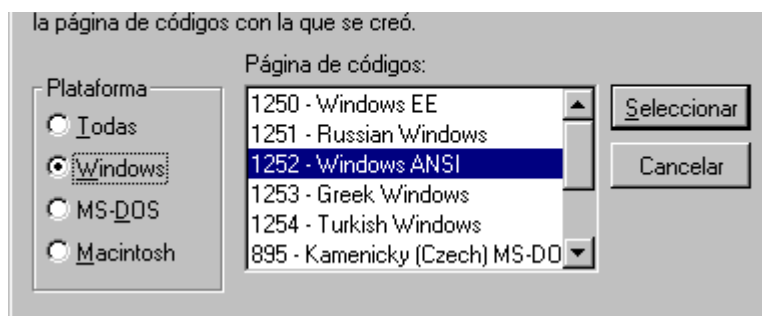
1. Compruebe si está vigente la comprobación automática de página de códigos (consulte el procedimiento anterior).
2. Abra el archivo.

Si el archivo no tiene marca de página de códigos, aparecerá el cuadro de diálogo [Página de códigos](#).

#### El cuadro de diálogo Página de códigos







3. Elija la página de códigos apropiada.
4. Examine el archivo para comprobar si ha asignado la página de códigos correcta.

Si no puede ver o reconocer algunos datos, la página de códigos no será correcta.

5. Si la página de códigos es incorrecta, quite la marca de página de códigos empleando el programa [CPZERO](#) del directorio Tools\Cpzero de Visual FoxPro. Para obtener más información al respecto, consulte [Eliminar marcas de página de códigos](#) más adelante en este mismo capítulo.
6. Repita este procedimiento hasta que la página de códigos sea correcta.

**Nota** Los archivos de texto, como por ejemplo los de programa (.prg) y consulta (.qpr), no tienen marcas de página de códigos. Esto significa que no es posible distinguir qué páginas de códigos utilizan estos archivos. Sin embargo, si los incluye en un proyecto, éste podrá mantener un registro de las páginas de códigos empleadas. Para obtener más detalles, consulte [Especificar la página de códigos de un archivo de texto](#) más adelante en este capítulo.

## Eliminar marcas de página de códigos

Si un archivo .DBF no se muestra correctamente, puede deberse a que tiene una marca de página de códigos incorrecta. Para eliminar la marca de página de códigos, utilice el programa CPZERO ubicado en Tools\Cpzero. Al ejecutar CPZERO se establecerá la página de códigos como 0, es decir, ninguna.

### Para eliminar una marca de página de códigos

- Ejecute [CPZERO](#) con la sintaxis siguiente:

DO CPZERO WITH "*nombrearchivo*", 0

**Nota** Cuando elimine la marca de página de códigos de un archivo .DBF, los datos del archivo no cambiarán. Para cambiar la página de códigos de los datos, deberá marcar el archivo con la página de códigos correcta. Para obtener más información al respecto, consulte [Agregar marcas de página de códigos](#) en una sección anterior en este mismo capítulo.

## Modificar las marcas de la página de códigos

Puede cambiar la página de códigos de un archivo .dbf si quita su marca de página de códigos y le

agrega una nueva, si copia el archivo a otro archivo o si usa el programa CPZERO.

### Para cambiar la página de códigos de un archivo .dbf copiando el archivo

- Utilice el comando [COPY TO](#) y especifique la página de códigos de destino con la cláusula AS. Para establecer la página de códigos a la página de códigos actual del sistema, omita la cláusula AS.

Por ejemplo, para copiar Prueba.dbf a Prueb866.dbf, al tiempo que cambia la página de códigos a 866, utilice los comandos siguientes:

```
USE TEST.DBF  
COPY TO TEST866.DBF AS 866
```

Cuando se complete COPY TO, los datos del archivo resultante tendrán la nueva página de códigos.

### Para cambiar una marca de página de códigos usando CPZERO

- Ejecute [CPZERO](#) con la sintaxis siguiente:

DO CPZERO WITH "*nombrearchivo*", *nuevaPáginaCódigos*

**Nota** Algunos caracteres no pueden traducirse correctamente entre páginas de códigos. Además, algunas traducciones de páginas de códigos no se admiten en Visual FoxPro. Compruebe siempre los resultados de un cambio de página de códigos para comprobar que los datos se han traducido correctamente.

### Especificar la página de códigos de un archivo de texto

Si olvida la página de códigos de un archivo de texto que no forma parte de un proyecto, no podrá determinar cuál es esta página de códigos, ya que un archivo de texto no tiene marcas de página de códigos como ocurre con los archivos .DBF. La mejor forma de recordar la página de códigos de un archivo de texto consiste en agregar el archivo a un proyecto.

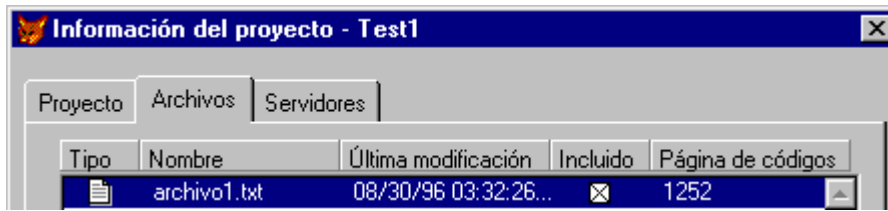
### Para especificar la página de códigos de un archivo de texto

1. Abra el [Administrador de proyectos](#).
2. Seleccione el archivo de texto cuya página de códigos desea especificar.
3. En el menú **Proyecto**, elija **Información del proyecto**.
4. En el cuadro de diálogo [Información del proyecto](#), haga clic en la ficha **Archivos**.
5. Haga clic con el botón secundario del *mouse* en el archivo seleccionado.
6. En el submenú, elija **Página de códigos**.

Visual FoxPro mostrará el cuadro de diálogo [Página de códigos](#).

7. Elija la página de códigos apropiada.

Visual FoxPro muestra las páginas de códigos disponibles.



Si sabe cuál es la página de códigos de un archivo de texto, puede especificarlo usando la cláusula **AS** del comando apropiado de Visual FoxPro. Para los archivos que desee importar o anexar, puede especificar la página de códigos en los comandos [IMPORT](#) o [APPEND](#). Para los archivos de consulta, programa o de texto de otro tipo que ya estén en su PC, puede cambiar la página de códigos mediante los comandos [MODIFY QUERY](#), [MODIFY COMMAND](#) y [MODIFY FILE](#). Para obtener información detallada sobre estos comandos, vea los temas correspondientes de la Ayuda.

Si no sabe con seguridad qué página de códigos debe aplicar, sustituya la función [GETTCP\(\)](#) por el número de página de códigos del comando. [GETTCP\(\)](#) mostrará el cuadro de diálogo Página de códigos para que seleccione la más adecuada.

**Nota** Algunos caracteres no pueden traducirse correctamente entre páginas de códigos. Además, algunas traducciones de páginas de códigos no se admiten en Visual FoxPro. Compruebe siempre los resultados de un cambio en la página de códigos para garantizar que los datos se han traducido correctamente.

## Determinar la página de códigos de un archivo de proyecto

Después de agregar un archivo a un proyecto, puede determinar su página de códigos. El método que debe utilizar depende de si el archivo es una tabla (archivo .dbf) o un archivo de texto.

### Para determinar la página de códigos de un archivo de texto

1. Abra el [Administrador de proyectos](#).
2. En **Otros** seleccione el archivo cuya página de códigos desea conocer.
3. En el menú **Proyecto**, elija **Información del proyecto**.

### Para determinar la página de códigos de una tabla

- Use la función [CPDBF\(\)](#).

Cuando genere una aplicación a partir un proyecto, el Administrador de proyectos integrará automáticamente los archivos en el proyecto, independientemente de cuántas páginas de códigos distintas tengan. La aplicación resultante tendrá la página de códigos actual.

**Nota** Cuando agregue un archivo .dbf a un proyecto, no será necesario que especifique una página de códigos para el archivo, ya que Visual FoxPro determina automáticamente la página de códigos a partir de la marca de página de códigos del archivo. Sin embargo, cuando agregue un archivo de texto a un proyecto, deberá especificar una página de códigos para el archivo, ya que Visual FoxPro no puede determinar automáticamente la página de códigos.

Para preparar un programa para que utilice otra página de códigos, especifique la página de códigos original cuando guarde o compile el programa en la nueva plataforma. Por ejemplo, si desea preparar un programa creado con Visual FoxPro para Macintosh con el fin de utilizarlo con Visual FoxPro, especifique la página de códigos apropiada de MS-DOS cuando guarde o compile el proyecto con Visual FoxPro. Si usa el comando [COMPILE](#), especifique la página de códigos mediante la cláusula AS. Otra alternativa es especificar la página de códigos con [SET CPCOMPILE](#) antes de compilar el programa.

## Especificar páginas de códigos para variables

Tal vez desee manipular datos internacionales de una determinada manera. Por ejemplo, podría traducir los datos de una variable a otra página de códigos o evitar la traducción de datos de un campo de tipo Character o Memo.

### Traducir datos de variables

Si el código de la aplicación incluye una variable que contiene datos de otra página de códigos, puede traducir los datos a la página de códigos apropiada mediante la función [CPCONVERT\(\)](#). Por ejemplo, suponga que la variable x contiene datos creados con la página de códigos de Macintosh® (10000). Para traducir los datos a la página de códigos de Windows (1252), ejecute el comando siguiente:

```
cConvert=CPCONVERT(10000,1252,x)
```

En Windows, los datos convertidos se presentan exactamente igual que en Macintosh. Por ejemplo, un carácter que se presente como "ä" en Macintosh tendrá el mismo aspecto en Windows.

## Evitar la traducción de datos de campos del tipo Character o Memo

En algunos casos, no deseará la traducción automática de páginas de códigos. Por ejemplo, si un campo de tipo Character contiene una contraseña codificada, no deseará que Visual FoxPro la traduzca automáticamente porque se podría modificar.

### Para evitar la traducción de datos en campos de tipo Character o Memo

1. Abra el proyecto que contiene la tabla.
2. Seleccione la tabla.
3. Elija el botón **Modificar**.

Aparece el [Diseñador de tablas](#).

4. Seleccione el campo cuyos datos desee proteger.
5. En la lista **Tipo**, seleccione **Carácter (binario)** para un campo del tipo CARÁCTER o **Memo (binario)** para un campo memo.
6. Elija **Aceptar** y después **Sí** para hacer que los cambios sean definitivos.
7. Compruebe los cambios mostrando la estructura de la tabla con el comando [DISPLAY STRUCTURE](#).

Otra alternativa es usar el comando [MODIFY STRUCTURE](#) para proteger los campos apropiados.

También puede evitar la traducción de caracteres seleccionados en archivos de texto mediante la función [CHR\(\)](#).

## Páginas de códigos compatibles con Visual FoxPro

Una página de códigos es un conjunto de caracteres específico de un idioma o una plataforma de hardware. Los caracteres acentuados no se representan con los mismos valores en distintas plataformas y distintas páginas de códigos. Además, algunos caracteres disponibles en una página de códigos no están disponibles en otras.

Página de códigos	Plataforma	Identificador de página de códigos
437	MS-DOS USA	x01
620 *	MS-DOS polaco	x69
737 *	MS-DOS griego (437G)	x6A
850	MS-DOS internacional	x02
852	MS-DOS de Europa del Este	x64
861	MS-DOS islandés	x67
865	MS-DOS nórdico	x66
866	MS-DOS ruso	x65
895 *	MS-DOS checo	x68
857	MS-DOS turco	x6B
1250	Windows de Europa del Este	xC8
1251	Windows ruso	xC9
1252	ANSI de Windows	x03
1253	Windows griego	xCB

Windows	Macintosh griego	Macintosh
1254	Windows turco	xCA
10000	Macintosh estándar	x04
10006	Macintosh griego	x98
10007 *	Macintosh ruso	x96
10029	Macintosh de Europa del Este	x97

\* No detectada cuando se incluye CODEPAGE=AUTO en el archivo de configuración.

## Ordenar datos en aplicaciones internacionales

Después de crear una página de datos internacionales, compruebe si la aplicación ordena correctamente los datos. El modo en que se ordenan los datos depende de la página de códigos asociada a la tabla, ya que especifica el orden o las secuencias de ordenación disponibles.

### Descripción de los tipos de ordenación

Los distintos tipos de ordenación incorporan las reglas de ordenación de la configuración de los distintos países, permitiendo ordenar correctamente los datos en esos idiomas. En Visual FoxPro, el orden actual determina los resultados de comparaciones entre expresiones de caracteres y el orden en el que los registros aparecen en tablas indexadas u ordenadas.

**Nota** La clasificación funciona de forma diferente en entornos de caracteres codificados en dos bytes (DBCS). Para obtener detalles, consulte [Ordenar datos DBCS](#) más adelante en este capítulo.

Utilice el orden adecuado, ya que cada orden produce resultados distintos, como se muestra en la tabla siguiente.

Sin ordenar	Máquina	General	Español
!@#\$	Space	space	space
1234	!@#\$	!@#\$	!@#\$
space	1234	1234	1234
Caesar	Caesar	äa	äa
cæsar	Car	ab	ab
Strasse	Char	äb	äb
straße	Czech	Caesar	Caesar
Car	Strasse	cæsar	cæsar
Char	Ab	Car	Car

Czech	Cæsar	Çar	Çar
ab	Straße	Char	Czech
Çar	Çar	Czech	Char
äa	Äa	Strasse	Strasse
äb	Äb	straße	straße

## Directrices para ordenar

Tenga en cuenta estas directrices a la hora de elegir un orden:

- Evite el orden Máquina si desea ordenar correctamente los caracteres internacionales, ya que Máquina ordena los caracteres internacionales según el orden ASCII. Por ejemplo `çar` va detrás de `straße`.
- Los caracteres con marcas diacríticas se ordenan de forma distinta que los caracteres sin estas marcas. Por ejemplo, en los órdenes General y Español `äa` se ordena antes que `ab` pero `ab` va antes que `äb`.
- Los caracteres del tipo ß se ordenan igual que los caracteres ampliados equivalentes. Por ejemplo, `straße` se ordena igual que `Strasse`, y `cæsar` igual que `Caesar`.
- En algunos idiomas, dos caracteres se ordenan como un único carácter. Por ejemplo, en Español, la `ch` de `Char` se ordena como si fuera un carácter entre `C` y `D`.

En las secciones siguientes se describe cómo especificar ordenaciones, comprobar la ordenación actual y reconocer sus resultados.

## Especificar la ordenación

Puede especificar una ordenación para los campos del tipo CARÁCTER que empleará en operaciones posteriores de indexación y ordenación.

### Para especificar un orden

1. En el menú **Herramientas**, elija **Opciones**.
2. Seleccione la ficha [Datos](#).
3. En el cuadro **Secuencia de ordenación**, seleccione el método de ordenación correspondiente.

Para guardar esta configuración para futuras sesiones de Visual FoxPro, elija **Establecer como predeterminado**.

**Sugerencia** También puede especificar un orden mediante el comando [SET COLLATE TO](#) o la instrucción COLLATE en el archivo Config.fpw. Para obtener más información sobre Config.fpw, consulte el capítulo 3, [Configurar Visual FoxPro](#), de la *Guía de instalación*.

El orden actual no afecta a los índices creados anteriormente, aunque sí afecta a los resultados de operaciones y comandos como SEEK y SELECT. SQL. Para obtener más información, el

comparaciones y comandos como [SEEK](#) y [SELECT - SQL](#). Para obtener más información al respecto, consulte [Reconocer los resultados de los métodos de ordenación](#) más adelante en este mismo capítulo.

Puede cambiar el orden en cualquier momento. Por ejemplo, después de abrir una tabla de clientes, puede crear etiquetas de índice que representen distintos métodos de ordenación, como se muestra en el código siguiente. A continuación, podrá cambiar el orden utilizando simplemente una etiqueta distinta:

```
USE cliente
SET COLLATE TO "GENERAL"
INDEX ON fnombre TAG migeneral ADDITIVE
SET COLLATE TO "MACHINE"
INDEX ON custid TAG mimáquina ADDITIVE
SET COLLATE TO "DUTCH"
INDEX ON lnombre TAG miholandés ADDITIVE
```

**Nota** El orden de un índice anula el orden actual.

La página de códigos actual determina qué órdenes hay disponibles. Si utiliza [SET COLLATE](#) para especificar un orden no admitido por la página de códigos actual, Visual FoxPro generará un error. Asimismo, si especifica en Config.fpw un orden no admitido en la página de códigos actual, se usará de forma predeterminada el orden Máquina.

## Comprobar órdenes

Para determinar el orden actual, utilice la función [SET \('COLLATE'\)](#). Por ejemplo, puede guardar el actual, establecer el orden actual en Máquina, realizar los trabajos necesarios y, a continuación, restaurar el orden original mediante el código siguiente:

```
cCurrentOrder=SET('COLLATE')
SET COLLATE TO 'MACHINE'
*
* código que requiere el orden Máquina
*
SET COLLATE TO cOrdenActual && vuelve al orden anterior
```

Además, puede determinar el orden de un índice o una etiqueta de índice utilizando la función [IDXCOLLATE\(\)](#).

## Reconocer los resultados de los métodos de ordenación

El orden afecta a los resultados de la comparación de cadenas, [SEEK](#), y [SELECT - SQL](#), como se describe en las secciones siguientes.

### Comparar cadenas

Todos los órdenes, salvo Máquina y Peso único, ignoran el uso de mayúsculas o minúsculas. Esto significa que no es necesario utilizar [UPPER\(\)](#) en las expresiones de índice.

El orden actual afecta a las comparaciones de cadenas. Por ejemplo, si establece el orden en General, las instrucciones siguientes devolverán verdadero (.T.):



```
? "A" = "a"  
? "Straße" = "Strasse"  
? "æ" = "ae"
```

Sin embargo, cuando utilice el orden Máquina, todas estas instrucciones devolverán falso (.F.) porque se ha especificado que las cadenas coincidan en una comparación exacta, byte a byte.

El operador de comparación de cadenas de caracteres (=) proporciona el mismo resultado cuando se compara por valor que cuando se compara con el orden Máquina. Por ejemplo, la instrucción siguiente devuelve falso (.F.):

```
? "Straße" == "Strasse"
```

**Nota** Visual FoxPro ignora SET EXACT cuando se utiliza el operador de comparación de cadenas de caracteres (=).

### Usar SEEK

Visual FoxPro ignora las marcas diacríticas cuando se realiza una *búsqueda parcial*. Una búsqueda parcial se produce cuando la longitud de la expresión es inferior a la de la clave. Si las marcas diacríticas son importantes, considere la posibilidad de utilizar [SCAN FOR...ENDSCAN](#) o [LOCATE FOR...CONTINUE](#) en lugar de [SEEK](#).

Entre las ventajas de utilizar SCAN y LOCATE en lugar de SEEK cabe citar las siguientes:

- SCAN y LOCATE reconocen las marcas diacríticas.
- Visual FoxPro optimiza plenamente los resultados de SCAN o LOCATE si el orden actual es Máquina o Peso único, mientras que sólo optimiza parcialmente los resultados de SEEK.
- SCAN y LOCATE recuerdan la condición que los ha invocado, lo que permite utilizarlos para hacer un bucle en una condición. Por el contrario, SEEK le sitúa en algún lugar del índice y [SKIP](#) continúa hacia abajo desde ese punto del índice. En consecuencia, es posible que SEEK no produzca los resultados deseados con datos internacionales.

### Usar SELECT - SQL

El comando [SELECT - SQL](#) utiliza el orden actual. Por ejemplo, si tiene una etiqueta de índice basada en el orden General y el orden actual (devuelto por [SET \('COLLATE'\)](#)) es Máquina, el resultado de SELECT - SQL se basará en Máquina.

Para emplear el orden actual, utilice la cláusula ORDER BY de SELECT - SQL.

### Usar índices

Los métodos de ordenación determinan el orden de los registros de las tablas indexadas. Tenga en cuenta las siguientes directrices para utilizar índices con órdenes:

- Vuelva a crear los índices creados en versiones anteriores de FoxPro si desea que éstos utilicen un orden distinto que Máquina.
- Vuelva a crear los índices de dBASE para aprovechar los órdenes de Visual FoxPro.
- Utilice el comando [REINDEX](#) para volver a generar un índice, ya que REINDEX no modifica

- Oprime el comando [REINDEXA](#) para volver a generar un índice, ya que REINDEXA no modifica el orden.

## Trabajar con juegos de caracteres codificados en dos bytes

Visual FoxPro admite [DBCS](#) (juegos de caracteres codificados en dos bytes), que son juegos de caracteres que requieren más de un byte para representar un carácter. Algunos ejemplos de idiomas que requieren este tipo de juego de caracteres son el chino simplificado, el chino tradicional, el japonés y el coreano.

El soporte de DBCS de Visual FoxPro DBCS permite la creación de aplicaciones internacionales. Por ejemplo, puede crear una aplicación en japonés con una versión USA de Visual FoxPro si ejecuta la versión japonesa de Windows. Las funciones de DBCS de Visual FoxPro DBCS operan correctamente en el juego de caracteres japonés y admiten la secuencia de ordenación en japonés.

**Nota** Visual FoxPro proporciona funciones de programación especiales para utilizarlas con cadenas en entornos DBCS. Para obtener más detalles, consulte [Trabajar con cadenas en entornos DBCS](#) más adelante en este mismo capítulo.

### Usar caracteres DBCS al asignar nombres a objetos

Visual FoxPro permite el uso de caracteres DBCS cuando se asignan nombres a elementos de las aplicaciones. Al igual que con Visual FoxPro, normalmente los elementos pueden:

- Tener una longitud máxima de 254 caracteres con la combinación de caracteres codificados en dos bytes y caracteres únicos. Por ejemplo, si usa caracteres codificados en dos bytes, el nombre que cree sólo puede tener 127 caracteres de longitud.
- Comenzar por una letra, número, guión bajo o combinación de bytes iniciales o finales.
- Contener sólo letras, números, guiones bajos o caracteres DBCS.

Estas reglas se aplican a variables, objetos (ventanas, menús, etc.), nombres de funciones y procedimientos, nombres de clases y subclases, alias y constantes. También puede usar los caracteres codificados en dos bytes para los nombres de archivos. Para evitar la posibilidad de que los caracteres del nombre del archivo se traten involuntariamente como delimitadores, es más seguro incluir siempre el nombre del archivo entre comillas.

**Nota** Los límites de longitud de Visual FoxPro se expresan mediante caracteres de un único byte. Si se utilizan caracteres codificados en dos bytes en nombres de campos, expresiones de índices, nombres de variables y ventanas, etc., se acorta la longitud del nombre. Por ejemplo, un nombre de campo puede ser de 10 bytes de longitud como máximo en una tabla libre; por tanto, un nombre de campo puede constar de diez caracteres de un byte único pero solamente de 5 caracteres codificados en dos bytes. Para obtener más información sobre las capacidades del sistema de Visual FoxPro, consulte [Capacidades del sistema](#).

### Ordenar datos DBCS

Para facilitar la tarea de ordenar información en entornos DBCS, Visual FoxPro admite secuencias de ordenación para el chino simplificado, el chino tradicional, el japonés y el coreano. Las secuencias de

ordenación permiten ordenar correctamente campos del tipo Character de tablas para cada idioma.

En la tabla siguiente se muestran las opciones de secuencias de ordenación de Visual FoxPro y el idioma correspondiente.

Opciones	Idioma
JAPANESE	Japonés
KOREAN	Coreano
PINYIN	Chino simplificado
STROKE	Chino simplificado y tradicional

Para obtener más información sobre la especificación de secuencias de ordenación, consulte [Especificar la ordenación](#) en este capítulo.

## Crear o modificar programas

Para evitar problemas con el código en la traducción, siga las directrices que se describen en las secciones siguientes.

### Probar versiones internacionales

Si es importante que la aplicación pueda determinar en qué idioma se está ejecutando Visual FoxPro, puede llamar a [VERSION\(\)](#). Conocer el entorno del idioma puede ayudarle a determinar qué texto mostrar, cómo dar formato a los datos, etc. Por ejemplo, el código siguiente determina en qué idioma se está ejecutando Visual FoxPro y después ejecuta un formulario específico del idioma:

```
IF VERSION(3) = 34 THEN
    * Ejecutándose en español; mostrar formulario español
    DO FORM CST_SPN.SCX
ELSE
    * Mostrar formulario inglés
    DO FORM CST_ENU.SCX
ENDIF
```

**Nota** La compatibilidad de cadenas de caracteres codificados en dos bytes sólo ha estado disponible en Visual FoxPro desde la versión 3.0b. Si la aplicación dispone de las funciones de DBCS, también deberá llamar a la función [VERSION\(1\)](#) para comprobar el número de versión de Visual FoxPro.

### Usar cadenas

Evite incluir cadenas directamente en el código, ya que dificultan la traducción. Por ejemplo, no incluya fechas ni monedas como cadenas en el código. Si es posible, escriba el código de modo que recupere las cadenas de archivos o tablas independientes del programa.

**Nota** El rendimiento del programa puede verse afectado si elimina todas las cadenas, por ejemplo, si busca cadenas mientras está dentro de un bucle.

Una forma de trabajar con cadenas en una aplicación que vaya a traducirse es usar constantes de cadenas en la aplicación. Después puede definir el texto de estas constantes en un archivo de texto diferente al que se hace referencia en los programas mediante la directiva del preprocesador [#INCLUDE](#). Por ejemplo, en lugar de incrustar el mensaje de error “no se encuentra el archivo”, puede usar la constante `ERR_FILE_NOT_FOUND`. El texto de esta constante podría encontrarse en un archivo denominado `ERR_TEXT.H`. Un programa donde se utilice esta técnica podría asemejarse a éste:

```
#INCLUDE ERR_TEXT.H

* procesar aquí

IF ERR THEN
    MESSAGEBOX( ERR_FILE_NOT_FOUND )
ENDIF
```

Cuando la aplicación se traduzca, el traductor puede crear una versión del archivo de texto de errores específica de la configuración regional y después volver a compilar la aplicación.

### Trabajar con cadenas en entornos DBCS

Visual FoxPro incluye funciones para manipular expresiones de caracteres que contengan cualquier combinación de caracteres de un byte o dos bytes. Usando funciones de cadena DBCS, puede programar aplicaciones sin tener que escribir código adicional que pruebe caracteres de dos bytes al contar, buscar, insertar o quitar caracteres en una cadena.

La mayor parte de las funciones son equivalentes a sus homólogas de un byte con la excepción de que sus nombres tienen un sufijo `C` para diferenciarse. Puede usar estas funciones con datos de un byte o de dos bytes; las funciones DBCS devuelven exactamente el mismo valor que sus homólogas de un byte cuando se les pasan datos de un byte. Otras funciones le ayudan a trabajar con cadenas específicamente en entornos de dos bytes.

Funciones de cadena DBCS	Descripción
<a href="#">AT_C()</a>	Devuelve la posición de una cadena dentro de otra (distingue mayúsculas de minúsculas), empezando por la izquierda.
<a href="#">ATCC()</a>	Devuelve la posición de una cadena dentro de otra (no distingue mayúsculas de minúsculas).
<a href="#">CHRTRANC()</a>	Reemplaza caracteres en una cadena.
<a href="#">IMESTATUS()</a>	Alterna la edición de dos bytes en la ventana Examinar.
<a href="#">ISLEADBYTE()</a>	Comprueba si un carácter es un carácter DBCS.
<a href="#">LEFTC()</a>	Devuelve los caracteres de la izquierda de una cadena.
<a href="#">LENC()</a>	Devuelve el número de caracteres de una cadena.
<a href="#">LIKEC()</a>	Determina si dos cadenas coinciden.

<a href="#">RATC()</a>	Devuelve la posición de una cadena dentro de otra (distingue mayúsculas de minúsculas), empezando por la derecha.
<a href="#">RIGHTC()</a>	Devuelve los caracteres de la derecha de una cadena.
<a href="#">STRCONV()</a>	Convierte caracteres entre representaciones de un byte y dos bytes.
<a href="#">STUFFC()</a>	Reemplaza caracteres de una cadena con otra cadena.
<a href="#">SUBSTRC()</a>	Devuelve una subcadena.

Al trabajar con funciones de cadena de dos bytes, recuerde que el límite de longitud máximo para variables, nombres, etcétera, se reduce a la mitad. Para obtener información sobre las capacidades de sistema de Visual FoxPro, vea [Capacidades de sistema](#).

**Nota** Las funciones DBCS de Visual FoxPro no son compatibles con versiones anteriores de Visual FoxPro y al llamarlas desde estas versiones se pueden producir resultados impredecibles. Si usa funciones DBCS en su aplicación, use [VERSION\(1\)](#) para comprobar que la versión de Visual FoxPro es posterior a la versión 3.0.

## Trabajar con formatos Date, Time y Currency

Para ayudarle a dar formato a fechas, horas y moneda para que se ajusten a las costumbres de los usuarios, puede usar varias técnicas de formato. Puede:

- Permitir a Visual FoxPro usar la configuración establecida en el Panel de control.
- Especificar un idioma o un formato específico que desee usar en el cuadro de diálogo Opciones de Visual FoxPro.
- Dar formato a información de fechas, horas y moneda en el código.

### Para establecer un formato para fecha, hora o moneda

1. En el menú **Herramientas**, elija **Opciones** y, a continuación, haga clic en la ficha [Regional](#).
2. Para usar la configuración establecida en el Panel de control, elija **Usar la configuración del sistema**.

–O bien–

Elija un idioma o formato para fechas y horas y, a continuación, elija opciones para dar formato a moneda y números. Si elige el formato **Corta** o **Larga** para el formato de fecha, no puede especificar ninguna otra opción para el formato de fecha y se lee la configuración del Panel de control de Windows.

3. Elija **Aceptar** para usar las opciones para esta sesión o establezca **Establecer como predeterminado** para hacer que las modificaciones sean la configuración predeterminada para esta copia de Visual FoxPro.

También puede crear esta configuración mediante los comandos [SET SYSEFORMATS](#) y [SET DATE](#)

También puede crear esta configuración mediante los comandos [SET SYSFORMATS](#) y [SET DATE](#). Como regla general, debe ejecutar este comando en la inicialización de la aplicación (por ejemplo, en el archivo de configuración). El valor predeterminado de SET SYSFORMATS es OFF, por lo que debe establecerlo explícitamente como ON al iniciar la aplicación.

Puede establecer validación de datos en cuadros de texto individuales si establece la propiedad [Format](#) del cuadro de texto. Sin embargo, como el formato de cuadro de texto tiene prioridad superior al formato a nivel de sistema, esto puede dificultar la localización de su aplicación a un entorno que use un formato diferente para fechas, moneda, etcétera.

## Usar directivas del preprocesador

Puede crear variantes de la aplicación para distintas configuraciones propias de un país mediante directivas del preprocesador. Estas directivas controlan la compilación de código en la aplicación e incluyen las construcciones [#INCLUDE](#), [#DEFINE](#), [#UNDEF](#) y [#IF...#ENDIF](#).

El uso de las directivas del preprocesador puede generar variantes rápidamente. Sin embargo, estas directivas tienen los siguientes inconvenientes:

- Para usar las directivas del preprocesador debe incluir código entre corchetes y el uso abundante de los corchetes puede aumentar la complejidad del código.
- Las constantes de tiempo de compilación sólo están disponibles en el programa que las crea.

## Administrar archivos en una aplicación internacional

El Administrador de proyectos puede ayudarle a organizar una aplicación internacional. En un proyecto, puede integrar la partes de una aplicación como formularios, menús, programas e informes. El proyecto garantiza que las partes son actuales cuando se genera la aplicación para el mercado de destino.

A diferencia de los archivos .DBF, los archivos de texto (como los de consulta y programa) no tienen marcas de página de códigos. Esto significa que debe hacer un seguimiento de las páginas de códigos que utilizan los archivos de texto para poder utilizarlos correctamente. Con el Administrador de proyectos, puede hacer un seguimiento de las páginas de códigos utilizadas por los archivos de texto. Para ver detalles, consulte [Especificar la página de códigos de un archivo de texto](#) en una sección anterior de este capítulo.

## Distribuir archivos de tiempo de ejecución específicos de la configuración regional

Si va a distribuir la aplicación con la versión de tiempo de ejecución de Visual FoxPro, tiene que incluir un *archivo de recursos* específico de la configuración regional. Este archivo contiene los cuadros de diálogo y otros elementos de interfaz de usuario que Visual FoxPro usa para interactuar con el usuario. Hay un archivo de recursos de tiempo de ejecución diferente para cada idioma en el que Visual FoxPro está disponible.

Normalmente sólo tendrá que preocuparse de recursos de tiempo de ejecución específicos de configuración regional si se cumplen las siguientes condiciones:

- Incluye la versión de tiempo de ejecución de Visual FoxPro en la aplicación.

- Distribuye la aplicación a usuarios que usan un idioma diferente del utilizado al programarla. Por ejemplo, si programa en inglés para usuarios de habla inglesa, no tiene que preocuparse de incluir un archivo de recursos específico de la configuración regional. Sin embargo, si usa la versión inglesa de Visual FoxPro para programar pero va a distribuir la aplicación de tiempo de ejecución en un país de habla francesa, debe incluir el archivo de recursos de tiempo de ejecución.
- La aplicación muestra cuadros de diálogo de Visual FoxPro, menús o mensajes de error. Normalmente, si ha diseñado y localizado sus propias versiones de estos elementos de interfaz, no tiene que incluir el archivo de recursos específico de la configuración regional.

Para obtener información sobre la distribución de archivos de tiempo de ejecución con la aplicación, consulte el capítulo 25, [Generar una aplicación para su distribución](#) y el capítulo 26, [Crear discos de distribución](#).

A los archivos de recursos de tiempo de ejecución se les da nombre con el formato Vfpaaa.dll, en donde “aaa” es un código de tres letras que representa el idioma. Por ejemplo, el código ENU significa Estados Unidos, Inglés, el código DEU significa Alemán y el código FRA significa Francés. Los archivos de recursos de tiempo de ejecución para estos idiomas serán Vfpenu.dll, Vfpdeu.dll y Vfpfra.dll respectivamente.

Siempre debe usar al menos un archivo de recursos, incluso si no pretende usar ninguno de los elementos de interfaz de usuario de Visual FoxPro como parte de su aplicación. De forma predeterminada, Visual FoxPro incluye el archivo de recursos proporcionado con su copia del programa. Por ejemplo, si programa una aplicación con la versión de Estados Unidos de Visual FoxPro, Visual FoxPro incluirá automáticamente Vfpenu.dll si incluye archivos de tiempo de ejecución en la aplicación. Si no tiene ninguna razón para usar un archivo de recursos específico de la configuración regional, puede distribuir el archivo de recursos predeterminados como parte de la aplicación.

Cuando la aplicación está instalada, los usuarios pueden especificar el archivo de tiempo de ejecución creando una entrada en el registro del sistema de Windows o usando un modificador de línea de comandos.

### **Para especificar un archivo de recursos de tiempo de ejecución**

- En la línea de comandos que inicia su aplicación, incluya el modificador L y el nombre del archivo de recursos que quiere usar (incluyendo una ruta si es necesario). No coloque un espacio entre el modificador y el nombre de archivo.

Por ejemplo, el siguiente comando especifica el archivo Vfpdeu.dll como archivo de recursos:

```
C:\Archivos de programas\Microsoft Visual ;  
Studio\Vfp98\MYAPP.EXE -LC:\Myapp\Vfpdeu.dll
```

–O bien–

- Configure el registro de Windows del equipo del usuario (mediante código o mediante una aplicación como Regedit.exe) de forma que apunte al archivo de recursos que hay que usar. La entrada del Registro que contiene el nombre del archivo de recursos de tiempo de ejecución es:

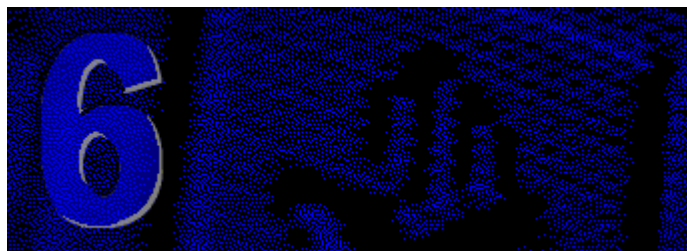
### HKEY\_CLASSES\_ROOT\VisualFoxProRuntime.5\RuntimeResource.5

Cuando se inicia la aplicación de tiempo de ejecución, Visual FoxPro busca en primer lugar un archivo de recursos según el modificador L y después según la configuración del Registro. Si ninguna de estas configuraciones especifican un archivo de recursos específico de la configuración regional, Visual FoxPro usa la configuración regional actual del sistema (Windows) para construir de forma dinámica un nombre de archivo DLL. Por tanto, si el archivo de recursos específico de la configuración regional para la aplicación coincide con la configuración regional de la versión de Windows del usuario, no tiene que especificar de forma explícita el nombre del archivo de recursos. Sin embargo, siempre es más seguro no confiar en la configuración predeterminada del sistema si quiere estar seguro de que se carga el archivo apropiado.





# Manual del programador, Parte 6: Crear soluciones cliente-servidor



Las aplicaciones cliente-servidor combinan la funcionalidad de Visual FoxPro en su equipo local con las ventajas de almacenamiento y seguridad proporcionadas por un servidor remoto. Puede hacer un prototipo de sus aplicaciones localmente y, a continuación, usar el Asistente para upsizing para transformar la aplicación para un entorno cliente-servidor.

## **Capítulo 19** [Diseñar aplicaciones cliente-servidor](#)

Aprenda a diseñar una eficaz aplicación cliente-servidor con tecnologías de programación multiusuario.

## **Capítulo 20** [Upsizing de bases de datos de Visual FoxPro](#)

La creación de prototipos locales de su diseño puede reducir el tiempo y el coste de programación. Cuando haya probado el prototipo local, es fácil y beneficioso hacer un upsizing de la aplicación de forma que pueda aprovechar todas las características proporcionadas por el servidor remoto.

## **Capítulo 21** [Implantación de una aplicación cliente-servidor](#)

Puede usar la tecnología de paso a través de SQL para mejorar la aplicación a la que ha hecho un upsizing. Mientras que las vistas remotas proporcionan acceso a datos del servidor, el paso a través de SQL le permite enviar comandos directamente al servidor con sintaxis de servidor nativa, lo que aumenta el control y la flexibilidad.

## **Capítulo 22** [Optimizar el rendimiento cliente-servidor](#)

Después de hacer el upsizing y la implantación, puede seguir otros pasos adicionales para optimizar el rendimiento de su aplicación. Averigüe qué puede hacer con Visual FoxPro y el servidor remoto para optimizar la aplicación cliente-servidor.

# capítulo 19: Diseñar aplicaciones cliente-servidor

Visual FoxPro le proporciona las herramientas necesarias para crear eficaces [aplicaciones cliente-](#)

[servidor](#). Una aplicación cliente-servidor de Visual FoxPro combina la eficacia, la velocidad, la interfaz gráfica de usuario y las sofisticadas funciones de consulta, informes y proceso de Visual FoxPro con el acceso multiusuario, almacenamiento masivo de datos, seguridad incorporada, robusto proceso de transacciones, inicio de sesiones y la sintaxis nativa del servidor de un origen de datos o servidor [ODBC](#). La sinergia de Visual FoxPro y las ventajas de los servidores proporcionan una eficaz solución cliente-servidor para sus usuarios.

El paso más importante a la hora de generar con éxito una aplicación cliente-servidor es crear un buen diseño. Este capítulo se basa en la información para programación de aplicaciones multiusuario proporcionada en el *Manual del programador*. Partiendo de esta base, definimos una metodología para la programación de aplicaciones cliente-servidor.

Si desea información acerca de la generación y el "upsizing" de un prototipo local, consulte el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#). Para obtener más información sobre el uso de la tecnología de paso a través de SQL, consulte el capítulo 21, [Implementar una aplicación cliente-servidor](#). Para acelerar la recuperación y el procesamiento de datos, consulte el capítulo 22, [Optimizar el rendimiento cliente-servidor](#).

Este capítulo trata los temas siguientes:

- [Objetivos para el diseño cliente-servidor](#)
- [Diseño para un elevado rendimiento](#)
- [Programación rápida de aplicaciones](#)
- [Incorporar precisión e integridad de datos](#)

## Objetivos para el diseño cliente-servidor

Al diseñar una aplicación cliente-servidor se deben equilibrar varios conjuntos de requisitos. Usted desea generar la aplicación más rápida y más productiva posible para sus usuarios. También desea garantizar la integridad de los datos de la aplicación, aprovechar al máximo las inversiones existentes en hardware e incorporar la posibilidad de ampliación en el futuro. Además, como programador de Visual FoxPro, desea que el proceso de programación sea lo más dinámico y económico posible.

La mejor forma de satisfacer estos requisitos es diseñar la aplicación con estos objetivos en mente. Vamos a preparar el terreno perfilando las técnicas que proporcionan el máximo rendimiento cliente-servidor.

## Diseño para un alto rendimiento

Generar una aplicación cliente-servidor rápida y de alto rendimiento con Visual FoxPro implica aprovechar la enorme velocidad del motor de Visual FoxPro. Esto se consigue con nuevas técnicas tales como el uso de acceso a datos basado en conjuntos, en lugar del desplazamiento Xbase tradicional, la generación de consultas parametrizadas para descargar solamente los datos necesarios, la ubicación de tablas en la plataforma óptima y el aprovechamiento de procedimientos tanto de Visual FoxPro como almacenados de forma remota.

Antes de que pueda hacer uso de las nuevas técnicas es necesario analizar los sistemas que piensa utilizar. Al diseñar una aplicación local o de servidor de archivos, debe determinar las consultas, los

formularios, los menús y los informes que la aplicación va a utilizar o crear. Cuando se diseña una aplicación cliente-servidor, se debe llevar a cabo el análisis habitual del sistema, así como un análisis adicional relacionado específicamente con las aplicaciones cliente-servidor. Es necesario plantearse dónde se ubicarán los datos utilizados por las consultas, los formularios, los menús y los informes, y cómo se tendrá acceso a esta información. Por ejemplo, puede plantearse cuestiones tales como:

- ¿Qué tablas se almacenarán en el servidor remoto una vez implantada la aplicación?
- ¿Qué tablas se almacenarían de forma más eficaz como tablas de búsqueda locales?
- ¿Qué vistas necesitará para tener acceso a los datos remotos?
- ¿Qué reglas corporativas exige el servidor y cómo interactúa su aplicación con estas reglas?

Cuando haya determinado los componentes básicos de su aplicación cliente-servidor, puede comenzar a diseñar la forma en que su aplicación tendrá acceso a los datos y los actualizará.

## Descargar solamente los datos necesarios

Uno de los factores más importantes a la hora de generar una aplicación cliente-servidor rápida y eficiente es reducir al mínimo la cantidad de datos que necesita extraer del servidor. Puesto que las aplicaciones cliente-servidor pueden tener acceso a cantidades de datos muy grandes en un servidor remoto, el uso de las técnicas tradicionales de desplazamiento Xbase puede dar como resultado una aplicación cliente-servidor lenta. Para acelerar el rendimiento se utilizan técnicas de acceso a datos basadas en conjuntos para filtrar la cantidad de datos descargados.

### Acceso eficaz a datos basados en conjuntos

Los datos remotos están [basados en conjuntos](#): el acceso a datos remotos se realiza seleccionando un *conjunto* de datos de un gran almacén de datos mediante instrucciones [SELECT - SQL](#). La diferencia más importante entre generar una aplicación local tradicional y una aplicación cliente-servidor es el contraste entre las técnicas tradicionales de desplazamiento en Visual FoxPro y las técnicas de acceso a datos del servidor basados en conjuntos.

### Usar las técnicas tradicionales de desplazamiento

En la programación tradicional Xbase, puede tener acceso a cantidades de datos discretas y, con frecuencia, de gran volumen, mediante el comando GO BOTTOM, para el cual puede, posteriormente, realizar una consulta. Puede desplazarse por los datos si ejecuta un comando [SET RELATION](#) para crear una relación temporal entre dos tablas y, a continuación, ejecuta un comando [SKIP](#) para moverse por los registros relacionados.

Si bien este método de desplazamiento por los registros podría utilizarse para datos remotos, no sería eficiente para grandes almacenes de datos remotos. Por ejemplo, si crea una vista remota que tiene acceso a una tabla grande en un origen de datos remotos y a continuación ejecuta el comando [GOTO BOTTOM](#) tendrá que esperar mientras todos los datos de la vista se recuperan desde el origen de datos, se envían a través de la red y se cargan en el cursor de la vista del sistema local.

### Usar consultas parametrizadas

Un enfoque más eficaz para el acceso a datos remotos es descargar únicamente los datos que necesita y entonces volver a consultarlos para obtener registros adicionales específicos o registros nuevos. Se

utiliza una instrucción **SELECT** basada en parámetros para descargar un pequeño conjunto de datos específico y después tener acceso a nuevos registros mediante la función [REQUERY\(\)](#) para solicitar un nuevo conjunto de datos.

No utilice el comando **GO BOTTOM** sobre los datos del servidor remoto porque se originaría:

- Una carga innecesaria en los recursos de la red al descargar enormes cantidades de datos.
- Disminución del rendimiento de la aplicación al tener que manejar datos innecesarios.
- Posible reducción de la precisión de los datos del cursor local porque los cambios en los datos remotos no se reflejan en este cursor local hasta que no se ejecuta una nueva consulta.

Por ejemplo, si desea crear una aplicación cliente-servidor que tenga acceso a los pedidos de un cliente determinado, cree una vista remota que tenga acceso a la tabla Customer. Cree otra vista remota que tenga acceso a la tabla Orders, pero parametrize la vista basándose en el campo `cust_id`. A continuación, emplee el registro del cliente actual como parámetro para la vista de la tabla Orders.

Puede utilizar el parámetro para establecer el alcance del conjunto de datos descargado a la cantidad justa de datos. Si solicita pocos datos, puede perder rendimiento porque necesitará volver a consultar el servidor remoto más frecuentemente. Si solicita demasiados datos, puede perder tiempo descargando datos que no va a utilizar.

### **Elegir el mejor diseño cliente-servidor**

Los siguientes ejemplos describen la forma de obtener las ventajas de la tecnología cliente-servidor y evitar los inconvenientes de técnicas de programación inadecuadas. El primer método utiliza técnicas de programación tradicionales para transferir todos los datos desde un origen de datos remoto hasta cursores locales, que se relacionan posteriormente con el comando [SET RELATION](#). Los métodos segundo, tercero y cuarto adoptan progresivamente técnicas de recuperación de datos cada vez más inteligentes, que limitan de forma efectiva la cantidad de datos descargados con una metodología 'justo en el momento' que proporciona los datos más actuales y el tiempo de respuesta más rápido a través de una red.

### **Usar una estrategia cliente-servidor no optimizada**

Una aplicación cliente-servidor sencilla y sin optimizar utiliza con los datos remotos las mismas técnicas de desplazamiento que usa para los datos locales. Por ejemplo, si tiene 10 millones de registros de clientes y 100 millones de registros de pedidos en un origen de datos remoto, puede crear una aplicación ineficaz que descargue todos los registros de las tablas Customer y Orders en los cursores locales. A continuación puede indexar basándose en 100 millones de registros de pedidos, crear una relación temporal entre las tablas Customer y Orders en los cursores locales, y utilizar el comando **SKIP** para desplazarse por los registros.

Este método no está optimizado para un gran rendimiento, pero podría ser útil si el extremo "uno" es local y el extremo "varios" es remoto.

### **Filtrar el extremo "varios"**

Una aplicación cliente-servidor ligeramente mejorada limita el extremo "varios" de la relación, pero transfiere todo el extremo "uno" para que pueda pasar por los registros. En este ejemplo se crea una

vista remota del extremo "varios" de la relación, la tabla Orders, parametrizada según el ID del cliente. A continuación se descarga toda la tabla Customer.

Aunque crear una vista parametrizada sobre la tabla Orders es una mejora con respecto a descargar todos los pedidos, continúa transfiriendo información innecesaria al descargar toda la tabla Customer. Además, la tabla Customer está cada vez menos actualizada a medida que otros usuarios del sistema hacen cambios en ella. Este método puede ser adecuado si el extremo "uno" de la relación contiene un pequeño conjunto de datos.

### **Filtrar el extremo "uno"**

Una técnica de programación cliente-servidor más apropiada crea vistas remotas para todos los datos remotos. El número de registros de Customer descargados a la vista remota de la tabla Customer se limita mediante la instrucción SELECT en esta vista para seleccionar solamente los clientes de una región. A continuación se crea una vista remota del extremo "varios" de la relación, la tabla Orders, parametrizada según el ID del cliente.

Este método transfiere un conjunto de registros más reducido. El comando [SKIP](#) se usa para ir al extremo "uno" de la relación (la vista Customer). El comando [REQUERY\(\)](#) se usa para tener acceso a nuevos datos en el extremo "varios" (Orders).

En este ejemplo se limita (se filtra) tanto el extremo "uno" como el extremo "varios" de la relación y también se puede utilizar el comando SKIP para desplazarse por los datos filtrados. Este método puede ser recomendable si el extremo "uno" de la relación, aún después de filtrarse, sigue siendo suficiente para proporcionar información para un conjunto de consultas sucesivas antes de volver a consultar el servidor remoto.

### **Usar la clave principal para tener acceso a la relación "uno a varios"**

El ejemplo de programación cliente-servidor más eficiente se olvida del lujo de utilizar el comando [SKIP](#) y crea un formulario que solicita la entrada o la selección del Id. de cliente, que se utiliza posteriormente como parámetro para una vista remota de la tabla Customer. Este parámetro se utiliza también para una vista remota de la tabla Orders.

Por ejemplo, podría crear un formulario "uno a varios" en el cual la información del cliente constituyera el extremo "uno" y un control Grid mostrara el extremo "varios" de la relación. El control Grid puede estar vinculado al Id. de cliente elegido en el extremo "uno" del formulario. Entonces puede establecer la propiedad MaxRecords de [CURSORSETPROP\(\)](#) a 1 y usar el código siguiente para llenar el lado "uno" del formulario:

```
SELECT * FROM customer WHERE customer.cust_id = ?cCust_id
```

Cuando el usuario desea ver el registro de otro cliente distinto, introduce o selecciona un nuevo Id. de cliente. El formulario vuelve a consultar en el origen de datos los pedidos del nuevo Id. de cliente y actualiza el control Grid con los nuevos datos de pedido.

Con estas técnicas, la aplicación descarga solamente los datos necesarios y en el momento en que se necesitan. La respuesta a través de la red se acelera si limita la cantidad de datos transferidos, y se proporciona al usuario información más actualizada si se vuelve a consultar el origen de datos justo

antes de mostrar la información solicitada.

Este método se recomienda cuando se desea tener acceso a la relación uno a varios de manera aleatoria mediante cualquier valor de clave principal. Quizás desee descargar las claves principales en un control, como una lista desplegable, al abrir el formulario, y después ofrecer un control que el usuario puede elegir para actualizar la lista de valores de clave principal cuando la requiera.

### Usar el entorno de datos en aplicaciones cliente-servidor

Cuando utilice datos remotos en un formulario o un conjunto de formularios, debe incluir las vistas en el entorno de datos del formulario o del conjunto de formularios. Puede establecer la propiedad `AutoOpenTables` para el entorno de datos como falsa (.F.), de forma que pueda especificar el momento en que la aplicación actualiza las vistas con los datos remotos. Establezca la propiedad `ControlSource` para los cuadros de texto y otros controles vinculados a datos después de haber llamado al método `OpenTables` del entorno de datos, normalmente en el código asociado con el evento `Init` del formulario. Para obtener más información sobre el establecimiento de las propiedades de formularios, consulte el capítulo 9, [Crear formularios](#).

### Localizar datos en la plataforma óptima

El máximo rendimiento se obtiene cuando los datos y otros atributos de la base de datos se almacenan en la plataforma óptima. La mejor plataforma para un elemento en concreto depende de la forma en que se tiene acceso y se actualiza dicho elemento. Por ejemplo, quizá desee almacenar una copia local de una tabla del servidor, como una guía de códigos postales, que se utiliza como tabla de búsqueda, y actualizar esta copia local solamente cuando cambie la tabla original.

En la tabla siguiente se muestran algunos elementos de aplicación habituales, así como ejemplos de dónde situarlos para obtener un rendimiento óptimo.

#### Ubicación de elementos por plataforma

Elemento	Ubicación	Tipo	Notas
Tablas	Local	Copias locales de tablas de búsqueda de servidor; tablas pequeñas con modificaciones poco frecuentes.	Use una marca de hora, si el servidor remoto lo admite, para comparar y, opcionalmente, actualizar la tabla local para que coincida con los cambios de la tabla original.
	Remota	Tablas grandes o modificadas con frecuencia	
Reglas	Local	Reglas en vistas	Puede utilizar <a href="#">DBSETPROP()</a>

		remotas	para almacenar en una vista remota reglas a nivel de campo y de registro. La aplicación puede usar estas reglas locales para comprobar la validez de los datos antes de enviarlos a la tabla original como actualización para tablas remotas.
	Remota	Reglas a nivel de fila y de columna en tablas base remotas	
Procedimientos almacenados	Local	Procedimientos almacenados de Visual FoxPro	
	Remota	Procedimientos almacenados en el servidor de apoyo	Use la función <a href="#">SQLEXPED()</a> para llamar a procedimientos almacenados en el servidor.
Transacciones	Local	Transacciones de Visual FoxPro	
	Remota	Transacciones del servidor	
Desencadenantes	Vistas locales	No hay desencadenantes en las vistas	
	Remota	Desencadenantes del servidor	

Para reducir el tráfico de la red durante las búsquedas, puede elegir entre almacenar localmente las tablas de búsqueda que se modifican con frecuencia o las que no cambian casi nunca. Por ejemplo, podría descargar la lista de clientes de su empresa y actualizarla solamente cuando cambiara la información de los clientes.

Para realizar esta tarea, puede programar la aplicación para que compare la marca de hora incluida en la copia local de la tabla con la marca de hora de los datos de apoyo (si el servidor remoto admite marcas de hora) y actualizar la copia local solamente si la tabla del servidor ha cambiado. Otra posibilidad es agregar al formulario un botón de comando que obligue a realizar una descarga inmediata de la tabla, lo que permite a los usuarios actualizar su copia de la tabla local cuando sea necesario.

## Elegir los métodos apropiados

Puede utilizar vistas remotas, paso a través de SQL o ambos para crear su aplicación cliente-servidor. La combinación de ambos métodos ofrece eficaces resultados: utilice las vistas para la mayoría de los requisitos de administración de datos y emplee paso a través de SQL para mejorar la potencia de la aplicación.



## Usar vistas

Puede utilizar las vistas como método básico para programar una eficaz aplicación cliente-servidor. Las vistas remotas constituyen una tecnología de grandes posibilidades, diseñada para permitirle seleccionar únicamente los datos que necesita desde un servidor remoto e incluirlos en un cursor local de Visual FoxPro, que puede utilizar posteriormente para ver y actualizar datos remotos. Una vista es, básicamente, un conjunto de resultados de una instrucción SELECT de SQL.

Las vistas son persistentes: la definición de la vista se almacena en una base de datos. Las definiciones de vistas tienen propiedades que usted puede establecer, y personalizar para el cursor de vista activa. Las vistas son la mejor herramienta para la definición de datos de un conjunto de resultados actualizable.

Puede utilizar las vistas locales para generar un prototipo local y posteriormente utilizar el Asistente para upsizing con el fin de transformar las vistas locales en vistas remotas. Para obtener información sobre el uso del Asistente para upsizing, consulte el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#)

Si los usuarios de la aplicación quieren usar datos para trabajar cuando están de viaje, puede utilizar vistas fuera de línea. Las vistas fuera de línea hacen portables los datos, permitiendo a los usuarios de equipos portátiles trabajar con una copia almacenada de un origen de datos que pueden actualizar mientras están de viaje. Cuando el usuario vuelve a conectarse al servidor, la aplicación puede combinar fácilmente cambios fuera de línea en las tablas de origen.

Es posible que también desee usar tecnología de vistas fuera de línea para permitir a los usuarios locales trabajar con datos "fuera de línea", combinando sus actualizaciones posteriormente. Para obtener información sobre trabajo con datos fuera de línea, consulte el capítulo 8, [Crear vistas](#).

## Usar paso a través de SQL

La tecnología de paso a través de SQL le proporciona acceso directo a un servidor remoto con las funciones de paso a través de SQL de Visual FoxPro. Estas funciones facilitan un acceso y un control adicionales del servidor que superan las capacidades de las vistas. Por ejemplo, puede efectuar definición de datos en el servidor remoto, establecer propiedades del servidor y tener acceso a procedimientos almacenados en el servidor.

El paso a través de SQL es la mejor herramienta para crear conjuntos de resultados de sólo lectura y para utilizar cualquier otra sintaxis nativa de SQL. A diferencia de las vistas, que son conjuntos de resultados de instrucciones SELECT de SQL, el paso a través de SQL le permite enviar al servidor todo lo que desee mediante la función [SQLEXEC\(\)](#). La tabla siguiente muestra las funciones de paso a través de SQL de Visual FoxPro.

## Funciones de paso a través de SQL

<a href="#">SQLCANCEL()</a>	<a href="#">SQLCOLUMNS()</a>	<a href="#">SQLCOMMIT()</a>
<a href="#">SQLCONNECT()</a>	<a href="#">SQLDISCONNECT()</a>	<a href="#">SQLEXEC()</a>
<a href="#">SQLGETPROP()</a>	<a href="#">SQLMORERESULTS()</a>	<a href="#">SQLPREPARE()</a>
<a href="#">SQLROLLBACK()</a>	<a href="#">SQLSETPROP()</a>	<a href="#">SQLSTRINGCONNECT()</a>
<a href="#">SQLTABLES()</a>		

Puede crear cursores personalmente mediante la tecnología de paso a través de SQL. Si bien el paso a través de SQL proporciona un acceso más directo al servidor, este acceso es menos persistente que el de las vistas. A diferencia de las vistas, cuyas definiciones se almacenan en una base de datos, los cursores creados mediante paso a través de SQL solamente existen durante la sesión actual. Para obtener más información sobre el uso de la tecnología de paso a través de SQL, consulte el capítulo 21, [Implementar una aplicación cliente-servidor](#)

### Combinar vistas y paso a través de SQL

El paradigma más eficaz de la generación de aplicaciones cliente-servidor con Visual FoxPro combina las tecnologías de vistas y de paso a través del SQL. Puesto que las vistas son fáciles de generar y proporcionan capacidades automáticas de almacenamiento en búfer y actualización, se utilizan para la mayoría de las tareas de administración de datos. Posteriormente puede usar el paso a través de SQL para llevar a cabo tareas específicas en el servidor remoto, como la definición de datos y la creación y ejecución de procedimientos almacenados en el servidor.

## Programación rápida de aplicaciones

Cualquiera que sea del método de programación que elija, necesita una buena estrategia para lograr que la programación de aplicaciones cliente-servidor sea rápida y eficaz. Puesto que Visual FoxPro facilita y acelera la generación de aplicaciones y el establecimiento de prototipos, puede elegir entre diseñar y crear un prototipo local para la aplicación, para después realizar un "upsizing" e implementarlo en etapas para un origen de datos remoto. Si tiene acceso a un origen de datos remoto durante el proceso de desarrollo, puede elegir realizar un prototipo de la aplicación basado el origen de datos remoto mediante vistas remotas.

### Generar un prototipo con vistas

El primer paso de la programación de una aplicación cliente-servidor de Visual FoxPro puede ser crear un prototipo. Al crear un prototipo de la aplicación, quizá módulo a módulo, se descubren posibles cambios y mejoras en el diseño durante las primeras etapas de la programación. De esta forma, es posible ajustar el diseño de manera eficaz sobre pequeños conjuntos de datos antes de agregar la capa adicional de complejidad inherente al trabajo con grandes conjuntos de datos remotos y heterogéneos. La generación de prototipos se describe en el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#)

### Crear un prototipo local con vistas locales

Un prototipo local para una aplicación cliente-servidor es una aplicación de Visual FoxPro operativa que utiliza vistas locales para tener acceso a tablas locales. Utilice vistas en su prototipo cliente-servidor porque la aplicación cliente-servidor final empleará vistas remotas para tener acceso a datos remotos. Al definir un prototipo para la aplicación con vistas locales, se acerca un poco a la aplicación final.

Generar un prototipo local es especialmente práctico si no dispone de acceso constante a un origen de datos remoto durante la programación o si no desea utilizar datos remotos para establecer el prototipo de la aplicación. Las vistas locales tienen acceso a tablas locales de Visual FoxPro, en lugar de tener acceso a tablas del origen de datos remoto. No obstante, debe crear datos locales de forma que imiten la estructura de los datos del servidor. Utilizar datos locales para representar datos remotos es un método eficaz para programar y probar rápidamente el diseño básico de la aplicación. También puede acelerar el desarrollo si limita la cantidad de datos seleccionados en las vistas. Para obtener más información acerca de la generación de vistas locales y remotas, consulte el capítulo 8, [Crear vistas](#).

### **Diseño de upsizing**

El [upsizing](#) es el proceso que crea en el servidor remoto una base de datos con la misma estructura de tabla, los mismos datos y muchos otros atributos de la base de datos original de Visual FoxPro. Mediante el upsizing, usted toma una aplicación existente de Visual FoxPro y la migración a una aplicación cliente-servidor. Para obtener más información sobre el upsizing, consulte el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#).

Cuando genere una aplicación en la que en un futuro realizará un "upsizing", elija el diseño de la arquitectura de la aplicación y el modelo de programación con el objetivo de extraer el máximo rendimiento para un origen de datos remoto. Estas selecciones se han descrito anteriormente en este capítulo, en la sección [Diseñar para un alto rendimiento](#).

### **Crear prototipos con vistas remotas**

Si tiene acceso a un origen de datos remoto y desea utilizar los datos remotos directamente a medida que programa su aplicación cliente-servidor, puede crear el prototipo con vistas remotas. Cuando se sigue este procedimiento con vistas remotas, se omite la fase de "upsizing", puesto que los datos se encuentran en un servidor remoto y ya dispone de vistas remotas para tener acceso a dichos datos.

### **Implementación de la aplicación cliente-servidor**

Puede simplificar las pruebas y la depuración de su aplicación si implementa por etapas la aplicación cuyo prototipo ha generado. Al implementar por etapas estas aplicaciones, se agregan mejoras multiusuario, se mueven los datos al origen de datos remoto, y se prueba y depura la aplicación, módulo a módulo, de forma sistemática.

Durante la implementación de la aplicación puede utilizar la sintaxis nativa del servidor y tener acceso a la funcionalidad específica del mismo como, por ejemplo, los procedimientos almacenados del servidor, con la tecnología de paso a través de SQL. Para obtener más información sobre paso a través de SQL, consulte el capítulo 21, [Implementación de una aplicación cliente-servidor](#).

### **Optimizar la aplicación**

Cuando la aplicación está plenamente implementada para datos remotos y ha completado la fase de prueba y depuración, puede ajustar la velocidad y el rendimiento de toda la aplicación. Para obtener más información sobre las mejoras que puede incluir en la aplicación implementada, consulte el capítulo 22, [Optimizar el rendimiento cliente-servidor](#)

## Incorporar precisión e integridad de datos

Puede combinar la eficacia de las reglas de validación de datos y los procedimientos almacenados de Visual FoxPro con las reglas de validación de datos y los procedimientos almacenados del origen de datos con el fin de generar aplicaciones cliente-servidor que protejan la integridad de los datos.

### Mantener la integridad de los datos

Puede crear versiones locales de las reglas de validación del servidor remoto para proporcionar al usuario mensajes comprensibles, por ejemplo, acerca de las actualizaciones que no se permiten cuando se envían a la tabla de apoyo ya que los datos introducidos han infringido alguna regla de validación de datos o de integridad relacional del servidor.

### Usar reglas de Visual FoxPro en una vista remota o en una vista fuera de línea

En las vistas remotas puede crear reglas a nivel de campo y de registro para validar datos introducidos localmente antes de enviarlos al origen de datos remoto. Puesto que el objetivo de estas reglas es impedir que se envíe al origen de datos cualquier dato que pueda ser rechazado por las reglas de integridad del servidor, debe reproducir las reglas del origen de datos en las reglas que usted cree para la vista remota. La función [DBSETPROP\(\)](#) sirve para crear reglas para las vistas.

**Sugerencia** Puede crear en una vista remota una regla de validación local que llame a un procedimiento almacenado del servidor remoto y que envíe al servidor en forma de parámetro el valor que desea validar. No obstante, el uso de un procedimiento almacenado remoto alarga el período de procesamiento durante la introducción de datos.

### Usar reglas del servidor

Quizá prefiera basarse en las reglas establecidas en el servidor para la validación de datos. Si ocurre un error, la rutina de tratamiento de errores puede llamar a la función [AERROR\(\)](#) para obtener información, incluyendo el número del mensaje de error, el texto del mensaje de error remoto y el controlador de conexión asociado al error.

### Usar desencadenantes del servidor

Aunque es posible crear desencadenantes de Visual FoxPro en tablas locales, no puede crearlos en las vistas. No obstante, sí puede utilizar desencadenantes en el origen de datos remoto. Los desencadenantes del servidor sirven para procesar actualizaciones secundarias de datos, como actualizaciones o eliminaciones en cascada. El uso de desencadenantes del servidor para procesar actualizaciones secundarias es más eficaz que el envío de múltiples comandos al servidor remoto desde la aplicación de Visual FoxPro.

## Protección contra pérdidas de datos

Visual FoxPro, como la mayoría de los orígenes de datos remotos, proporciona capacidades de registro de transacciones como protección frente a pérdidas de datos. Para obtener más información sobre el uso de las transacciones de Visual FoxPro, consulte el capítulo 17, [Programar para acceso compartido](#)

Puede utilizar las transacciones de Visual FoxPro para prototipos locales y para proceso de datos locales. Utilice las transacciones del servidor para actualizaciones, inserciones y eliminaciones de datos remotos. Para obtener más información acerca del uso de transacciones remotas, consulte el capítulo 22, [Optimizar el rendimiento cliente-servidor](#)

# Capítulo 20: Upsizing de bases de datos de Visual FoxPro

Una vez que haya diseñado la aplicación cliente-servidor, estará en condiciones de generar y realizar un upsizing del prototipo local. Un prototipo local es un modelo operativo de la aplicación que utiliza tablas, vistas y bases de datos de Visual FoxPro para representar datos a los que se tiene acceso en algún momento a través de un servidor remoto. El Asistente para upsizing se utiliza para mover bases de datos, tablas y vistas desde su sistema hasta un servidor SQL remoto o un servidor Oracle®.

Este capítulo trata los temas siguientes:

- [Objetivos para la creación de prototipos](#)
- [Generar un prototipo local de una aplicación](#)
- [Usar el Asistente para upsizing](#)
- [Upsizing a SQL Server](#)
- [Upsizing a Oracle](#)

## Objetivos para la creación de prototipos

Cuando utilice Visual FoxPro para generar un prototipo de su aplicación, aprovechará las posibilidades de los formularios, asistentes, generadores y diseñadores visuales y del Administrador de proyectos para programar rápidamente una aplicación operativa. Aunque su objetivo último es implementar la aplicación a través de plataformas cliente-servidor, obtendrá grandes ventajas si genera un buen prototipo.

## Reducir el tiempo de programación

Si construye un prototipo rápido, puede refinar el diseño y la arquitectura local de su aplicación de forma rápida y sencilla, sin necesidad de tener acceso al servidor remoto para volver a generar tablas y bases de datos del servidor. También puede probar y depurar los formularios de la aplicación frente a almacenes de datos de menor tamaño, lo que le permite corregir y mejorar con mayor rapidez la interfaz de usuario de la aplicación. Puesto que los costos generales de la arquitectura son reducidos, evita pérdidas de tiempo de programación en la regeneración, reindexación y reconexión de datos

remotos simplemente para probar el prototipo.

## **Reducir los costes de programación y aumentar la satisfacción del cliente**

Puesto que el prototipo local reside completamente en su PC, resulta sencillo hacer una demostración de un modelo operativo de la aplicación ante el usuario final en las primeras fases del ciclo de programación. Si el cliente ve la aplicación a medida que avanza, ganará confianza en su capacidad de proporcionar una solución que satisfaga sus necesidades. También le ofrece la oportunidad de obtener la opinión del cliente sobre la interfaz de usuario y los informes antes de invertir recursos en la implantación sobre un servidor remoto.

A medida que los usuarios vean e interactúen con su prototipo, pueden comenzar a identificar las áreas que les gustaría modificar, así como ver las posibilidades de agregar funcionalidad adicional a la aplicación. Puede implantar los cambios y volver a hacer demostraciones de la aplicación en un proceso iterativo hasta que usted y el cliente estén satisfechos con el diseño y las funciones de la aplicación cuyo prototipo se ha realizado. Posteriormente, el prototipo sirve como especificación operativa para la aplicación cliente-servidor final implementada.

## **Contribuir al éxito de la implementación**

También existe la posibilidad de proporcionar a los usuarios el prototipo de la aplicación como demostración, lo que les permite experimentar con el modelo operativo a medida que usted avanza en el proceso de implementación de la aplicación real. A medida que obtienen experiencia con el prototipo, su curva de aprendizaje se reduce y colaboran mejor a la hora de refinar y adaptar la aplicación. También logran una posición mejor para conseguir una mayor productividad y satisfacción en la etapa de la implementación final porque ya comprenden la estructura básica de la aplicación.

Disponer de un modelo operativo aumenta el tiempo previo para que el usuario final se familiarice y se sienta cómodo con la aplicación. También proporciona un marco que permite al personal de su empresa o del cliente diseñar y desarrollar un plan de formación para la aplicación. El prototipo se puede utilizar incluso para formar a los usuarios finales antes de la entrega de la aplicación final, contribuyendo de esta forma al éxito en la implementación de la aplicación cliente-servidor final.

## **Generar un prototipo local de una aplicación**

Para generar un prototipo local de la aplicación, puede comenzar desde cero o bien convertir una aplicación existente de Visual FoxPro en una aplicación cliente-servidor. La principal diferencia entre crear un prototipo local de una aplicación cliente-servidor y programar cualquier otra aplicación de Visual FoxPro radica en el uso de vistas y tablas locales para representar datos en los que, posteriormente, se realiza un upsizing.

### **Para crear y realizar un upsizing sobre un prototipo local**

1. Cree su aplicación con vistas y tablas locales para representar los datos que desea mover a un servidor remoto.
2. Utilice vistas locales en los formularios y el entorno de datos de la aplicación.

3. Realice un upsizing de las vistas y tablas locales mediante el [Asistente para upsizing a SQL Server](#) o [Asistente para upsizing a Oracle](#):
  - En el paso **Set Upsizing Options**, en el área **Changes to make locally**, seleccione **Redirect views to remote data**.

Cuando seleccione esta opción, el Asistente para upsizing copiará al servidor remoto las tablas locales que usted elija y redirigirá las vistas locales para utilizar datos remotos donde sea pertinente.

Para obtener más información sobre la creación de vistas, consulte el capítulo 8, [Crear vistas](#). Para obtener más información sobre la creación de formularios y el uso de un entorno de datos, consulte el capítulo 9, [Crear formularios](#). Para obtener información sobre la programación de una aplicación, consulte el capítulo 2, [Programar una aplicación](#).

## Usar el Asistente para upsizing

Visual FoxPro proporciona dos Asistentes para upsizing: el [Asistente para upsizing a SQL Server](#) y el [Asistente para upsizing a Oracle](#). Estos asistentes crean bases de datos de SQL Server u Oracle que duplican en la medida de lo posible las funciones de un conjunto de tablas de una base de datos de Visual FoxPro. También puede elegir si desea redirigir las vistas de Visual FoxPro de forma que utilicen los datos remotos recién creados en lugar de datos locales. Puede utilizar el Asistente para upsizing con el fin de:

- Mover datos locales a un servidor remoto.
- Transformar tablas base locales y vistas locales en tablas base remotas y vistas remotas.
- Migrar una aplicación local hacia una aplicación cliente-servidor.

**Nota** Aunque el Asistente para upsizing tiene acceso a los servidores SQL Server, puede crear una aplicación cliente-servidor para cualquier origen de datos ODBC remoto. Para otros servidores distintos de SQL Server, puede utilizar las funciones de paso a través de SQL para crear tablas remotas y después usar Visual FoxPro para crear vistas remotas que tengan acceso a las tablas del servidor. Para obtener más información sobre el uso de las funciones de paso a través de SQL, consulte el capítulo 21, [Implementar una aplicación cliente-servidor](#). Para obtener información sobre la creación de vistas remotas, consulte el capítulo 8, [Crear vistas](#).

## Upsizing a SQL Server

Antes de ejecutar el Asistente para upsizing, debe preparar tanto la parte del cliente como la parte del servidor.

### Preparar el lado SQL Server

Antes del upsizing, debe asegurarse de que dispone de los permisos necesarios en el servidor, estimar el tamaño de la base de datos y comprobar que el espacio en disco del servidor es suficiente. También existen preparativos especiales para un upsizing a múltiples discos o dispositivos.

### Comprobar el espacio libre en disco

Asegúrese de que el espacio en disco disponible en el servidor es suficiente.

**Precaución** Si el Asistente para upsizing a SQL Server se queda sin espacio libre en el servidor, se detendrá, dejando en el servidor una base de datos parcial y los dispositivos que haya creado. Puede eliminar los dispositivos, las bases de datos y las tablas con la herramienta de Administración de SQL Server.

### **Establecer permisos en las bases de datos de SQL Server**

Para ejecutar el Asistente para upsizing, tiene que disponer de determinados permisos en el servidor SQL Server hacia el que desea realizar el upsizing. Los permisos necesarios dependen de las tareas que desee realizar.

- Para un upsizing hacia una base de datos existente, necesita los permisos CREATE TABLE y CREATE DEFAULT.
- Para generar una nueva base de datos, necesita los permisos CREATE DATABASE y SELECT para las tablas del sistema de la base de datos principal.
- Para crear nuevos dispositivos, tiene que ser un administrador del sistema.

Para obtener más información sobre la concesión de permisos del servidor, consulte la documentación del servidor.

### **Estimar el tamaño de la base de datos y los dispositivos de SQL Server**

Cuando usted crea una nueva base de datos, el Asistente para upsizing a SQL Server le pide que seleccione dispositivos para la base de datos y el registro. También le solicita que defina el tamaño de la base de datos y de los dispositivos.

### **Estimar el tamaño de la base de datos de SQL Server**

Cuando SQL Server crea una base de datos, reserva una cantidad fija de espacio para dicha base de datos en uno o varios dispositivos. La base de datos no utiliza necesariamente todo este espacio; el tamaño solamente limita el crecimiento máximo de una base de datos grande antes de que se agote su espacio.

**Nota** Puede aumentar el tamaño de una base de datos de SQL Server después de haberla creado. Para obtener más información al respecto, consulte el comando ALTER DATABASE en la documentación de SQL Server.

Para estimar el espacio necesario para la base de datos, consulte el tamaño de los archivos .dbf de Visual FoxPro para las tablas en las que desea realizar el upsizing y calcule la velocidad a la que crecerá la nueva base de datos de SQL Server. En general, cada megabyte de datos de Visual FoxPro requiere por lo menos entre 1,3 y 1,5 megabytes en SQL Server.

Si dispone de mucho espacio en disco en el servidor, multiplique por dos el tamaño de las tablas de Visual FoxPro. Esto le garantiza que el Asistente para upsizing a SQL Server tendrá suficiente espacio para realizar el upsizing de la base de datos y dejará espacio para el crecimiento. Si piensa agregar gran cantidad de datos a la base de datos, aumente el múltiplo.



## **Estimar el tamaño de los dispositivos de SQL Server**

Todas las bases de datos y los registros de SQL Server se sitúan en dispositivos. Un dispositivo es una ubicación lógica donde se introducen bases de datos y registros, además de un archivo físico. Cuando se crea un dispositivo, SQL Server crea un archivo, reservando de esta forma una cantidad fija de espacio en disco para su uso propio.

El Asistente para upsizing a SQL Server muestra la cantidad de espacio libre disponible en los dispositivos de SQL Server existentes. Seleccione un dispositivo que tenga como mínimo suficiente espacio libre para el tamaño estimado de la base de datos.

Si ninguno de los dispositivos existentes tiene espacio libre suficiente, puede crear un nuevo dispositivo con el Asistente para upsizing a SQL Server. Los dispositivos nuevos deben tener como mínimo el tamaño estimado de su base de datos. Si es posible, defina el dispositivo con un tamaño mayor que el de la base de datos, lo que le permitirá ampliarla posteriormente o situar otras bases de datos o registros en el mismo dispositivo.

**Importante** El tamaño de los dispositivos no se puede modificar. Asegúrese de crear dispositivos cuyo tamaño sea suficientemente grande.

## **Usar múltiples discos o dispositivos de SQL Server**

En la mayoría de los casos, el Asistente para upsizing a SQL Server proporciona un control más que suficiente sobre los dispositivos de SQL Server. No obstante, si el servidor tiene múltiples discos o si desea situar una base de datos o un registro en múltiples dispositivos, quizá desee crear dispositivos antes de ejecutar el Asistente para upsizing a SQL Server.

### **Servidores con múltiples discos físicos**

Si su servidor tiene dos o más discos duros físicos, es posible que desee situar la base de datos en un disco y su registro en otro disco distinto. En caso de que falle el disco, tendrá más probabilidades de recuperar la base de datos si el registro y la base de datos están almacenados en distintos discos físicos.

El Asistente para upsizing a SQL Server le permite crear nuevos dispositivos, pero solamente en un disco físico: el mismo en el que está situado el dispositivo maestro de base de datos.

Para situar una base de datos y un registro en discos separados, asegúrese de que dispone en ambos discos de dispositivos con el tamaño suficiente, creando nuevos dispositivos en caso necesario. A continuación, ejecute el Asistente para upsizing a SQL Server.

### **Colocar bases de datos o registros en múltiples dispositivos**

SQL Server permite que las bases de datos y los registros abarquen varios dispositivos. No obstante, el Asistente para upsizing solamente le permite especificar un dispositivo para la base de datos y un dispositivo para el registro.

Para especificar múltiples dispositivos para una base de datos o un registro, convierta dichos

dispositivos (y ningún otro) en predeterminados. Posteriormente, ejecute el Asistente para upsizing y elija Predeterminado para el dispositivo de la base de datos o del registro.

**Nota** Si el tamaño de la nueva base de datos o del nuevo registro de SQL Server no necesita usar todos los dispositivos predeterminados, SQL Server solamente utilizará los dispositivos necesarios para alojar la base de datos o el registro.

## Preparar el cliente

Antes de realizar el upsizing, debe tener acceso a SQL Server a través de un origen de datos ODBC o de una conexión con nombre. También necesita tener una base de datos de Visual FoxPro, de la que debe realizar una copia de seguridad antes de ejecutar el Asistente para upsizing a SQL Server.

### Crear un origen de datos ODBC o una conexión con nombre

Al crear una nueva base de datos remota, se selecciona un origen de datos ODBC o una conexión con nombre en la base de datos de Visual FoxPro que tiene acceso al SQL Server al que quiere realizar el upsizing. Como no puede iniciar el Asistente para upsizing hasta que seleccione una conexión con nombre o un origen de datos, debe crear la conexión con nombre o el origen de datos adecuado antes de iniciar el proceso de upsizing.

Para obtener información sobre la creación de una conexión con nombre, consulte el 8, [Crear vistas](#). Para crear un origen de datos ODBC, ejecute el Administrador de ODBC. Para obtener información sobre la definición de orígenes de datos ODBC, consulte el capítulo 1, [Instalar Visual FoxPro](#), de la *Guía de instalación e Índice principal*.

### Copia de seguridad de la base de datos

Es conveniente crear una copia de seguridad de la base de datos (archivos .dbc, .dct y .dcx) antes de realizar el upsizing. Aunque el Asistente para upsizing no modifica los archivos .dbf, sí maneja el .dbc directamente abriéndolo como tabla a veces e indirectamente cambiando el nombre de las tablas y vistas al crear nuevas vistas remotas. Si realiza una copia de seguridad de la base de datos, podrá revertirla a su estado original previo al upsizing sobrescribiendo los archivos .dbc, .dct y .dcx modificados con las copias originales de la copia de seguridad, lo que invierte el cambio de nombres y la creación de vistas nuevas.

### Cerrar tablas

El Asistente para upsizing a SQL Server intenta abrir de forma exclusiva todas las tablas de la base de datos cuyo upsizing se desea realizar. Si alguna de las tablas ya está abierta y compartida, el asistente la cierra y la vuelve a abrir de forma exclusiva. Al abrir las tablas de forma exclusiva antes del upsizing, se protege contra usuarios que intenten modificar los registros de las tablas que se van a exportar durante la exportación de datos. Si alguna tabla no se puede abrir de forma exclusiva, el Asistente para upsizing a SQL Server mostrará un mensaje y estas tablas no estarán disponibles para el upsizing.

### Iniciar el Asistente para upsizing a SQL Server

Después de crear su origen de datos ODBC y de completar los preparativos necesarios en el cliente y

el servidor, podrá comenzar el upsizing.

### Para iniciar el Asistente para upsizing a SQL Server

1. En el menú **Herramientas**, elija **Asistentes** y, a continuación, elija **Upsizing**.
2. En el cuadro de diálogo **Selección de los asistentes**, elija **Asistente para upsizing a SQL Server**.
3. Siga las indicaciones de las pantallas del asistente, como se describe en las secciones posteriores.

Puede elegir el botón **Cancelar** en cualquier momento para salir del asistente; el asistente no ejecuta ninguna acción en el servidor hasta que usted elija el botón **Finalizar**.

4. Cuando vaya a hacer el upsizing, elija el botón **Finalizar**.

Tras elegir el botón Finalizar, el Asistente para upsizing a SQL Server comienza a exportar la base de datos al servidor.

El botón Finalizar está disponible después de proporcionar la información básica necesaria para el upsizing. Si elige el botón Finalizar antes de completar todas las pantallas del asistente, el Asistente para upsizing a SQL Server usará los valores predeterminados para las pantallas restantes.

### Funcionamiento del Asistente para upsizing a SQL Server

El Asistente para upsizing a SQL Server hace que la exportación mediante upsizing de una base de datos de Visual FoxPro a SQL Server sea casi transparente. Esta sección explica exactamente lo que ocurre al elegir el botón Finalizar: la forma en que el Asistente para upsizing exporta los datos y asigna objetos de Visual FoxPro a objetos de SQL Server.

#### Métodos de exportación de datos

El Asistente para upsizing a SQL Server exporta los datos utilizando un método entre dos disponibles. El primer método crea un procedimiento almacenado que ejecuta inserciones de múltiples filas. Este método puede ser muy rápido, puesto que los procedimientos almacenados se precompilan y se ejecutan rápidamente.

No obstante, los procedimientos almacenados no aceptan como parámetros variables binarias de longitud variable. Si desea exportar datos que se deben almacenar en tablas de SQL Server utilizando tipos de datos de texto o imagen, o bien tablas con más de 254 campos, el Asistente para upsizing a SQL Server utilizará otro método de exportación distinto. Este segundo método crea una instrucción SQL INSERT para cada fila de la tabla y después ejecuta la instrucción.

Si el Asistente para upsizing a SQL Server encuentra errores mientras exporta datos con el método SQL INSERT y el número de errores supera el 10% del número de registros de la tabla, o bien es superior a 100 registros (lo que sea mayor), el asistente cancelará la exportación de la tabla y guardará el número de errores de exportación para el informe de errores. No obstante, la tabla de servidor exportada no se borra y los registros que se lograron exportar sin problemas permanecen en la tabla.

del servidor.

## Introducción a la asignación de objetos

Para realizar el upsizing de una base de datos de Visual FoxPro hasta un servidor, el Asistente para upsizing a SQL Server crea objetos de servidor que, en la medida de lo posible, hagan lo mismo que la base de datos de Visual FoxPro. La asignación de algunos objetos de Visual FoxPro a objetos del servidor es muy directa: bases de datos, tablas, campos, valores predeterminados e índices de Visual FoxPro se asignan a bases de datos, tablas, campos, valores predeterminados e índices de SQL Server de forma directa, uno a uno.

Sin embargo, no todos los objetos locales se asignan directamente a objetos del servidor. Las reglas de validación y la integridad referencial de Visual FoxPro forman parte del diccionario de datos y se exigen a nivel del motor de base de datos. Las reglas de validación y la integridad referencial de SQL Server no forman parte del diccionario de datos y se exigen mediante código vinculado a una tabla. Estas diferencias, así como las decisiones de diseño realizadas por el Asistente para upsizing a SQL Server, significan que gran parte del diccionario de datos de Visual FoxPro no se puede asignar directamente a construcciones de SQL Server.

La siguiente tabla resume la forma en que se asignan los objetos desde Visual FoxPro hasta SQL Server:

<b>Objetos de Visual FoxPro</b>	<b>Objetos de SQL Server</b>
Base de datos	Base de datos
Tabla	Tabla
índices	índices
Campo	Campo
Valor predeterminado	Valor predeterminado
Regla de validación de tabla	Procedimientos almacenados de SQL Server, llamados desde desencadenantes UPDATE e INSERT
Regla de validación de campo	Procedimientos almacenados de SQL Server, llamados desde desencadenantes UPDATE e INSERT
Relaciones permanentes (cuando se usen para limitaciones de integridad referencial)	Desencadenantes Update, Insert y Delete

Las siguientes secciones tratan cada uno de los objetos de Visual FoxPro y el objeto (o los objetos) de SQL Server al que se asigna.

## Convenciones de nombres para objetos mediante upsizing

A medida que el Asistente para upsizing a SQL Server migra objetos a un origen de datos, crea objetos con nombre en el servidor. El Asistente utiliza prefijos para los objetos que necesitan

nombres nuevos debido a que no existía en Visual FoxPro ningún objeto autónomo de ese tipo (por ejemplo, valores predeterminados y reglas). Después del prefijo se incluye un nombre de tabla al que sigue un nombre de campo, en caso necesario. Esta convención de nombres permite que todos los objetos del mismo tipo tengan el mismo prefijo y se ordenen juntos cuando se presenten en pantalla con las herramientas de administración del origen de datos. Los objetos creados en la misma tabla también se agrupan cuando se muestran en pantalla.

## Objetos de base de datos y tabla

Una base de datos de Visual FoxPro se asigna directamente a una base de datos de SQL Server. Una tabla de Visual FoxPro, exceptuando parte de su diccionario de datos, se asigna a una tabla de SQL Server.

Los nombres de base de datos, tabla, índice y campo pueden cambiar durante el proceso de upsizing, en caso de que infrinjan las convenciones de nombres de SQL Server. Los nombres de SQL Server deben tener 30 caracteres como máximo, y el primero de ellos debe ser una letra o el símbolo “@”. Los demás caracteres pueden ser números, letras o los símbolos “\$”, “#” y “\_”; no se admiten espacios en blanco. El Asistente para upsizing sustituye los caracteres no válidos por el símbolo “\_”.

Los nombres que sean idénticos a palabras reservadas de SQL Server reciben el sufijo “\_”. Por ejemplo, FROM y GROUP se convierten en FROM\_ y GROUP\_. El Asistente para upsizing también incluye el símbolo “\_” delante de los nombres de objetos que comienzan por un número.

## Tablas

El Asistente para upsizing a SQL Server asigna a cada tabla exportada mediante upsizing el mismo nombre que la tabla local a menos que contenga un espacio o sea una palabra clave para el origen de datos.

## Vistas de nuevas tablas del servidor

Si selecciona “Crear vistas remotas de las tablas”, el Asistente para upsizing a SQL Server creará vistas remotas y les asignará muchas de las propiedades de los campos de la tabla local original.

## Asignar nombres de campos y tipos de datos de Visual FoxPro a sus homólogos de SQL Server

Los nombres de campos y los tipos de datos se convierten automáticamente en campos de SQL Server cuando se exporta una tabla de Visual FoxPro mediante el Asistente para upsizing a SQL Server.

Los tipos de datos de Visual FoxPro se asignan a tipos de datos de SQL Server de la siguiente forma:

Abreviatura	Tipo de datos de Visual FoxPro	Tipo de datos de SQL Server
C	<a href="#">Character</a>	char
Y	<a href="#">Currency</a>	money

D	<a href="#">Date</a>	datetime
T	<a href="#">DateTime</a>	datetime
B	<a href="#">Double</a>	float
F	<a href="#">Float</a>	float
G	<a href="#">General</a>	image
I	<a href="#">Integer</a>	int
L	<a href="#">Logical</a>	bit
M	<a href="#">Memo</a>	text
M (binario)	Memo (binary)	image
C (binario)	Character (binario)	binary
N	<a href="#">Numeric</a>	float

### Columnas marca de hora e identidad

Las columnas de marca de hora se crean con el tipo de datos marca de hora de Transact-SQL. Cuando selecciona la casilla de verificación de la columna Timestamp para una tabla específica en el paso 4-Map Field Data Types (Asignar tipos de datos de campos), el Asistente para upsizing a SQL Server crea un campo de marca de hora para la tabla.

Si la tabla contiene uno o más campos memo (M) o imagen (P), el Asistente para upsizing a SQL Server selecciona la casilla de verificación Timestamp para la tabla de forma predeterminada y crea un campo de marca de hora en la versión de la tabla resultante del upsizing.

Las columnas de identidad se crean con los campos de la propiedad Transact-SQL IDENTITY.

### Índices

Los índices de SQL Server y los de Visual FoxPro son muy parecidos. La siguiente tabla muestra la forma en que se convierten los tipos de índices de Visual FoxPro a tipos de índices de SQL Server:

### Conversión de tipos de índice

Tipo de índice de Visual FoxPro	Tipo de índice de SQL Server
<a href="#">Principal</a>	único agrupado
<a href="#">Candidato</a>	único
<a href="#">Único Normal</a>	No único

El Asistente para upsizing a SQL Server utiliza los nombres de etiquetas de Visual FoxPro como nombres para los índices de SQL Server. Si el nombre de etiqueta es una palabra reservada en el servidor, el asistente modificará el nombre de la etiqueta adjuntando el carácter “\_”.

**Nota** SQL Server no admite índices en orden ascendente ni descendente, ni permite expresiones dentro de los índices del servidor. El Asistente para upsizing a SQL Server elimina las expresiones de Visual FoxPro de las expresiones de índice al exportarlo; solamente envía al servidor los nombres de campos.

### Valores predeterminados de SQL Server

Una expresión predeterminada de Visual FoxPro se asigna directamente a un único valor predeterminado de SQL Server. El Asistente para upsizing a SQL Server intenta crear un valor predeterminado de SQL Server basándose en la expresión predeterminada para un campo de Visual FoxPro. Si logra crear el valor predeterminado, el Asistente para upsizing a SQL Server lo vinculará al campo apropiado de SQL Server. El informe de upsizing sobre los campos indica si el Asistente para upsizing a SQL Server ha logrado convertir la expresión de Visual FoxPro a Transact-SQL de SQL Server. Para ver más detalles sobre la conversión, consulte la sección [Asignar expresiones](#), más adelante en este mismo capítulo.

Aunque los valores predeterminados de SQL Server y de Visual FoxPro son muy parecidos, existen algunas diferencias en cuanto a su creación y comportamiento entre ambos productos. Los valores predeterminados de SQL Server son objetos autónomos, independientes de cualquier campo o tabla. Una vez creado un valor predeterminado, se puede utilizar o vincular a cualquier cantidad de campos distintos.

### Convenciones de nombres para valores predeterminados

El Asistente para upsizing a SQL Server asigna nombres a los valores predeterminados mediante el prefijo `Dflt_` más el nombre de la tabla y el del campo. Por ejemplo, un valor predeterminado para el campo `ordamt` de la tabla `Customer` se puede denominar `Dflt_Customer_Ordamt` en el servidor. Si la combinación del prefijo con los nombres de tabla y campo da como resultado un nombre superior a 30 caracteres, Visual FoxPro truncará los caracteres sobrantes.

Los campos con una expresión predeterminada de cero se vinculan a un valor predeterminado llamado `UW_ZeroDefault`. Si dos o más campos tienen la misma expresión predeterminada distinta de cero, el Asistente para upsizing a SQL Server creará dos valores predeterminados, con dos nombres distintos, cuya funcionalidad es idéntica.

### Valores predeterminados para campos lógicos de Visual FoxPro

Los campos lógicos de SQL Server no aceptan valores nulos, mientras que los campos lógicos de Visual FoxPro sí los admiten. Para solucionar esta diferencia, el Asistente para upsizing a SQL Server crea y vincula automáticamente un valor predeterminado denominado `UW_ZeroDefault` a cada campo lógico exportado, aunque usted no elija exportar los valores predeterminados. Este valor predeterminado establece el valor del campo en el servidor a 0 (o falso (.F.)), si muestra el campo en Visual FoxPro) cuando no se proporciona ningún valor.

Si la tabla local de Visual FoxPro contiene un valor predeterminado para un campo lógico que establece el campo igual a verdadero (.T.), el Asistente para upsizing a SQL Server no vinculará el valor predeterminado `UW_ZeroDefault` a la tabla del servidor. En su lugar, el asistente crea un valor predeterminado que establece el campo igual a 1 y asigna al valor predeterminado un nombre de acuerdo con las convenciones de nombres descritas anteriormente en este capítulo.

Los valores predeterminados de SQL Server se comportan de distinta forma que los de Visual FoxPro. Para obtener más información al respecto, consulte la sección [Valores predeterminados](#) más adelante en este mismo capítulo.

### Desencadenantes de SQL Server

Un desencadenante de SQL Server es una serie de instrucciones Transact-SQL asociadas con una tabla determinada de SQL Server. Cuando elija exportar mediante upsizing reglas de validación y relaciones en el paso 8, el Asistente para upsizing a SQL Server convertirá las reglas de validación a nivel de campo y de registro de Visual FoxPro, así como las relaciones permanentes entre tablas en procedimientos almacenados que se llaman desde los desencadenantes de SQL Server. Cada desencadenante del servidor contiene código para emular la funcionalidad de varias reglas de validación e integridad referencial.

**Nota** El Asistente para upsizing a SQL Server no exporta los desencadenantes de Visual FoxPro.

Una tabla del servidor puede tener tres desencadenantes, uno para cada comando que puede modificar datos en la tabla: UPDATE, INSERT y DELETE. El desencadenante se ejecuta automáticamente al ejecutar el comando asociado.

La siguiente tabla describe los desencadenantes creados por el Asistente para upsizing a SQL Server. Un desencadenante específico puede contener código para emular las funciones indicadas de Visual FoxPro, en su totalidad o en parte.

Desencadenante	Funcionalidad de Visual FoxPro emulada
UPDATE	Reglas de validación (validación a nivel de campo y de registro)
	Integridad referencial
INSERT	Reglas de validación (validación a nivel de campo y de registro)
	Integridad referencial (sólo desencadenantes de tablas secundarias)
DELETE (sólo tabla primaria)	Integridad referencial



---

## Convenciones de nombres para desencadenantes

El Asistente para upsizing asigna nombres a los desencadenantes del servidor combinando un prefijo, que indica el tipo de desencadenante que se va a crear, con el nombre de la tabla de SQL Server a la que pertenece el desencadenante. El prefijo (“TrigU\_” para desencadenantes UPDATE, “TrigD\_” para desencadenantes DELETE y “TrigI\_” para desencadenantes INSERT) se sitúa delante del nombre de la tabla. Por ejemplo, el desencadenante UPDATE de la tabla Customer se podría denominar TrigU\_Customer.

## Reglas de validación

El Asistente para upsizing puede exportar reglas de validación a nivel de campo y de registro de Visual FoxPro, que convierte a procedimientos almacenados en el SQL Server. El asistente asigna nombres a las reglas a nivel de campo combinando un prefijo “vrf” (del inglés, “regla de validación, campo”) con los nombres de la tabla y del campo; un ejemplo puede ser vrf\_customer\_lname. Las reglas de validación de tabla se denominan con el prefijo “vrt” (del inglés, “regla de validación, tabla”) más el nombre de la tabla, creando así un nombre como vrt\_customer.

El Asistente para upsizing a SQL Server utiliza desencadenantes que llaman a procedimientos almacenados en lugar de reglas de SQL Server para exigir una validación a nivel de campo porque las reglas de SQL Server no permiten ver mensajes de error personalizados. Para obtener más información sobre las reglas de SQL Server, consulte el comando CREATE RULE en la documentación de SQL Server.

## Integridad referencial

Visual FoxPro admite la integridad referencial a través de desencadenantes sobre los eventos UPDATE, DELETE e INSERT en relaciones persistentes de tablas que se exigen a nivel de motor. Puede elegir implementar restricciones de integridad referencial en SQL Server usando dos métodos:

- Integridad referencial basada en desencadenantes

–O bien–

- Integridad referencial declarativa

Cuando elige integridad referencial basada en desencadenantes, el Asistente para upsizing a SQL Server crea desencadenantes que incluyen el código Transact-SQL necesario para duplicar las restricciones de integridad referencial de Visual FoxPro. Si elige implementar integridad referencial declarativa, el Asistente para upsizing a SQL Server crea restricciones de SQL Server mediante el comando ALTER TABLE con la palabra clave CONSTRAINT.

## Integridad referencial basada en desencadenantes

En el método basado en desencadenantes, se hace cumplir la integridad referencial en SQL Server mediante código Transact-SQL en desencadenantes. Puede usar desencadenantes para proporcionar restricciones a instrucciones UPDATE, DELETE y INSERT, y para realizar en cascada las

modificaciones resultantes de las instrucciones DELETE y INSERT.

El Asistente para upsizing a SQL Server crea desencadenantes de SQL Server evaluando los desencadenantes de Visual FoxPro usados para hacer cumplir la integridad referencial en relaciones persistentes de la base de datos de Visual FoxPro. La siguiente tabla presenta la asignación entre las restricciones de integridad referencial de Visual FoxPro y los desencadenantes de SQL Server generados por el Asistente para upsizing a SQL Server.

**Restricción de integridad referencial de Visual FoxPro**

**Desencadenante de SQL Server**

UPDATE	Aplicar en cascada	Aplicar en cascada el desencadenante UPDATE
	Restringir	Restringir el desencadenante UPDATE
	Ignorar	No se genera ningún desencadenante
DELETE	Aplicar en cascada	Aplicar en cascada el desencadenante DELETE
	Restringir	Restringir el desencadenante DELETE
	Ignorar	No se genera ningún desencadenante
INSERT	Restringir	Restringir el desencadenante INSERT
	Ignorar	No se genera ningún desencadenante

Una relación persistente de Visual FoxPro utilizada en una limitación de integridad referencial puede convertirse en hasta cuatro desencadenantes de SQL Server: dos para la tabla primaria y dos para la tabla secundaria.

**Nota** Si solamente se exporta mediante upsizing una de las tablas de una relación o si no se exige la integridad referencial en Visual FoxPro, la relación no se exportará.

### **Tabla primaria**

El Asistente para upsizing a SQL Server crea un desencadenante UPDATE que impide que el usuario cambie la clave principal de la tabla primaria, o bien envía en cascada ese cambio a través de la tabla secundaria, dependiendo del tipo de relación que se creó en Visual FoxPro.

El Asistente para upsizing a SQL Server también crea un desencadenante DELETE que impide que el usuario elimine un registro con registros secundarios relacionados, o que elimina los registros secundarios, dependiendo siempre del tipo de relación existente entre las tablas de Visual FoxPro.

### **Tabla secundaria**

El Asistente para upsizing a SQL Server crea un desencadenante UPDATE que impide al usuario realizar en la clave externa cambios que convertirían el registro en huérfano. De la misma forma, un desencadenante INSERT se crea para impedir que el usuario agregue un registro nuevo que no tenga

tabla primaria.

### Valores de error personalizados

Cuando se infringe la integridad referencial establecida por el desencadenante creado por el asistente, el Asistente para upsizing a SQL Server incluye un valor de error personalizado en la variable @@ERROR. El Asistente para upsizing a SQL Server define valores de errores posibles como parte del código del desencadenante. El valor depende de la acción que el usuario intentaba realizar: actualizar, insertar o eliminar.

La siguiente tabla indica los números de error generados para cada acción:

<b>Acción</b>	<b>Error de SQL Server</b>
Regla de validación infringida	44444
Intento de eliminación	44445
Intento de actualización	44446
Intento de inserción	44447
Instrucción Update o Delete que afecta a dos o más filas; la instrucción se deshace automáticamente	44448

### Integridad referencial declarativa

Si elige implementar integridad referencial declarativa, el Asistente para upsizing a SQL Server crea restricciones de SQL Server mediante el comando ALTER TABLE con la palabra clave CONSTRAINT. La restricción de la tabla primaria usa la palabra clave PRIMARY KEY. La tabla secundaria usa las palabras clave FOREIGN KEY y REFERENCES. La integridad referencial declarativa es compatible con los niveles RESTRICT, actualizaciones RESTRICT y eliminaciones RESTRICT.

Puede usar restricciones de SQL Server para proporcionar restricciones a instrucciones UPDATE, DELETE y INSERT.

### Asignar expresiones

A pesar de que Visual FoxPro y Transact-SQL tienen algunas funciones en común, SQL Server no admite muchas de las funciones de Visual FoxPro. El Asistente para upsizing a SQL Server intenta convertir las expresiones de Visual FoxPro en reglas de validación a nivel de campo y de registro y en valores predeterminados para Transact-SQL, utilizando para ello la siguiente asignación de expresiones.

Expresión de Visual FoxPro	Expresión de SQL Server
Verdadero (.T.)	1
Falso (.F.)	0
#	<>
.AND.	AND
.NOT.	NOT
.NULL.	NULL
.OR.	OR
=<	<=
=>	>=
<a href="#">ASC()</a>	ASCII()
<a href="#">AT()</a>	CHARINDEX()
<a href="#">CDOW()</a>	DATENAME(dw, ...)
<a href="#">CHR()</a>	CHAR()
<a href="#">CMONTH()</a>	DATENAME(mm, ...)
<a href="#">CTOD()</a>	CONVERT(datetime, ...)
<a href="#">CTOT()</a>	CONVERT(datetime, ...)
<a href="#">DATE()</a>	GETDATE()
<a href="#">DATETIME()</a>	GETDATE()
<a href="#">DAY()</a>	DATEPART(dd, ...)
<a href="#">DOW()</a>	DATEPART(dw, ...)
<a href="#">DTOC()</a>	CONVERT(varchar, ...)
<a href="#">DTOR()</a>	RADIANS()
<a href="#">DTOT()</a>	CONVERT(datetime, ...)
<a href="#">HOUR()</a>	DATEPART(hh, ...)
<a href="#">LIKE()</a>	PATINDEX()
<a href="#">MINUTE()</a>	DATEPART(mi, ...)
<a href="#">MONTH()</a>	DATEPART(mm, ...)
<a href="#">MTON()</a>	CONVERT(money, ...)

<a href="#">NTOM()</a>	CONVERT(float, ...)
<a href="#">RTOD()</a>	DEGREES( )
<a href="#">SUBSTR()</a>	SUBSTRING( )
<a href="#">TTOC()</a>	CONVERT(char, ...)
<a href="#">TTOD()</a>	CONVERT(datetime, ...)
<a href="#">YEAR()</a>	DATEPART(yy, ...)

Las siguientes expresiones son iguales para Visual FoxPro y para SQL Server.

### Expresiones que se asignan directamente desde Visual FoxPro hasta SQL Server

<a href="#">CEILING()</a>	<a href="#">LOG()</a>	<a href="#">LOWER()</a>
<a href="#">LTRIM()</a>	<a href="#">RIGHT()</a>	<a href="#">RTRIM()</a>
<a href="#">SOUNDEX()</a>	<a href="#">SPACE()</a>	<a href="#">STR()</a>
<a href="#">STUFF()</a>	<a href="#">UPPER()</a>	

### Archivos creados por el Asistente para upsizing a SQL Server

El Asistente para upsizing a SQL Server crea tablas para su propio uso durante el proceso de upsizing. Estos archivos se eliminan del disco duro a menos que:

- Elija producir un informe de upsizing.
- Desee guardar el código SQL generado.
- Surjan errores durante el proceso de upsizing y elija guardar la información de error.

Si se producen algunas de las condiciones anteriores, el Asistente para upsizing a SQL Server creará un proyecto (denominado Informe, Informe1, Informe2, etc.) y una base de datos (denominada Upsize, Upsize1, etc.) en un subdirectorio (llamado UPSIZE) del directorio definido por el comando SET DEFAULT para la sesión de Visual FoxPro. El asistente agrega a la base de datos las tablas utilizadas para producir el Informe de upsizing, una tabla para almacenar el código SQL generado y cualquier tabla de error existente. La siguiente tabla indica los archivos que se pueden crear durante el proceso de upsizing.

### Tablas locales creadas durante el proceso de upsizing

Objetivo del archivo	Nombre de tabla	Contenido
Tablas de informe	Errors_uw	Información sobre cualquier error ocurrido durante el proceso de upsizing.

	Fields_uw	Información sobre todas las tablas exportadas mediante upsizing.
	Indexes_uw	Información sobre todos los índices exportados mediante upsizing.
	Misc_uw	Información diversa sobre el proceso de upsizing.
	Relations_uw	Información sobre todas las limitaciones de integridad referencial almacenadas en la base de datos de Visual FoxPro.
	Tables_uw	Información sobre todas las tablas de la base de datos que desea exportar mediante upsizing.
	Views_uw	Información sobre las vistas locales redirigidas para tener acceso a datos remotos.
Tabla de archivos de comandos	SQL_uw	Un campo memo que contiene todo el código SQL generado por el Asistente para upsizing a SQL Server.
Tablas de error de exportación de datos	ExportErrors_ <i>nombre_tabla</i>	Para cada tabla que experimente errores de exportación de datos durante el upsizing, el Asistente para upsizing a SQL Server genera una tabla que contiene los registros que no se han logrado exportar.

Si el asistente se cancela durante el procesamiento o si se detiene debido a un error, no dejará ninguna de estas tablas en el disco duro.

### Usar el código SQL generado

La tabla de archivos de comandos almacenada en el disco duro contiene todo el código SQL generado por el Asistente para upsizing, ya se ejecute con o sin errores en el servidor. Si desea utilizar este código, el mejor enfoque es examinar el SQL generado, copiar las partes que desee utilizar, ejecutar los fragmentos de código extraídos y repetir el proceso hasta obtener los resultados deseados. No puede ejecutar todo el archivo de comandos SQL en lugar del Asistente para upsizing a SQL Server, puesto que el asistente ejecuta pasos adicionales que no quedan reflejados en el código SQL generado.

### Fin del proceso de upsizing a SQL Server

Ahora puede ejecutar diversos pasos adicionales, tanto en el servidor como en la aplicación de Visual FoxPro, para asegurarse de que la aplicación y los datos están seguros y funcionan de forma

apropiada.

También puede utilizar la información de esta sección cuando genere una aplicación desde vistas remotas en lugar de mediante upsizing. Independientemente de la forma en que se crearon las tablas remotas, es posible llevar a cabo determinados pasos para garantizar que el servidor y el cliente están preparados para colaborar en la aplicación cliente-servidor.

## **Pasos en SQL Server**

Puede finalizar el proceso de upsizing en el servidor:

- Asegurándose de que las tablas desde Visual FoxPro que desea modificar son actualizables.
- Estableciendo permisos en la base de datos de forma que los usuarios puedan tener acceso a los objetos que necesitan.
- Protegiendo el trabajo realizado transformando en recuperable la nueva base de datos, por si se daña o se pierde.

### **Agregar índices únicos para permitir actualizaciones**

Una tabla remota debe tener un índice único para que se considere actualizable en Visual FoxPro. El Asistente para upsizing a SQL Server puede exportar un índice único existente, pero no crea ninguno cuando no existe. Asegúrese de que las tablas que desea modificar desde Visual FoxPro son actualizables.

### **Establecer permisos**

La nueva base de datos de SQL Server y sus objetos reciben un conjunto de permisos predeterminados de SQL Server. Establezca en la base de datos remota los permisos necesarios para que los usuarios puedan tener acceso a los objetos que necesitan.

### **Permisos de inicio de sesión en la base de datos**

Los permisos predeterminados de una base de datos nueva la hacen accesible solamente a los administradores del sistema y al propietario de la base de datos.

Puede agregar nuevos usuarios y grupos mediante el Administrador de seguridad de SQL Server o los procedimientos del sistema `sp_adduser` y `sp_addgroup`.

Para obtener más información sobre la forma de agregar usuarios y grupos, consulte la Ayuda del Administrador de seguridad de SQL Server y la documentación de los procedimientos del sistema `sp_adduser` y `sp_addgroup` en *Transact-SQL Reference* de *Microsoft SQL Server*.

### **Permisos de objetos**

Todos los objetos creados por el Asistente para upsizing a SQL Server, incluyendo las tablas, los desencadenantes y los valores predeterminados, en principio solamente son accesibles para el propietario de la base de datos y para los administradores del sistema. Esto se cumple cuando se realiza una exportación mediante upsizing hasta una base de datos nueva o existente. Si sobrescribe los objetos existentes, también sobrescribirá todos los permisos de objetos.

Para conceder permisos para tablas, utilice el Administrador de objetos SQL o los comandos GRANT y REVOKE. Para obtener más información sobre el establecimiento de permisos de objetos, consulte la sección “Managing Object Permissions” en el *Microsoft SQL Server Administrator's Companion* y los comandos GRANT y REVOKE de la *Microsoft SQL Server Transact-SQL Reference*.

### **Posibilidad de recuperación garantizada**

Proteja su trabajo convirtiendo la nueva base de datos en recuperable por si se daña o se pierde.

### **Volcar la base de datos maestra**

Cuando se crea una base de datos en un SQL Server, se agregan nuevos registros a las tablas del sistema de la base de datos maestra. El volcado de esta base de datos maestra le proporciona una copia de seguridad que incluye todos los cambios más recientes. Para obtener más información sobre el volcado de la base de datos maestra, consulte “Backing Up the master Database” del *Microsoft SQL Server Administrator's Companion*, y la instrucción DUMP y “Dumping the master Database” de la *Microsoft SQL Server Transact-SQL Reference*.

### **Programar copias de seguridad**

Programe copias de seguridad periódicas de la base de datos de forma que pueda restaurarla a partir de esta copia de seguridad en caso de que surja algún problema grave. Para realizar copias de seguridad de bases de datos SQL Server, vea “Database Maintenance Plan Wizard” y “Backup and Recovery” en *What's New in SQL Server 6.5* y “Database Design and Backup Strategy” en la *Microsoft SQL Server Transact-SQL Reference*.

### **Duplicar dispositivos**

La duplicación de dispositivos copia continuamente la información desde un dispositivo SQL Server hasta otro. En caso de que falle un dispositivo, el otro contiene una copia actualizada de todas las transacciones.

Si prevé que se realizarán muchos cambios en una base de datos entre una copia de seguridad y la siguiente, y no puede permitirse la pérdida de tales cambios, piense en la duplicación de dispositivos. Este proceso de duplicación es más efectivo cuando los dispositivos se encuentran en distintos discos, puesto que si están en el mismo disco y éste falla, se perderán ambos dispositivos.

Para obtener más información sobre la duplicación de dispositivos, consulte “Mirroring a Database Device”, “About SQL Server Device Mirroring” y “Using SQL Server Mirroring” en *Microsoft SQL Server Administrator's Companion*.

### **Pasos en el cliente de Visual FoxPro**

Cuando haya transferido los objetos desde Visual FoxPro hasta un SQL Server, probablemente necesitará modificar el código de la base de datos de Visual FoxPro original de manera que funcione adecuadamente con la nueva base de datos de SQL Server.

### **Optimizar vistas**



Las vistas creadas por el Asistente para upsizing a SQL Server no están parametrizadas, por lo que no están optimizadas. Para lograr un procesamiento más eficaz, agregue parámetros a las vistas creadas por el Asistente para upsizing a SQL Server con el fin de descargar solamente los datos necesarios. Para obtener información sobre la forma de agregar un parámetro a una vista, consulte el capítulo 8, [Crear vistas](#).

SQL Server no admite algunas funciones de Visual FoxPro. Si la vista remota creada por el Asistente para upsizing utiliza funciones que no se pudieron asignar a funciones de Transact-SQL, la vista no funcionará. Para obtener más información sobre la asignación de expresiones de Visual FoxPro a expresiones de Transact-SQL, consulte la sección [Asignar expresiones](#) anteriormente en este mismo capítulo.

### **Crear procedimientos almacenados y desencadenantes**

El Asistente para upsizing no exporta mediante upsizing los procedimientos almacenados ni los desencadenantes de Visual FoxPro. Si desea crear procedimientos almacenados o desencadenantes de SQL Server, puede utilizar Transact-SQL en el servidor, o bien paso a través de SQL en Visual FoxPro. Para obtener más información sobre el uso de Transact-SQL, consulte la documentación de SQL Server. Para obtener información sobre el uso de paso a través de SQL, consulte el capítulo 21, [Implementar una aplicación cliente-servidor](#).

### **Comparar el orden de eventos**

En Visual FoxPro, algunos eventos se producen en un orden distinto, dependiendo de si la aplicación utiliza datos de SQL Server o datos de Visual FoxPro. Estas diferencias pueden requerir cambios en el código.

### **Valores predeterminados**

Los valores predeterminados de campo de Visual FoxPro aparecen cuando comienza a modificar un registro nuevo. Los valores predeterminados generados por SQL Server aparecen solamente después de insertar un registro. Necesita cambiar el código que dependa de si hay valores antes de confirmar el registro, como puede ser el código para búsquedas.

### **Reglas de validación**

En Visual FoxPro, la validación de campo se produce cuando el enfoque abandona el campo. Al modificar datos de SQL Server en tablas adjuntas, los desencadenantes y las reglas no se activan hasta que no sale del registro. Quizá necesite modificar algunas reglas de validación de registros que dependen de que la validación del campo se realice al salir de dicho campo.

### **Tratar expresiones no convertidas**

El informe de upsizing indica si se han logrado convertir cada una de las reglas de validación de tabla, reglas de validación de campo y expresiones predeterminadas de Visual FoxPro. Si una expresión o una regla de validación predeterminada no se ha convertido, deberá volver a escribirla en Transact-SQL.

También puede realizar la validación a nivel de formulario en Visual FoxPro. No obstante, si los datos del servidor se modifican posteriormente sin utilizar un formulario determinado, la validación no se aplicará y quizá se introduzcan datos no válidos.

Para obtener más información sobre la conversión de expresiones, consulte la sección [Asignar expresiones](#), anteriormente en este mismo capítulo. Para obtener más información sobre las funciones de Transact-SQL, consulte la documentación de SQL Server.

## Bloquear registros

Visual FoxPro utiliza el bloqueo optimista de forma interna al tener acceso a tablas de un servidor SQL Server. El bloqueo optimista significa que la fila se bloquea solamente mientras se confirma el valor modificado y se realiza el proceso de actualización, lo que suele ser un intervalo muy breve.

En SQL Server se utiliza el bloqueo optimista en lugar del bloqueo pesimista, porque este último se realiza en SQL Server bloqueando páginas, lo que puede bloquear gran cantidad de registros simultáneamente. Aunque el bloqueo de páginas impide a otros usuarios realizar cambios en el mismo registro que se está modificando, también puede impedir que los usuarios tengan acceso a muchos otros registros de la misma página (bloqueada). El bloqueo optimista proporciona el mejor acceso multiusuario para una aplicación cliente-servidor de Visual FoxPro.

Puede optimizar las actualizaciones y controlar la forma en que se tratan los conflictos de actualización con la propiedad WhereType de SQL. Para obtener más información sobre la forma de controlar conflictos de actualización, consulte el capítulo 8, [Crear vistas](#).

## Upsizing a Oracle

El Asistente para upsizing a Oracle tiene un comportamiento similar al del Asistente para upsizing a SQL Server. Para obtener instrucciones paso a paso, vea [Asistente para upsizing a Oracle](#). Para obtener información sobre servidores Oracle, consulte la documentación de Oracle.

### Inicio del asistente para upsizing a Oracle

Después de haber creado una conexión con nombre o un origen de datos ODBC que se conecta a un servidor Oracle y haber terminado la preparación necesaria en el cliente y en el servidor, puede iniciar el upsizing.

#### Para iniciar el Asistente para upsizing Oracle

1. En el menú **Herramientas**, elija **Asistentes** y, a continuación, elija **Upsizing**.
2. En el cuadro de diálogo **Selección de los asistentes**, elija **Asistente para upsizing a Oracle**.
3. Siga las indicaciones de las pantallas del Asistente.

Puede elegir el botón **Cancelar** en cualquier momento para salir del Asistente; el asistente no ejecuta ninguna acción en el servidor hasta que usted elija el botón **Finalizar**.

4. Cuando vaya a hacer el upsizing, elija el botón **Finalizar**.

El botón **Finalizar** está disponible después de proporcionar la información básica para el upsizing. Si elige el botón **Finalizar** antes de completar todas las pantallas del asistente, el Asistente para upsizing a Oracle utiliza los valores predeterminados para las pantallas restantes.

Tras elegir el botón Finalizar, el Asistente para upsizing a Oracle comienza a exportar la base de datos al servidor.

## Capítulo 21: Implementar una aplicación cliente-servidor

Si ha creado un prototipo de trabajo local y ha realizado un upsizing del mismo, o ha programado la aplicación para a datos remotos con vistas remotas, tendrá acceso a los grandes almacenes de datos disponibles normalmente en una base de datos de servidor. También puede aprovechar las posibilidades de procesamiento de transacciones y de seguridad del servidor remoto. A la vez que las vistas remotas controlan las tareas de administración locales, puede mejorar la aplicación mediante la tecnología de paso a través de SQL (SQL pass-through, SPT) para crear objetos en el servidor, ejecutar procedimientos almacenados de servidor y ejecutar comandos con la sintaxis nativa de servidor.

Este capítulo trata técnicas para implementar tecnología cliente-servidor en una aplicación que usa vistas remotas. Si desea saber más sobre cómo diseñar y crear una aplicación cliente-servidor, consulte el capítulo 19, [Diseñar aplicaciones cliente-servidor](#) y el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#). Para obtener más información sobre cómo crear vistas remotas, consulte el capítulo 8, [Crear vistas](#).

En este capítulo se trata lo siguiente:

- [Usar la tecnología de paso a través de SQL](#)
- [Trabajar con datos remotos mediante paso a través de SQL](#)
- [Controlar errores de paso a través de SQL](#)

### Usar la tecnología de paso a través de SQL

Su aplicación cliente-servidor puede tener acceso a datos del servidor mediante:

- Vistas remotas
- Paso a través de SQL

Las vistas remotas proporcionan el método más común y más fácil para tener acceso a datos remotos y actualizarlos. Los asistentes para upsizing pueden crear automáticamente vistas remotas en la base de datos como parte del upsizing, o puede usar Visual FoxPro para crear vistas remotas después de hacer el upsizing. Para obtener más información sobre vistas remotas, consulte el capítulo 8, [Crear vistas](#).

La tecnología de paso a través de SQL le permite enviar instrucciones SQL directamente a un servidor. Las instrucciones de paso a través de SQL, porque se ejecutan en el servidor de aplicaciones, son formas eficaces de mejorar el rendimiento de las aplicaciones cliente-servidor. La tabla siguiente compara vistas remotas con paso a través de SQL:

### Comparación de las tecnologías vista remota y paso a través de SQL

Vista remota	Paso a través de SQL
Se basa en una instrucción SQL SELECT.	Se basa en cualquier instrucción nativa de SQL Server, permitiendo instrucciones de definición de datos o la ejecución de procedimientos almacenados de servidor.
Se puede usar como origen de datos para controles en tiempo de diseño.	No se puede usar como origen de datos para controles.
No ofrece la posibilidad de ejecutar comandos DDL sobre el origen de datos.	Proporciona un método para usar comandos DDL sobre el origen de datos.
Busca un conjunto de resultados.	Busca uno o varios conjuntos de resultados.
Proporciona administración de conexiones incorporadas.	Requiere administración de conexiones explícita.
Proporciona información de actualización predeterminada incorporada para actualizar, insertar y eliminar.	No proporciona información de actualización predeterminada.
Proporciona ejecución SQL implícita y búsqueda de datos.	Proporciona ejecución SQL explícita y control de búsqueda de resultados.
No proporciona control de transacciones.	Proporciona control de transacciones explícito.
Almacena propiedades de forma persistente en bases de datos.	Proporciona propiedades temporales para cursor de paso a través de SQL, en base a propiedades de sesiones.
Utiliza búsqueda progresiva asíncrona al ejecutar SQL.	Es compatible con búsqueda asíncrona a través de programa.

La tecnología de paso a través de SQL ofrece las siguientes ventajas frente a las vistas remotas:

- Puede usar funcionalidad específica de servidor, como procedimientos almacenados y funciones intrínsecas basadas en servidor.
- Puede usar extensiones a SQL admitidas por el servidor, así como definición de datos, administración de servidor y comandos de seguridad.
- Tiene más control sobre instrucciones de paso a través de SQL para actualizar, eliminar e insertar.
- Tiene más control sobre transacciones remotas.

**Sugerencia** Visual FoxPro puede controlar consultas de paso a través de SQL que devuelven más de un conjunto de resultados. Para obtener más información, consulte [Procesar múltiples conjuntos de resultados](#) más adelante en este mismo capítulo.

Las consultas de paso a través de SQL también tienen desventajas:

- De forma predeterminada una consulta de paso a través de SQL siempre devuelve un snapshot no actualizable de datos remotos, que se almacena en un cursor de vista activo. Puede convertir el cursor en actualizable con la función [CURSORSETPROP\(\)](#). En cambio, una vista remota actualizable normalmente no requiere que establezca propiedades antes de poder actualizar datos remotos, porque los valores de las propiedades se almacenan en la base de datos con la definición de la vista.
- Tiene que escribir comandos SQL directamente en la ventana Comandos o en un programa, en lugar de usar el Diseñador de vistas gráfico.
- Es usted quién crea y administra la conexión al origen de datos.

Si usa vistas remotas o paso a través de SQL, puede consultar y actualizar datos remotos. En muchas aplicaciones utilizará tanto vistas remotas como paso a través de SQL.

## Usar funciones de paso a través de SQL

Para usar el paso a través de SQL para conectarse a un origen de datos ODBC remoto, debe llamar en primer lugar a la función [SQLCONNECT\(\)](#) de Visual FoxPro para crear una conexión. Una vez hecho esto, use las funciones de paso a través de SQL para enviar comandos al origen de datos remoto para su ejecución.

### Para usar funciones de paso a través de SQL de Visual FoxPro

1. Confirme que el sistema puede conectar su equipo al origen de datos. Use una utilidad como ODBC Test para ODBC.
2. Establezca una conexión con su origen de datos con las funciones [SQLCONNECT\(\)](#) o [SQLSTRINGCONNECT\(\)](#).

Por ejemplo, si está conectando Visual FoxPro al origen de datos de SQL Server `sqlremote`, puede iniciar una sesión como administrador del sistema (id de usuario `sa`) con la contraseña `secret` con el siguiente comando:

```
nConnectionHandle = SQLCONNECT('sqlremote','sa','secret')
```

**Nota** También puede usar la función `SQLCONNECT( )` para conectarse a una conexión con nombre.

3. Use funciones de paso a través de SQL de Visual FoxPro para obtener datos en cursores de Visual FoxPro y procesar la obtención de datos comandos y funciones estándar de Visual FoxPro.

Por ejemplo, puede consultar a la tabla `authors` y examinar el cursor resultante usando este

comando:

```
? SQLEXP(nConnectionHandle,"select * from authors","mycursorname")
BROWSE
```

4. Desconéctese del origen de datos con la función `SQLDISCONNECT()`.

### Funciones de paso a través de SQL de Visual FoxPro

La siguiente tabla muestra las funciones SQL de Visual FoxPro que admiten el trabajo con orígenes de datos remotos, agrupados según la tarea.

Tarea	Función	Propósito
Administración de conexiones	<a href="#">SQLCONNECT()</a>	Conectarse a un origen de datos para operaciones de paso a través de SQL.
	<a href="#">SQLSTRINGCONNECT()</a>	Conectarse a un origen de datos con sintaxis de cadena de conexión ODBC.
	<a href="#">SQLDISCONNECT()</a>	Romper una conexión a un origen de datos ODBC, haciendo obsoleto el controlador de conexión especificado.
Control y ejecución de instrucciones SQL	<a href="#">SQLCANCEL()</a>	Cancelar una consulta SQL que se ejecuta de forma asíncrona en una conexión activa.
	<a href="#">SQLEXP()</a>	Ejecutar una consulta de paso a través SQL en una conexión activa; devuelve el número de resultados generados o 0 si <code>SQLEXP()</code> se sigue ejecutando (procesamiento asíncrono).
	<a href="#">SQLMORERESULTS()</a>	Colocar otro conjunto de resultados en un cursor. Devuelve 0 si la instrucción que está creando el conjunto de resultados se sigue ejecutando.
	<a href="#">SQLPREPARE()</a>	Precompilar la instrucción SQL en el origen de datos y vincular los parámetros de Visual FoxPro, es decir, guardar las expresiones de parámetros para todos los parámetros de la instrucción SQL.
	<a href="#">SQLCOMMIT()</a>	Sugerir un compromiso de

		transacción.
	<a href="#">SQLROLLBACK()</a>	Solicitar que se deshaga una transacción.
Información de origen de datos	<a href="#">SQLCOLUMNS()</a>	Almacenar una lista de nombres de columnas e información sobre cada uno en un cursor. Devuelve 1 si la función tiene éxito o 0 si la función se sigue ejecutando.
	<a href="#">SQLTABLES()</a>	Almacenar los nombres de tablas del origen en un cursor. Devuelve 1 si la función tiene éxito o 0 si la función se sigue ejecutando.
Control variado	<a href="#">SQLGETPROP()</a>	Obtener una propiedad de conexión de una conexión activa.
	<a href="#">SQLSETPROP()</a>	Establecer una propiedad de una conexión activa.

Las instrucciones `SQLEXP()`, `SQLMORERESULTS()`, `SQLTABLES()` y `SQLCOLUMNS()` se pueden cancelar en modo síncrono al presionar ESC si SET ESCAPE está establecido a ON. Puede cancelar estas instrucciones en cualquier momento en modo asíncrono si ejecuta `SQLCANCEL()`. Todas las demás instrucciones de paso a través de SQL funcionan de forma síncrona y no se pueden interrumpir.

### Crear conjuntos de resultados

Cuando usa las funciones de paso a través de SQL [SQLEXP\(\)](#) o [SQLMORERESULTS\(\)](#) para consultar datos, Visual FoxPro le devuelve los datos en uno o muchos conjuntos de resultados. Los conjuntos de resultados se originan a partir de cursores del origen de datos servidor y se convierten en cursores de Visual FoxPro. El nombre predeterminado para un conjunto de resultados es `SQLRESULT`.

### Acceso a procedimientos almacenados en servidor con funciones de paso a través de SQL

Puede usar la tecnología de paso a través de SQL de Visual FoxPro para crear y ejecutar procedimientos almacenados en un servidor remoto. Los procedimientos almacenados puede mejorar en gran medida la eficacia y flexibilidad de SQL y mejorar mucho el rendimiento de instrucciones SQL y lotes. Muchos servidores proporcionan procedimientos almacenados para definir y manipular objetos de base de datos servidor y para realizar administración de sistema servidor y de usuarios.

**Nota** Los ejemplos de este capítulo usan la sintaxis de Microsoft SQL Server, a no ser que se indique lo contrario.

### Para llamar a un procedimiento almacenado en servidor

- Use la función [SQLEXP\(\)](#) con el nombre del procedimiento almacenado.

Por ejemplo, el código siguiente muestra el resultado de llamar a un procedimiento almacenado llamado `sp_who` en SQL Server mediante una conexión activa al origen de datos `sqlremote`:

```
nConnectionHandle = SQLCONNECT('sqlremote')
? SQLEXPED(nConnectionHandle, 'use pubs')
? SQLEXPED(nConnectionHandle, 'sp_who')
BROWSE
```

Para obtener más información sobre la creación y ejecución de procedimientos almacenados, consulte la documentación del servidor.

## Devolver múltiples conjuntos de resultados

Si ejecuta un procedimiento almacenado que contiene instrucciones `SELECT` con sintaxis de servidor nativa, cada conjunto de resultados se devuelve a un cursor de Visual FoxPro distinto. Puede usar estos cursores para devolver valores o parámetros de un procedimiento almacenado de servidor al cliente de Visual FoxPro.

### Para devolver múltiples conjuntos de resultados

- Use la función [SQLEXPED\(\)](#) para seleccionar múltiples conjuntos de resultados usando su sintaxis de servidor nativa.

Por ejemplo, el código siguiente crea y ejecuta un procedimiento almacenado de servidor SQL, `my_procedure`, que devuelve tres cursores de Visual FoxPro: `sqlresult`, `sqlresult1` y `sqlresult2`:

```
=SQLEXPED(nConnectionHandle, 'create procedure my_procedure as ;
    select * from sales; select * from authors;
    select * from titles')
=SQLEXPED(nConnectionHandle, 'execute my_procedure')
```

## Cómo procesa el servidor conjuntos de resultados y errores

Como el servidor compila cada procedimiento almacenado cuando lo crea, recibe los errores de sintaxis de servidor en tiempo de creación. Cuando ejecute el procedimiento almacenado, el servidor ejecuta las instrucciones SQL compiladas secuencialmente (como en un programa de Visual FoxPro) y Visual FoxPro busca cada conjunto de resultados de cada instrucción SQL en el procedimiento almacenado de forma independiente, en el orden en que se ejecutan.

Los conjuntos de resultados y los errores se devuelven en el orden de recepción y el procesamiento se detiene si se encuentra un error. Por ejemplo, si ocurre un error en tiempo de ejecución cuando el servidor ejecuta la tercera instrucción de un procedimiento almacenado de cuatro instrucciones, recibirá los dos primeros conjuntos de resultados y, a continuación, recibirá el error que ocurrió al procesar el tercer conjunto de resultados. El procesamiento se detiene cuando se devuelve el error; el cuarto conjunto de resultados no se obtiene. Puede usar la función [AERROR\(\)](#) para obtener información sobre el error más reciente.

**Nota** Sólo puede ejecutar procedimientos almacenados de servidor de Visual FoxPro mediante las funciones de paso a través de SQL de Visual FoxPro. Las vistas no admiten procedimientos



almacenados de servidor, porque cada vista contiene una instrucción SQL SELECT explícita en su definición SQL.

## Transferir una instrucción SQL al origen de datos

La función [SQLEEXEC\(\)](#) le permite enviar una instrucción SQL al origen de datos sin interpretación. En el caso más sencillo, cualquier cadena que escriba en el segundo parámetro de la función [SQLEEXEC\(\)](#) se pasa al origen de datos sin interpretación. Esto le permite ejecutar cualquier instrucción mediante el SQL nativo del origen de datos.

También puede usar la función [SQLEEXEC\(\)](#) para crear una consulta parametrizada o pasar extensiones ODBC a SQL al origen de datos.

## Crear una consulta parametrizada

Así como puede crear vistas parametrizadas mediante el [Diseñador de vistas](#) o el lenguaje, puede crear una consulta de paso a través de SQL parametrizada.

### Para crear una consulta parametrizada con paso a través de SQL

- Coloque un signo de cierre de interrogación (?) antes de un parámetro de Visual FoxPro y, a continuación, incluya el parámetro en una cadena SQL que envíe con [SQLEEXEC\(\)](#).

El parámetro que suministre se evalúa como expresión de Visual FoxPro y el valor se envía como parte de la instrucción SQL de la vista. Si la evaluación falla, Visual FoxPro le pide el valor del parámetro.

**Sugerencia** Si el parámetro es una expresión, escriba la expresión del parámetro entre paréntesis. Esto asegura que toda la expresión se evalúa como parte del parámetro.

Por ejemplo, si tiene la tabla `customer` de la base de datos `Testdata` en un servidor remoto, el código siguiente crea una consulta parametrizada que limita la vista a aquellos clientes cuyo país coincida con el valor suministrado para el parámetro `?cCountry`:

```
? SQLEEXEC(1, 'SELECT * FROM customer WHERE customer.country = ?cCountry')
```

Si desea pedir al usuario el valor de un parámetro, escriba la expresión de parámetro entre comillas. Para obtener más información sobre la petición de un valor de parámetro, consulte el capítulo 8, [Crear vistas](#).

Los orígenes de datos ODBC no aceptan parámetros en las siguientes ubicaciones:

- En una lista de tablas o campos SELECT.
- Como las dos expresiones de un predicado de comparación.
- Como los dos operandos de un operador binario.

Un origen de datos ODBC no aceptará parámetros en las siguientes ubicaciones en las cláusulas WHERE o HAVING de una instrucción SELECT:

- Como los operandos primero y segundo de un predicado BETWEEN.
- Como los operandos primero y tercero de un predicado BETWEEN.
- Como la expresión y el primer valor de un predicado IN.
- Como el operando de un operador unario + o -.
- Como el argumento de una función SET.

## Usar parámetros de entrada-salida de SQL Server

Puede usar parámetros de entrada-salida para pasar valores entre Visual FoxPro y SQL Server. Los parámetros de entrada-salida sólo están disponibles mediante paso a través de SQL; no se pueden usar en vistas.

La tabla siguiente proporciona un ejemplo del uso de parámetros de entrada-salida para pasar valores desde Visual FoxPro a un procedimiento almacenado de SQL, devolviendo el resultado a una variable de Visual FoxPro.

### Usar parámetros de entrada-salida en un procedimiento almacenado de SQL Server

Código	Comentarios
<pre>resultCode = SQLExec(connHand,     "CREATE PROCEDURE sp_test;       @mult1 int, @mult2 int, @result int;     OUTPUT AS SELECT       @result = @mult1 * @mult2")</pre>	<p>Crea un procedimiento almacenado, sp_test, que multiplica dos variables (mult1 y mult2) y, a continuación, almacena la cantidad resultante en la variable result.</p>
<pre>outParam = 0</pre>	<p>Crea una variable de Visual FoxPro para recibir el valor del parámetro de salida cuando se pasa desde SQL Server a Visual FoxPro.</p>
<pre>resultCode = SQLExec(connHand, ;" {CALL sp_test (2, 4, ?@outParam)}")</pre>	<p>Ejecuta el procedimiento almacenado en SQL Server, pasando los valores '2' y '4' para que se multipliquen en el procedimiento almacenado.</p>
<pre>? "outParam =", outParam &amp;&amp; el valor es 8</pre>	<p>Muestra el valor del parámetro de salida.</p>

## Definir parámetros

La sintaxis para los parámetros de salida es:

*?@nombre\_de\_parámetro*

Cuando implemente parámetros de entrada-salida, defina las variables de Visual FoxPro que desee incluir en su comando de paso a través de SQL antes de usar las variables en su instrucción SQL. Para enviar y recibir correctamente información con parámetros de entrada-salida, debe definir:

- Un parámetro de procedimiento almacenado, con un tipo de salida, que devuelva un valor.

Por ejemplo, si el parámetro del procedimiento almacenado es `@result`, debe asignar un tipo de salida, como `int`, a `@result` y debe asignar un valor a `@result`.

- Una expresión de parámetro de salida (*@nombre\_de\_parámetro*) que se evalúa a una variable existente de Visual FoxPro.

Por ejemplo, si la expresión de parámetro de salida es `?@outParam`, la aplicación debe tener definida la variable `outParam` de Visual FoxPro.

**Nota** Si no usa un parámetro de salida, en Visual FoxPro o en el procedimiento almacenado, o no define una variable de Visual FoxPro para recibir el valor devuelto, el parámetro de Visual FoxPro no cambiará.

## Convertir tipos de datos

Visual FoxPro convierte valores devueltos de variable siguiendo las reglas siguientes:

- Las variables de tipo de datos de coma flotante (N, F, B) se convierten en N.
- El tamaño de presentación se establece como 20.
- La configuración decimal se establece como la actual. La configuración decimal sólo afecta al formato de presentación predeterminado y no afecta a la precisión decimal.
- Las variables de fecha y hora (D, T) se convierten a variables de hora (T).

No puede usar tipos de datos Memo, General, Picture o NULL en parámetros de entrada-salida.

Si la aplicación usa campos de cursor como parámetros, Visual FoxPro intentará convertir el resultado otra vez al tipo de datos original.

## Devolver valores de parámetros

Los parámetros de entrada-salida sólo están disponibles después de que se ha buscado el último conjunto de resultados de una instrucción. Esto significa que los valores de entrada-salida se devuelven a Visual FoxPro sólo después de que:

- [SQLEXEC\(\)](#) devuelva (1) en modo de proceso por lotes

–O bien–

- [SQLMORERESULTS\(\)](#) devuelva (2) en modo de no-proceso por lotes.

Si la instrucción `SQLEEXEC()` solicita múltiples conjuntos de resultados, los parámetros de salida sólo estarán disponibles con seguridad después de que se haya buscado el último conjunto de resultados en el origen de datos.

## Crear combinaciones externas con datos remotos

Puede usar el paso a través de SQL para realizar combinaciones externas sobre datos remotos con sintaxis de servidor nativo, si el servidor no admite combinaciones externas. Una combinación externa combina información de una o más tablas independientemente de si se encuentran filas coincidentes.

### Para realizar una combinación externa en un servidor

- Use la función [SQLEEXEC\(\)](#) con la sintaxis de combinación externa de servidor.

Por ejemplo, el código siguiente usa la función de paso a través de SQL de Visual FoxPro `SQLEEXEC()` para mostrar los resultados de una combinación externa en SQL Server mediante la conexión con nombre activa `sqlremote`:

```
? SQLEEXEC(sqlremote, 'select au_fname, au_lname, pub_name ;  
                      from authors, publishers ;  
                      where authors.city *= publishers.city')  
BROWSE
```

Para obtener más información sobre la sintaxis de combinación externa y los tipos de combinaciones externas, consulte la documentación del servidor. Para obtener información sobre la creación de una conexión con nombre, consulte "Definir una conexión" en el capítulo 8, [Crear vistas](#).

## Usar extensiones ODBC de SQL

Puede usar [SQLEEXEC\(\)](#) para ejecutar extensiones ODBC de SQL si encierra la instrucción SQL con sintaxis de escape estándar o extendida de grupo de acceso SQL. Para obtener más información sobre extensiones ODBC a SQL, consulte el apéndice Gramática SQL de la documentación de ODBC.

### Crear combinaciones externas mediante cláusula de escape ODBC

Puede usar paso a través de SQL para realizar combinaciones externas en datos remotos con la sintaxis de escape ODBC, si su servidor admite combinaciones externas. Una combinación externa combina información de una o más tablas independientemente de si se han encontrado filas coincidentes.

La sintaxis para combinaciones externas con la cláusula de escape de ODBC es:

{oj *expresión de combinación externa*}

El ejemplo siguiente crea un conjunto de resultados de los nombres y departamentos de los empleados que trabajan en el proyecto 544:

```
SELECT employee.name, dept.deptname;
```

```
FROM {oj employee LEFT OUTER JOIN dept;  
      ON employee.deptid = dept.deptid};  
WHERE employee.projid = 544
```

Para obtener más información sobre sintaxis de combinaciones externas y tipos de combinaciones externas, consulte la documentación del servidor. Para obtener más sobre la creación de conexiones con nombre, consulte "Definir una conexión" en el capítulo 8, [Crear vistas](#).

## Administrar conexiones con paso a través de SQL

Cuando crea una vista remota, debe elegir un nombre de origen de datos ODBC o un nombre de conexión que posteriormente se utiliza como canalización para el servidor remoto, una vez activada la vista. Para tener acceso a datos remotos directamente con paso a través de SQL, es necesario disponer del controlador para una conexión activa. Un controlador es un valor que hace referencia a un objeto; en este caso, el controlador hace referencia a una conexión de origen de datos. Para obtener un controlador, debe solicitar una conexión con el origen de datos utilizando la función [SQLCONNECT\(\)](#) o [SQLSTRINGCONNECT\(\)](#). Si la conexión se realiza correctamente, la aplicación recibirá un controlador de conexión para su uso en llamadas posteriores de Visual FoxPro.

La aplicación puede solicitar múltiples conexiones para un origen de datos. También puede trabajar con múltiples orígenes de datos ODBC solicitando una conexión con cada origen de datos al que desea tener acceso. Si desea reducir el número de conexiones utilizadas, puede configurar vistas remotas para compartir la misma conexión. Puede desconectarse de un origen de datos mediante la función [SQLDISCONNECT\(\)](#).

**Sugerencia** Visual FoxPro se basa en la definición del origen de datos ODBC almacenada en el archivo Odbc.ini de Windows o en el registro de Windows NT para conectarse a un origen de datos. Si cambia el nombre de la información de inicio de sesión para un origen de datos, tenga presente que estos cambios pueden influir en que una aplicación que utilice ese origen de datos pueda conectarse o no al servidor remoto deseado.

## Controlar propiedades de entorno y conexión

El entorno cliente-servidor se establece cada vez que se abre Visual FoxPro. El entorno existe para esa sesión y desaparece al cerrar Visual FoxPro. El entorno cliente-servidor contiene:

- Propiedades globales que actúan como prototipos para las nuevas conexiones.
- Valores de error para los errores que ocurren fuera de una conexión especificada.

Puede utilizar un controlador de 0, el controlador de "entorno", para hacer referencia al valor de propiedad global. La función [SQLSETPROP\(\)](#) sirve para controlar el valor de la propiedad predeterminada en el entorno de conexión y de las propiedades dentro de conexiones individuales. Los métodos empleados para introducir valores [SQLSETPROP\(\)](#) son coherentes para las conexiones de entorno e individuales:

- Las propiedades especificadas con uno de los dos valores posibles pueden emplear un valor lógico (.F. o .T.) para *expresión*.
- Los nombres de propiedad se pueden abreviar hasta la forma más breve que no dé lugar a ambigüedades. Por ejemplo, puede utilizar "Asynchronous", "Asynch" o "A" para especificar la propiedad Asynchronous. Los nombres de propiedades no distinguen entre mayúsculas y

minúsculas.

Cuando se inicia una conexión, ésta hereda valores predeterminados para las propiedades de la conexión. Puede usar `SQLSETPROP( )` para modificar estos valores.

### Establecer propiedades de conexión

Para ver el valor actual de propiedades para una conexión, utilice [SQLGETPROP\(\)](#) con el controlador de conexión adecuado. La siguiente tabla indica las propiedades de conexión a las que puede tener acceso con [SQLGETPROP\(\)](#).

### Propiedades de conexión de Visual FoxPro

Para	Use esta propiedad	Objetivo
Mostrar la información utilizada para crear la conexión activa	ConnectionString	La cadena de conexión de inicio de sesión.
	DataSource	El nombre del origen de datos, según lo ha definido ODBC.
	Password	La contraseña de conexión.
	UserID	La identificación del usuario.
Trabajar con conexiones compartidas	ConnectBusy	Verdadero (.T.) si una conexión compartida está ocupada; falso si no lo está (.F.).
Controlar la apariencia de la interfaz	DispLogin	Controla cuándo se muestra el cuadro de diálogo Inicio de sesión de ODBC.
	DispWarnings	Controla si se muestran o no los mensajes de error.
Controlar los intervalos de tiempo	ConnectTimeout	Especifica el tiempo (en segundos) que hay que esperar antes de devolver un error de fin de tiempo de espera para la conexión.
	IdleTimeout	Especifica el intervalo de tiempo de espera inactivo (en segundos). Las conexiones activas en proceso de cualificación se desactivan tras el intervalo de tiempo especificado. <sup>1</sup>
	WaitTime	Controla la cantidad de tiempo en milisegundos que transcurre antes de que Visual FoxPro compruebe si la instrucción SQL ha terminado de ejecutarse.

	QueryTimeout	Controla el tiempo (en segundos) que hay que esperar antes de devolver un error general de fin de tiempo de espera.
Administrar transacciones	Transactions	Determina la forma en que la conexión administra las transacciones en la tabla remota.
Controlar la recopilación de conjuntos de resultados en los cursores de presentación	Asynchronous	Especifica si los conjuntos de resultados se devuelven de forma síncrona (valor predeterminado) o asíncrona.
	BatchMode	Especifica si SQLEXP( ) devuelve todos los conjuntos de resultados a la vez (valor predeterminado), o individualmente con SQLMORERESULTS( ).
	PacketSize	Especifica el tamaño del paquete de red utilizado por la conexión.
Mostrar controladores ODBC internos	ODBCChdbc <sup>2</sup>	El controlador interno de conexión ODBC que pueden utilizar los archivos de bibliotecas externas (archivos .fl) para llamar a las funciones API de ODBC.
	ODBCchstmt <sup>2</sup>	El controlador interno de instrucciones ODBC que pueden utilizar los archivos de bibliotecas externas (archivos .fl) para llamar a las funciones API de ODBC.

1. En el modo de transacción manual, la conexión no se desactiva.

2. Si una conexión se desactiva, los valores ODBCChdbc y ODBCchstmt dejan de ser válidos. No libere estos valores en una biblioteca de usuario.

Para obtener más información sobre las propiedades de conexión y su valor predeterminado, vea [SQLSETPROP\(\)](#).

### Controlar valores de propiedades de entorno

Los valores que establezca en el entorno Visual FoxPro con el controlador 0 se usarán como prototipos o valores predeterminados para todas las conexiones o los adjuntos posteriores.

### Para ver el valor actual de las propiedades de entorno

- Utilice [SQLGETPROP\(\)](#) con 0 como valor para el controlador.

El ejemplo siguiente muestra en la pantalla el valor de la propiedad WaitTime del entorno actual:

```
? SQLGETPROP(0, "WaitTime")
```

Si establece la propiedad DispWarnings como verdadera (.T.), Visual FoxPro mostrará los errores de entorno a partir de ese punto y también establecerá DispWarnings como verdadera (.T.) para las conexiones recién creadas.

Aunque los valores que establece para el controlador 0 se utilizan como valores prototipo para cada conexión, también puede establecer propiedades personalizadas para una conexión individual si ejecuta [SQLSETPROP\(\)](#) para ese controlador de conexión. La excepción es la propiedad ConnectTimeout, cuya configuración hereda la conexión en el momento de realizarse. Si cambia el valor de la propiedad ConnectTimeout, este nuevo valor no se utilizará hasta que establezca una nueva conexión.

### Controlar objetos conexión y vista

Puede controlar las conexiones y vistas estableciendo propiedades en el objeto conexión o vista. Las propiedades que controlan bases de datos, tablas, campos de tablas, definiciones de vistas, campos de vistas, conexiones con nombre, conexiones activas o cursores de vista activos se llaman *propiedades de motor*. Puede mostrar o establecer propiedades de motor con una de las siguientes funciones de Visual FoxPro:

Para mostrar propiedades de motor use	Para establecer propiedades de motor use
<a href="#">CURSORGETPROP()</a>	<a href="#">CURSORSETPROP()</a>
<a href="#">DBGETPROP()</a>	<a href="#">DBSETPROP()</a>
<a href="#">SQLGETPROP()</a>	<a href="#">SQLSETPROP()</a>

La función que use depende de si desea establecer propiedades en el objeto 0 (conexión 0 y cursor 0), la definición de objeto de una base de datos (conexión con nombre o definición de vista) o el objeto activo (conexión activa o cursor de vista activo). La tabla siguiente muestra objetos y las funciones que se usan para establecer propiedades de cada objeto:

Para establecer propiedades para	Conexión	Vista
Objeto 0	<a href="#">SQLSETPROP()</a>	<a href="#">CURSORSETPROP()</a>
Definición de objeto de una base de datos	<a href="#">DBSETPROP()</a>	<a href="#">DBSETPROP()</a>
Objeto activo	<a href="#">SQLSETPROP()</a>	<a href="#">CURSORSETPROP()</a>

### Propiedades de motor



La tabla siguiente presenta propiedades de motor ordenadas alfabéticamente con los objetos que usan cada propiedad.

## LEYENDA

○ Sólo lectura

● Sólo lectura local; lectura-escritura remota

● Escritura-lectura

**Propiedades**

	Base de datos	Tabla	Campo de tabla	Definición de vista	Campo de vista	Definición de conexión	Conexión activa	Cursor activo
Asynchronous						●	●	
BatchMode						●	●	
BatchUpdateCount			●					●
Buffering								●
Caption			●	●				
Comment	●	●	●	●	●	●		
CompareMemo			●					●
ConnectBusy							○	
ConnectHandle								○
ConnectName			●				○	○
ConnectionString						●	○	
ConnectTimeout						●	●	
Database								○
DataSource						●	○	
DataType				○				
DefaultValue			○	●				
DeleteTrigger		○						
DispLogin						●	●	
DispWarnings						●	●	
FetchAsNeeded			●					●
FetchMemo			●					○
FetchSize			●					●
IdleTimeout						●	●	
InsertTrigger		○						
KeyField				●				
KeyFieldList								●
MaxRecords			●					●
ODBCdbc							○	
ODBCstmt							○	
Offline			○					○
PacketSize						●	●	
ParameterList			●					●
Password						●	○	
Path		○						
Prepared			●					●
PrimaryKey		○						
QueryTimeout						●	●	
RuleExpression		○	○	●	●			
RuleText		○	○	●	●			

SendUpdates				●				●
ShareConnection				●				○
SourceName								○
SourceType				○				○
SQL				○				○
Tables				●				●
Transactions						●	●	
Updatable					●			
UpdatableFieldList								●
UpdateName					●			
UpdateNameList								●
UpdateTrigger		○						
UpdateType				●				●
UseMemoSize				●				○
UserID						●	○	
Version	○							
WaitTime						●	●	
WhereType				●				●

Propiedad de motor	Aplicable a
Asynchronous	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
Batchmode	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
BatchUpdateCount <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
Buffering	Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
Caption	Campos de tablas, campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
Comment	Bases de datos, tablas, campos de tablas, definiciones de vista, campos de definiciones de vista, definiciones de conexión: vea <a href="#">DBSETPROP()</a> .
CompareMemo	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
ConnectBusy	Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
ConnectHandle	Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
ConnectName <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLGETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
ConnectionString	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> .

	Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
ConnectTimeout	Definiciones de conexiones: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
Database	Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
DataSource	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
DataType	Campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
DefaultValue	Campos de tablas, campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
DeleteTrigger	Tablas: vea <a href="#">DBGETPROP()</a> .
DispLogin	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
DispWarnings	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
FetchAsNeeded	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
FetchMemo <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
FetchSize <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
IdleTimeout	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
InsertTrigger	Tablas: vea <a href="#">DBGETPROP()</a> .
KeyField	Campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
KeyFieldList <sup>2</sup>	Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
MaxRecords <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
ODBCHdbc	Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
ODBCHstmt	Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
Offline	Definiciones de vista: vea <a href="#">DBGETPROP()</a> .
PacketSize	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
ParameterList	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .

Password	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
Path	Tablas: vea <a href="#">DBGETPROP()</a> .
Prepared	Definiciones de vista: vea <a href="#">DBSETPROP()</a> .
PrimaryKey	Tablas: vea <a href="#">DBGETPROP()</a> .
QueryTimeOut	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
RuleExpression	Tablas, campos de tablas, definiciones de vista, campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
RuleText	Tablas, campos de tablas, definiciones de vista, campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
SendUpdates <sup>2</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
ShareConnection	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
SourceName	Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
SourceType	Definiciones de vista: vea <a href="#">DBGETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
SQL	Definiciones de vista: vea <a href="#">DBGETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
Tables <sup>2</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
Transactions	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
Updatable	Campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
UpdatableFieldList <sup>2</sup>	Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
UpdateName	Campos de definiciones de vista: vea <a href="#">DBSETPROP()</a> .
UpdateNameList <sup>2</sup>	Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
UpdateTrigger	Tablas: vea <a href="#">DBGETPROP()</a> .
UpdateType	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .
UseMemoSize <sup>1</sup>	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORGETPROP()</a> .
UserID	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> .

	Conexiones activas: vea <a href="#">SQLGETPROP()</a> .
Version	Bases de datos: vea <a href="#">DBGETPROP()</a> .
WaitTime	Definiciones de conexión: vea <a href="#">DBSETPROP()</a> . Conexiones activas: vea <a href="#">SQLSETPROP()</a> .
WhereType	Definiciones de vista: vea <a href="#">DBSETPROP()</a> . Cursores de vista activa: vea <a href="#">CURSORSETPROP()</a> .

1. Propiedad principalmente útil para vistas remotas; el valor no tiene efecto sobre el rendimiento de vistas locales. Puede configurar esta propiedad en vistas locales si desea preestablecer la propiedad en la vista local y, a continuación, hacer un upsizing para crear una vista remota.

2. La propiedad se tiene que establecer para que se envíen actualizaciones al origen de datos remoto.

### Usar transacciones con datos remotos

Puede ajustar las transacciones alrededor de actualizaciones, eliminaciones e inserciones de datos remotos mediante uno de estos dos métodos:

- Modo de transacción automática
- Modo de transacción manual

El modo de transacción que seleccione determina cómo controla Visual FoxPro las transacciones en su equipo local.

### Usar el modo de transacción automática

De forma predeterminada, Visual FoxPro ajusta automáticamente en una transacción todos los comandos que se pueden incluir en una transacción enviados al servidor remoto. Este tratamiento automático de transacciones predeterminado se proporciona cuando la propiedad Transactions está establecida a 1 o a DB\_TRANSAUTO.

### Para usar el modo de transacción automática

- Utilice la función [DBSETPROP\(\)](#) para establecer la propiedad Transactions de la conexión a 1 o a DB\_TRANSAUTO.

–O bien–

- Utilice la función [SQLSETPROP\(\)](#) para establecer la propiedad Transactions de la conexión activa a 1 o a DB\_TRANSAUTO.

El proceso de transacciones para la tabla remota se controla automáticamente.

**Nota** Los comandos de Visual FoxPro [BEGIN TRANSACTION](#) y [END TRANSACTION](#) crean una transacción únicamente para el cursor local de Visual FoxPro. No extienden la transacción al servidor remoto.

## Control manual de las transacciones

Si desea controlar las transacciones de forma manual, puede establecer la propiedad Transactions como 2 o como DB\_TRANSMANUAL. Con el tratamiento manual de las transacciones, Visual FoxPro inicia automáticamente una transacción cuando se ejecuta la primera instrucción SQL que se puede incluir en transacciones, pero es necesario ejecutar la función `SQLCOMMIT()` o `SQLROLLBACK()` de Visual FoxPro para finalizar la transacción.

### Para usar el modo de transacción manual

- Utilice el comando [DBSETPROP\(\)](#) para establecer la propiedad Transactions de la conexión como 2 o DB\_TRANSMANUAL.  
  
–O bien–
- Utilice el comando [SQLSETPROP\(\)](#) para establecer la propiedad Transactions en la conexión activa como 2 o DB\_TRANSMANUAL.

El proceso de transacciones se controla automáticamente mediante [SQLCOMMIT\(\)](#) y [SQLROLLBACK\(\)](#).

Tras confirmar o deshacer la transacción previa, Visual FoxPro inicia automáticamente una nueva transacción cuando se ejecuta la siguiente instrucción SQL que puede incluir en transacciones. Para obtener más información acerca de las transacciones, consulte el capítulo 17, [Programar para acceso compartido](#)

## Transacciones anidadas

Visual FoxPro admite transacciones anidadas hasta un máximo de cinco niveles. El paso a través de SQL incorpora un solo nivel de anidamiento de transacciones.

Si su servidor admite múltiples niveles de transacciones, puede utilizar paso a través de SQL con el fin de administrar explícitamente los niveles de transacciones. No obstante, la administración explícita de transacciones no es recomendable, ya que puede dificultar el control de la interacción entre la transacción incorporada y la temporización de las transacciones del servidor remoto. Para obtener más información sobre la administración explícita de transacciones, consulte la documentación de ODBC.

## Trabajar con datos remotos mediante paso a través de SQL

Cuando recupere un conjunto de resultados mediante paso a través de SQL, puede ver y controlar las propiedades de su cursor de conjunto de resultados mediante las funciones [CURSORGETPROP\(\)](#) y [CURSORSETPROP\(\)](#) de Visual FoxPro. Estas funciones son las mismas que las que se emplean para establecer las propiedades en un cursor de vista activo.

**Nota** Los cursores no son objetos y no están vinculados al modelo de objetos. No obstante, puede ver sus propiedades o atributos, con `CURSORGETPROP()` y establecer sus propiedades con

CURSORSETPROP( ).

## Establecer las propiedades del cursor para datos remotos

La siguiente tabla indica las propiedades de cursor de Visual FoxPro que permiten trabajar con vistas y conjuntos de resultados conectados, agrupadas por categorías de tareas.

### Propiedades de cursor de Visual FoxPro

Tarea	Propiedad	Objetivo
Ver la definición del cursor	SQL	Contiene la instrucción SQL a partir de la cual se creó el cursor.
Controlar las interacciones entre Visual FoxPro y ODBC	ConnectHandle	Controlador para la conexión remota empleada por el cursor.
	ConnectName	Nombre de la conexión empleada por el cursor.
	Prepare	Especifica si la consulta para la vista se prepara antes de ejecutarse.
	FetchAsNeeded	Especifica si las filas se buscan automáticamente durante el bucle inactivo o sólo cuando es necesario.
	CompareMemo	Especifica si los campos Memo y General participan en la cláusula WHERE de una instrucción UPDATE, independientemente del valor de la propiedad UpdateType
	FetchMemo	Especifica si los campos Memo y General se buscan automáticamente con los conjuntos de resultados o si se buscan después, previa petición, cuando se abra el campo Memo o General.
	UseMemoSize	Especifica el tamaño mínimo de columna (1 a 255) de los conjuntos de resultados cuyas columnas se devuelven en los campos Memo.
	FetchSize	Especifica el número de filas que se buscan simultáneamente desde el conjunto de resultados remoto.
	MaxRecords	Especifica el número máximo de filas

		buscadas cuando se devuelven los conjuntos de resultados.
Actualizar datos	SendUpdates*	Especifica si las actualizaciones del cursor se envían a las tablas en las que se basa el cursor.
	BatchUpdateCount	Especifica el número de instrucciones de actualización enviadas al servidor para tablas almacenadas en búfer.
	Tables*	Lista delimitada por comas de nombres de tablas del origen de datos; se usa para definir el alcance de las propiedades UpdateNameList y UpdatableFieldsList.
	KeyFieldList*	Lista delimitada por comas de campos de Visual FoxPro que representan la clave principal del conjunto de resultados utilizado para las actualizaciones.
	UpdateNameList*	Lista delimitada por comas que combina los campos de Visual FoxPro del cursor con los nombres de tabla y columna de los campos a los que desea enviar las actualizaciones.
	UpdatableFieldList*	Lista delimitada por comas de los campos de Visual FoxPro para los que se envían actualizaciones.
	Buffering	Especifica el tipo de almacenamiento en búfer que se ejecuta en el cursor.
	UpdateType	Especifica si se debe realizar una actualización utilizando cláusulas UPDATE o DELETE, y luego INSERT.
	WhereType	Especifica qué se debe incluir en la cláusula WHERE para las actualizaciones de los datos de la tabla.

\* Propiedades que es necesario establecer antes de que se permita actualizar datos.

Estas propiedades se utilizan para controlar la forma en que la aplicación interactúa con datos remotos, como el establecimiento del número de filas recuperadas durante la búsqueda progresiva y el control de almacenamiento en búfer y de actualizaciones de datos remotos.

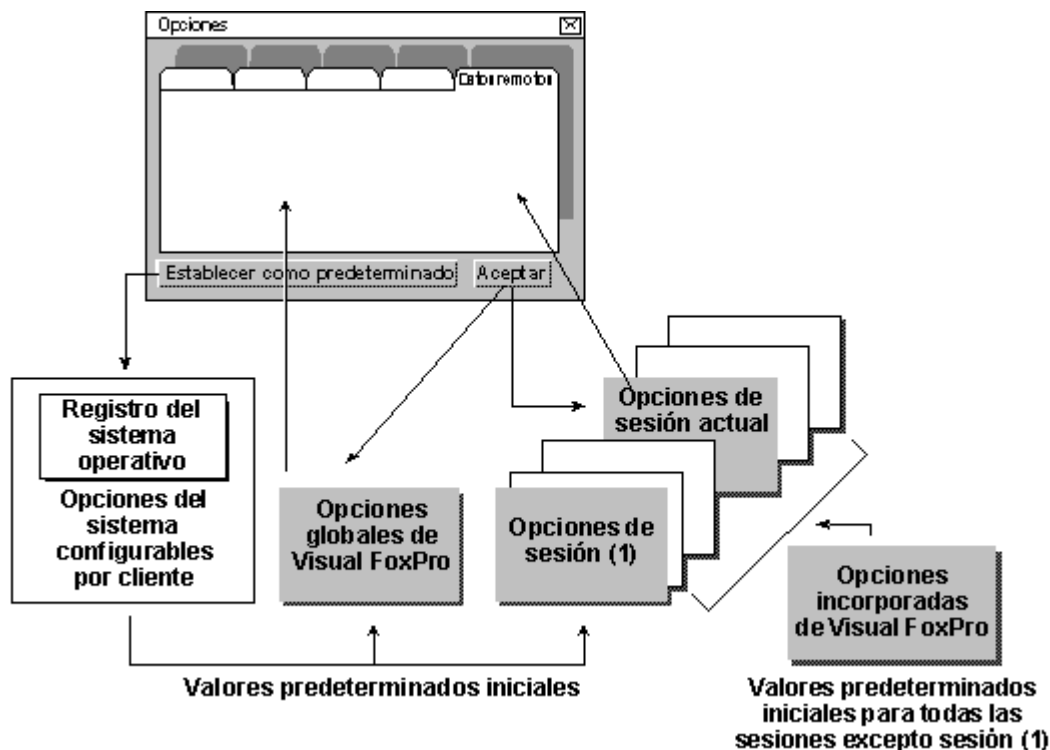
### Usar la ficha Datos remotos del cuadro de diálogo Opciones



Algunas propiedades de cursor heredan sus valores iniciales del entorno, mientras que otras solamente están disponibles a nivel del cursor. Algunas propiedades están disponibles para cursores que representan vistas remotas y tablas conectadas de paso a través de SQL u ODBC.

Puede controlar algunos valores de conexión y de cursor a través de la ficha **Datos remotos** del cuadro de diálogo **Opciones**. Al presentar la ficha **Datos remotos**, los valores del cuadro de diálogo representan las configuraciones del cursor para la sesión actual y las configuraciones predeterminadas globales de Visual FoxPro para la conexión. Cuando modifique los valores de la ficha **Datos remotos** y elija **Aceptar**, los nuevos valores se guardarán en la sesión actual del cursor y en la configuración predeterminada global de la conexión. Si elige **Establecer como predeterminado**, los valores se incluirán en las opciones configurables del sistema en su equipo. El siguiente diagrama ilustra estas interacciones.

### Ver y establecer valores globales y de sesión con el cuadro de diálogo Opciones



### Establecer propiedades con paso a través de SQL

Cuando cree un cursor, éste heredará los valores de sus propiedades, tales como UpdateType y UseMemoSize, del cursor del entorno o del cursor 0 de la sesión actual. Puede modificar este valor predeterminado de propiedad utilizando la función [CURSORSETPROP\(\)](#) con el número de cursor 0.

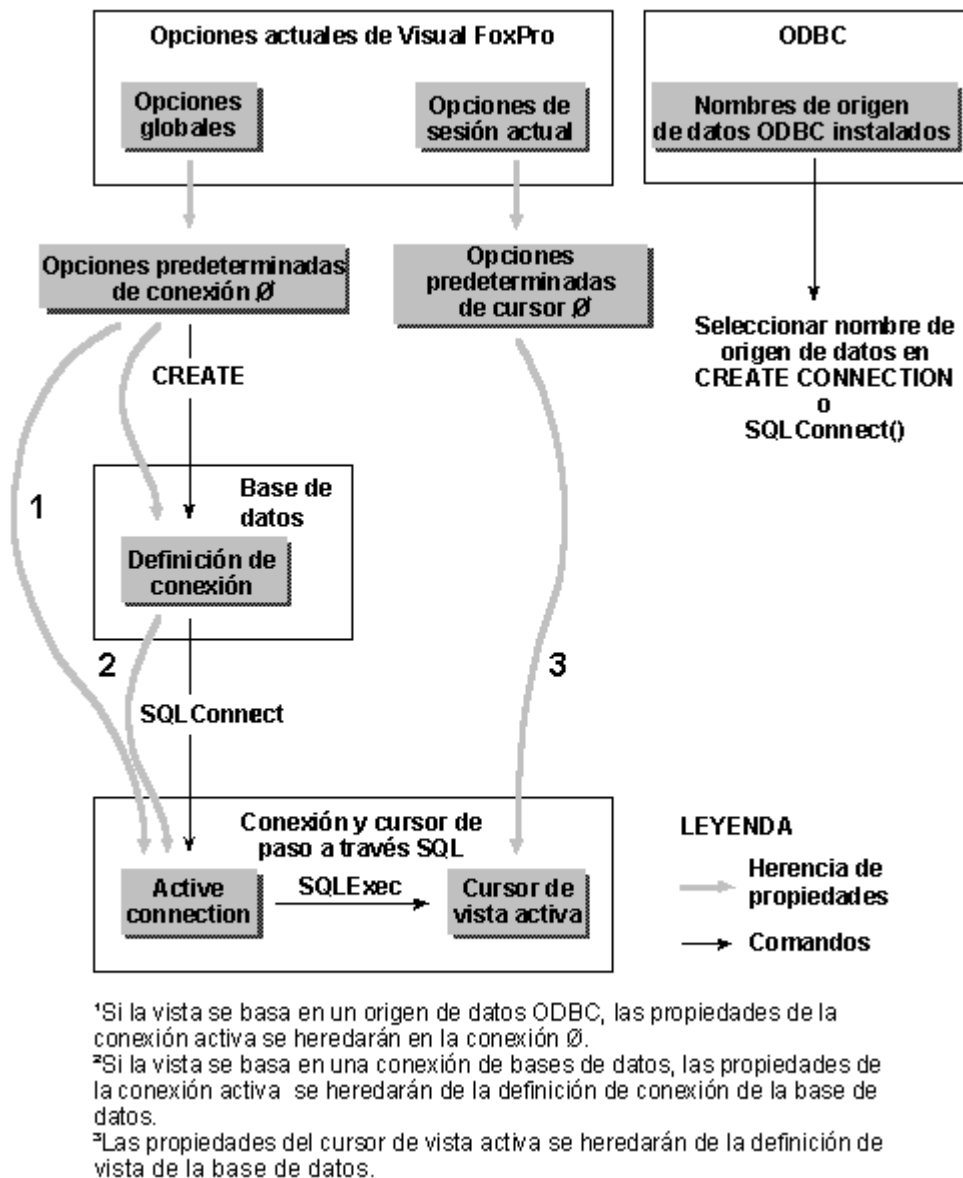
Tras crear un cursor de vista con paso a través de SQL, puede modificar los valores de las propiedades del cursor activo mediante la función `CURSORSETPROP()` para el cursor de vista. Los cambios que realice con `CURSORSETPROP()` serán temporales: la configuración temporal para la vista activa desaparece cuando se cierra dicha vista y la configuración temporal para el cursor 0 queda eliminada cuando se cierra la sesión de Visual FoxPro.

Las conexiones heredan las propiedades de forma similar. Las propiedades predeterminadas para la conexión 0 se heredan al crear y almacenar una conexión con nombre en una base de datos. Puede modificar estos valores predeterminados de las propiedades para la conexión 0 con la función [SQLSETPROP\(\)](#). Una vez creada la conexión y almacenada en una base de datos, puede modificar las propiedades de la conexión con la función [DBSETPROP\(\)](#). Cuando utilice una conexión, la conexión activa heredará los valores de las propiedades almacenados en la base de datos. Puede modificar estas propiedades en la conexión activa mediante la función [SQLSETPROP\(\)](#) para el controlador de conexión.

Tanto los cursores de vista de paso a través de SQL como las conexiones con nombre pueden utilizar un origen de datos ODBC con nombre. Si utiliza un origen de datos ODBC en un cursor de vista de paso a través de SQL, la conexión heredará sus propiedades de los valores predeterminados de la sesión.

El siguiente diagrama ilustra las propiedades heredadas de los cursores y las conexiones creados con paso a través de SQL. Las líneas grises representan el flujo de herencia de las propiedades y las líneas negras representan los comandos de Visual FoxPro.

### **Conexión de paso a través de SQL (SPT) y herencia de propiedades de cursor**



## Actualizar datos remotos con paso a través de SQL

Al utilizar funciones de paso a través de SQL para actualizar datos en un servidor remoto, se controla si se desean actualizar los datos, así como detalles específicos acerca de las actualizaciones, estableciendo propiedades en el cursor del conjunto de resultados. Visual FoxPro comprueba estas propiedades cuando se solicita una actualización antes de realizarla.

Para actualizar datos remotos es necesario definir cinco propiedades: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList y SendUpdates. Puede especificar propiedades adicionales, tales como Buffering, UpdateType y WhereType, para adecuarse mejor a los requisitos de la aplicación.

### Para permitir actualizaciones en un cursor de vista activo

- Utilice la función [CURSORSETPROP\(\)](#) para especificar las propiedades de actualización del

cursor de vista: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList y SendUpdates.

**Sugerencia** Los cursores de vista de paso a través de SQL no admiten actualizaciones hasta que no especifique propiedades de actualización para el cursor de vista. Si desea almacenar los valores de las propiedades de actualización de forma definitiva, cree una definición de vista. Visual FoxPro suministra valores predeterminados que preparan la vista para que sea actualizable cuando se crea mediante el Diseñador de vistas o el lenguaje. Puede utilizar la función `CURSORSETPROP()` con el fin de agregar información adicional para personalizar o aumentar la cantidad de valores predeterminados.

Las propiedades de actualización que se establecen en el cursor de vista activo tienen nombres ligeramente distintos que sus correspondientes en `DBSETPROP()`. La siguiente tabla incluye los nombres utilizados para definiciones de vista y para cursores activos.

### Propiedades de actualización de vistas y cursores

Objetivo	Propiedades de definición de vista <sup>1</sup>	Propiedades del cursor activo <sup>2</sup>
Hacer actualizables las tablas remotas.	Tablas	Tablas
Especificar nombres remotos para campos de vista.	UpdateName (propiedad a nivel de campo)	UpdateNameList
Especificar campos de vista que desea usar como claves.	KeyField (propiedad a nivel de campo)	KeyFieldList
Especificar los campos de vista que son actualizables.	Updatable (propiedad a nivel de campo)	UpdatableFieldList
Activar actualizaciones.	SendUpdates	SendUpdates

1 Se establecen con `DBSETPROP()`.

2 Se establecen con `CURSORSETPROP()`.

Para obtener más información sobre cómo establecer propiedades de actualización, consulte el capítulo 8, [Crear vistas](#) o vea [DBSETPROP\(\)](#) o [CURSORSETPROP\(\)](#).

### Controlar la temporización de actualizaciones remotas

El control del almacenamiento en búfer de las actualizaciones de datos remotos se realiza estableciendo la propiedad Buffering del cursor. De los cinco posibles valores para la propiedad de almacenamiento en búfer, dos son válidos para vistas remotas:

- 3, o `DB_BUFOPTROW`, el valor predeterminado, que bloquea la fila de forma optimista.

- 5, o DB\_BUFOPTTABLE, que bloquea la tabla de forma optimista.

Visual FoxPro solamente admite el bloqueo optimista en cursores remotos.

**Nota** Los valores de almacenamiento pesimista en búfer de tabla y fila, 2 y 4, no se aplican a vistas remotas, porque Visual FoxPro no bloquea los datos del servidor. El valor de la propiedad de almacenamiento en búfer 1 no se aplica a vistas remotas porque las vistas siempre se almacenan en búfer.

### Usar el almacenamiento optimista de filas en búfer

El valor predeterminado de Buffering, DB\_BUFOPTROW, bloquea de forma optimista datos remotos de fila en fila. Por ejemplo, si desea que los cambios realizados en la tabla `titles` se realicen fila a fila, como cuando se utiliza el comando `SKIP`, puede establecer la propiedad `Buffering` con el valor 3:

```
CURSORSETPROP('buffering', 3, 'titles')
```

Cuando establece `Buffering` para el almacenamiento de filas en búfer, dispone de dos métodos para enviar actualizaciones al servidor remoto. Puede:

- Llamar a la función [TABLEUPDATE\(\)](#).
- Usar un comando que mueva el puntero de registro a un lugar fuera de la fila, como [SKIP](#) o [GO BOTTOM](#).

La función `TABLEUPDATE()` actualiza el servidor sin mover el puntero de registro. Los comandos que mueven el puntero de registro envían actualizaciones al servidor remoto como efecto secundario de su salida del registro actualizado.

Si utiliza el almacenamiento de filas en búfer y desea tener la posibilidad de revertir los cambios realizados en las filas, debe ajustar los cambios en una transacción utilizando funciones de transacción de paso a través de SQL.

### Usar almacenamiento optimista de tablas en búfer

Si desea que los cambios realizados en una tabla se realicen por lotes, como cuando el usuario hace clic en un botón Guardar o Aceptar en un formulario, puede establecer la propiedad `Buffering` con el valor 5, o `DB_BUFOPTTABLE`. Es necesario llamar a la función `TABLEUPDATE()` para enviar la actualización al servidor.

En el ejemplo siguiente, se establece la propiedad `Buffering` como el código de inicialización del formulario y luego se realizan los cambios en el código de almacenamiento.

Código	Comentarios
<code>CURSORSETPROP('buffering', 5, 'sqltitles')</code>	Establecer en código Init
* Actualizar cambios en lotes;	

\* sin importar los realizados por otros

---



---

```
TABLEUPDATE(.T., .T., 'titles')
```

---



---

Establecer en código  
Save

Para restaurar en una tabla los valores originales e impedir que las actualizaciones se envíen al servidor remoto, llame a [TABLEREVERT\(\)](#). Puede controlar si se revierte una sola fila o varias combinando el valor de la propiedad Buffering del cursor con el comando TABLEUPDATE(). El siguiente ejemplo revierte solamente la fila actual. Quizá le resulte conveniente invocar este código cuando el usuario haga clic en un botón Cancelar de un formulario:

```
= TABLEREVERT(.F., 'titles')      && Revertir fila actual
```

Si desea revertir todas las filas, por ejemplo cuando el usuario presiona esc para salir de un formulario, puede utilizar el mismo ejemplo, pero cambiando el valor de la propiedad Buffering y el comando TABLEUPDATE() para revertir todas las filas, con toda la tabla almacenada en búfer:

```
= TABLEREVERT(.T., 'titles')      && Revertir todas las filas
```

Para obtener más información sobre el almacenamiento en búfer, consulte el capítulo 17, [Programar para acceso compartido](#).

## Detectar cambios realizados por otros usuarios

En aplicaciones multiusuario, los conflictos con las actualizaciones de otros usuarios se detectan mediante la consulta SQL Update, que se genera cuando se intenta realizar una escritura localmente. El nivel de detección depende del valor de la propiedad WhereType. Para obtener más información sobre los valores de la propiedad WhereType, consulte el capítulo 8, [Crear vistas](#).

## Imponer actualizaciones

Puede utilizar la función [TABLEUPDATE\(\)](#) para controlar si los cambios realizados en una tabla o cursor por otro usuario de la red se sobrescriben cuando usted envía sus propias actualizaciones. Si establece el parámetro Force de TABLEUPDATE() como verdadero (.T.) y la propiedad [CURSORSETPROP\(\)](#) UpdateType está establecida al valor predeterminado 2, los datos anteriores se actualizarán con los nuevos datos enviados, siempre y cuando el valor del campo clave del registro en la tabla remota no se haya modificado. Si el valor del campo clave de la tabla remota ha cambiado o si la propiedad UpdateType está establecida a 1, Visual FoxPro enviará una instrucción DELETE y luego una instrucción INSERT a la tabla remota.

## Mensajes de error de actualización

La tabla siguiente muestra los mensajes de error de Visual FoxPro y ODBC que se aplican específicamente a actualizaciones remotas. La columna Acción contiene la acción que debe llevar a cabo para resolver la condición de error.

Mensaje de error	Significado	Acción
1004	El cursor no se puede actualizar porque el campo clave del registro en la tabla remota ha cambiado.	Verificar que el campo clave del registro en la tabla remota no se haya modificado.

No especificó tablas de actualización. Use la propiedad de cursor <code>Tables</code> .	La propiedad de cursor <code>Tables</code> no contiene nombres de tablas remotas. Se necesita al menos una tabla para permitir actualizaciones en el servidor remoto.	Use la propiedad <code>Tables</code> para especificar al menos una tabla para el cursor.
No hay columnas clave especificadas para la tabla de actualización <i>nombre_tabla</i> . Use la propiedad de cursor <code>KeyFieldList</code> .	La clave principal para la tabla remota especificada en el mensaje de error no está incluida en la propiedad <code>KeyFieldList</code> para el cursor; se necesita una clave principal para cada tabla que se va a actualizar.	Use la propiedad <code>KeyFieldList</code> para especificar la clave principal para la tabla remota.
No se especificó tabla de actualización válida para la columna <i>nombre_columna</i> . Use las propiedades de cursor <code>UpdateNameList</code> y <code>Tables</code> .	La propiedad <code>UpdateName</code> para la columna <i>nombre_columna</i> tiene un cualificador de tabla no válido.	Establezca el cualificador de tabla con la propiedad <code>UpdateNameList</code> o agregue el cualificador de tabla al valor de la propiedad <code>Tables</code> , o ambas cosas.
La propiedad de cursor <code>KeyFieldList</code> no define una clave única.	Varios registros remotos tienen la misma clave.	Use la propiedad <code>KeyFieldList</code> para definir una clave única para la tabla remota.
De ODBC: objeto ODBC no válido.	ODBC no encuentra la tabla o columna remota porque no existe con ese nombre. Visual FoxPro valida los nombres de campos de Visual FoxPro; los nombres de tabla y columna remotas solamente se validan en el servidor remoto.	Compruebe el nombre del objeto.

Para obtener más información sobre control de errores, consulte [Controlar errores de paso a través de SQL](#) más adelante en este mismo capítulo.

## Elegir un modo de procesamiento eficaz con paso a través de SQL

Visual FoxPro proporciona dos modos de procesamiento para recuperar y actualizar datos remotos mediante paso a través de SQL: síncrono y asíncrono. Si utiliza las funciones de paso a través de SQL, puede elegir el método que prefiera. No necesita elegir ningún método para las vistas remotas, pues Visual FoxPro utiliza automáticamente la recopilación progresiva y administra el modo de procesamiento para las vistas remotas.

### Ventajas del modo síncrono

De forma predeterminada, las funciones SQL de Visual FoxPro se procesan de manera síncrona: Visual FoxPro no devuelve el control a una aplicación hasta que no termine la llamada a la función. El procesamiento síncrono es útil cuando se trabaja con Visual FoxPro de forma interactiva.

## Ventajas del modo asíncrono

El procesamiento asíncrono proporciona mayor flexibilidad que el síncrono. Por ejemplo, cuando su aplicación procesa una función de forma asíncrona, puede generar un indicador que muestre el progreso de la instrucción en ejecución, mostrar el movimiento del puntero del *mouse*, crear bucles y establecer cronómetros para permitir la interrupción del procesamiento que tarda demasiado tiempo.

## Uso asíncrono de paso a través de SQL

Su aplicación puede solicitar el procesamiento asíncrono para las cuatro funciones que envían peticiones a un origen de datos y que recuperan datos: [SQLEXPED\(\)](#), [SQLMORERESULTS\(\)](#), [SQLTABLES\(\)](#) y [SQLCOLUMNS\(\)](#). El procesamiento asíncrono se activa estableciendo la propiedad Asynchronous de la conexión mediante la función [SQLSETPROP\(\)](#); cuando se establece la comunicación asíncrona para la conexión, estas cuatro funciones operan de manera asíncrona.

## Para comprobar el valor de la propiedad Asynchronous

- Utilice la función [SQLGETPROP\(\)](#) para ver el valor de la propiedad Asynchronous. En el ejemplo siguiente, `nConnectionHandle` representa el número de controlador para la conexión activa:

```
? SQLGETPROP(nConnectionHandle, 'Asynchronous')
```

## Para activar el procesamiento asíncrono

- Utilice la función [SQLSETPROP\(\)](#) para especificar la propiedad Asynchronous:

```
? SQLSETPROP(nConnectionHandle, 'Asynchronous', .T.)
```

En el modo asíncrono, es necesario llamar a cada función varias veces hasta que devuelve un valor distinto de 0 (en ejecución). Mientras la función esté en ejecución, podrá cancelar el procesamiento de la función si presiona la tecla ESC en caso de que la propiedad SET ESCAPE esté definida como verdadera (.T.).

Hasta que la función haya terminado de procesarse, la aplicación solamente podrá utilizar un controlador de conexión con [SQLCANCEL\(\)](#) o con la función asíncrona ([SQLEXPED\(\)](#), [SQLMORERESULTS\(\)](#), [SQLTABLES\(\)](#) o [SQLCOLUMNS\(\)](#)) asociada originalmente al controlador. No puede llamar a ninguna de las otras tres funciones asíncronas ni a [SQLDISCONNECT\(\)](#) con el mismo controlador de conexión hasta que la función haya terminado.

## Procesar múltiples conjuntos de resultados

Su aplicación recupera múltiples conjuntos de resultados cuando usted utiliza la función [SQLEXPED\(\)](#) para ejecutar dos o más instrucciones SELECT de SQL, o para ejecutar un procedimiento almacenado que ejecuta múltiples instrucciones SELECT. Los resultados de cada instrucción SELECT de SQL se devuelven en un cursor de Visual FoxPro distinto.

El nombre que determina el SQL RESULT se utiliza para el primer cursor. Los demás cursores utilizan



El nombre predeterminado `SQLRESULT1` se utiliza para el primer cursor; los demás cursores reciben nombres únicos indexando el nombre predeterminado. Por ejemplo, los nombres predeterminados para los cursores devueltos por una instrucción `SQLEXEC()` que solicita tres conjuntos de resultados son `Sqlresult`, `Sqlresult1` y `Sqlresult2`.

En el modo por lotes, si una función devuelve múltiples conjuntos de resultados, sus respectivos nombres de cursor en Visual FoxPro tendrán sufijos únicos y podrán incluir un máximo de 255 caracteres. Por ejemplo, el ejemplo siguiente establece la propiedad `BatchMode` en el modo por lotes y después ejecuta una instrucción `SQLEXEC()` que contiene cuatro instrucciones `SELECT` de SQL que generan cuatro conjuntos de resultados:

```
? SQLSETPROP(nConnectionHandle,'BatchMode',.T.)
? SQLEXEC(nConnectionHandle,'select * from authors ;
      select * from titles ;
      select * from roysched ;
      select * from titleauthor','ITEM')
```

Cuando termina de procesarse la función anterior, Visual FoxPro devuelve los cuatro conjuntos de resultados como los cursores de Visual FoxPro `Item`, `Item1`, `Item2` e `Item3`.

Puede cambiar el nombre predeterminado mediante el parámetro *cNombreCursor* con las funciones [SQLEXEC\(\)](#) o [SQLMORERESULTS\(\)](#). Si el nombre especificado para un conjunto de resultados ya se ha utilizado, el nuevo conjunto de resultados sobrescribirá el cursor existente.

Cuando la aplicación recupera múltiples conjuntos de resultados, puede elegir entre el procesamiento síncrono o asíncrono y entre los modos por lotes o sin lotes.

### Usar el modo de procesamiento por lotes

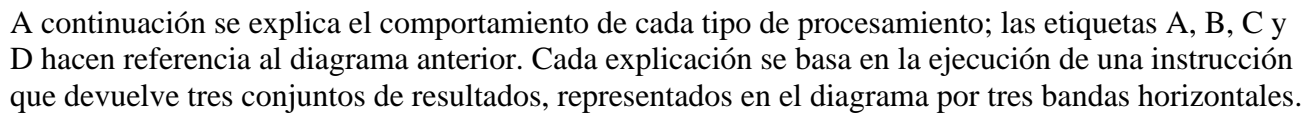
La propiedad `BatchMode`, establecida con [SQLSETPROP\(\)](#), controla la forma en que [SQLEXEC\(\)](#) devuelve múltiples conjuntos de resultados. Su valor predeterminado es `.T.`, para el modo por lotes. El procesamiento mediante el modo por lotes significa que Visual FoxPro no devuelve ningún resultado desde una llamada `SQLEXEC()` en ejecución hasta que no se hayan recuperado todos los conjuntos de resultados individuales.

### Usar el modo de procesamiento sin lotes

Si utiliza [SQLSETPROP\(\)](#) para establecer la propiedad `BatchMode` al valor `0`, para el modo sin lotes, cada conjunto de resultados se devolverá de manera individual. La llamada a la función [SQLEXEC\(\)](#) devuelve el primer conjunto de resultados. A continuación, la aplicación debe llamar a [SQLMORERESULTS\(\)](#) repetidamente hasta que devuelva un valor de `2`, lo que indica que no hay más resultados disponibles.

En el modo sin lotes, el nombre del cursor se puede modificar en cada llamada posterior a [SQLMORERESULTS\(\)](#). De esta forma, si para el ejemplo anterior el nombre del primer cursor de una secuencia `SQLEXEC()` es `Item`, y la segunda llamada a [SQLMORERESULTS\(\)](#) cambia el parámetro *cCursorName* a `Otheritem`, los cursores resultantes se denominarán `Item`, `Item1`, `Otheritem` y `Otheritem1`.

## Modos de procesamiento síncrono y asíncrono de Visual FoxPro



En el modo síncrono, el control no vuelve a la aplicación hasta que no se haya completado la ejecución de la función.

file:///C:/temp/~hh8A68.htm

Cuando se ejecuta una instrucción de paso a través de SQL de forma síncrona en el modo por lotes, el control no vuelve hasta que no se hayan recuperado todos los conjuntos de resultados. El nombre del primer cursor se especifica mediante el parámetro *cNombreCursor* en la función original. Si el cursor especificado ya existe, el conjunto de resultados sobrescribirá el cursor existente. Cuando se solicitan múltiples conjuntos de resultados en el modo síncrono por lotes, Visual FoxPro crea los nombres de cursores adicionales indexando de forma exclusiva el nombre del primer cursor.

### **B: Modo síncrono sin lotes**

Cuando se ejecuta una instrucción de paso a través de SQL de manera síncrona en el modo sin lotes, la primera instrucción recupera el primer conjunto de resultados y devuelve un 1. A continuación, debe llamar a la función [SQLMORERESULTS\(\)](#) repetidas veces y, si lo desea, especificar un nuevo nombre para el cursor. Si no especifica ningún nombre para el cursor, se crearán nombres distintos para los conjuntos de resultados indexando de forma única el nombre básico. Cuando [SQLMORERESULTS\(\)](#) devuelve un valor de 2, no hay más resultados disponibles.

### **Usar el procesamiento asíncrono**

En el modo asíncrono, la aplicación debe continuar llamando a la misma función de paso a través de SQL hasta que devuelva un valor distinto de 0 (en ejecución). El nombre predeterminado para el conjunto de resultados, *Sqlresult*, se puede cambiar explícitamente con el parámetro *cNombreCursor* la primera vez que llama a la función. Si el nombre especificado para un conjunto de resultados ya se ha utilizado, el nuevo conjunto de resultados sobrescribirá la información del cursor existente.

### **C: Modo asíncrono por lotes**

Cuando ejecuta el modo por lotes de forma asíncrona, cada llamada repetida de la función original devuelve un 0 (en ejecución) hasta que se devuelven todos los conjuntos de resultados a los cursores especificados. Una vez recuperados todos los resultados, el valor de retorno es el número de cursores o bien un número negativo que indica un error.

### **D: Modo asíncrono sin lotes**

Cuando se procesa en modo asíncrono sin lotes, [SQLEXEC\(\)](#) devuelve un valor de 1 al completar la recuperación de cada conjunto de resultados. A continuación, la aplicación debe llamar a [SQLMORERESULTS\(\)](#) repetidas veces hasta que devuelva un valor de 2, indicando que no hay más resultados disponibles.

**Sugerencia** Los conjuntos de resultados remotos se recuperan en dos etapas: primero se prepara el conjunto de resultados en el servidor; a continuación, el conjunto de resultados se recopila en un cursor local de Visual FoxPro. En el modo asíncrono, puede llamar a la función [USED\(\)](#) para ver si Visual FoxPro ha comenzado a recopilar el cursor solicitado.

## **Controlar la conversión de tipos de datos**

Al mover datos entre un servidor remoto y Visual FoxPro, pueden surgir diferencias en cuanto a la cantidad de tipos de datos disponibles en el servidor o en Visual FoxPro, porque raras veces hay una

correlación exacta entre los tipos de datos disponibles en un origen de datos remoto y los de Visual FoxPro. Para resolver estas diferencias, Visual FoxPro utiliza los tipos de datos de ODBC para asignar tipos de datos remotos a tipos de datos locales de Visual FoxPro. Si comprende la forma en que se asignan los tipos de datos entre ODBC y Visual FoxPro, puede prever la forma en que se manejarán los datos remotos del servidor en su aplicación de Visual FoxPro.

Si lo necesita, también puede ajustar los tipos de datos utilizados en el servidor o en la aplicación. El tipo de datos predeterminado para los campos en Visual FoxPro se puede anular creando una vista para el conjunto de datos remoto y luego estableciendo la propiedad del campo de presentación `DataType` en la base de datos. La propiedad `DataType` es una propiedad de tipo carácter que indica el tipo de datos deseado para cada campo de una vista remota. Para obtener más información sobre la propiedad `DataType`, vea [DBSETPROP\(\)](#).

### Transferir datos de vistas remotas

Cuando se recuperan datos desde un origen de datos ODBC remoto, Visual FoxPro convierte el tipo de datos de cada campo ODBC en un tipo de datos equivalente de Visual FoxPro para el cursor del conjunto de resultados. La siguiente tabla indica los tipos de datos disponibles en orígenes de datos de ODBC y sus equivalentes en Visual FoxPro.

Tipo de datos ODBC del campo remoto	Tipo de datos de campo en el cursor de Visual FoxPro
SQL_CHAR SQL_VARCHAR SQL_LONGVARCHAR	Character o Memo <sup>1</sup>
SQL_BINARY SQL_VARBINARY SQL_LONGVARBINARY	Memo
SQL_DECIMAL SQL_NUMERIC	Numeric o Currency <sup>2</sup>
SQL_BIT	Logical
SQL_TINYINT SQL_SMALLINT SQL_INTEGER	Integer
SQL_BIGINT	Character
SQL_REAL SQL_FLOAT SQL_DOUBLE	Double; el número de lugares decimales es el valor de SET DECIMAL en Visual FoxPro
SQL_DATE	Date
SQL_TIME	DateTime <sup>3</sup>
SQL_TIMESTAMP	DateTime <sup>4</sup>

<sup>1</sup> Si el ancho del campo ODBC es menor que el valor de la propiedad del cursor UseMemoSize, se convertirá en un campo de tipo Character en el cursor de Visual FoxPro; de lo contrario, se convertirá en un campo Memo.

<sup>2</sup> Si el campo del servidor es del tipo de datos money, se convertirá en un tipo de datos Currency en Visual FoxPro.

<sup>3</sup> El día toma el valor predeterminado de 1/1/1900.

<sup>4</sup> Si el valor del campo SQL\_TIMESTAMP contiene fracciones de segundos, estas fracciones se truncarán al convertir el valor a un tipo de datos DateTime de Visual FoxPro.

**Nota** Los valores nulos de los campos del origen de datos ODBC se convierten en valores nulos en el cursor de Visual FoxPro, sin importar el valor de SET NULL en Visual FoxPro en el momento en que la aplicación recupera los datos remotos.

### Convertir parámetros de Visual FoxPro a tipos de datos de vista remota

Si hay datos de Visual FoxPro en un cursor que se originó a partir de datos remotos, los datos volverán a su tipo ODBC original cuando se envíen al servidor remoto. Si envía datos originados en Visual FoxPro al servidor remoto mediante paso a través de SQL, se aplicarán las siguientes conversiones.

Tipo de datos de Visual FoxPro	Tipo de datos de ODBC
Character	SQL_CHAR o SQL_LONGVARCHAR <sup>1</sup>
Currency	SQL_DECIMAL
Date	SQL_DATE o SQL_TIMESTAMP <sup>2</sup>
DateTime	SQL_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
General	SQL_LONGVARBINARY
Logical	SQL_BIT
Memo	SQL_LONGVARCHAR
Numeric	SQL_DOUBLE

1. Si la variable de memoria de Visual FoxPro que se asigna a un parámetro crea una expresión cuyo ancho es menor que 255, se convertirá en un tipo SQL\_CHAR en el origen de datos ODBC; de lo contrario, se convertirá en un tipo SQL\_LONGVARCHAR.

2. Los datos de tipo Fecha de Visual FoxPro se convierten a SQL\_DATE para todos los orígenes de datos ODBC excepto para SQL Server, donde se convierten en SQL\_TIMESTAMP.

### Asignar un parámetro de Visual FoxPro a un tipo de datos remoto

Puede asignar un valor de parámetro de Visual FoxPro a un tipo de datos remoto concreto si aplica al parámetro el formato como una expresión de caracteres que utilice la sintaxis para el tipo de datos remoto deseado. Por ejemplo, si el servidor proporciona un tipo de datos DateTime, puede crear el parámetro de Visual FoxPro como una expresión de caracteres en el formato utilizado por el servidor para representar los datos de tipo DateTime. Cuando el servidor recibe el valor del parámetro, intenta asignar los datos con formato al tipo de datos DateTime.

**Nota** Cuando envíe un parámetro al servidor remoto, asegúrese de que el tipo de datos de la cláusula WHERE coincide con el tipo de datos utilizado para la expresión del parámetro.

## Controlar errores de paso a través de SQL

Si una función de paso a través de SQL devuelve un error, Visual FoxPro almacenará el mensaje de error en una matriz. La función [AERROR\(\)](#) proporciona información relativa a los errores detectados en cualquiera de los niveles de componentes: Visual FoxPro, el origen de datos ODBC o el servidor remoto. Examinando los valores devueltos por AERROR( ), puede determinar el error de servidor producido y el texto de su mensaje de error.

**Importante** Es necesario llamar a AERROR( ) inmediatamente para obtener información de error. Si genera un segundo error antes de llamar a AERROR( ), la información del primer error se perderá.

# Capítulo 22: Optimizar el rendimiento cliente-servidor

Cuando haya implementado su aplicación cliente-servidor, podría encontrar ciertas áreas en las que le gustaría mejorar el rendimiento. Puede ajustar su aplicación para obtener el máximo rendimiento si, por ejemplo, acelera los formularios y las consultas, y aumenta la velocidad de los datos.

En este capítulo se explican estrategias para optimizar el rendimiento de las aplicaciones en el cliente, la red y el servidor. Para obtener información acerca de la implementación de aplicaciones cliente-servidor, consulte los demás capítulos de este manual.

En este capítulo se trata lo siguiente:

- [Optimizar el uso de las conexiones](#)
- [Acelerar la recuperación de datos](#)
- [Acelerar las consultas y las vistas](#)
- [Acelerar los formularios](#)
- [Mejorar el rendimiento en actualizaciones y eliminaciones](#)

## Optimizar el uso de las conexiones

Establecer una conexión emplea tiempo y memoria tanto en el cliente como en el servidor. Cuando optimice las conexiones, equilibrará su necesidad de alto rendimiento con los requisitos de recursos de su aplicación.

-- -- -- -- --

El número de conexiones que utiliza Visual FoxPro depende de si usted fuerza o no el cierre de las conexiones no usadas y de qué duración establezca para el tiempo de espera de inactividad de la conexión.

## Usar conexiones compartidas

Puede utilizar conexiones de forma exclusiva o bien compartir una conexión. Cada método presenta sus ventajas. Cuando use una conexión de forma exclusiva, su aplicación no experimentará ninguna contención en cuanto a recursos de conexión una vez establecida una conexión. Si cada conjunto de resultados usa una conexión exclusiva, también podrá entremezclar el proceso asíncrono en varios conjuntos de resultados.

Cuando use una conexión compartida, tendrá una conexión para múltiples conjuntos de resultados. Debe serializar las operaciones de manipulación de datos sobre los conjuntos de resultados que comparten la misma conexión y diseñar la aplicación para probar el nivel de ocupación de la conexión cada vez que pueda producirse un conflicto. Para obtener información sobre la manera de compartir una conexión, consulte el capítulo 8, [Crear vistas](#).

## Controlar los tiempos de espera de conexión

Si su aplicación no realiza ninguna acción durante un periodo de tiempo prolongado, puede reducir el uso de la conexión si establece la propiedad IdleTimeout de la conexión. La propiedad IdleTimeout controla el intervalo de tiempo que se permite que las conexiones estén inactivas antes de que Visual FoxPro las cierre. De forma predeterminada, las conexiones esperan indefinidamente y no se desactivan hasta que no las cierre específicamente el usuario.

El tiempo de conectividad para una definición de conexión se establece con la propiedad IdleTimeout de la función [DBSETPROP\(\)](#); puede establecer la propiedad IdleTimeout para una conexión activa con la función [SQLSETPROP\(\)](#).

Visual FoxPro cerrará las conexiones aunque las ventanas Examinar y los formularios que muestran datos remotos sigan abiertos y, después, volverá a conectar automáticamente cuando se necesite de nuevo la conexión. Sin embargo, Visual FoxPro no puede cerrar una conexión si:

- Hay pendientes resultados de una consulta desde el servidor.
- La conexión está en modo de transacción manual. Debe realizar o deshacer la transacción y cambiar al modo de transacción automática para poder cerrar la conexión.

Para establecer el modo de transacción para una definición de conexión, utilice la propiedad Transactions de la función [DBSETPROP\(\)](#); puede establecer el modo de transacción para una conexión activa mediante la función [SQLSETPROP\(\)](#).

## Liberar conexiones

Puede mejorar el rendimiento si cierra las conexiones que su aplicación ya no utilice. Las conexiones se cerrarán automáticamente cuando cierre una vista. Si la conexión está compartida por múltiples vistas, Visual FoxPro cerrará la conexión cuando se cierre la última vista que usa la conexión.

Puede cerrar la conexión para una consulta si no desea actualizar los datos de un cursor. Use una consulta de paso a través de SQL para seleccionar los datos que necesita en un cursor local y, a continuación, cierre la conexión.

## Acelerar la recuperación de datos

Puede acelerar la recuperación de datos si administra el número de filas buscadas durante la [búsqueda progresiva](#), controla el tamaño de búsqueda y usa la búsqueda diferida de Memo.

También puede usar la propiedad de vista UseMemoSize para devolver campos de caracteres como campos memo y después desactivar FetchMemo para permitir a su aplicación buscar los campos de caracteres convertidos a campos memo.

### Usar la búsqueda progresiva

Cuando consulta un origen de datos remoto, Visual FoxPro recupera filas completas de datos y genera un cursor de Visual FoxPro. Para acelerar la recuperación de datos remotos, Visual FoxPro emplea la [búsqueda progresiva](#) de los cursores de vista y de los creados asíncronamente mediante paso a través de SQL. En lugar de pedirle a usted o a su aplicación que espere mientras se recupera un conjunto entero de datos, Visual FoxPro ejecuta una consulta y busca únicamente un pequeño subconjunto de las filas del conjunto de resultados en el cursor local. El tamaño de este subconjunto es de 100 filas de forma predeterminada.

**Nota** Las instrucciones síncronas de paso a través de SQL no emplean la búsqueda progresiva. Se recupera todo el conjunto de resultados solicitado por una instrucción [SQLEXEC\(\)](#) antes de que el control vuelva a su aplicación.

A medida que Visual FoxPro recupera filas adicionales de datos, el cursor local contiene cada vez más datos consultados. Puesto que las filas se recuperan en distintos momentos desde el origen de datos, la información de las filas no se actualiza automáticamente. Si su conexión funciona en modo asíncrono, Visual FoxPro le devolverá el control a usted o a su programa en cuanto busque el primer subconjunto de datos. Durante el tiempo de inactividad, Visual FoxPro realiza una búsqueda en segundo plano de las filas restantes de los datos consultados, de subconjunto en subconjunto, en el cursor local. Esta situación le permite usar los datos ya buscados del cursor sin tener que esperar al resto de los datos.

**Nota** El aumento del número de filas buscadas aumenta el rendimiento, pero disminuye la respuesta de la interfaz del usuario. La disminución del número de filas buscadas produce el efecto contrario.

### Buscar datos bajo petición

Puede desactivar la búsqueda progresiva y buscar filas únicamente cuando sea necesario si usa de forma apropiada la propiedad de cursor de vista o de base de datos FetchAsNeeded. Esto produce una obtención de datos más eficiente para vistas remotas o vistas al recuperar conjuntos de datos muy grandes.

La propiedad FetchAsNeeded se establece de forma predeterminada a falso (.F.), por lo que la



búsqueda progresiva se utiliza de forma predeterminada. Cuando se establece la propiedad `FetchAsNeeded` como verdadero (.T.), sólo se busca en las filas cuando sea necesario. Cuando se establece la propiedad `FetchAsNeeded` a verdadero, no se puede realizar una actualización hasta que se complete la búsqueda, se llame a la función [SQLCANCEL\(\)](#) en el controlador de conexión actual o se cierre la vista.

Si desea ver el impacto del uso de la propiedad `FetchAsNeeded`, establezca la propiedad `FetchAsNeeded` de una vista que recupera un conjunto de resultados grande a .T. y, a continuación, abra una ventana Examinar en la vista y desplácese. La barra de estado se actualiza para mostrar el número de filas recuperadas cuando se desplaza por la ventana Examinar.

### Controlar la búsqueda del cursor

Si desea buscar todo el cursor, puede ejecutar el comando [GOTO BOTTOM](#) o cualquier comando que requiera acceso a todo el conjunto de datos.

**Sugerencia** Si bien puede usar el comando `GO BOTTOM` para buscar todo el cursor, normalmente es más eficiente generar una vista parametrizada que busque únicamente una sola fila cada vez y vuelva a ejecutar la consulta a medida que el usuario cambie registros. Para obtener más información acerca de la generación de vistas de alto rendimiento, consulte el capítulo 8, [Crear vistas](#).

Los programas no ofrecen procesamiento de bucle inactivo. Para buscar en cursores de vista mediante programación, use los comandos `GO nNúmeroRegistro` o `GOTO BOTTOM`. Para buscar cursores creados mediante paso a través de SQL en modo asíncrono, llame a la función asíncrona de paso a través de SQL una vez por cada subconjunto de fila.

### Cancelar una instrucción SQLEXEC( )

Puede usar la función [SQLCANCEL\(\)](#) para cancelar una instrucción [SQLEXEC\(\)](#) o una vista en cualquier momento. Sin embargo, si el servidor ha terminado de generar el conjunto de resultados remoto y Visual FoxPro ha empezado a buscar el conjunto de resultados remoto en un cursor local, la función `SQLCANCEL()` cancelará la instrucción `SQLEXEC()` y abandonará el cursor local. Si desea eliminar el cursor local puede ejecutar el comando [USE](#), que cierra el cursor y cancela la búsqueda.

El comando `USE` no cancelará una instrucción `SQLEXEC()` si dicha instrucción todavía no ha creado un cursor local. Para determinar si Visual FoxPro ha creado un cursor local, puede llamar a la función [USED\(\)](#).

### Controlar el tamaño de búsqueda

Puede controlar el número de filas buscadas de una vez por su aplicación en un servidor remoto si establece la propiedad `FetchSize` de su vista. Esta propiedad especifica el número de registros que se buscan en el cursor local desde el servidor remoto cada vez, mediante la búsqueda progresiva o llamadas asíncronas de paso a través de SQL. El valor predeterminado es 100 filas.

### Para controlar el número de registros buscados de una vez en una vista

- En el [Diseñador de vistas](#), elija **Opciones avanzadas** en el menú **Consulta**. En el área

**Búsqueda de datos** del cuadro de diálogo [Opciones avanzadas](#), use el control numérico para establecer un valor para el **Número de registros que se van a buscar cada vez**.

–O bien–

- Establezca la propiedad FetchSize mediante la función [DBSETPROP\(\)](#) para establecer el tamaño de búsqueda en la definición de vista.

–O bien–

- Establezca la propiedad FetchSize mediante la función [CURSORSETPROP\(\)](#) para establecer el tamaño de búsqueda del cursor de la vista activa.

Por ejemplo, el código siguiente establece la definición de la vista para buscar progresivamente 50 filas de una vez en Customer\_remote\_view:

```
? DBSETPROP('Customer_remote_view', 'View', 'FetchSize', 50)
```

## Usar la búsqueda diferida de Memo

Una aplicación bien diseñada suele utilizar la búsqueda diferida de Memo para acelerar la transferencia de conjuntos de resultados que contienen campos de tipo Memo o General. La búsqueda diferida de Memo significa que el contenido de los campos de tipo Memo y General no se transferirá automáticamente cuando transfiere una fila. En su lugar, se transfieren rápidamente el resto de los campos de la fila, y no se busca el contenido de los campos de tipo Memo y General hasta que no lo pide abriendo el campo Memo o General. La búsqueda diferida de Memo constituye el método más rápido de transferencia de filas y permite buscar únicamente el contenido de los campos de tipo Memo o General, que puede ser muy grande, si lo necesita el usuario.

Por ejemplo, su formulario podría incluir un campo de tipo General que muestre una imagen. Para acelerar el rendimiento, puede usar la búsqueda diferida de Memo para impedir la transferencia de la imagen hasta que el usuario elija el botón “Vista previa” de su formulario. El código subyacente al botón “Vista previa” busca en el campo General y lo muestra en el formulario.

Para controlar la búsqueda diferida de Memo, utilice la propiedad FetchMemo de su vista o su cursor. La propiedad FetchMemo especifica si se debe buscar el contenido de los campos tipo Memo o General cuando se transfiere la fila. El valor predeterminado es verdadero (.T.), que significa que los campos de tipo Memo y General se transfieren automáticamente. Si sus datos contienen grandes cantidades de datos de campos tipo Memo o General, quizá observe un mejor rendimiento si establece como falso (.F.) la propiedad FetchMemo.

**Nota** La vista debe ser actualizable para permitir la búsqueda diferida de Memo, ya que Visual FoxPro usa los valores de campos clave establecidos por las propiedades de actualización para encontrar la fila de origen en el servidor cuando recupere el campo de tipo Memo o General. Para obtener información acerca de la creación de una vista actualizable, consulte el capítulo 8, [Crear vistas](#).

Use [DBSETPROP\(\)](#) para establecer la propiedad FetchMemo de una vista y [CURSORSETPROP\(\)](#)

para establecer la propiedad FetchMemo de un cursor.

## Optimizar el rendimiento de búsqueda de datos

Puede usar las siguientes recomendaciones para establecer propiedades de conexión y vista con el fin de optimizar la búsqueda de datos. La propiedad PacketSize de su conexión es la que tiene más influencia en el rendimiento. Además, puede optimizar el rendimiento de búsqueda mediante conexiones síncronas.

Objeto	Propiedad	Valor
Conexión	PacketSize	4K a 12K <sup>1</sup>
Conexión	Asynchronous <sup>2</sup>	.F.
Vista	FetchSize <sup>3</sup>	máximo

1. Establezca un valor más alto para filas que contengan más datos; debe experimentar para encontrar el mejor valor.
2. Use conexiones síncronas para mejorar el rendimiento hasta un 50%, a no ser que quiera poder cancelar instrucciones SQL mientras se ejecutan en el servidor.
3. El efecto de FetchSize es muy dependiente del tamaño de registros del conjunto de resultados de búsqueda. En modo síncrono, no afecta al rendimiento de forma significativa, por que puede establecerlo como sea necesario para la búsqueda progresiva de vistas de procesamiento asíncrono de paso a través de SQL. Si se reduce el valor de FetchSize, proporciona mejor respuesta al buscar de forma progresiva una vista, pero disminuye la velocidad de búsqueda. Si aumenta, mejora el rendimiento de búsqueda de vista.

El rendimiento depende en gran manera de la configuración del sistema y de los requisitos de la aplicación. Estas recomendaciones se basan en un equipo cliente que ejecuta Windows NT versión 3.5, con ODBC versión 2.10 y controlador ODBC de SQL Server versión 2.05; y un equipo servidor que ejecuta Windows NT versión 3.5 con SQL Server versión 4.21 y versión 6.0.

## Acelerar consultas y vistas

Puede mejorar el rendimiento de las consultas y las vistas si agrega índices, optimiza el proceso local y remoto, y optimiza expresiones con parámetros.

### Agregar índices a tablas remotas

Los índices remotos pueden hacer las consultas mucho más eficaces. Las consultas de múltiples tablas son mucho más rápidas si las tablas están indexadas según los campos de combinación. Tener índices en campos incluidos en la cláusula WHERE de una consulta también puede mejorar el rendimiento.

Los índices agrupados consiguen el máximo rendimiento. En SQL Server, cada tabla puede tener un índice agrupado. El [Asistente para upsizing a SQL Server](#) crea automáticamente índices agrupados en tablas que tenían una clave principal en Visual FoxPro.

**Sugerencia** Si bien los índices de campos de tablas empleados en consultas pueden acelerar el proceso, los índices sobre conjuntos de resultados pueden disminuir el rendimiento. Use con cuidado los índices sobre conjuntos de resultados.

## Optimizar el procesamiento local y remoto

Si necesita procesar una combinación de datos locales y remotos, cree una vista remota que combine todos los datos remotos en una única vista. Entonces, puede combinar la vista remota con los datos locales de una vista local. Como Visual FoxPro hace una búsqueda completa de ambas vistas antes combinarlas y filtrar la vista combinada, es importante limitar el tamaño del conjunto de resultados de la vista.

Para optimizar el procesamiento remoto, limite el conjunto de resultados remoto a la cantidad mínima de datos que necesite su aplicación. Cuando recupere menos datos en un conjunto de resultados remoto, minimizará el tiempo necesario para transferir datos remotos al cursor de la vista o la consulta local.

## Optimizar vistas parametrizadas

Puede acelerar la recuperación de datos durante operaciones [REQUERY\(\)](#) en una vista parametrizada abierta si compila la vista antes de que se ejecute. Para precompilar o “preparar” una vista, establezca la propiedad Prepared de la vista a verdadero (.T.).

## Optimizar expresiones de parámetros

Los parámetros de vistas y de paso a través de SQL son expresiones de Visual FoxPro y se evalúan en Visual FoxPro antes de enviarse al servidor remoto. El tiempo de evaluación para la expresión es importante, ya que aumenta el tiempo de ejecución de la consulta.

## Acelerar formularios

Cuando diseñe un formulario basado principalmente en datos del servidor, adopte un enfoque minimalista para un mejor rendimiento. Determine los datos y la funcionalidad necesarios y no pida al servidor estos datos y esta funcionalidad hasta que no lo solicite el usuario. Solicitar datos al servidor usa tiempo de procesamiento y crea tráfico en la red. Para solicitar menos datos en sus formularios:

- Solicite el mínimo número de registros posibles. Por ejemplo, use un filtro o una consulta para limitar el tamaño del conjunto de registros. Asegúrese de que el servidor pueda procesar todas las restricciones que usted use.
- Use el mínimo número de campos remotos posible en las vistas subyacentes a sus formularios.
- Use el menor número posible de formularios que tengan acceso a vistas remotas en su conjunto de formularios. Cuando abra un conjunto de formularios, todos los formularios del conjunto se abrirán y completarán con los datos aplicables. Si limita el número de formularios del conjunto, especialmente aquellos que deben conectarse a un servidor y recuperar datos remotos, reducirá el tiempo de carga del conjunto de formularios.
- Use menos controles dependientes que tengan acceso a datos remotos. Cada cuadro combinado, cuadro de lista y cuadrícula que dependa de una tabla o una consulta remota necesita una

cuadro de lista y cuadrícula que dependa de una tabla o una consulta remota necesita una consulta distinta al servidor cuando se abra el formulario. Evite usar controles que contengan totales o cuadros de lista y cuadros combinados que tengan orígenes de filas grandes.

- Si los usuarios necesitan comparar múltiples conjuntos de datos, considere la posibilidad de almacenar en tablas locales temporales los datos devueltos por el servidor. Proporcione un formulario en el cual el usuario pueda usar los datos almacenados previamente o ejecutar una nueva consulta.

## Almacenamiento local de tablas de consulta

Con frecuencia, una aplicación contiene varios formularios que usan la misma tabla remota. Si los datos de la tabla no cambian con frecuencia, puede acelerar la carga del formulario y reducir la carga del servidor mediante una de las técnicas siguientes:

- Almacene en la aplicación de base de datos local Visual FoxPro las tablas que nunca cambian y que no son demasiado grandes (como los nombres y las abreviaturas de las regiones de su país). Si la tabla se combina en consultas o vistas con tablas remotas, también debería conservar una copia de ella en el servidor para evitar combinar datos locales y remotos.
- Almacene en el servidor y en la aplicación de base de datos local las tablas que cambien con poca frecuencia (como las listas de edificios de la compañía). Ofrezca un método para que el usuario transfiera la tabla cuando cambien los datos.
- Almacene en el servidor y en la aplicación de base de datos local las tablas que cambien ocasionalmente pero no a diario (como una lista de empleados en una pequeña compañía o departamento). Su aplicación debe actualizar automáticamente la versión local cada vez que se inicie. Este método emplea tiempo adicional cuando se inicia la aplicación, pero acelera las consultas cuando la aplicación está en ejecución.

## Mostrar campos únicamente bajo petición

Presente los campos que tardan mucho tiempo en recuperarse del servidor únicamente cuando se soliciten, como los campos de tipo Memo o General. Puede usar las técnicas siguientes:

- Si su formulario se basa en una vista, coloque los campos de tipo Memo o General fuera de la pantalla de otra página del formulario. Agregue una etiqueta al formulario, como “Avanzar página para ver notas e imágenes”, que informen al usuario de cómo ver la información. Establezca la propiedad FetchMemo de la vista o del cursor como falsa (.F.), de forma que Visual FoxPro no recupere campos de tipo Memo o General hasta que se muestren en la pantalla.
- Establezca la propiedad [Visible](#) como falsa (.F.) para los controles dependientes de los campos de tipo Memo o General. Agregue un botón de alternar o un botón de comando que establezca la propiedad como verdadera (.T.), de forma que el usuario pueda elegir ver el contenido de estos controles.
- Muestre los campos más importantes en el formulario principal y ofrezca un botón con la etiqueta “Más información” que abra otro formulario que contenga otros campos. Base el segundo formulario en una vista parametrizada por el campo de clave principal del formulario principal. Por ejemplo, suponga que tiene un formulario principal basado en una vista cuya instrucción SQL SELECT incluye el código siguiente:

```
SELECT customer_id, company_name, address, city, region, country
FROM customers
```

```
FROM customers
```

En el formulario anterior, `cust_id` depende `thisform.txtCust_id`. Podría basar el segundo formulario en la vista siguiente, que sólo se utiliza cuando el usuario elige el botón “Más información”:

```
SELECT orders.order_id, orders.order_date, orders.shipper_id, ;
employee.emp_id, employee.last_name, employee.first_name ;
FROM orders, employee ;
WHERE orders.cust_id = ?THISFORM.txtCust_id ;
AND orders.employee_id = employees.emp_id
```

## Mejorar el rendimiento en actualizaciones y eliminaciones

Puede acelerar las instrucciones Update y Delete:

- Si agrega marcas de hora a sus tablas remotas.
- Si usa la propiedad CompareMemo.
- Si usa el modo de transacción manual.
- Si usa procedimientos almacenados en el servidor.
- Si crea lotes de actualizaciones.

### Agregar marcas de hora

Puede mejorar el rendimiento cuando actualice, inserte o elimine datos de una tabla remota que contenga muchos campos si agrega un campo de marca de hora a la tabla remota, siempre y cuando su servidor ofrezca el tipo de campo MarcaHora.

La presencia de un campo de tipo MarcaHora en una tabla remota le permite usar la opción `DB_KEYANDTIMESTAMP` de actualización WhereType de SQL de Visual FoxPro. Esta opción ahorra tiempo de proceso porque Visual FoxPro sólo compara dos campos de la vista, el campo de clave y el campo de marca-hora, contra una tabla remota para detectar conflictos de actualización. Al comparar únicamente dos campos, en lugar de todos los campos actualizables (mediante la opción `DB_KEYANDUPDATABLE`) o todos los campos modificados (mediante la opción `DB_KEYANDMODIFIED`), la opción `DB_KEYANDTIMESTAMP` reduce el tiempo necesario para actualizar datos remotos. Para obtener más información acerca de las opciones de WhereType, consulte el capítulo 8, [Crear vistas](#).

**Nota** La opción `DB_KEYANDTIMESTAMP` compara los campos clave y de marca-hora únicamente cuando la tabla remota contiene un campo de marca-hora. Si usa la opción `DB_KEYANDTIMESTAMP` frente a una tabla remota que no contiene un campo de marca-hora, Visual FoxPro sólo compara los campos clave.

El Asistente para upsizing puede agregar automáticamente a las tablas que exporte campos de marca-hora según proceda. Para obtener más información al respecto, consulte "Columnas de marca de hora" en el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#).

**Sugerencia** Si hace algo que modifique la estructura de una tabla base de una vista, como agregar un campo de marca de hora, quizá tenga que volver a crear la vista. Los campos de una definición de vista se almacenan en la base de datos, y todos los cambios realizados a las tablas base para una vista



después de usarla no quedarán reflejados en la definición de la vista hasta que no vuelva a crearla.

## **Excluir campos memo de la cláusula WHERE de actualización**

Cuando sea apropiado, puede acelerar las actualizaciones evitando que los campos memo (campos de tipo Memo, General o Picture) se comparen con sus homólogos de tabla de base. De forma predeterminada, la propiedad CompareMemo se establece como verdadero (.T.), lo cual incluye automáticamente los campos memo en la cláusula WHERE de SQL generada al crear una vista actualizable. Puede establecer la propiedad CompareMemo como falso (.F.) para excluir memos de la cláusula WHERE de SQL.

## **Usar transacciones**

Para conseguir un rendimiento óptimo, use el modo de transacción manual y administre las transacciones. El modo de transacción manual le permite controlar cuándo realizar un grupo de transacciones, lo que permite al servidor procesar más instrucciones rápidamente.

El modo de transacción automática lleva más tiempo, ya que de forma predeterminada cada instrucción de actualización se incluye en una transacción distinta. Este método ofrece el máximo control sobre cada instrucción individual de actualización, pero también aumenta la sobrecarga.

Puede mejorar el rendimiento en el modo de transacción automática si incrementa el valor de la propiedad BatchUpdateCount de la vista o del cursor. Cuando usa un valor grande para BatchUpdateCount, muchas instrucciones de actualización se incluyen por lotes en una única instrucción de actualización, que a su vez se incluye en una única transacción. Sin embargo, si falla alguna instrucción del lote, se deshará todo el lote.

**Sugerencia** Algunos servidores no admiten la propiedad BatchUpdateCount; debe probar esta propiedad en cada servidor remoto antes de desplegarla en su aplicación.

## **Usar procedimientos almacenados en el servidor**

Puede crear procedimientos almacenados en el servidor, que están precompilados y, por tanto, se ejecutan muy rápidamente. Puede ejecutar procedimientos almacenados, enviar parámetros mediante paso a través de SQL y mover procesos adicionales al servidor según sea apropiado para su aplicación.

Por ejemplo, quizá desee recopilar localmente la entrada del usuario y ejecutar una consulta de paso a través de SQL para enviar los datos al servidor, llamando al procedimiento almacenado apropiado. Para ello, cree un formulario en un cursor o una matriz local para recopilar los datos y escriba código que cree una instrucción [SQLEXEC\(\)](#) mediante el nombre del procedimiento almacenado en el servidor y los parámetros que se deben proporcionar. Podría agregar este código al evento Click de un botón de comando cuyo título fuera “Aceptar” o “Realizar”. Cuando el usuario elija el botón se ejecutará la instrucción [SQLEXEC\(\)](#). Usar procedimientos almacenados en el servidor para actualizar datos remotos puede ser más eficiente, ya que los procedimientos almacenados se compilan en el servidor.

## **Crear lotes de actualizaciones**

Si su aplicación actualiza una serie de registros, quizá desee crear lotes de actualizaciones de modo que la red y el servidor las gestionen más eficazmente. Las instrucciones Update o Insert se agrupan por lotes antes de enviarse al servidor, de acuerdo con el valor de la propiedad BatchUpdateCount de la vista. El valor predeterminado es 1, que significa que cada registro se envía al servidor con una instrucción de actualización. Puede reducir el tráfico de la red si incrementa el valor para empaquetar múltiples actualizaciones en una instrucción.

**Sugerencia** Algunos servidores no admiten la propiedad BatchUpdateCount; debe probar esta propiedad en cada servidor remoto antes de desplegarla en su aplicación.

Para usar esta característica eficazmente, la conexión de la vista debe establecerse para el modo 5 de almacenamiento en búfer, para el almacenamiento optimista de tablas en búfer y los cambios deben confinarse desde el punto de vista ideal a los mismos campos en cada fila del cursor. Puede usar [DBSETPROP\(\)](#) para establecer la propiedad BatchUpdateCount para la definición de la vista; para cambiar el valor para el cursor de una vista activa, use [CURSORSETPROP\(\)](#).

## Optimizar el rendimiento de actualizaciones y eliminaciones

Puede usar las siguientes instrucciones para establecer propiedades de vista y conexión para optimizar el rendimiento de actualizaciones y eliminaciones. La propiedad BatchSize de la vista es la que tiene más influencia en el rendimiento.

Objeto	Propiedad	Valor	Notas
Vista	BatchUpdateCount	10 a 30 filas	Establezca un valor superior para actualizaciones de pequeño tamaño. <sup>1</sup> Se establece para aumentar el rendimiento hasta un 50%. El valor predeterminado es 1.
Conexión	Asynchronous	(.F.)	Use conexiones síncronas para aumentar el rendimiento hasta un 50%, a no ser que pueda cancelar instrucciones SQL mientras se ejecutan en el servidor. De forma predeterminada la conexión es síncrona.
Conexión	WaitTime	N/D	Para aumentar el rendimiento en modo asíncrono, use un tiempo de espera más corto; para reducir el tráfico de red, aumente el tiempo de espera.
Conexión	PacketSize	4K a 12K	Tiene poca influencia en el rendimiento.

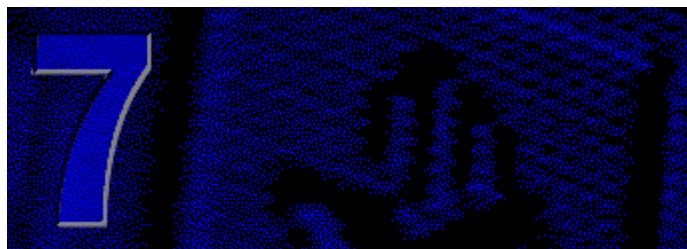
<sup>1</sup> El mayor valor también depende de la velocidad del servidor.



1 El mejor valor también depende de la velocidad del servidor.

El rendimiento real depende en gran medida de la configuración del sistema y de los requisitos de la aplicación. Pruebe con los valores de la tabla para determinar los mejores valores para su configuración. Las recomendaciones anteriores son óptimas para un equipo cliente que ejecuta Windows NT versión 3.5 con ODBC 2.10 y el controlador ODBC de SQL 2.05; y un equipo servidor que ejecute Windows NT, Versión 3.5 con Microsoft SQL Server 4.21 y 6.0.

# Manual del programador, Parte 7: Crear archivos de Ayuda



Los archivos de Ayuda son un origen de información valiosa para los usuarios de la aplicación. Con Visual FoxPro, puede elegir crear Winhelp, Ayuda HTML o Ayuda de tipo .DBF.

## Capítulo 23 [Crear Ayuda gráfica](#)

La Ayuda gráfica puede ser Winhelp si desea una apariencia Windows, o una Ayuda HTML si desea una apariencia de Web.

## Capítulo 24 [Crear Ayuda de tipo .DBF](#)

La Ayuda de tipo .DBF está basada en caracteres y le proporciona flexibilidad para migrar el archivo de Ayuda a otras plataformas. Este tipo de Ayuda es fácil de crear porque está basada en una tabla de Visual FoxPro.

## Capítulo 23: Crear Ayuda gráfica

Puede agregar un toque profesional a su aplicación al incluir un archivo de Ayuda gráfico en forma de Ayuda HTML o Ayuda WinHelp. La Ayuda gráfica puede incluir gráficos y texto con formato; la Ayuda de tipo dbf está limitada a una sola fuente sin gráficos. Para obtener información acerca de la creación de Ayuda de tipo .DBF, vea el capítulo 24, [Crear Ayuda de tipo .DBF](#).

**Nota** Microsoft Visual Studio 6.0 incluye el Microsoft HTML Workshop (Hhw.exe) para crear archivos de Ayuda HTML. No incluye Microsoft Help Workshop 4.0 (Hcw.exe) para crear archivos Winhelp. Las versiones anteriores de Microsoft Visual FoxPro incluían Microsoft Help Workshop 4.0.

Los temas descritos en este capítulo son:

- [Ayuda HTML](#)
- [WinHelp 4.0](#)

### Ayuda HTML

La Ayuda HTML proporciona muchas de las características de Winhelp y agrega las siguientes:

- Soporte para HTML.
- Soporte para ActiveX, Java y secuencias de comandos (Javascript y Microsoft Visual Basic Script).
- Soporte para los formatos de imagen de HTML (.jpg, .gif, .png).
- Capacidad para ir desde un tema de Ayuda a un sitio de Internet.
- Capacidad para ver el código HTML de un tema de Ayuda.

La Ayuda HTML se crea con Microsoft HTML Help Workshop, que se incluye en Visual Studio y en Visual FoxPro. HTML Help Workshop proporciona un completo sistema de creación de Ayuda HTML e incluye una compatibilidad con versiones anteriores que le permite crear fácilmente archivos de Ayuda HTML a partir de proyectos Winhelp existentes. Para crear archivos de Ayuda HTML en su aplicación, consulte la Ayuda en pantalla de HTML Help Workshop.

En ...\\Samples\\Vfp98\\Solution\\Help se incluye un proyecto de Ayuda HTML de ejemplo, parte del ejemplo Solutions. Incluye los siguientes archivos:

<b>Archivo</b>	<b>Descripción</b>
Solution.chm	Archivo de Ayuda compilado.
Solution.hhp	Archivo de proyecto (un archivo de texto que reúne todos los elementos de un proyecto de Ayuda y contiene información acerca de cómo va a aparecer un archivo de Ayuda compilado).
Solution.hhk	Archivo de índice; contiene las entradas del índice (palabras clave).
Solution.hhc	Archivo de Tabla de contenido.
Solution.ali	Archivo de alias para el soporte de Ayuda interactiva. Asocia Id. de producto con temas de Ayuda.
Solution.hh	Archivo de encabezados para el soporte de Ayuda interactiva. Incluye los Id. de producto.
Solution.chi	Archivo de índice utilizado cuando se suministran archivos .chm que van a permanecer en un CD-ROM, como en el caso de la biblioteca MSDN. El archivo .chi permite la instalación local de cierta información de exploración en el disco duro que permite tener acceso de forma rápida, mientras que el contenido principal reside en el CD-ROM. Los archivos .chi no se usan en entornos sin CD-ROM. Cuando no se utilizan archivos .chi, toda la información que podría almacenar permanece en el propio archivo .chm.
MSDN_ie3.css	Hoja con el estilo de cascada.
MSDN_ie4.css	Hoja con el estilo de cascada.
Archivo.htm	Archivos de contenido.

---

*Archivo.gif*Archivos gráficos.

---

## Diseñar el acceso a la Ayuda HTML

Además de crear un archivo de Ayuda HTML que contenga información útil, tiene que proporcionar un medio para que los usuarios de su aplicación tengan acceso a dicha Ayuda. Hay tres maneras de proporcionar Ayuda:

- Un menú Ayuda (un menú que aparece en la barra del menú principal de su aplicación).
- Ayuda interactiva (Ayuda que aparece cuando el usuario presiona F1, u otra tecla especificada, mientras está seleccionado un objeto, control o opción de menú determinado).
- Ayuda "¿Qué es esto?" (Ayuda que aparece cuando el usuario pide Ayuda sobre un objeto o control determinado).

La implementación de la Ayuda HTML es similar a la de WinHelp. Las siguientes secciones describen cómo puede implementar la Ayuda HTML en sus aplicaciones.

### Diseñar el menú Ayuda

El menú Ayuda suele contener comandos que proporcionan acceso a los temas de su sistema de Ayuda. Se recomienda que sólo un comando del menú Ayuda abra su sistema de Ayuda HTML. Además de dicho comando, puede incluir otros comandos en el menú Ayuda para proporcionar información del sistema o la información de derechos de autor y la versión de su aplicación.

## Interactividad

La Ayuda interactiva permite que los usuarios tengan acceso a los temas de Ayuda relacionados con lo que estén haciendo o viendo en su aplicación en cualquier momento dado. Por ejemplo, si un usuario está viendo un formulario de introducción de datos, la Ayuda interactiva podría proporcionar un tema relacionado específicamente con dicho formulario.

Usted decide el nivel de detalle de la implementación de la Ayuda interactiva en su aplicación. Por ejemplo, puede asociar un tema de Ayuda interactiva con un formulario o puede asociar temas de Ayuda más detallados para cada uno de los controles y campos del formulario.

Se suele llegar a la Ayuda interactiva al presionar F1, pero puede especificar que se active mediante cualquier tecla con [ON KEY LABEL](#).

### Usar la Ayuda interactiva en un formulario

Para implementar Ayuda interactiva, tiene que especificar el archivo de Ayuda de su aplicación y después asociar temas de Ayuda determinados con objetos de su aplicación.

### Para agregar Ayuda interactiva

1. Especifique el archivo de Ayuda de su aplicación.
2. Asigne un tema de Ayuda a cada objeto para el que vaya a proporcionar Ayuda interactiva.

## Especificar el archivo de Ayuda

El archivo de Ayuda al que se tiene acceso en su aplicación se determina al incluir el comando [SET HELP TO archivo](#) en el código, donde *archivo* es el nombre del archivo de Ayuda. Por ejemplo, si el archivo de Ayuda es MiAyuda.chm, puede utilizar el siguiente comando:

```
SET HELP TO MIAYUDA.CHM
```

Este comando suele estar incluido en el código de configuración del programa principal de su aplicación.

## Asignar temas de Ayuda

Puede asignar un tema de Ayuda a objetos específicos de su aplicación de Visual FoxPro.

### Para asignar un tema de Ayuda a un objeto

1. En modo Diseño, abra el objeto (por ejemplo, un formulario, un control o una barra de herramientas) al que vaya a asignar Ayuda interactiva.
2. Vea las propiedades del objeto.
3. Establezca la propiedad [HelpContextID](#) al número correspondiente al tema asociado de su archivo de Ayuda HTML.

Para obtener más información acerca de la asociación de temas de Ayuda HTML con Id. de contexto, consulte la Ayuda de HTML Help Workshop.

**Nota** Para asignar temas de Ayuda a títulos de menú o a comandos de menú, tiene que incluir el comando [SET TOPIC TO](#) en el procedimiento asociado con el título de menú o el comando de menú.

## Implementar la Ayuda "¿Qué es esto?"

La Ayuda "¿Qué es esto?" es similar a la Ayuda interactiva puesto que proporciona Ayuda relacionada con el objeto o control específico que actualmente tiene el enfoque.

En WinHelp, en lugar de invocar el archivo de Ayuda y mostrar el tema de Ayuda en la ventana de Ayuda con su tamaño predeterminado, la Ayuda "¿Qué es esto?" presenta el tema en una pequeña ventana emergente que desaparece en cuanto el usuario hace clic en cualquier parte de la pantalla. La Ayuda "¿Qué es esto?" es útil para proporcionar breves descripciones, definiciones o sugerencias sobre controles específicos.

Al contrario que en WinHelp, la Ayuda HTML "¿Qué es esto?" se presenta en la ventana de Ayuda con su tamaño predeterminado.

La Ayuda "¿Qué es esto?" se asocia con un formulario, control de formulario o barra de herramientas particular estableciendo su propiedad [WhatsThisHelpID](#) al número correspondiente al tema asociado de su archivo de Ayuda.

Para implementar la Ayuda "¿Qué es esto?", utilice las siguientes propiedades y métodos:

Propiedad	Descripción
<a href="#">WhatsThisHelp</a>	Establezca esta propiedad a True (.T.) en un formulario para activar la Ayuda "¿Qué es esto?" en el formulario y en los controles del formulario.
<a href="#">WhatsThisButton</a>	Establezca esta propiedad a True (.T.) si quiere que aparezca un botón "¿Qué es esto?" en la barra de título del formulario.
<a href="#">WhatsThisHelpID</a>	En un formulario, control o barra de herramientas, establezca esta propiedad al ID correspondiente al tema asociado de su archivo de Ayuda HTML.
<a href="#">WhatsThisMode</a>	Utilice este método para mostrar el puntero con la interrogación para la Ayuda "¿Qué es esto?" y activar dicho modo de Ayuda. Al hacer clic en un objeto se muestra el tema de Ayuda "¿Qué es esto?" especificado por la propiedad WhatsThisHelpID del objeto.

### Para implementar la Ayuda "¿Qué es esto?"

1. En modo Diseño, abra el formulario en el que vaya a activar la Ayuda "¿Qué es esto?".
2. Establezca la propiedad [WhatsThisHelp](#) del formulario a True (.T.).
3. Para mostrar un botón "¿Qué es esto?" en la barra de título del formulario, establezca la propiedad [WhatsThisButton](#) del formulario a True (.T.).
4. Para asociar un tema de Ayuda "¿Qué es esto?" al formulario, establezca la propiedad [WhatsThisHelpID](#) del formulario al Id. correspondiente al tema asociado de su archivo de Ayuda HTML.
5. Para asociar un tema de Ayuda "¿Qué es esto?" con un control concreto del formulario, seleccione el control y establezca su propiedad [WhatsThisHelpID](#) al Id. correspondiente al tema asociado de su archivo de Ayuda HTML.

### Programar las funciones de Ayuda

Puede programar su aplicación para que los usuarios tengan acceso a su sistema de Ayuda HTML. Aunque un sistema de Ayuda HTML puede consistir en uno o varios archivos separados, la Ayuda aparece ante los usuarios como parte integrante de su aplicación.

Puede programar su aplicación de Visual FoxPro para utilizar Ayuda gráfica y Ayuda de tipo .dbf con los comandos SET HELP TO y SET TOPIC TO. [SET HELP TO](#) especifica el nombre de un archivo de Ayuda personalizado para su aplicación. [SET TOPIC TO](#) establece la palabra clave identificativa de un tema del archivo de Ayuda personalizado.

## Reservar F1

Cuando un usuario presiona F1 dentro de su aplicación, Visual FoxPro puede mostrar un tema de Ayuda interactiva. Para hacerlo así, asigne un Id. de contexto con un tema de su tabla de Ayuda y asigne el mismo valor a la propiedad [HelpContextID](#) del formulario o el control. Cuando el formulario o el control tengan el enfoque y el usuario presione F1, Visual FoxPro muestra el tema asociado.

**Nota** De forma predeterminada, F1 está activada para la Ayuda interactiva. Como es un estándar admitido para la Ayuda, no se recomienda la modificación de esta tecla.

## Incluir botones de Ayuda en los formularios

Si agrega botones de Ayuda a sus formularios, los usuarios pueden tener acceso a la Ayuda con más facilidad. Especialmente, debe considerar la inclusión de un botón de Ayuda si sus usuarios no son muy experimentados.

### Para establecer la sensibilidad al contexto y agregar un botón de Ayuda

1. En el evento [Init](#) de su formulario, establezca la propiedad [HelpContextID](#) de todos los objetos del formulario al mismo valor, correspondiente a un tema de Ayuda. Por ejemplo, si dicho valor es 7, podría utilizar el siguiente comando:

```
THIS.SetAll("HelpContextID", 7)
```

2. Agregue un botón de comando al formulario.
3. Establezca la propiedad [Caption](#) del botón de comando a "Ayuda".
4. En el evento [Click](#) del botón de comando, agregue el siguiente comando:

```
HELP ID THIS.HelpContextID
```

**Sugerencia** Guarde el botón de Ayuda como una clase de forma que pueda incluirlo fácilmente en cualquier otro formulario. Para obtener más información acerca de cómo guardar objetos como clases, vea el capítulo 9, [Crear formularios](#).

## Distribuir un sistema de Ayuda HTML compilado

Además del archivo .chm que se crea para el sistema de Ayuda HTML, puede utilizar un programa de instalación de uso y distribución gratuita, Hhupd.exe, que instala y registra los componentes de ejecución de la Ayuda HTML mostrados a continuación. El Explorador de Internet o el motor de ejecución del Explorador de Internet tienen que estar instalados en los equipos de los usuarios.

Componente	Descripción
Hhctrl.ocx	Control ActiveX de la Ayuda HTML
Itss.dll	Biblioteca de vínculos dinámicos que controla el HTML compilado
Itircl.dll	Biblioteca de vínculos dinámicos para búsquedas de texto
Hh.exe	Visor de Ayuda HTML

Este programa de instalación se encuentra en la carpeta Redist de la carpeta de instalación del Taller de Ayuda HTML. Puede llamar a dicho programa de instalación desde otros programas de instalación y ejecutarlo en modo 'silencioso' para que no interfiera con el programa de instalación que haya creado. Para obtener la lista completa de las opciones de línea de comandos, ejecute Hhupd.exe/?.

## WinHelp 4.0

Utilice Microsoft Help Workshop, proporcionado en las versiones anteriores de Visual FoxPro, para crear archivos Winhelp. Microsoft Help Workshop incluye una Guía de diseño de Ayudas. La Guía de diseño de Ayudas (Hcw.hlp) es un archivo de Ayuda gráfica que contiene gran parte de la información necesaria para diseñar sistemas de Ayuda robustos.

### Elegir las funciones de Ayuda

Los sistemas WinHelp puede tener algunas o todas las funciones siguientes:

- Una página de contenido que proporciona una vista jerárquica de los temas del sistema de Ayuda.
- Un índice, basado en las palabras clave que usted proporcione, que guía al usuario hasta una información específica.
- Funciones de búsqueda de texto que permiten que los usuarios busquen palabras o frases en la Ayuda.
- Texto con varias fuentes, tamaños de fuente y colores.
- Gráficos, incluyendo mapas de bits con varias resoluciones.
- Macros que automatizan o amplían la operación del sistema de Ayuda.
- Zonas activas (áreas sensibles al *mouse* que se crean para proporcionar a los usuarios saltos que vinculan temas; ventanas emergentes que muestran texto adicional; y macros incluidas en el sistema de Ayuda).
- Hipergráficos segmentados: gráficos con una o varias zonas activas.
- Ventanas secundarias.
- Menús personalizables.
- Gráficos en formato metarchivo de Windows.
- .DLLs.

### Diseñar el acceso a la Ayuda en pantalla

Además de crear un archivo WinHelp que contenga la información necesaria, tiene que proporcionar los medios a través de los cuales los usuarios de su aplicación van a tener acceso a la Ayuda. Hay tres



maneras de proporcionar Ayuda:

- Un menú Ayuda (un menú que aparece en la barra del menú principal de su aplicación).
- Ayuda interactiva (Ayuda que aparece cuando el usuario presiona F1, u otra tecla especificada, mientras está seleccionado un objeto, control u opción de un menú determinado).
- Ayuda "¿Qué es esto?" (Ayuda que aparece como sugerencia emergente cuando el usuario pide Ayuda sobre un objeto o control específico).

## Diseñar el menú Ayuda

El menú Ayuda suele contener comandos que proporcionan acceso a los temas de su sistema de Ayuda. WinHelp 4.0 incorpora la ventana Buscador de Ayuda, que es un único cuadro de diálogo que proporciona acceso al contenido, al índice y a la búsqueda de texto.

## La ventana Buscador de Ayuda



Se recomienda que sólo un comando del menú Ayuda abra la ventana Buscador de Ayuda. Además de dicho comando, puede incluir otros comandos en el menú Ayuda para proporcionar información del sistema o la información de derechos de autor y la versión de su aplicación.

Puede invocar la ventana Buscador de Ayuda desde los programas mediante la función WinHelp con el parámetro HELP FINDER. Para obtener más información, vea "Usar la función WinHelp" más adelante en este capítulo y el tema WinHelp de la Guía de diseño de Ayudas.

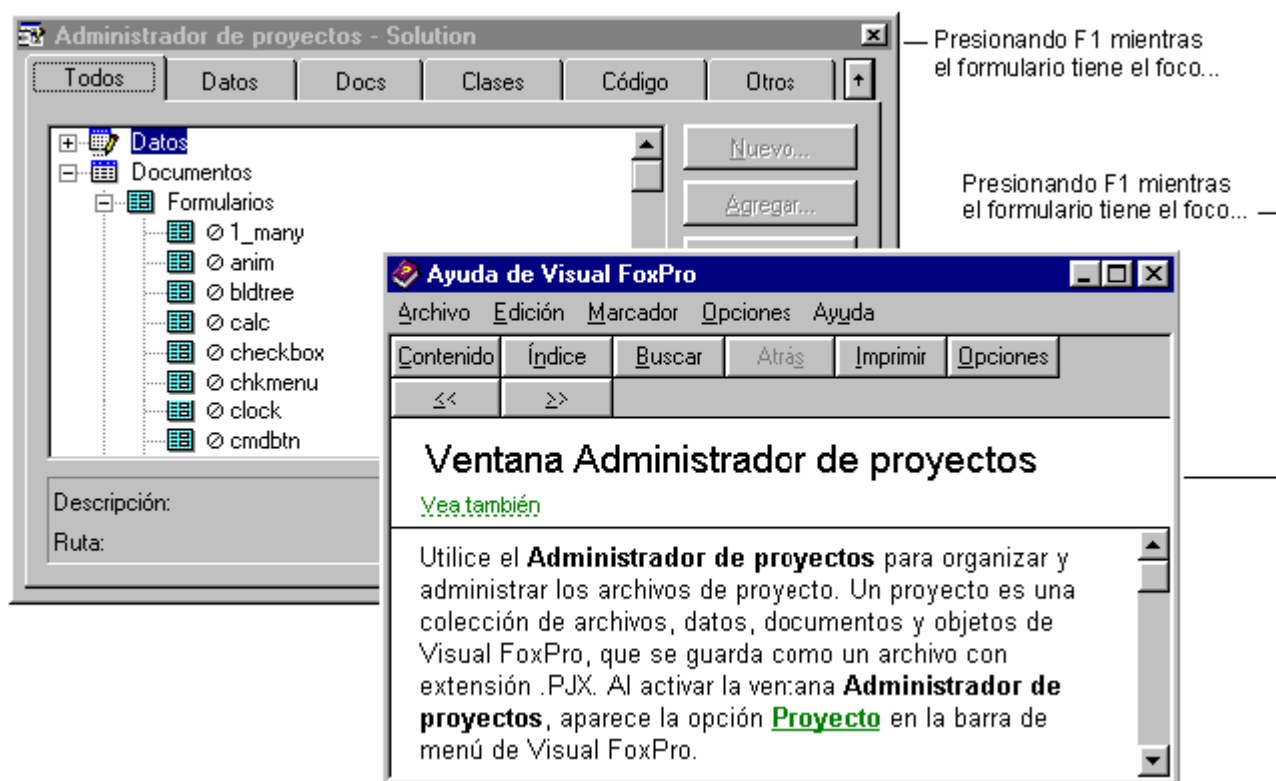
## Interactividad de la Ayuda

La Ayuda interactiva permite que los usuarios tengan acceso a los temas de Ayuda relacionados con lo que estén haciendo o viendo en su aplicación en cualquier momento dado. Por ejemplo, si un usuario está viendo un formulario de introducción de datos, la Ayuda interactiva podría proporcionar un tema relacionado específicamente con dicho formulario

Usted decide el nivel de detalle de la implementación de la Ayuda interactiva en su aplicación. Por ejemplo, puede asociar un tema de Ayuda interactiva con un formulario, o puede asociar temas de Ayuda más detallados para cada uno de los controles y campos del formulario.

Se suele llegar a la Ayuda interactiva al presionar F1, pero puede especificar la activación de la Ayuda interactiva mediante cualquier tecla con [ON KEY LABEL](#).

### Usar WinHelp interactiva en un formulario



Para implementar Ayuda interactiva, tiene que especificar el archivo de Ayuda de su aplicación y, a continuación, asociar temas de Ayuda específicos a objetos de su aplicación.

### Para agregar Ayuda interactiva

1. Especifique el archivo de Ayuda de su aplicación.
2. Asigne un tema de Ayuda a cada objeto para el que vaya a proporcionar Ayuda interactiva.

### Especificar el archivo de Ayuda

El archivo de Ayuda al que se tiene acceso en su aplicación se determina al incluir el comando [SET](#)

[HELP TO archivo](#) en el código, donde *archivo* es el nombre del archivo de Ayuda. Por ejemplo, si el archivo de Ayuda es MiAyuda.hlp, puede utilizar el siguiente comando:

```
SET HELP TO MIAYUDA.HLP
```

Este comando suele estar incluido en el código de configuración del programa principal de su aplicación.

### Asignar temas de Ayuda

Puede asignar un tema de Ayuda a objetos específicos de su aplicación de Visual FoxPro.

#### Para asignar un tema de Ayuda a un objeto

1. En modo Diseño, abra el objeto (por ejemplo, un formulario, un control o una barra de herramientas) al que vaya a asignar Ayuda interactiva.
2. Vea las propiedades del objeto.
3. Establezca la propiedad [HelpContextID](#) al número correspondiente al tema asociado de su archivo de Ayuda.

Para obtener más información acerca de la asociación de temas de Ayuda con Id. de contexto, consulte la Guía de diseño de Ayuda.

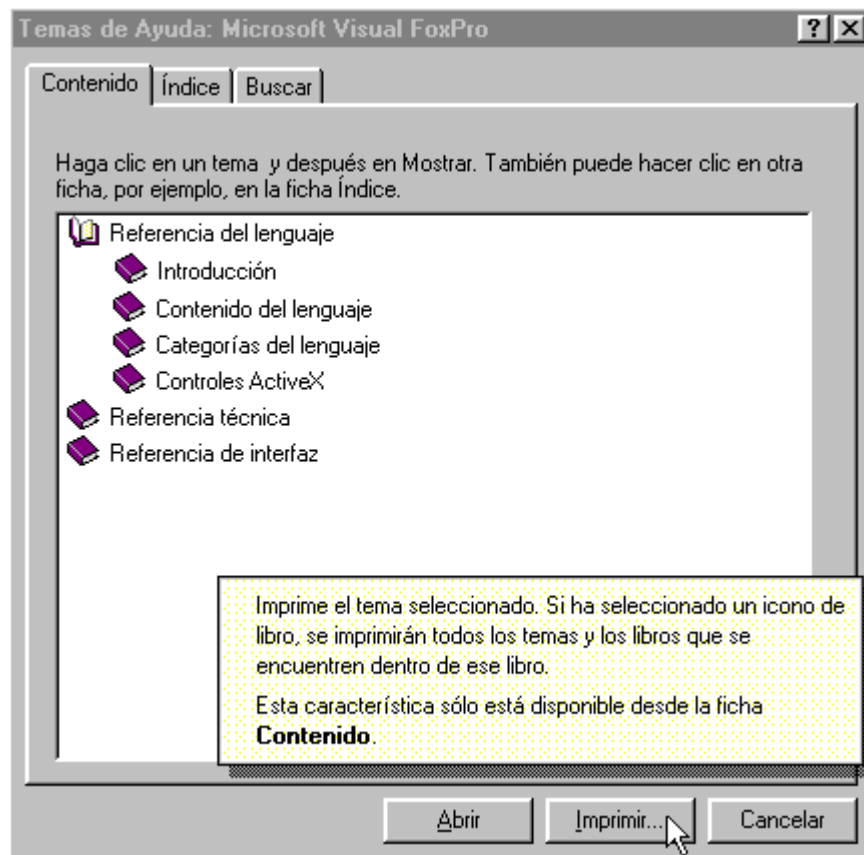
**Nota** Para asignar temas de Ayuda a títulos de menú o a comandos de menú, tiene que incluir el comando [SET TOPIC TO](#) en el procedimiento asociado con el título de menú o el comando de menú.

### Implementar la Ayuda "¿Qué es esto?"

La Ayuda "¿Qué es esto?" es similar a la Ayuda interactiva puesto que proporciona Ayuda relacionada con el objeto o control determinado que tiene actualmente el enfoque. Sin embargo, en lugar de invocar el archivo de Ayuda y mostrar el tema de Ayuda en la ventana de Ayuda con su tamaño predeterminado, la Ayuda "¿Qué es esto?" muestra el tema en una pequeña ventana emergente que desaparece en cuanto el usuario haga clic en cualquier parte de la pantalla. La Ayuda "¿Qué es esto?" es útil para proporcionar breves descripciones, definiciones o sugerencias sobre controles concretos.

**Sugerencia** Los temas de Ayuda "¿Qué es esto?" deben ser breves y concisos para que la ventana no sea tan grande como para ocultar el componente que describe.

#### Ayuda "¿Qué es esto?"

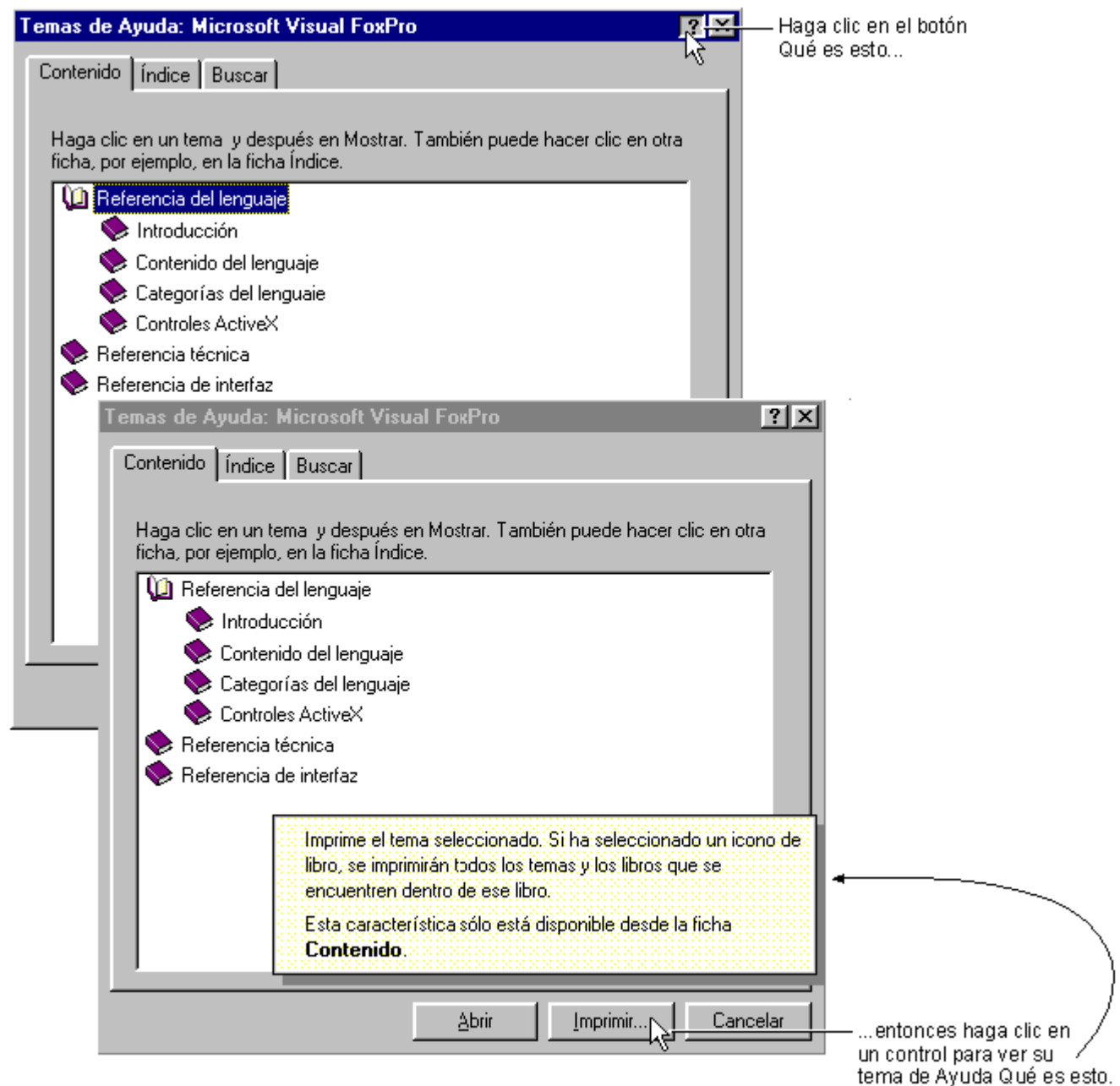


La Ayuda "¿Qué es esto?" se asocia a un formulario, control de formulario o barra de herramientas particular al establecer su propiedad [WhatsThisHelpID](#) al número correspondiente al tema asociado de su archivo de Ayuda

Para implementar la Ayuda "¿Qué es esto?", utilice las siguientes propiedades y métodos:

Propiedad	Descripción
<a href="#">WhatsThisHelp</a>	Establezca esta propiedad a True (.T.) en un formulario para activar la Ayuda "¿Qué es esto?" en el formulario y en los controles del formulario.
<a href="#">WhatsThisButton</a>	Establezca esta propiedad a True (.T.) si desea que aparezca un botón "¿Qué es esto?" en la barra de título del formulario.
<a href="#">WhatsThisHelpID</a>	En un formulario, control o barra de herramientas, establezca esta propiedad al Id. correspondiente al tema asociado de su archivo de Ayuda.
<a href="#">WhatsThisMode</a>	Utilice este método para mostrar el puntero con la interrogación para la Ayuda "¿Qué es esto?" y activar dicho modo de Ayuda. Al hacer clic en un objeto se presenta el tema de Ayuda "¿Qué es esto?" especificado por la propiedad WhatsThisHelpID del objeto.

## Usar un botón "¿Qué es esto?"



## Para implementar la Ayuda "¿Qué es esto?"

1. En modo Diseño, abra el formulario en el que vaya a activar la Ayuda "¿Qué es esto?".
2. Establezca la propiedad [WhatsThisHelp](#) del formulario a True (.T.).
3. Para mostrar un botón "¿Qué es esto?" en la barra de título del formulario, establezca la propiedad [WhatsThisButton](#) del formulario a True (.T.).
4. Para asociar un tema de Ayuda "¿Qué es esto?" al formulario, establezca la propiedad [WhatsThisHelpID](#) del formulario al Id. correspondiente al tema asociado de su archivo de

Ayuda.

5. Para asociar un tema de Ayuda "¿Qué es esto?" a un control específico del formulario, seleccione el control y establezca su propiedad [WhatsThisHelpID](#) al Id. correspondiente al tema asociado de su archivo de Ayuda.

## Programar las funciones de Ayuda

Puede programar su aplicación para que los usuarios tengan acceso a su sistema de Ayuda en la Ayuda de Microsoft. Aunque un sistema de Ayuda puede consistir en uno o varios archivos separados, la Ayuda aparece ante los usuarios como parte integrante de su aplicación.

Puede programar su aplicación Visual FoxPro para utilizar Ayuda gráfica y Ayuda de tipo .dbf con los comandos SET HELP TO y SET TOPIC TO o con la función WinHelp, descrita más adelante en este capítulo. [SET HELP TO](#) especifica el nombre de un archivo de Ayuda personalizado para su aplicación. [SET TOPIC TO](#) establece la palabra clave identificativa de un tema del archivo de Ayuda personalizado.

## Usar la función WinHelp

Otra forma de programar su aplicación para utilizar Ayuda es la utilización de la función WinHelp. La función WinHelp forma parte de la interfaz de programación de aplicaciones (API) de Windows. La función WinHelp sólo está disponible en las plataformas Windows.

Puede utilizar la función WinHelp junto con la propiedad HelpContextID, para invocar un segundo archivo de Ayuda.

**Sugerencia** Si utiliza SET HELP TO, HELP ID y SET TOPIC TO, no necesita utilizar la función WinHelp.

## Para utilizar la función WinHelp

1. Defina los parámetros que va a pasar desde su aplicación.

Para obtener la descripción de dichos parámetros, vea "El parámetro wCmd", más adelante en este mismo capítulo.

2. Establezca la biblioteca con [SET LIBRARY TO](#) y defina las variables que vaya a utilizar, normalmente en el código de inicialización del archivo principal de la aplicación.

```
SET LIBRARY TO SYS(2004) + "FOXTOOLS.FLL" ADDITIVE  
Help = RegFn("Help", "LCIC", "I")
```

La biblioteca tiene que establecerse a Foxtools.fll. [SYS\(2004\)](#) devuelve el directorio raíz de Visual FoxPro, en donde está instalada Foxtools.fll.

Si quiere abrir un tema de Ayuda enviando su palabra clave K, defina una variable con RegFn ( ), de igual modo que la variable Help del ejemplo anterior. Si desea abrir un tema de Ayuda asociado a un número, define una variable con RegFn de igual modo que la variable Help del

ejemplo anterior y utilice un número en lugar de una cadena de texto en *dwData*. Si pasa números, tiene que asociarlos en la sección [MAP] del archivo .hlp con cadenas de contexto únicas definidas con la nota al pie (#).

### 3. Utilice CallFn( ) para invocar la función.

Por ejemplo, si su archivo de Ayuda es MiAyuda.hlp, utilice CallFn( ) para abrir un tema en MiAyuda.hlp incluyendo la palabra clave K del tema:

```
#define HELP_KEY 0x0101
wCmd = HELP_KEY
cNombreArchivo = MiAyuda.hlp
dwData = "Agregar elementos de menú en tiempo de ejecución"
CallFn(Help, MainHWND(), cFileName, wCmd, dwData)
```

Para obtener más información sobre las funciones de FoxTools, consulte Foxtools.chm en el directorio Vfp98\Tools.

## Especificar los parámetros de WinHelp

Los siguientes parámetros especifican las opciones de la función WinHelp.

### El parámetro hWnd

El parámetro *hWnd* identifica la ventana que solicita la Ayuda. La Ayuda utiliza este identificador para saber qué aplicaciones han solicitado Ayuda. En Visual FoxPro, utilice la función MainHWND( ), incluida en la biblioteca Foxtools.fll, para obtener el parámetro *hWnd*.

### El argumento lpzFileName

El argumento *lpzFileName* representa una cadena de texto que especifica la ruta y el nombre del archivo de Ayuda que contiene el tema deseado. Se pasa por valor.

### El parámetro wCmd

El parámetro *wCmd* especifica el tipo de búsqueda que la Ayuda utiliza para buscar el tema especificado o que la aplicación ya no requiere Ayuda. Puede establecerse a uno de los siguientes valores.

Constante	Valor	Significado
HELP_FINDER	0x000B	Muestra la ventana Buscador de Ayuda.
HELP_CONTEXT	0x0001	Muestra Ayuda sobre un tema específico identificado por un número de contexto.
HELP_HELPONHELP	0x0004	Carga Help.hlp y muestra el tema Usar el índice de Ayuda.

HELP_INDEX	0x0003	Muestra el tema de índice de Ayuda definido en la sección [OPTIONS] del archivo de proyecto de Ayuda (.hlp).
HELP_KEY	0x0101	Muestra el primer tema encontrado entre los que contienen la palabra clave especificada en el parámetro <i>dwData</i> .
HELP_QUIT	0x0002	Informa a la aplicación de Ayuda que ya no es necesaria. Si ninguna otra aplicación ha solicitado Ayuda, Windows cierra la aplicación de Ayuda.
HELP_SETINDEX	0x0005	Establece un tema específico como tema de índice.

### El parámetro *dwData*

El parámetro *dwData* representa el tema para el que la aplicación solicita Ayuda. Su contenido y su formato dependen del valor de *wCmd* pasado cuando la aplicación invoca la función WinHelp.

En la mayor parte de las llamadas a Ayuda, el argumento *dwData* se pasa por valor. Es la forma predeterminada de Visual FoxPro.

Dependiendo de las circunstancias, *dwData* puede representar una cadena de texto, indicando la palabra clave que se busca, o un valor numérico, indicando el número de contexto que identifica al tema concreto.

La siguiente lista describe el formato de *dwData* para cada valor de *wCmd*.

Valor de <i>wCmd</i>	Formato de <i>dwData</i>
HELP_CONTEXT	Un valor numérico que contiene el número de contexto del tema. En lugar de HELP_INDEX, HELP_CONTEXT puede utilizar el valor -1.
HELP_HELPONHELP	Ignorado.
HELP_INDEX	Ignorado.
HELP_KEY	Un puntero a una cadena que contiene la palabra clave del tema deseado.
HELP_QUIT	Ignorado.
HELP_SETINDEX	Un valor numérico que contiene el número de contexto del tema que se va a establecer como índice.

Como la función WinHelp puede especificar un número de contexto o una palabra clave, acepta la



Ayuda interactiva y la búsqueda de temas en el archivo de Ayuda.

**Nota** Si un archivo de Ayuda contiene dos o más índices, la aplicación tiene que asignar uno como índice predeterminado. Para asegurarse de que el índice correcto esté establecido, la aplicación tiene que invocar la Ayuda con *wCmd* establecido a *help\_setindex* (y *dwData* especificando el identificador de contexto adecuado). Todas las llamadas a Ayuda tienen que ir seguidas de un comando establecido a *help\_context*. Nunca se debe utilizar *HELP\_index* con *help\_setindex*.

### Reservar F1 para la Ayuda

Cuando un usuario presiona F1 en su aplicación, Visual FoxPro puede mostrar un tema de Ayuda interactiva. Para hacerlo, asigne un Id. de contexto de Ayuda a un tema de su tabla de Ayuda y asigne el mismo valor a la propiedad [HelpContextID](#) del formulario o el control. Cuando el formulario o el control tiene el enfoque y el usuario presiona F1, Visual FoxPro muestra el tema asociado.

**Nota** De forma predeterminada, F1 está activada para la Ayuda interactiva. Como es un estándar admitido para la Ayuda, no se recomienda la modificación de esta tecla.

### Incluir botones de Ayuda en los formularios

Si agrega botones de Ayuda a sus formularios, los usuarios pueden tener acceso a la Ayuda con más facilidad. Especialmente, debe considerar la inclusión de un botón de Ayuda si sus usuarios no son muy experimentados.

### Para establecer la sensibilidad al contexto y agregar un botón de Ayuda

1. En el evento [Init](#) de su formulario, establezca la propiedad [HelpContextID](#) de todos los objetos del formulario al mismo valor, correspondiente a un tema de Ayuda. Por ejemplo, si dicho valor es 7, podría utilizar el siguiente comando:

```
THIS.SetAll("HelpContextID", 7)
```

2. Agregue un botón de comando al formulario.
3. Establezca la propiedad [Caption](#) del botón de comando a "Ayuda".
4. En el evento [Click](#) del botón de comando, agregue el siguiente comando:

```
HELP ID THIS.HelpContextID
```

**Sugerencia** Guarde el botón de Ayuda como una clase de forma que pueda incluirlo fácilmente en cualquier otro formulario. Para obtener más información acerca de cómo guardar objetos como clases, vea el capítulo 9, [Crear formularios](#).

### Salir de la Ayuda

La aplicación de Ayuda es un recurso compartido disponible por todas las aplicaciones Windows. Como también es una aplicación independiente, el usuario puede ejecutarla como cualquier otra aplicación. Como resultado, su aplicación tiene un control limitado sobre la aplicación de Ayuda.

Aunque su aplicación no puede cerrar directamente la aplicación de Ayuda, puede informar a la aplicación de Ayuda que ya no la va a necesitar. Antes de cerrar su ventana principal, su aplicación tiene que invocar a Ayuda con el parámetro *wCmd* establecido a *help\_quit*, que informa a la Ayuda de que su aplicación ya no va a necesitarla.

Las aplicaciones que hayan llamado a la Ayuda en algún punto de su ejecución tienen que llamar a la Ayuda con el parámetro *wCmd* establecido a *help\_quit* antes de terminar.

Si una aplicación abre más de un archivo de Ayuda, tiene que invocar la función *WinHelp* para salir de la aplicación de Ayuda una vez por cada archivo abierto.

Si una aplicación o una biblioteca de vínculos dinámico (DLL) ha abierto un archivo de Ayuda pero no quiere seguir manteniendo la asociación con la instancia de la aplicación de Ayuda, la aplicación o la DLL tiene que invocar *WinHelp* con el parámetro *wCmd* establecido a *help\_quit* para salir de dicha instancia de la aplicación de Ayuda.

**Nota** Antes de terminar, las aplicaciones o las DLL tienen que llamar siempre a *WinHelp* por cada uno de los archivos de Ayuda abiertos. Un archivo de Ayuda se abre si hace cualquier llamada a la Ayuda con el nombre del archivo de Ayuda.

La aplicación de Ayuda no termina hasta que todas las ventanas que hayan solicitado la Ayuda la hayan invocado con *wCmd* establecido a *help\_quit*. Si alguna aplicación no lo hiciera, la aplicación de Ayuda seguiría ejecutándose, incluso después de que todas las aplicaciones que solicitaron Ayuda hubieran terminado.

## Capítulo 24: Crear Ayuda de tipo .DBF

La Ayuda de tipo .DBF es fácil de crear y utiliza tablas estándar de Visual FoxPro que se transfieren fácilmente a otras plataformas. Si quiere una solución sencilla para proporcionar ayuda, si programa aplicaciones para varias plataformas o si prefiere un archivo de ayuda compatible con versiones anteriores, puede proporcionar una ayuda de tipo .DBF a sus usuarios.

En este capítulo se tratan los temas siguientes:

- [Diseñar Ayuda de tipo .DBF](#)
- [Ver el archivo de ejemplo de Ayuda de tipo .DBF](#)
- [Usar la Ayuda de tipo .DBF](#)
- [Personalizar la Ayuda de tipo .DBF](#)

### Diseñar la Ayuda de tipo .DBF

Los archivos de Ayuda de tipo .DBF, o las tablas de Ayuda, son [tablas libres](#) que se muestran en la ventana de Ayuda de tipo .DBF. Con este tipo de Ayuda, los usuarios pueden:

- Obtener Ayuda interactiva sobre el cuadro de diálogo, el comando de menú o el objeto actual al presionar F1.

- Saltar a temas relacionados del archivo de Ayuda seleccionando un tema en la lista desplegable "Vea también".
- Seleccionar una palabra o frase clave en medio de un tema y saltar a dicha palabra o frase haciendo clic en el botón "Buscar".
- Copiar al Portapapeles el texto seleccionado en la ventana de Ayuda.

En Visual FoxPro se incluye un archivo de Ayuda de tipo .DBF, Ttrade.dbf, ubicado en el directorio ...\\Samples\\Vfp98\\Tastrade\\Help de Visual Studio. Las secciones siguientes usan el ejemplo Ttrade.dbf para describir el diseño y el desplazamiento por la ayuda de tipo .DBF.

## Ver el archivo de Ayuda de tipo .DBF

### Para ver el archivo de ayuda de ejemplo Ttrade.dbf

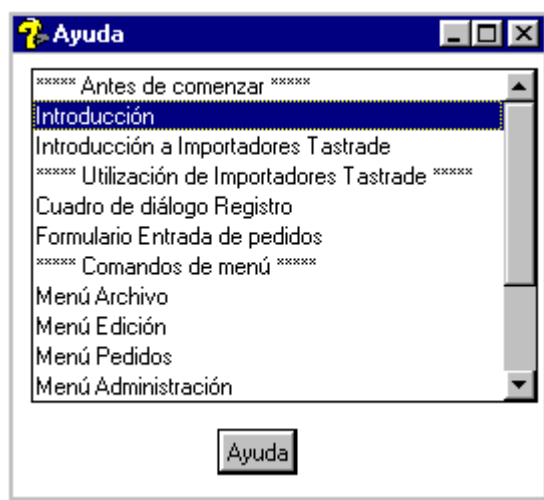
- En la ventana **Comandos**, con el directorio predeterminado establecido al que contiene Ttrade.dbf, escriba:

```
SET HELP TO TTRADE.DBF  
HELP
```

Visual FoxPro muestra la Ayuda de tipo .DBF Ttrade en su propia ventana.

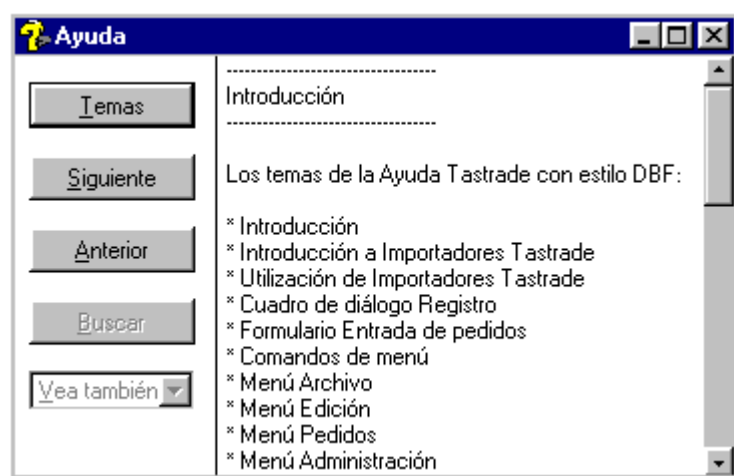
La ventana de Ayuda .DBF tiene dos modos: Temas y Detalles. El modo Temas muestra una lista de todos los temas del archivo de Ayuda. Observe que la fuente de la ventana Ayuda .DBF es MS Sans Serif y no se puede cambiar.

### Ventana de Ayuda .DBF en modo Temas



Al hacer doble clic en un tema, el contenido de ese tema aparece en el modo Detalles.

### Ventana de Ayuda .DBF en modo Detalles



## Requisitos de las tablas de Ayuda

Las tablas de Ayuda tienen capacidad para un máximo de 32.767 registros y deben contener al menos dos campos. La tabla puede tener un primer campo opcional para interactividad. El ejemplo siguiente muestra la configuración de una tabla de Ayuda típica:

### Tabla TTRADE en modo Examinar

Contextid	Topic	Details
0	***** Antes de comenzar *****	Memo
12	Introducción	Memo
0	Introducción a Importadores Tastrade	Memo
0	***** Utilización de Importadores Tastrade *****	Memo
10	Cuadro de diálogo Registro	Memo

No hay requisitos para los nombres de campo. Los tipos de campo son, en orden:

**Numéricos** Este campo es opcional y contiene el identificador de contexto (HelpContextID) empleado con la Ayuda interactiva.

**Carácter** Es el nombre del tema que aparece en el modo Temas de la ventana de Ayuda.

**Memo** Este campo contiene la información que aparece en el modo Detalles de la ventana de Ayuda.

Aparte de estos requisitos, puede agregar tantos campos adicionales como desee.

## Descripción de Ttrade.dbf

Ttrade.dbf es un buen ejemplo de un archivo de Ayuda de tipo .DBF. Puede diseñar su archivo de Ayuda basándose en Ttrade.dbf, o bien puede crear su propio diseño. Puesto que un archivo de Ayuda es una tabla, puede crear su propio archivo de Ayuda creando una tabla nueva o copiando y

modificando una tabla existente.

Para ver o modificar la tabla Ttrade.dbf, ábrala y examine su contenido como cualquier otra tabla. Si ha ejecutado anteriormente el comando SET HELP TO TTRADE.DBF, deberá utilizar primero el comando [SET HELP OFF](#) antes de abrir la tabla Ttrade.dbf.

## Temas de TTRADE

Ttrade.dbf incluye varios tipos de temas. Son los siguientes:

- Instrucciones paso a paso.
- Temas de interfaz para Ayuda interactiva, incluidos comandos de menú y cuadros de diálogo.
- Temas de referencia del lenguaje.

En su archivo de Ayuda puede utilizar estas categorías y otras distintas.

## Detalles de TTRADE

Cuando un usuario selecciona un tema en el modo Temas, la ventana de Ayuda muestra el contenido del campo memo llamado Details.

## Referencias cruzadas de TTRADE

En la mayoría de los temas de Ayuda, las referencias cruzadas de tipo "Vea también" aparecen al final de la información de Detalles. Estas referencias se muestran automáticamente en el cuadro "Vea también" y actúan como vínculos directos a los temas relacionados.

### Para crear una referencia cruzada tipo "Vea también"

1. Al final del campo memo, escriba **Vea también** seguido de un signo de dos puntos y espacios opcionales.
2. En la misma línea, escriba una lista delimitada por comas de los temas deseados.
3. Introduzca un retorno de carro para indicar el final de la lista.

La distinción entre mayúsculas y minúsculas no importa en la lista "Vea también"; Visual FoxPro elimina los espacios en blanco de cada tema al que se hace referencia. Por ejemplo, las referencias cruzadas para el tema Introducción aparecen a continuación.

### Contenido del campo memo del tema Introducción



**Sugerencia** Agregue líneas por encima y por debajo de la sección "Vea también" al final del tema para separarla visualmente del contenido.

Al buscar un tema en el cuadro "Vea también", Visual FoxPro intenta hacer coincidir la selección del usuario con el primer tema de Ayuda que conste o comience por la misma cadena de texto. Si Visual FoxPro no encuentra ninguna coincidencia, mostrará "No hay Ayuda para *tema*" en la barra de estado.

## Usar la Ayuda de tipo .DBF

Los usuarios podrán tener acceso fácilmente a la Ayuda si incluye un comando Contenido en el menú Ayuda que utilice el comando HELP. Cuando el usuario elija el comando Contenido, aparecerá la ventana de Ayuda en modo Temas. El usuario puede desplazarse por la lista para buscar el tema deseado o escribir una letra para seleccionar el primer tema que comience por esa letra. Una vez seleccionado un tema, hay tres formas de mostrar información sobre él:

- Hacer doble clic en el tema de la lista.
- Hacer clic en el botón Ayuda.
- Presionar la tecla ENTRAR.

## Personalizar la Ayuda de tipo .DBF

En el código de la aplicación, especifique el archivo de Ayuda que utiliza, los temas que se muestran y cuándo se muestran, y otras configuraciones opcionales. Si incluye Ayuda interactiva, los usuarios podrán obtener Ayuda al solicitarla desde cuadros de diálogo y los comandos de menú de la aplicación.

### Especificar una tabla de Ayuda

Especifique la tabla de Ayuda; para ello, ejecute el comando [SET HELP](#) TO *nombrearchivo*. De este modo se cerrará la tabla de Ayuda actual, si había alguna abierta, y se abrirá *nombrearchivo* como nueva tabla de Ayuda.

En una situación típica de programación, guarde en una variable el nombre del archivo de Ayuda actual y especifique el nombre del archivo de Ayuda en el código de inicialización, como en el ejemplo siguiente:

```
cAyudaUsuario = SET("HELP", 1)
SET HELP TO MIAYUDA.DBF
```

Al salir de la aplicación, podrá restaurar el archivo de Ayuda original:

```
SET HELP TO (cAyudaUsuario)
```

## Mostrar temas en la ventana de Ayuda

Después de especificar la tabla de Ayuda, puede especificar los temas que desea mostrar de esta forma:

- Para seleccionar temas por nombre, utilice los comandos [HELP Tema](#) o [SET TOPIC TO cNombreTemaAyuda](#).
- Para temas de Ayuda interactiva, utilice la propiedad [HelpContextID](#).
- Para mostrar un subconjunto de temas, use el comando [SET HELPFILTER](#).

### Seleccionar temas por nombre

Para seleccionar temas por nombre, utilice el comando [HELP Tema](#). Cuando utilice este comando, Visual FoxPro buscará en la tabla de Ayuda el registro cuyo campo de tema coincide con *Tema*. En la búsqueda no se distinguen mayúsculas de minúsculas.

Cuando Visual FoxPro encuentre una coincidencia, mostrará el contenido del campo memo Detalles en el modo Detalles de la ventana de Ayuda. Si Visual FoxPro no encuentra ninguna coincidencia, mostrará todos los temas de una lista en el cuadro de diálogo Temas de Ayuda con la coincidencia más cercana resaltada.

### Activar la Ayuda interactiva

Puede diseñar la aplicación de modo que el usuario pueda obtener Ayuda interactiva de dos formas:

- Al presionar la tecla F1 en cualquier momento.
- Al hacer clic en un botón "Ayuda" que se incluye en los formularios y los cuadros de diálogo.

### Reserva de F1

Cuando un usuario presiona F1 en la aplicación, Visual FoxPro puede mostrar un tema de Ayuda interactiva. Para ello, asigne un HelpcontextID a un tema en la tabla de Ayuda y asigne el mismo valor a la propiedad [HelpContextID](#) del formulario o el control. Cuando el formulario o el control tenga el enfoque y el usuario presione F1, Visual FoxPro mostrará el tema correspondiente.

**Nota** F1 está activada para Ayuda interactiva de forma predeterminada. Puesto que se trata de un estándar reconocido para la Ayuda, no se recomienda redefinir esta tecla.

## Agregar botones Ayuda a formularios

Si agrega botones Ayuda a los formularios, los usuarios podrán tener acceso a la Ayuda de forma más fácil. El botón Ayuda puede resultar especialmente útil si el usuario no tiene mucha experiencia.

### Para crear un tema de Ayuda interactiva

1. En el primer campo de un registro de la tabla de Ayuda, escriba un valor numérico.
2. Rellene los campos Tema y Detalles del registro.

Ahora ya puede asignar el tema de Ayuda al formulario. Es conveniente asignar un botón Ayuda, el formulario y sus objetos al mismo tema de Ayuda.

### Para establecer Ayuda interactiva y agregar un botón Ayuda

1. En el evento [Init](#) del formulario, establezca la propiedad [HelpContextID](#) de todos los objetos del formulario con el mismo valor que haya asignado al tema de Ayuda. Por ejemplo, si el valor es 7, podrá emplear el comando siguiente:

```
THIS.SetAll("HelpContextID", 7)
```

2. Agregue un botón de comando al formulario.
3. Establezca como Ayuda la propiedad [Caption](#) del botón de comando.
4. En el evento [Click](#) del botón de comando, agregue el comando siguiente:

```
HELP ID THIS.HelpContextID
```

**Sugerencia** Guarde el botón de Ayuda como una clase de manera que pueda agregarlo fácilmente a cualquier formulario. En el [Diseñador de formularios](#), elija Guardar como en el menú Archivo. Para obtener más información sobre cómo guardar objetos como clases, consulte el capítulo 9, [Crear formularios](#).

## Controlar la ubicación de la ventana de Ayuda

Para especificar una ubicación para la Ayuda, debe crear su propia ventana mediante el comando [DEFINE WINDOW](#). Utilice este comando en el código de inicialización para especificar el tamaño y la ubicación de la ventana. A continuación, muestre la ventana activándola o mostrándola.

Por ejemplo, los comandos siguientes definen una ventana llamada prueba y muestran la tabla de Ayuda actual dentro de esa ventana:

```
DEFINE WINDOW prueba FROM 1,1 TO 35,60 SYSTEM  
ACTIVATE WINDOW prueba  
HELP IN WINDOW prueba
```

## Adaptar la Ayuda a su aplicación

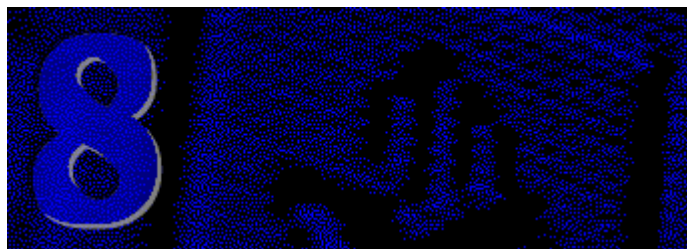


Puesto que es posible agregar cualquier número de campos a una tabla de Ayuda, y que puede emplearse cualquier expresión lógica para seleccionar temas de Ayuda, la imaginación es el único límite para el sistema de Ayuda que puede crear.

Por ejemplo, puede:

- Definir una o muchas variables de programa que controlan el comportamiento del sistema de Ayuda y, a continuación, asignar valores a estas variables de acuerdo con el modo operativo del programa.
- Proporcionar más detalles en los archivos de Ayuda para los usuarios sin experiencia que para los usuarios experimentados.
- Permitir que los usuarios tengan acceso a la Ayuda sólo si introducen una contraseña determinada.

# Manual del programador, Parte 8: Distribuir aplicaciones



Después de haber programado una aplicación, puede prepararla para su distribución con todos los archivos necesarios y crear discos de distribución.

## Capítulo 25 [Generar una aplicación para su distribución](#)

Aprenda a personalizar su aplicación y a asegurarse de que dispone de todos los archivos y recursos necesarios antes de crear un archivo ejecutable para su distribución.

## Capítulo 26 [Crear discos de distribución](#)

La creación de discos y una rutina de instalación para los usuarios se convierte en un proceso sencillo gracias al Asistente para instalación de Visual FoxPro.

## Capítulo 25: Generar una aplicación para su distribución

Crear una aplicación para distribuirla es similar a programar una aplicación estándar de Visual FoxPro. Para ello, se trabaja en el entorno de programación de Visual FoxPro de la forma habitual, pero posteriormente se crea un archivo ejecutable o servidor de Automatización (un componente COM ) y se prueba en el entorno de tiempo de ejecución. A continuación debe distribuir su aplicación y todos los archivos relacionados con ella a las ubicaciones de distribución.

Este capítulo describe las modificaciones que necesita efectuar para preparar una aplicación para su distribución, así como algunas sugerencias de cambios que ayudan a lograr que una aplicación distribuida tenga una apariencia única.

Los temas tratados en este capítulo incluyen:

- [El proceso de distribución](#)
- [Preparar una aplicación para su distribución](#)
- [Personalizar una aplicación para su distribución](#)
- [Preparar la creación de discos de distribución](#)

## El proceso de distribución

La lista siguiente identifica los pasos que necesita seguir para distribuir una aplicación de Visual FoxPro.

- Crear y depurar la aplicación mediante el entorno de programación de Visual FoxPro.
- Preparar y personalizar la aplicación para el entorno en tiempo de ejecución. Para obtener más detalles, consulte [Personalizar una aplicación para su distribución](#) y [Preparar una aplicación para su distribución](#).

**Importante** Determinadas características del entorno de programación no están disponibles en el entorno de tiempo de ejecución, por lo que debe eliminarlas de su aplicación. Estas características se muestran en [Eliminar características y archivos restringidos de Visual FoxPro](#), más adelante en este mismo capítulo.

- Crear documentación y Ayuda en pantalla. Para obtener más información sobre la creación de Ayuda para su aplicación, consulte la parte 7, [Crear archivos de Ayuda](#).
- Crear un archivo de aplicación o ejecutable. Para obtener más información acerca de la generación de una aplicación, consulte el capítulo 13, [Compilar una aplicación](#).
- Crear un directorio de distribución que contenga todos los archivos que necesiten los usuarios para ejecutar la aplicación.
- Crear discos de distribución y una rutina de instalación mediante el [Asistente para instalación](#). Para obtener información adicional al respecto, consulte el capítulo 26, [Crear discos de distribución](#).
- Empaquetar y distribuir los discos de la aplicación, junto con la documentación impresa creada.

## Preparar una aplicación para su distribución

En las secciones siguientes se describen los pasos que posiblemente necesite seguir con el fin de preparar la aplicación para el entorno de tiempo de ejecución. Estos pasos incluyen:

- Elegir el tipo de generación.
- Considerar los problemas del entorno.
- Asegurar el correcto comportamiento en tiempo de ejecución.
- Incluir recursos en la aplicación.
- Eliminar características y archivos restringidos.
- Personalización de la aplicación.

### Elegir el tipo de generación

Antes de que pueda distribuir la aplicación, debe generar un archivo de aplicación, con la extensión .app, o bien un archivo ejecutable, con la extensión .exe. La siguiente tabla muestra las diferencias entre ambos tipos de generación.

Tipo de generación	Características
Archivo de aplicación (.app)	Entre 10 y 15 KB más pequeño que un archivo .exe. El usuario debe ser propietario de una copia de Visual FoxPro.
Archivo ejecutable (.exe)	La aplicación incluye el cargador de Visual FoxPro, por lo que el usuario no necesita ser propietario de ninguna copia de Visual FoxPro. Debe proporcionar los dos archivos de soporte Vfp6r.dll y Vfp6renu.dll (EN denota la versión inglesa). Debe colocar estos archivos en el mismo directorio que el archivo ejecutable o en la ruta de MS-DOS path. Vea <a href="#">BUILD EXE</a> para obtener más información acerca de cómo crear y distribuir ejecutables.
DLL OLE	Se usa para crear un archivo al que se puede llamar desde otras aplicaciones. Para ver detalles sobre el uso de esta opción de generación, consulte el capítulo 16, <a href="#">Agregar OLE</a> .

Al elegir el tipo de generación, tenga en cuenta el tamaño del archivo final de la aplicación y si los usuarios disponen o no de una copia de Visual FoxPro.

## Considerar problemas de hardware, memoria y red

Debe considerar y probar el entorno mínimo en el que puede funcionar la aplicación, incluyendo la cantidad de espacio en disco y la memoria. Los resultados de las pruebas y la resolución de otros problemas tratados en este capítulo pueden ayudarle a determinar el tipo de generación que debe elegir, los archivos que necesita incluir en la aplicación y la manera de estructurar el directorio de distribución.

Las aplicaciones que cree tendrán los mismos requisitos de hardware, memoria y red que Visual FoxPro. Para obtener más información acerca de estos requisitos, consulte "Requisitos del sistema" en el capítulo 1, [Instalar Visual FoxPro](#), de la *Guía de instalación*. Si desea información adicional acerca de la creación de aplicaciones para entornos multiusuario, consulte el capítulo 17, [Programar para acceso compartido](#).

Los archivos ejecutables siempre comprueban la presencia de la biblioteca de tiempo de ejecución de Visual FoxPro, Vfp6r.dll. Para ejecutar un archivo .exe de aplicación mediante la versión de programación de Visual FoxPro, debe obligar a la aplicación a usar el archivo .exe de Visual FoxPro.

## Para ejecutar una aplicación en Visual FoxPro

- Inicie Visual FoxPro y, a continuación, en el menú **Programa**, elija **Ejecutar**. En el cuadro de diálogo [Ejecutar](#), seleccione el nombre del archivo .exe de la aplicación.

–O bien–

- En la ventana [Comandos](#), escriba [Ejecutar](#) seguido del nombre del archivo .exe de la aplicación.

–O bien–

- En la línea de comandos que inicia Visual FoxPro, especifique el modificador E. Por ejemplo, si su aplicación se llama MIAPLI, puede ejecutarla con la siguiente línea de comandos:

```
MIAPLI.EXE -E
```

Este modificador de línea de comandos obliga a la aplicación a utilizar el archivo ejecutable Vfp6.exe. Para que este modificador funcione, Vfp6.exe debe estar en la ruta de búsqueda.

## Asegurar el correcto comportamiento en tiempo de ejecución

Una aplicación que sólo contenga formularios o conjuntos de formularios no funcionará correctamente en un entorno de tiempo de ejecución a menos que ofrezca un comando READ EVENTS. Puede asegurar que la aplicación se ejecutará correctamente si agrega un programa que llama o establece la propiedad WindowType.

### Para ejecutar un formulario en un entorno de tiempo de ejecución

- Ejecute el formulario desde un programa que contenga un comando [READ EVENTS](#).

–O bien–

- Establezca como **Modal** la propiedad [WindowType](#).

Algunas aplicaciones de Visual FoxPro se basan en gran medida en los menús de sistema de Visual FoxPro. En tiempo de ejecución, algunos menús y comandos no están disponibles y si no se proporciona un comando READ EVENTS, las aplicaciones controladas por menús terminarán tan rápidamente como se iniciaron. Utilice las secciones siguientes para repasar los menús que incluya en la aplicación.

Para obtener más información sobre la estructuración de una aplicación con el comando READ EVENTS, consulte "Control del bucle de eventos" y "Ejemplos de cómo estructurar una aplicación" en el capítulo 13, [Compilar una aplicación](#).

### Opciones de menú

Si utiliza el menú de sistema de Visual FoxPro, su archivo incluirá solamente los siguientes menús y comandos predeterminados.

Menú	Elementos del menú
Archivo	Cerrar, Guardar, Guardar como, Salir
Edición	Deshacer, Rehacer, Cortar, Copiar, Pegar, Pegado especial, Seleccionar todo, Buscar, Reemplazar
Ventana	Organizar todo, Ocultar, Ocultar todo, Mostrar todo, Borrar, Recorrer,

todas las ventanas abiertas

---

Ayuda

índice, Buscar Ayuda sobre, Asistencia técnica, Acerca de Visual FoxPro

---

Puede desactivar o eliminar cualquiera de los menús o comandos predeterminados, así como agregar sus propios menús y comandos a las aplicaciones de tiempo de ejecución.

**Solución de problemas** Si el sistema de menús funciona en el entorno de programación, pero se cierra prematuramente la aplicación, asegúrese de que tiene un comando [READ EVENTS](#) activo durante la ejecución del sistema de menús. También debe asegurarse de incluir un comando [CLEAR EVENTS](#) cuando salga del sistema de menús.

Para obtener información adicional acerca de la personalización de menús, consulte el capítulo 11, [Diseñar menús y barras de herramientas](#)

## Incluir recursos en sus aplicaciones

Visual FoxPro proporciona varios archivos de recursos que amplían la funcionalidad básica de las aplicaciones, entre los que se incluyen archivos de recursos FOXUSER, bibliotecas API y controles ActiveX. Si utiliza estos archivos, necesitará incluirlos en el árbol del proyecto o de distribución.

### Incluir archivos de recursos FOXUSER

Los archivos de recursos de Visual FoxPro almacenan información útil para las aplicaciones, incluyendo las posiciones de las ventanas, las configuraciones de la ventana Examinar y definiciones de etiquetas. Si su aplicación utiliza cualquiera de estos elementos de recursos, deberá distribuir con ella la base de datos FOXUSER y los archivos memo o los archivos de recursos que cree específicamente para su aplicación. Estos archivos de recursos constan de una tabla de Visual FoxPro asociada a un archivo memo, normalmente denominados Foxuser.dbf y Foxuser.fpt.

**Nota** El archivo de recursos FOXUSER no es el mismo que el archivo de recursos que depende de la configuración regional que contiene cuadros de diálogo y mensajes de error. El archivo de recursos FOXUSER almacena información de aplicación como macros definidas por usted; el archivo de recursos que depende de la configuración regional almacena cadenas de texto del sistema. Para obtener más información, consulte [Incluir un archivo de recursos que depende de la configuración regional](#), más adelante en este mismo capítulo.

### Incluir archivos de biblioteca externa

Si su aplicación incluye archivos de biblioteca externa como controles ActiveX (archivos .ocx) o bibliotecas de API de Visual FoxPro (archivos .fil), use el [Asistente para instalación](#) para asegurarse de que se colocan en el directorio apropiado. Puede distribuir el archivo Foxtools.fil de Visual FoxPro con sus aplicaciones. Para obtener más información sobre la creación de bibliotecas externas para tener acceso a la API de Visual FoxPro, consulte la parte 9, [Acceso a las API](#).

### Incluir componentes COM

Si incluye controles ActiveX o ha creado un servidor de Automatización (un componente COM) como parte de su aplicación, incluya cualquier archivo .ocx en su proyecto y asegúrese de que se

instalan los archivos de soporte necesarios en el equipo del usuario en el directorio System de Windows. Tenga en cuenta que sólo puede distribuir controles ActiveX de los que tenga licencia. Para los servidores de Automatización, también debe incluir archivos de registro, como bibliotecas de tipos (archivos .tlb) y archivos de registro (archivos .vbr) en su aplicación.

Si usa el [Asistente para instalación](#) para crear los discos de distribución, puede incluir estos archivos automáticamente. En el Paso 6, asegúrese de que la columna ActiveX contiene marcas de verificación para los controles ActiveX que va a distribuir. Cuando haga esto, el programa Instalar creado por el Asistente para instalación asegurará que los componentes COM se registran correctamente en el equipo del usuario cuando se instale la aplicación. Para obtener más información sobre el Asistente para instalación, consulte el capítulo 26, [Crear discos de distribución](#).

Todos los usuarios pueden ejecutar formularios que contengan controles ActiveX; sin embargo, su aplicación no puede llevar a cabo ciertas tareas si se ejecuta con la versión de tiempo de ejecución de Visual FoxPro. Recuerde las siguientes instrucciones:

- La aplicación se tiene que ejecutar en una versión completa de Visual FoxPro para modificar formularios, clases o subclases que incluyan controles ActiveX.
- La aplicación debe ejecutarse bajo una versión completa de Visual FoxPro para agregar controles ActiveX a formularios en tiempo de ejecución. Por ejemplo, la versión completa de Visual FoxPro es necesaria para agregar el control Outline a un formulario ejecutando el código siguiente:

```
PUBLIC frmOleNewForm
frmOleNewForm = CREATEOBJECT("form")
frmOleNewForm.Show
frmOleNewForm.ScaleMode = 3
frmOleNewForm.Addobject("NewOutline", "OLEControl", ;
"MSOutl.Outline")
```

**Nota** Cuando se cierra un formulario, los controles agregados en tiempo de ejecución no se guardan.

- Su aplicación se puede ejecutar con la versión de tiempo de ejecución o con la versión completa de Visual FoxPro para agregar subclases de controles ActiveX a un formulario en tiempo de ejecución. Por ejemplo, puede definir la subclase RedOutline a partir de la clase Outline y distribuir la subclase en Olelib.vcx; todos los usuarios pueden agregar entonces el control RedOutline a un formulario ejecutando el código siguiente:

```
PUBLIC frmOleNewForm
frmOleNewForm = CREATEOBJECT("form")
frmOleNewForm.Show
frmOleNewForm.ScaleMode = 3
SET CLASSLIB TO CURR() + OLELIB.VCX
frmOleNewForm.Addobject("NewOutline", "RedOutline")
```

## Incluir un archivo de configuración

El archivo de configuración, Config.fpw, puede establecer numerosos valores predeterminados de Visual FoxPro. Por ejemplo, es posible cambiar el título de Visual FoxPro, su color de fondo y la forma en que el usuario se desplaza mediante el teclado.

Si desea que el archivo de configuración sea de sólo lectura, sitúelo en el proyecto y márkelo como incluido. Si desea que la configuración sea modificable, incluya el archivo en el proyecto y márkelo como excluido. A continuación, distribuya el archivo de configuración con el archivo de aplicación o con el ejecutable, en forma de archivo independiente. En cualquiera de los casos, es necesario que el archivo de configuración se denomine Config.fpw. Sin embargo, puede especificar un nombre de archivo de configuración diferente mediante el modificador de línea de comandos C al iniciar Visual FoxPro.

Para obtener más información acerca de las opciones que puede establecer en el archivo de configuración, consulte el capítulo 3, [Configurar Visual FoxPro](#), en la *Guía de instalación*.

### Incluir un archivo de recursos dependiente de la configuración regional

Si va a distribuir su aplicación con la versión de tiempo de ejecución de Visual FoxPro, es posible que tenga que incluir una *archivo de recursos* que dependa de la configuración regional. Este archivo contiene los cuadros de diálogo y otros elementos de interfaz de usuario que Visual FoxPro usa para interactuar con el usuario. Hay un archivo de recursos de tiempo de ejecución diferente para cada idioma que tenga una versión disponible de Visual FoxPro.

Para obtener más información sobre el uso de archivos de tiempo de ejecución dependientes de la configuración regional, consulte "Distribuir archivos de tiempo de ejecución dependientes de la configuración regional" en el capítulo 18, Programación de aplicaciones internacionales.

**Nota** El archivo de recursos dependiente de la configuración regional no es el mismo que el archivo de recursos FOXUSER, que almacena información de la aplicación, como macros definidas por usted. El archivos de recursos dependiente de la configuración regional almacena cadenas de texto del sistema. Para obtener más información, consulte [Incluir archivos de recursos FOXUSER](#) en una sección anterior de este capítulo.

### Incluir todos los archivos

Puede reproducir y distribuir libremente algunos de los archivos, gráficos y programas de Visual FoxPro con las aplicaciones que cree. Para obtener información detallada, consulte [Quitar características y archivos restringidos de Visual FoxPro](#) más adelante en este capítulo.

Antes de generar una aplicación, asegúrese de que su proyecto incluye los archivos necesarios para la aplicación, así como cualquier archivo de recursos adicional, como pueden ser archivos gráficos o plantillas.

La tabla siguiente muestra los archivos que puede agregar al proyecto.

Si va a	Agregue estos archivos a sus proyectos
Aplicar una configuración personalizada a su aplicación	Config.fpw
Aplicar una configuración personalizada a la aplicación	Foxuser.dbf y Foxuser.fpt



---

Distribuir un archivo de Ayuda de tipo .dbf

El archivo de Ayuda de tipo .dbf

---

### Para agregar archivos a su aplicación

- Incluya los archivos en el proyecto.

Si no desea modificarlos en la aplicación distribuida, sitúelos en el proyecto y márkelos como incluidos. Estos archivos serán de sólo lectura y no admitirán cambios.

–O bien–

- Agregue los archivos al directorio de la aplicación. Para obtener detalles, consulte el capítulo 26, [Crear discos de distribución](#).

Si desea modificar los archivos, sitúelos dentro del proyecto y márkelos como excluidos. A continuación, distribúyalos con su aplicación como archivos independientes.

Para obtener más información acerca de la creación de un proyecto, y de la inclusión o exclusión de archivos en un proyecto, consulte el capítulo 13, [Compilar una aplicación](#).

### Quitar características y archivos restringidos de Visual FoxPro

El entorno de programación de Visual FoxPro contiene numerosas características y muchos archivos con licencia para su uso exclusivo. Si su aplicación contiene alguna de estas características o archivos, elimínelos.

#### Características restringidas de Visual FoxPro

No se permite incluir los siguientes menús de Visual FoxPro, ni sus comandos, en un archivo ejecutable distribuido.

#### Menús restringidos

Base de datos	Proyecto
Formulario	Consulta
Menú	Tabla
Programa	

Si su aplicación incluye alguno de los siguientes comandos, devolverá el error "Característica no disponible". Aunque no se permite incluir comandos que creen o modifiquen menús, formularios ni consultas, sí es posible ejecutar programas compilados de menú, formulario o consulta en la aplicación.

**Comandos no disponibles**

<a href="#">BUILD APP</a>	<a href="#">MODIFY FORM</a>
<a href="#">BUILD EXE</a>	<a href="#">MODIFY MENU</a>
<a href="#">BUILD PROJECT</a>	<a href="#">MODIFY PROCEDURE</a>
<a href="#">COMPILE</a>	<a href="#">MODIFY PROJECT</a>
<a href="#">CREATE FORM</a>	<a href="#">MODIFY QUERY</a>
<a href="#">CREATE MENU</a>	<a href="#">MODIFY SCREEN</a>
<a href="#">CREATE QUERY</a>	<a href="#">MODIFY STRUCTURE</a>
<a href="#">CREATE SCREEN</a>	<a href="#">MODIFY VIEW</a>
<a href="#">CREATE VIEW</a>	<a href="#">SUSPEND</a>
<a href="#">MODIFY CONNECTION</a>	<a href="#">SET STEP</a>
<a href="#">MODIFY DATABASE</a>	

Los comandos siguientes se pasarán por alto cuando se utilicen en una aplicación distribuida.

**Comandos que se pasan por alto**

<a href="#">SET DEBUG</a>	<a href="#">SET DOHISTORY</a>
<a href="#">SET DEVELOPMENT</a>	<a href="#">SET ECHO</a>

**Archivos restringidos de Visual FoxPro**

Visual FoxPro instala en el PC archivos que están restringidos y no se pueden reproducir ni distribuir. Entre ellos se incluyen:

- Archivos de asistentes
- Fuentes TrueType
- Archivos de la utilidad SpellCheck
- Archivos de Ayuda

Aunque no se permite distribuir las aplicaciones de ejemplo de Visual FoxPro con sus aplicaciones, puede hacer referencia a partes del código de estas aplicaciones como ejemplos para generar su propia aplicación. También puede incluir en su aplicación la biblioteca de clases de asistentes Wizstyle.vcx y las bibliotecas de clases de ejemplos.

**License.txt**

Visual FoxPro contiene muchos archivos que se conceden en licencia para su uso exclusivamente en tareas de diseño, programación y prueba. Vea License.txt, que se encuentra en el directorio Visual

FoxPro, para obtener una lista de archivos restringidos.

Si su aplicación contiene cualquiera de estos archivos, quítelo. Bajo los términos de Contrato de licencia de Microsoft que ha recibido con este producto, no puede distribuir estos archivos en su aplicación ni en sus discos.

### **Archivos distribuibles**

Puede distribuir cualquier archivo de Visual FoxPro que no esté restringido. Según el Contrato de licencia que ha recibido con este producto, los archivos se pueden distribuir con la aplicación correspondiente. Las instrucciones siguientes se aplican a archivos distribuibles.

### **Asistente para instalación**

El [Asistente para instalación](#) comprueba si hay archivos restringidos y los excluye del conjunto de discos distribuibles. No asigne estos nombres de archivo a ninguno de los archivos que vaya a distribuir. El Asistente para instalación excluirá cualquier archivo que tenga un nombre idéntico a alguno de los de la lista.

Se puede distribuir cualquier archivo de los directorios Distrib.src y SETUP de Visual FoxPro necesarios para una aplicación. Cuando utilice el [Asistente para instalación](#) para crear discos de distribución, éste colocará automáticamente los archivos necesarios de estos directorios en los discos distribuibles, en un formato comprimido. Después de la instalación, estos archivos se descomprimen y se instalan con su nombre en los directorios apropiados del equipo del usuario. No es necesario copiar estos archivos en su árbol de distribución.

### **Ejemplos**

Los archivos de las carpetas ...\\Samples\\Vfp98 y Vfp98\\Api\\Samples de Visual Studio le servirán para aprender y para obtener ideas para sus aplicaciones. Aunque no puede distribuir código de [aplicaciones de ejemplo](#) Visual FoxPro sin modificar, puede consultar partes de código de aplicaciones de ejemplo para crear su propia aplicación.

Si utiliza cualquiera de los archivos de estos directorios (incluyendo todos los archivos .bmp, .ico y .cur), debe incluirlos en su proyecto y al generar la aplicación. No pueden aparecer con su nombre en los discos distribuibles y no se pueden distribuir independientemente de sus aplicaciones.

### **Bibliotecas de clases**

Puede utilizar sin modificación en sus aplicaciones cualquier archivo.vcx, incluyendo los de los directorios Vfp98\\Ffc y Vfp98\\Gallery. Debe incluir las bibliotecas en su proyecto y en la aplicación generada.

### **Archivos ODBC**

Consulte el Contrato de licencia de Microsoft que recibió con este producto para ver restricciones específicas relativas a la redistribución de archivos ODBC.

### **Controles ActiveX**

Visual FoxPro incluye un conjunto de controles ActiveX (archivos .ocx) que puede agregar a sus aplicaciones y distribuirlos con ellas.

## Personalizar una aplicación para su distribución

El entorno de tiempo de ejecución predeterminado de Visual FoxPro tiene la misma apariencia que el entorno de programación: muestra los iconos y menús de Visual FoxPro. Probablemente deseará proporcionar a su aplicación una apariencia exclusiva, personalizando alguna de sus características con los siguientes métodos:

- Protección y documentación del código fuente.
- Llamada a rutinas de tratamiento de errores y cierre de sesión.
- Modificación de los menús y comandos predeterminados de Visual FoxPro.
- Inclusión de un archivo de configuración para especificar configuraciones personalizadas para el título, los iconos, el teclado y la Ayuda.
- Modificación de la ventana principal de Visual FoxPro.
- Adición de Ayuda a la aplicación.

### Protección y documentación del código fuente

Para evitar que los usuarios vean o modifiquen el código fuente de su aplicación, codifique dicho código y elimine la información de depuración.

**Sugerencia** Haga siempre una copia de seguridad del código fuente antes de codificarlo.

#### Para proteger el código fuente

1. Abra el proyecto de su aplicación.
2. En el menú **Proyecto**, elija **Información del proyecto**.
3. En el cuadro de diálogo [Información del proyecto](#), seleccione **Codificado** y desactive **Información de depuración**.
4. En el [Administrador de proyectos](#), elija **Generar**.
5. En el cuadro de diálogo [Opciones para generar](#), seleccione **Volver a compilar todos los archivos** y elija **Aceptar**.
6. En la sección de instalación de la aplicación, incluya el comando [SET DEBUG OFF](#).

Antes de comenzar el proceso de distribución, si no lo ha hecho todavía, puede comentar y dar formato al código para que tenga una apariencia coherente y su mantenimiento sea más sencillo. Puede usar la opción [Presentación](#) del menú Herramientas o el [Asistente para documentación](#) para personalizar la documentación de la aplicación de varias maneras, entre las que se incluyen:

- Definición de mayúsculas para palabras clave y variables.

- Definición de sangrías en el código fuente.
- Adición de encabezados a los archivos, procedimientos y métodos.

### Para utilizar el Asistente para documentación

1. En el menú **Herramientas**, elija **Asistentes**.
2. En el submenú que aparecerá, elija **Documentación**.

También puede utilizar el [Asistente para documentación](#) para crear referencias cruzadas de los símbolos que ha utilizado en su aplicación y para producir un resumen analítico del proyecto.

### Llamada a rutinas de tratamiento de errores y cierre de sesión

A veces surgen errores cuando los usuarios ejecutan su aplicación. Puede llamar a su propia rutina de tratamiento de errores incluyendo [ON ERROR](#). Normalmente, ON ERROR utiliza un comando [DO](#) para ejecutar una rutina que trata el error, como en este ejemplo:

```
ON ERROR DO Mi_Error
```

Si su aplicación no contiene ninguna rutina de tratamiento de errores y se produce un error, la aplicación se interrumpirá y Visual FoxPro presentará un mensaje de error con las siguientes opciones:

- **Cancelar** Si el usuario elige "Cancelar", Visual FoxPro detendrá inmediatamente la ejecución de la aplicación y devolverá el control al sistema.
- **Ignorar** Si el usuario elige "Ignorar", Visual FoxPro pasará por alto la línea que originó el error y continuará con la línea siguiente del programa.

Para obtener información adicional acerca del tratamiento de errores, consulte "Controlar errores en tiempo de ejecución" en el capítulo 14, [Probar y depurar aplicaciones](#)

Para ver una lista completa de los mensajes de error de Visual FoxPro y una explicación de los mismos, vea [Mensajes de error](#).

**Sugerencia** Asegúrese de proporcionar documentación a los usuarios que describa los errores que sean visibles y que sugiera posibles formas de corregirlos.

Puede crear su propia rutina de cierre de sesión si incluye el comando [ON SHUTDOWN](#) en el código. Normalmente, ON SHUTDOWN utiliza un comando DO para llamar a una rutina si intenta salir de la aplicación, como en este ejemplo:

```
ON SHUTDOWN DO Mi_Cierre
```

Esta rutina suele incluir un cuadro de diálogo que pregunta al usuario si realmente desea salir de la aplicación actual. Si el usuario desea salir de la aplicación, la rutina puede cerrar los archivos abiertos y limpiar el entorno y posteriormente ejecutar el comando [QUIT](#). Si el usuario no desea salir de la aplicación actual, la rutina puede devolver el control a la aplicación.

## Agregar Ayuda a la aplicación

Puede integrar Ayuda interactiva en sus aplicaciones, de forma que los usuarios puedan presionar F1 o elegir la Ayuda en un menú para obtener ayuda para su aplicación. La Ayuda que proporciona con su aplicación tiene las mismas características que la Ayuda de Visual FoxPro. Para obtener más información, consulte la parte 7, [Crear archivos de Ayuda](#).

Si crea Ayuda gráfica para su aplicación, incluya el archivo .chm o .hlp en el directorio de distribución de la aplicación de forma que el Asistente para instalación lo incluya en sus discos de distribución.

**Nota** No puede distribuir Winhelp.exe o los archivos de Ayuda distribuidos con Visual FoxPro. Para obtener más información, vea [Quitar características y archivos restringidos de Visual FoxPro](#), más adelante en este capítulo

## Modificar la apariencia de la aplicación

Puede modificar la apariencia de su aplicación sin necesidad de cambiar su código, mediante los métodos siguientes:

- Cambiar el sistema de menús predeterminado.
- Cambiar el título predeterminado.
- Cambiar el icono predeterminado de aplicación.
- Especificar el desplazamiento por teclado específico para la plataforma.

### Cambiar los menús predeterminados de Visual FoxPro

Puede agregar sus propios menús y comandos de menú a las aplicaciones distribuidas, empleando para ello el Diseñador de menús. Si no crea ningún menú propio, el entorno de tiempo de ejecución mostrará un menú predeterminado de Visual FoxPro.

Para obtener detalles acerca de los menús predeterminados, consulte [Asegurar el correcto comportamiento en tiempo de ejecución](#), en una sección anterior de este capítulo. Para obtener más información acerca del Diseñador de menús, consulte el capítulo 11, [Diseñar menús y barras de herramientas](#).

### Cambiar el título predeterminado

La aplicación se ejecuta en la ventana principal de Visual FoxPro. El texto "Microsoft Visual FoxPro" aparece de forma predeterminada en la barra de título.

### Para personalizar el título de la ventana principal de Visual FoxPro

- Agregue la siguiente instrucción a su archivo de configuración:

```
TITLE = cMiTítulo
```

Sustituya cMiTítulo por el título de la ventana principal de su aplicación.

Para incluir una función de Visual FoxPro como parte del título, use la propiedad [Caption](#) de la ventana principal como se ilustra en el siguiente ejemplo.

```
COMMAND=_SCREEN.Caption=;  
"Visual FoxPro " + SUBSTR(VERSION(),25,3)
```

### **Cambiar el icono predeterminado de aplicación**

Una vez compilada la aplicación, el icono predeterminado de Visual FoxPro aparecerá en el Explorador de Windows o el menú Inicio como icono para la aplicación. Puede utilizar el icono genérico suministrado por Visual FoxPro o bien diseñar uno propio.

Si quiere mostrar su propio icono, cree un archivo de icono (.ico) con dos imágenes: una pequeña (16 por 16) y una estándar (32 por 32). Cree las dos imágenes como iconos de 16 colores.

Puede cambiar el icono predeterminado de Visual FoxPro en el cuadro de diálogo Información del proyecto. Si usa el Asistente para instalación para crear discos de distribución para su aplicación, también puede especificar un icono de aplicación en este cuadro de diálogo.

### **Para cambiar el icono de aplicación predeterminado mediante el Administrador de proyectos**

1. En el [Administrador de proyectos](#), seleccione el [archivo principal](#) para su proyecto.
2. En el menú **Proyecto**, elija **Información del proyecto** y, a continuación, seleccione la ficha **Proyecto**.
3. Elija **Adjuntar icono**.
4. Elija **Icono** y, a continuación, seleccione un archivo de icono (.ico) para asignar a su proyecto.

### **Copia de seguridad del código fuente**

En la programación de toda aplicación, es recomendable realizar copias de seguridad completas de los archivos originales de programa antes de generar una aplicación. Almacene las copias de seguridad en un sitio distinto que las aplicaciones compiladas.

**Importante** Asegúrese de conservar copias independientes de los programas originales para su uso posterior. No es posible volver a crear los programas de origen a partir del código compilado.

### **Generar la aplicación**

Una vez que el proyecto de aplicación contenga todos los archivos necesarios, estará en condiciones de generar un archivo distribuible. Puede generar su proyecto como una aplicación estándar que solamente se ejecuta cuando se dispone de Visual FoxPro, o bien como una aplicación ejecutable que funcione sin Visual FoxPro.

**Nota** También puede generar su aplicación como servidor de Automatización OLE. Para ver más detalles, consulte, [Crear servidores de Automatización](#), en el Capítulo 16, "Agregar OLE".

## Generar una aplicación estándar de Visual FoxPro

Puede generar una aplicación estándar de Visual FoxPro con el Administrador de proyectos o bien con el comando BUILD APP. No obstante, el Administrador de proyectos pone a su disposición más opciones para la generación de aplicaciones.

### Para generar una aplicación estándar

- En el [Administrador de proyectos](#), elija **Generar**. En el cuadro de diálogo [Opciones para generar](#), elija **Generar aplicación**.

–O bien–

- Utilice el comando [BUILD APP](#).

### Generar un archivo ejecutable

Puede generar un archivo ejecutable de Visual FoxPro con el Administrador de proyectos o con el comando BUILD EXE.

### Para generar un archivo ejecutable

- En el [Administrador de proyectos](#), elija **Generar**. En el cuadro de diálogo [Opciones para generar](#), elija **Generar ejecutable**.

–O bien–

- Utilice el comando [BUILD EXE](#).

Si incluye las cláusulas STANDALONE o EXTENDED del comando BUILD EXE, Visual FoxPro generará un mensaje de error "Característica no disponible".

También puede generar un servidor de Automatización, que crea una DLL que se puede llamar desde otros programas para Windows.

### Para compilar un servidor de Automatización

- En el [Administrador de proyectos](#), elija **Generar**. En el cuadro de diálogo [Opciones para generar](#), elija **Generar DLL OLE**.

–O bien–

- Utilice el comando [BUILD DLL](#).

## Preparar la creación de discos de distribución

Ahora que ha tenido en cuenta todos los requisitos y opciones proporcionados por Visual FoxPro y



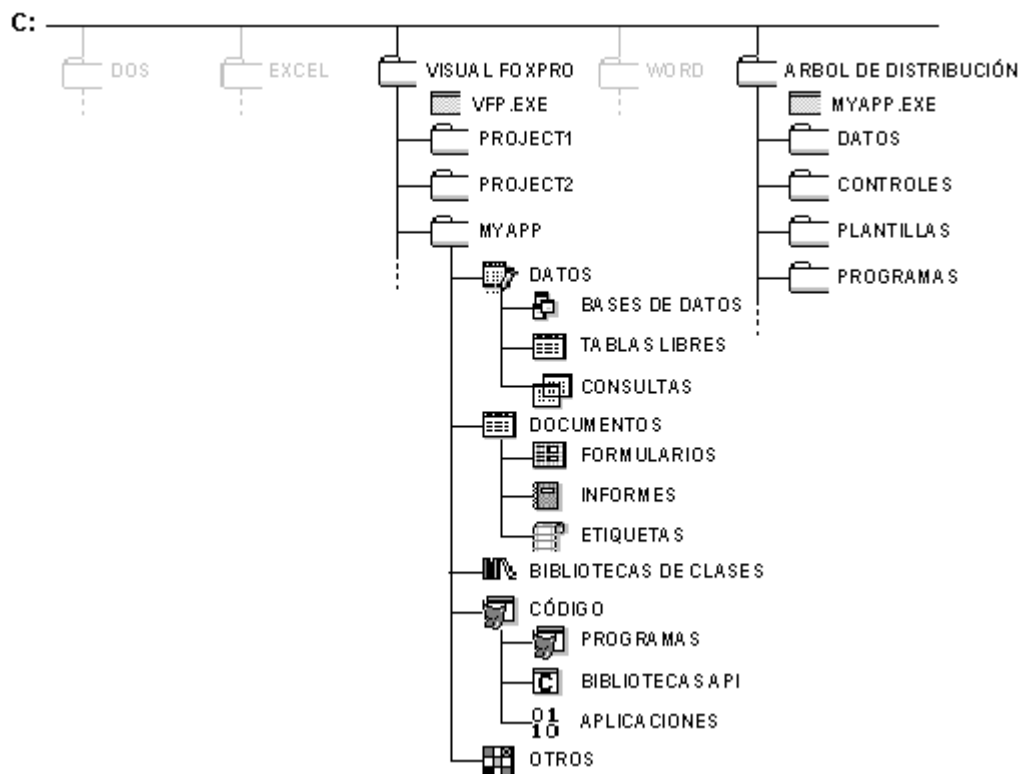
que ha generado una aplicación a partir de sus archivos, siga estos pasos:

- Cree un directorio de distribución.
- Copie los archivos de aplicación desde el proyecto hasta las ubicaciones adecuadas del directorio de distribución.
- Cree los discos de distribución.

## Crear un directorio de distribución

El directorio de distribución contiene copias de todos los archivos del proyecto que constituyen su aplicación. La estructura de este *árbol de distribución* representa la forma en que la rutina de Instalación creada por el Asistente para instalar instalará los archivos en el equipo de un usuario.

## Asignar los archivos del proyecto al árbol de distribución



## Para crear el directorio de distribución

1. Cree un directorio con el nombre que desee que aparezca en el equipo del usuario.
2. Divida el directorio de distribución en los subdirectorios adecuados para su aplicación.
3. Copie al directorio los archivos del proyecto de la aplicación.

Puede utilizar este directorio para comprobar su aplicación en el entorno de tiempo de ejecución. Si es necesario, restablezca temporalmente los valores predeterminados en el equipo en el que ha programado la aplicación para reflejar la configuración mínima de un equipo de destino de usuario. Cuando todo funcione correctamente, use el Asistente para instalar con el fin de crear imágenes de

disco que reproducirán el entorno correcto cuando distribuya copias de la aplicación.

## Crear los discos de distribución

Para crear los discos de distribución, utilice el [Asistente para instalación](#). El Asistente para instalación comprime los archivos en el árbol de distribución y copia los archivos comprimidos al directorio imagen de los discos, colocándolos en un subdirectorío distinto para cada disco. Después de usar el Asistente para instalación para crear imágenes de sus discos de aplicación, copie el contenido de cada directorio imagen de disco a un disco distinto.

Al distribuir el paquete, el usuario puede instalar todos los archivos de la aplicación al ejecutar SETUP.EXE desde el disco 1.

Para obtener detalles acerca del uso del Asistente para instalación, consulte el capítulo 26, [Crear discos de distribución](#).

# Capítulo 26: Crear discos de distribución

Después de haber diseñado y probado una aplicación, puede usar el Asistente para instalación para crear una rutina de instalación y discos de distribución para su aplicación. Si pretende distribuir su aplicación en más de un formato de disco, el Asistente para instalación crea rutinas y discos para todos los formatos que especifique.

Para obtener información detallada sobre cómo preparar una aplicación para su distribución, consulte el capítulo 25, [Generar una aplicación para su distribución](#). Para obtener más información sobre cómo crear una aplicación, consulte el capítulo 13, [Compilar una aplicación](#).

Este capítulo explica cómo crear discos de distribución:

- [Descripción del proceso de distribución](#)
- [Usar el Asistente para instalación](#)

## Descripción del proceso de distribución

Cuando distribuye una aplicación, copia toda la aplicación y los archivos de soporte a un medio de distribución, normalmente discos y proporciona un método para que los usuarios instalen la aplicación en sus equipos. Como copiar e instalar los archivos apropiados puede ser complicado, use el Administrador de proyectos y el Asistente para instalación para racionalizar este proceso.

En el [Administrador de proyectos](#), cree y administre sus archivos de aplicación e identifique los archivos que desea distribuir.

Con el [Asistente para instalación](#), cree uno o más conjuntos de discos distribuibles que contienen una rutina de instalación para la aplicación. Para simplificar la tarea de crear una rutina de instalación, el Asistente para instalación le hace una serie de preguntas sobre la aplicación y qué apariencia le gustaría que tuviera la rutina de instalación. Cuando responda a las preguntas, el Asistente para instalación creará una rutina de instalación personalizada.

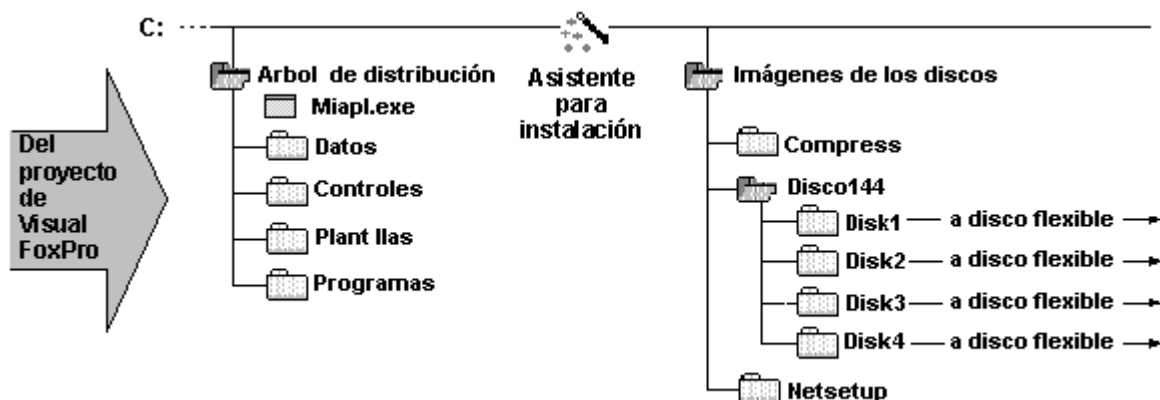
Cada vez que ejecuta el Asistente para instalación, se registran las opciones que seleccione para el árbol de distribución. De esta forma, la próxima vez que ejecute el Asistente para instalación, el proceso es aún más sencillo.

**Nota** Si sólo copia los archivos de aplicación al equipo de un usuario, es posible que la aplicación no funcione de forma apropiada. Las rutinas de instalación de Windows, como la creada por el Asistente para instalación, requiere la comprobación de versiones y el registro de varios archivos DLL y ActiveX. Para asegurar una instalación correcta, use el [Asistente para instalación](#). Para obtener más información, consulte [El asistente para instalación](#), más adelante en este mismo capítulo.

## El árbol de distribución

Antes de crear discos con el Asistente para instalación, tiene que crear una estructura de directorios o árbol de distribución, que contenga todos los archivos de distribución como le gustaría que apareciesen en el disco duro del usuario. Coloque en el árbol de distribución todos los archivos que quiera que estén en los discos de distribución.

### Asignar el árbol de distribución a las imágenes de los discos



El árbol de distribución puede adoptar casi cualquier formato. Sin embargo, el archivo de aplicación o ejecutable debe estar en el directorio raíz del árbol.

Muchas aplicaciones de Visual FoxPro requieren archivos de recursos adicionales. Por ejemplo, es posible que desee incluir un archivo de configuración o un archivo de Ayuda. Si tiene que agregar un archivo de recursos y no lo ha incluido en su proyecto, coloque el archivo en la estructura de directorios de la aplicación.

La tabla siguiente muestra algunos archivos típicos que se colocan en el directorio de la aplicación.

Si	Agregue estos archivos al directorio de la aplicación
Aplica una configuración personalizada a la aplicación	Config.fpw u otro archivo de configuración
Suministra una configuración personalizada a la aplicación	Foxuser.dbf y Foxuser.fpt
Distribuye fuentes de Visual FoxPro	Foxfont Foxprint
Distribuye una biblioteca de soporte	<i>NombreBiblioteca.ocx o NombreBiblioteca.fl</i>
Incluye un archivo de recursos que depende de la configuración regional	Vfp6raaa.dll, en donde “aaa” es el código de idioma de tres letras.

Cuando ejecuta el Asistente para instalación, crea un directorio de distribución independiente para cada formato de disco que especifique. Estos directorios contienen todos los archivos necesarios para las imágenes de disco.

Por ejemplo, si especifica imágenes de discos de 1,44 MB y 1,2 MB, el Asistente para instalación crea dos subdirectorios llamado DISK144 y NETSETUP. Si su aplicación requiere cuatro discos de distribución, el Asistente para instalación crea cuatro subdirectorios llamados DISK1, DISK2, DISK3 y DISK4 en el directorio DISK144.

**Importante** Como el Asistente para instalación crea dos subdirectorios nuevos en el disco duro, asegúrese de tener espacio de disco suficiente para tres copias comprimidas de la aplicación.

## El Asistente para instalación

El [Asistente para instalación](#) crea una rutina de instalación para la aplicación, que incluye un archivo Instalar.exe, algunos archivos de información, y los archivos comprimidos o descomprimidos de la aplicación (guardados en archivos .cab). El resultado final es un conjunto de archivos que puede copiar en disquetes, en la red o en un sitio web. Entonces los usuarios pueden instalar la aplicación de la misma forma que instalan cualquier aplicación para Windows. Mientras estén instalando la aplicación, los usuarios verán opciones que usted especifica al usar el Asistente para instalación.

Después de crear el árbol de distribución, use el Asistente para instalación para crear un conjunto de subdirectorios de imágenes de disco que contengan todos los archivos necesarios para instalar la aplicación. Copie los archivos de estos subdirectorios para crear discos de distribución para la aplicación.

El Asistente para instalación realiza los pasos siguientes:

1. Crea un archivo llamado Wzsetup.ini que contiene sus elecciones de Asistente para instalaciones para este árbol de distribución.
2. Se asegura de que todos los archivos necesarios para ejecutar su aplicación distribuida están

incluidos en la aplicación.

3. Copia los archivos comprimidos a subdirectorios que crea en el directorio disco de distribución.
4. Crea archivos de instalación en los directorios de imágenes especificados, incluyendo Setup.inf y Setup.stf, que especifican los parámetros de instalación para la rutina de instalación.
5. Crea archivos llamados Dkcontrl.dbf y Dkcontrl.cdx en el árbol de distribución. Estos archivos contienen información estadística sobre cómo se han comprimido, y asignado a subdirectorios de disco los archivos.

## Usar el Asistente para instalación

Use el [Asistente para instalación](#) para crear discos de distribución a partir de los archivos del árbol de distribución. El Asistente para instalación le permite crear nuevas rutinas de instalación o usar información del árbol de distribución como valores predeterminados.

El Asistente para instalación requiere un directorio de trabajo llamado Distrib.src. Si ésta es la primera vez que usa el Asistente para instalación o si por alguna razón el directorio Distrib.src está en una ubicación diferente de la que está buscando el Asistente para instalación, verá un mensaje que le indica que no se puede encontrar el directorio.

### Para crear una rutina de instalación y un directorio de distribución

1. En el menú **Herramientas**, elija **Asistentes**.
2. En el submenú **Asistentes**, elija **Instalación**.
3. Si el Asistente para instalación le pide que cree o busque el directorio Distrib.src, confirme que desea crearlo o elija **Buscar** y especifique la ubicación del directorio.

Para obtener detalles sobre las opciones disponibles en cada pantalla del Asistente para instalación, haga clic en el botón Ayuda de la pantalla o presione F1.

### Especificar el árbol de distribución

Para especificar el árbol de distribución, use el [Paso 1](#) del Asistente para instalación. Debería identificar el directorio del árbol de distribución que creó para modelar la instalación por parte del usuario de sus archivos de aplicación.

El Asistente para instalación espera que el directorio especificado contenga todos los archivos y subdirectorios que desea crear en un entorno de usuario. El Asistente para instalación usa este directorio como origen para archivos que comprimirá en el directorio imagen de disco.

El Asistente para instalación registra las opciones que establezca para cada árbol de distribución y los usa como valores predeterminados la próxima vez que desee crear una rutina de instalación a partir del mismo árbol de distribución.

## Seleccionar componentes opcionales

Para especificar los componentes opcionales que su aplicación usa o con los que es compatible, use el [Paso 2](#) del Asistente para instalación. Por ejemplo, si desea distribuir su aplicación con la versión de tiempo de ejecución (run-time) de Visual FoxPro, elija Visual FoxPro Runtime de forma que el Asistente para instalación incluya el archivo de soporte runtime (Vfp6r.dll). Si distribuye su aplicación como servidor de Automatización, elija esa opción.

**Nota** Si su aplicación incluye un servidor de Automatización, el programa Instalar se registrará automáticamente en el equipo del usuario cuando éste instale la aplicación.

## Especificar imágenes de disco

Para especificar los distintos tipos de disco desde los que se puede cargar su aplicación, use el [Paso 3](#) del Asistente para instalación.

El Asistente para instalación le pide que especifique los distintos tipos de disco desde los que se puede cargar la aplicación. Puede elegir cualquiera de las siguientes opciones:

- Discos de 1,44 MB (3,5 pulgadas).
- Compress Wetsetup.
- Instalación a partir de la red (Netsetup).

El Asistente para instalación también le pide el nombre de un subdirectorio de distribución que contenga las imágenes de disco para cada tipo de disco especificado. Puede crear el directorio de imagen de disco antes de ejecutar el Asistente para instalación o puede dejar que el Asistente para instalación cree este directorio.

Si selecciona la opción Netsetup, el Asistente para instalación crea un único directorio que contendrá todos los archivos.

## Personalizar cuadros de diálogo de instalación distribuida

Para personalizar los cuadros de diálogo de instalación distribuida, use el [Paso 4](#) del Asistente para instalación.

El Asistente para instalación le pedirá que especifique los títulos de los cuadros de diálogo de la rutina de instalación y el contenido del copyright.

El Asistente para instalación crea cuadros de diálogo de instalación con los títulos especificados.

## Especificar una acción postinstalación

Para especificar un programa o acción que Instalar debería ejecutar cuando la instalación esté terminada, use el [Paso 4](#) del Asistente para instalación. Las acciones típicas postinstalación pueden ser mostrar un archivo léame o ejecutar el proceso de instalación para un producto relacionado.

El Asistente para instalación le pide que especifique el nombre del ejecutable. Escriba la línea de comandos completa necesaria para ejecutar el ejecutable, incluyendo la ruta de acceso completa del ejecutable, el nombre de los archivos que hay que pasar al programa y cualquier cambio de línea de comandos.

**Nota** El programa que especifique debe existir en el sistema del usuario o se producirá un error.

## Identificar los destinos predeterminados de archivos

Para identificar los destinos predeterminados para los archivos de la aplicación, use el [Paso 5](#) del Asistente para instalación.

El Asistente para instalación le pedirá que especifique:

- El nombre predeterminado del directorio en el que el programa de instalación colocará la aplicación en el equipo del usuario.
- El grupo de programas predeterminado en el que colocar el icono de inicio para su aplicación en el equipo del usuario.

La rutina de instalación colocará la aplicación en el directorio que especifique y el icono predeterminado de la aplicación (o cualquier icono que especifique) en el grupo de programas especificado.

Puede especificar si la rutina de instalación terminada permitirá al usuario modificar el grupo de programas predeterminado o modificar el directorio de destino predeterminado y el grupo de programas.

## Revisar las propiedades del archivo

Para mostrar los resultados de todas sus elecciones, use el [Paso 6](#) del Asistente para instalación.

El Asistente para instalación muestra un resumen de los archivos y el resultado de todas las elecciones y le permite modificar los nombres de los archivos, los destinos de archivos y otras especificaciones.

## Terminar el proceso del Asistente para instalación

Para empezar a crear la rutina de instalación para la aplicación, elija Terminar en el [Paso 7](#) del Asistente para instalación. En este paso también puede crear un archivo de dependencias (.dep) que le permitirá usar otras utilidades de instalación para instalar su aplicación.

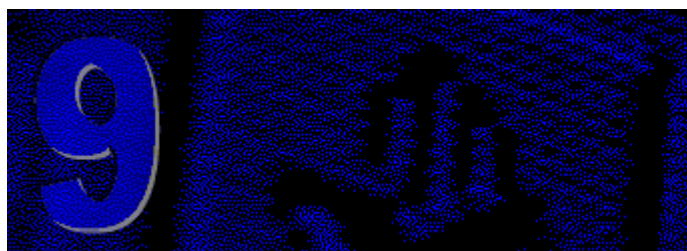
Cuando elija Finalizar, el Asistente para instalación realizará los pasos siguientes:

- Registrará la configuración para usar la próxima vez que cree discos de distribución a partir del mismo árbol de distribución.
- Iniciará el proceso de crear las imágenes de los discos de aplicación.

Cuando el Asistente para instalación haya creado las imágenes de disco especificadas, puede copiarlas a discos maestros y, a continuación, copiar y combinar los discos con el resto del paquete de distribución. Después de crear un conjunto de discos maestros, puede eliminar los directorios de las imágenes de los discos.



# Manual del programador, Parte 9: Acceso a las bibliotecas API



Si su aplicación exige requisitos que no pueden cumplir las características integradas en Visual FoxPro, puede ampliar el programa con bibliotecas externas, controles ActiveX o bibliotecas de vínculos dinámicos (DLL). Gracias a las bibliotecas externas podrá agregar objetos a su aplicación, desde cuadros de texto mejorados hasta calendarios y otras aplicaciones completas, y también aprovechar la funcionalidad ofrecida por otros programas (como Windows) a través de sus interfaces de programación de aplicaciones (API).

## Capítulo 27 [Extender Visual FoxPro con bibliotecas externas](#)

Puede agregar fácilmente controles ActiveX (archivos .ocx) a su aplicación, lo que le ofrece nuevos objetos que puede usar en formularios para crear subclases y administrar su forma de trabajar con controles nativos de Visual FoxPro. Además, puede vincular una biblioteca externa como una DLL y llamar a las funciones de la biblioteca para usarlas en sus propios programas. Si ya tiene bibliotecas externas de Visual FoxPro (archivos .flx), también las puede vincular y llamar a sus funciones.

## Capítulo 28 [Acceso a la API de Visual FoxPro](#)

Si no encuentra una biblioteca externa adecuada, puede escribir su propio control ActiveX o biblioteca de vínculos dinámicos (archivo .flx) específicos de Visual FoxPro. Puede llamar a las funciones disponibles en la API de Visual FoxPro y crear controles o bibliotecas muy integradas con Visual FoxPro y optimizadas para su uso.

## Capítulo 27: Extender Visual FoxPro con bibliotecas externas

Puede extender las posibilidades nativas de Visual FoxPro aprovechando las facilidades de los controles ActiveX (archivos .ocx), objetos ActiveX y las bibliotecas de vínculos dinámicos (DLL). Las bibliotecas externas le permiten tener acceso no sólo a las posibilidades de otros programas, sino a las del mismo Windows. Por ejemplo, puede usar un control ActiveX para leer y actualizar el registro de Windows directamente o llamar a las funciones a nivel de sistema vinculándose a una de las DLL de Windows.

Si la funcionalidad necesaria no está aún disponible en una biblioteca externa, puede crear su propio control ActiveX en C++ mediante un compilador de 32 bits como Visual C++<sup>®</sup> versión 4.0 o

posterior o con Microsoft Visual Basic® Control Creation Edition versión 5.0. Para obtener detalles, consulte el capítulo 28, [Acceso a la API de Visual FoxPro](#).

En este capítulo se describe:

- [Usar bibliotecas externas](#)
- [Acceso a controles ActiveX](#)
- [Acceso a bibliotecas de vínculos dinámicos](#)
- [Acceso a la biblioteca de Visual FoxPro](#)

## Usar bibliotecas externas

En la mayor parte de los casos, Visual FoxPro proporciona todas las herramientas necesarias para terminar la aplicación. Sin embargo, ocasionalmente verá que la aplicación necesita funcionalidad adicional que aún no está disponible en Visual FoxPro. En estos casos, puede salir de Visual FoxPro y aprovechar las posibilidades de las bibliotecas externas.

Visual FoxPro le permite tener acceso a estos tipos de bibliotecas externas:

- **Controles ActiveX (archivos .ocx).** Los controles y objetos ActiveX son programas que incluyen objetos diseñados para realizar tareas específicas. La mayor parte de los controles y objetos ActiveX agregan nuevos objetos a Visual FoxPro, todo desde un nuevo tipo de cuadro de texto hasta un calendario, una calculadora u otro objeto complejo. Algunos controles y objetos ActiveX también incorporan funcionalidades adicionales, como acceso al sistema de correo electrónico o al puerto de comunicaciones de su equipo. Como regla, después de incorporar un control u objeto ActiveX a Visual FoxPro, puede usar los objetos que contienen igual que lo haría con cualquier clase de base de Visual FoxPro.
- **Bibliotecas de vínculos dinámicos (archivos .dll).** Un archivo .dll es una biblioteca de funciones que puede llamar desde programas de Visual FoxPro de la misma manera que cualquier función definida por el usuario de Visual FoxPro. Muchos programas para Windows y el mismo Windows, ofrecen su funcionalidad mediante archivos .dll. Por ejemplo, puede tener acceso a la configuración de colores del sistema para Windows vinculándose a un archivo .dll del sistema y llamando a funciones del mismo.
- **Bibliotecas externas de Visual FoxPro (archivos .fll).** Un archivo .fll es como un archivo .dll, pero usa un protocolo especial para compartir datos con Visual FoxPro y, a menudo, llama a funciones internas de Visual FoxPro. Como consecuencia, los archivos .fll son específicos de Visual FoxPro, a diferencia de los archivos .dll, que se pueden llamar desde cualquier programa para Windows. Puede llamar a las funciones de un .fll igual que a cualquier función definida por el usuario de Visual FoxPro.

Antes de usar cualquier biblioteca, tiene que estar familiarizado con las convenciones usadas para tener acceso a sus controles o funciones. Por ejemplo, si desea incluir un control ActiveX en un formulario, debe saber qué propiedades, eventos y métodos puede usar para administrar el control. Para un control ActiveX, puede usar un Explorador de clases de Visual FoxPro para determinar las propiedades, eventos y métodos que puede utilizar. De forma similar, si desea llamar a una función de un archivo .dll, tiene que saber el nombre de función, el número y el tipo de datos del parámetro que requiere, y el tipo de datos del valor devuelto. En general, puede obtener este tipo de información de la documentación que acompaña a una biblioteca, ya sea un libro o un sistema de Ayuda. Para

obtener información sobre archivos .DLL del sistema para Windows, puede consultar el Software Development Kit (SDK) apropiado para su versión de Windows.

## Acceso a controles y objetos ActiveX

Puede usar cualquier control ActiveX disponible en su equipo. Para usar un control ActiveX, lo agrega a un formulario y establece sus propiedades, escribe controladores para sus eventos o llama a sus métodos. Puede agregar un control ActiveX a un formulario con la barra de herramientas Controles de formularios o con el control Contenedor OLE, o mediante código. Para obtener detalles sobre cómo agregar un control ActiveX en el Diseñador de formularios, consulte el capítulo 16, [Agregar OLE](#).

Puede crear un control ActiveX mediante código de la misma forma que crearía cualquier control de Visual FoxPro. Sin embargo, antes de crear el control debe determinar el nombre de la biblioteca de clases del control, que está almacenada en el Registro de Windows. Si no tiene otra forma de determinar el nombre de la biblioteca de clases, use el Diseñador de formularios para crear el control (como se describe en la sección anterior) y, a continuación, obtenga la propiedad OLEClass del control.

Los objetos ActiveX se pueden crear directamente con `CREATEOBJECT()` y no requieren una instancia de un formulario.

### Para crear un control ActiveX mediante código

1. Llame a [CREATEOBJECT\(\)](#) para crear un formulario.
2. Llame al método [AddObject](#) del nuevo formulario para agregar el control, especificando `olecontrol` como la clase. Debe pasar el nombre de la biblioteca de clases del control como tercer parámetro del método `AddObject`.

Por ejemplo, el programa siguiente crea un formulario nuevo y le agrega un control de esquema:

```
oMiFormulario = CREATEOBJECT("form")
oMiFormulario.AddObject("oleOutline","olecontrol", ;
    "MSOutl.Outline")
```

Después de crear el formulario y el control, puede mostrar el formulario llamando a su método [Show](#) y mostrar el control estableciendo su propiedad [Visible](#) a verdadero:

```
oMiFormulario.oleOutline.Visible = .T.
oMiFormulario.Show
```

Algunos controles ActiveX no están diseñados principalmente para su uso de forma interactiva por parte de un usuario. Por ejemplo, un control cronómetro no admite métodos para la interacción con el usuario. Incluso en este caso, aún puede crear el control en un formulario porque el control normalmente ofrecerá un componente visible predeterminado, como un icono. Frecuentemente no podrá cambiar o redimensionar el icono.

Si no desea que la aplicación muestre el icono para controles que no interactúan, puede ocultar el control si establece la propiedad [Visible](#) de su control contenedor OLE a falso o establece su

propiedad [Left](#) a un valor negativo (como -100) que lo saca de la parte visible de la pantalla. De forma alternativa, puede colocar el control en un formulario que nunca se hace visible (es decir, para el que nunca se llama el método Show). En todos los casos, aún puede llamar a los métodos del control como si éste fuese visible.

## Acceso a bibliotecas de vínculos dinámicos

Si la funcionalidad que requiere está disponible en una DLL, puede vincularla a la biblioteca y llamar a sus funciones. Antes de llamar a una función DLL tiene que determinar su protocolo de llamada, incluyendo el nombre de la función, el número y los tipos de datos de sus parámetros y el tipo de datos del valor que devuelve.

En Visual FoxPro, sólo puede usar bibliotecas DLL que se hayan programado para un entorno de 32 bits. Sin embargo, si necesita acceso a una DLL de 16 bits, puede llamarlas mediante funciones disponibles en Foxtools.fll. Para obtener más detalles, vea la Ayuda para Foxtools (Foxtools.chm).

### Para llamar a una función de DLL

1. Registre la función de DLL mediante el comando [DECLARE - DLL](#). Los nombres de función distinguen mayúsculas de minúsculas.

**Nota** Si especifica WIN32API como nombre de biblioteca, Visual FoxPro busca la función de la DLL de Windows de 32 bits en Kernel32.dll, Gdi32.dll, User32.dll, Mpr.dll y Advapi32.dll.

2. Llame a la función de igual forma que llamaría a una función de Visual FoxPro.

Por ejemplo, el siguiente programa registra la función GetActiveWindow( ) de la DLL del sistema USER de Windows, que muestra el controlador de la ventana principal de Visual FoxPro. GetActiveWindow( ) no requiere parámetros y devuelve un valor entero:

```
DECLARE INTEGER GetActiveWindow IN win32api  
MESSAGEBOX(STR( GetActiveWindow() ) )
```

La DLL que contiene la función que registra debe estar disponible en el directorio predeterminado, en los directorios Windows o System, o en la ruta de acceso de DOS.

Si la función que desea llamar tiene el mismo nombre que otra función ya disponible en Visual FoxPro (una función nativa o una función de DLL declarada previamente), puede asignar un alias a la función con el nombre duplicado y, a continuación, llamarla con el alias.

```
DECLARE INTEGER GetActiveWindow IN win32api AS GetWinHndl  
MESSAGEBOX(STR( GetWinHndl() ) )
```

Las funciones de DLL vinculadas permanecerán disponibles hasta que salga de Visual FoxPro, por lo que sólo tiene que declararlas una vez por sesión. Si no pretende volver a llamar a las funciones de una DLL, puede ejecutar el comando [CLEAR DLLS](#) para quitarla de la memoria y liberar recursos.

**Nota** Ejecutar CLEAR DLLS borra de la memoria todas las funciones de DLL declaradas.

## Transferir parámetros a una DLL

Cuando registra una función de DLL, tiene que especificar el nombre y el tipo de datos de sus parámetros. De forma predeterminada, los datos se pasan por valor. Puede hacer que un parámetro se pase por referencia si incluye un signo at (@) delante del parámetro.

En general, las funciones siguen las convenciones de tipos de datos usadas en C, que son diferentes de las usadas en Visual FoxPro. Por ejemplo, las funciones de DLL no admiten un tipo de datos para date (fecha) o currency (moneda). Si los datos que pasa a una función de una DLL están en un tipo de datos no admitido por la función, tiene que convertirlos a un tipo de datos apropiado antes de pasarlos. Por ejemplo, puede convertir una fecha a un formato numérico de fecha Juliana con comandos como el siguiente:

```
cFecha = sys(11, date())
nFecha = val( cDate )
```

Algunas funciones de DLL requieren parámetros más complejos, como estructuras o matrices. Si la función requiere un puntero a una estructura, debe determinar la distribución de la estructura y emularla como una cadena en Visual FoxPro antes de transferirla o recibirla de la función de DLL. Por ejemplo, la función del sistema de Windows GetSystemTime() espera un puntero a una estructura formada por ocho palabras o enteros de 16 bits sin signo que indican el año, el mes, el día, etc. La estructura se define de esta forma:

```
typedef struct _SYSTEMTIME {
    WORD wYear ;
    WORD wMonth ;
    WORD wDayOfWeek ;
    WORD wDay ;
    WORD wHour ;
    WORD wMinute ;
    WORD wSecond ;
    WORD wMilliseconds ;
} SYSTEMTIME
```

Para pasar datos entre Visual FoxPro y la función GetSystemTime(), tiene que crear un búfer de cadena de 40 bytes (formado inicialmente por espacios) y, a continuación, pasar la dirección de esta cadena a la función para llenarla. Cuando se devuelve la cadena, debe dividirla en trozos de 2 bytes para extraer los campos individuales de la estructura. El siguiente fragmento demuestra cómo podría extraer tres de los campos de la estructura:

```
DECLARE INTEGER GetSystemTime IN win32api STRING @
cBuff=SPACE(40)
=GetSystemTime(@cBuff)

tYear = ALLTRIM(STR(ASC(SUBSTR(cBuff,2)) *
    256 + ASC(SUBSTR(cBuff,1))))
tMonth = ALLTRIM(STR(ASC(SUBSTR(cBuff,4)) *
    256 + ASC(SUBSTR(cBuff,3))))
tDOW = ALLTRIM(STR(ASC(SUBSTR(cBuff,6)) *
    256 + ASC(SUBSTR(cBuff,5))))
```

Para obtener más información, puede examinar el formulario de ejemplo Systime.scx del directorio ... \Samples\Vfp98\Solution\Winapi de Visual Studio. Para ver otros ejemplos de cómo pasar

parámetros a funciones de DLL, vea el programa Registry.prg que se encuentra en el directorio ...  
\Samples\Vfp98\Classes de Visual Studio.

Si los datos con los que está trabajando en Visual FoxPro son una matriz, debe recorrer varias veces la matriz y concatenarlos en una única cadena que representa una matriz de tipo C antes de pasarlos a la función de DLL. Si la función de Windows espera valores de 16 bits o de 32 bits, tiene que convertir los valores a sus equivalentes hexadecimales antes de concatenarlos en una cadena. Cuando pasa la cadena que contiene los datos de la matriz, Visual FoxPro pasa la dirección de la variable de tipo cadena a la DLL, que puede manipularla como matriz. Para ver un ejemplo de esto, vea el formulario de ejemplo Syscolor.scx que se encuentra en el directorio ...  
\Samples\Vfp98  
\Solution\Winapi de Visual Studio.

## Acceso a la biblioteca de Visual FoxPro

Como una DLL, una biblioteca de Visual FoxPro (archivo .fll) contiene funciones que puede llamar de la misma manera que haría con cualquier otra función. Como los archivos .fll están creados específicamente para llamarlos desde Visual FoxPro, generalmente es más fácil pasar parámetros a y desde funciones .fll.

Para usar una biblioteca de Visual FoxPro, se especifica el nombre del archivo .fll y, a continuación, se llama a la función normalmente. A diferencia del registro de funciones de DLL, no tiene que registrar las funciones individuales del archivo .fll, ni tiene que especificar información sobre los parámetros o los tipos de datos usados por la función.

**Nota** Si quiere usar una biblioteca .fll de una versión anterior de Visual FoxPro, hay que recompilar la biblioteca de modo que funcione con Visual FoxPro versión 5.0.

### Para llamar a una función .FLL

1. Registre la función .fll ejecutando el comando [SET LIBRARY](#).
2. Llame a cualquiera de las funciones de la biblioteca de la misma manera en que lo haría para cualquier función.

Por ejemplo, el siguiente programa llama a una función de la biblioteca Foxtools.fl l para determinar qué tipo de unidad es la unidad C:.

```
SET LIBRARY TO "C:\Program Files\Microsoft ;  
Visual Studio\Vfp98\Foxtools.fl l"  
? DriveType("C:")
```

Si tiene que registrar más de un archivo .fl l, incluya la palabra clave ADDITIVE en el comando SET LIBRARY. Si no lo hace, se borrará el archivo .fl l previamente registrado y se reemplazará por el último registro.

Si un nombre de función es incompatible con el de otra función ya disponible en Visual FoxPro, la última función definida tiene la prioridad. Si el nombre de una biblioteca vinculada tiene el mismo nombre que el de una función intrínseca de Visual FoxPro, la función de Visual FoxPro tiene la prioridad.

Las funciones de un archivo .fll permanecen disponibles hasta que salga de Visual FoxPro, por lo que sólo tiene que registrarlas una vez por sesión. Si no pretende volver a llamar a las funciones de un archivo .fll, ejecute [RELEASE LIBRARY](#), [RELEASE ALL](#), o [SET LIBRARY TO](#) para eliminarlas de la memoria y liberar recursos.

## Capítulo 28: Acceso a la API de Visual FoxPro

Si Visual FoxPro no incluye ya las características que necesita para su aplicación, puede extender sus posibilidades creando un control ActiveX (archivo .ocx) o una biblioteca (archivo .fll) específica de Visual FoxPro, con un compilador de 32 bits como Microsoft Visual C o C++<sup>®</sup> versión 4.0 o posterior. La información de este capítulo trata ambos tipos de programas.

**Nota** Si usa Visual C o C++ versión 2.x para programar un control ActiveX necesita el Control Development Kit. Los procedimientos de este capítulo son para la versión 4.0 de Visual C++.

Para obtener información sobre cómo usar controles ActiveX o bibliotecas FLL, consulte el capítulo 27, [Extender Visual FoxPro con bibliotecas externas](#).

En este capítulo se describe:

- [Crear una biblioteca o un control ActiveX](#)
- [Agregar llamadas a la API de Visual FoxPro](#)
- [Transferir y recibir parámetros](#)
- [Devolver un valor a Visual FoxPro](#)
- [Transferir parámetros a funciones de la API de Visual FoxPro](#)
- [Acceso a variables y campos de Visual FoxPro](#)
- [Administrar la memoria](#)
- [Generar y depurar bibliotecas y controles ActiveX](#)

### Crear una biblioteca o un objeto ActiveX

Puede extender las posibilidades de Visual FoxPro creando programas en C o C++ que cumplan tareas necesarias para su aplicación. Por ejemplo, si la aplicación requiere acceso directo a las posibilidades de Windows, puede escribir un programa C o C++ que haga las llamadas a la API de Windows y, a continuación, devuelva información a Visual FoxPro.

Puede crear tres tipos de programas que tengan acceso a la API de Visual FoxPro:

- Un control ActiveX (archivo .ocx).
- Un objeto COM.
- Una DLL específica de Visual FoxPro. Como la DLL sólo se puede llamar desde Visual FoxPro, se le suele dar la extensión .fll.

Cada tipo de programa tiene sus ventajas. Un control ActiveX:

- Se puede utilizar mediante técnicas orientadas a objetos estándar, como establecer sus

- propiedades y llamar a sus métodos.
- Se puede convertir en subclase y se puede pasar por alto sus métodos.
- Está encapsulado y se puede llamar (crear instancias) varias veces sin administración compleja del entorno para conservar estados de usuarios.
- Tiene un paso de parámetros simple.
- También se puede llamar desde otros programas para Windows, si lo programa para esto.

Las ventajas de los objetos COM:

- Puede tener acceso a los mismos mediante técnicas estándar orientadas a objetos, como establecer sus propiedades y llamar a sus métodos.
- No se pueden pasar por alto sus métodos.
- Están encapsulados y se pueden llamar (crear instancias) múltiples veces sin administración compleja del entorno para conservar los estados de usuarios.
- La transferencia de parámetros es más sencilla.
- También se pueden llamar desde otros programas para Windows, si lo programa para esto.

Por otra parte, una biblioteca.fll:

- Puede resultarle familiar si ha utilizado versiones anteriores de Visual FoxPro.

**Nota** Si quiere usar una biblioteca .fll de una versión de Visual FoxPro anterior a la 5.0, la biblioteca se tiene que volver a compilar para que funcione con Visual FoxPro versión 6.0.

## Crear un objeto ActiveX básico

Puede crear objetos COM con la ActiveX Template Library proporcionada con Microsoft Visual C++ 5.0. Para obtener más información acerca de cómo crear objetos COM con Visual C++ 5.0, busque "ATL" en Microsoft Developer's Network.

Los controles ActiveX específicos de Visual FoxPro se crean de la misma manera que cualquier control similar. La mayor parte de los compiladores de C++ le permiten crear la estructura del control y también se pueden utilizar con Microsoft Visual Basic® Control Creation Edition versión 5.0.

La sección siguiente describe los pasos para crear un control ActiveX con Microsoft Visual C++ para utilizarlo con Visual FoxPro.

### Para crear un proyecto para un control ActiveX

1. Inicie Microsoft Visual C++.
2. En el menú **File**, elija **New**.
3. En el cuadro de diálogo **New**, elija **Project Workspace**.
4. En el cuadro de diálogo **New Project Workspace**, especifique un nombre de proyecto.
5. En la lista **Type**, elija **OLE ControlWizard**.



6. Elija **Create** y, a continuación, siga los pasos del Asistente.

Cuando el asistente haya terminado puede generar inmediatamente el control ActiveX. Sin embargo, también tendrá que definir propiedades y métodos para el control.

### Para agregar propiedades y métodos al control ActiveX

1. En el menú **View**, elija **ClassWizard**.
2. Elija la ficha **OLEAutomation**.
3. Elija **Add Method** o **Add Property**.
4. Escriba el nombre, el parámetro y demás información necesaria para el elemento que está creando y, a continuación, elija **OK**.
5. Elija **Edit Code** para mostrar el editor y, a continuación, escriba el código que define la propiedad o método que está creando.

Por ejemplo, para crear una propiedad `Version` que devuelva la versión del archivo `.ocx` como un número entero (como 101), se crea la propiedad con un tipo devuelto `long` y se agrega código similar al siguiente:

```
#define VERSION 101

long CPyCtrl::GetVersion()
{
    // establecer el número de versión aquí
    return VERSION;
}
```

Como el número de versión suele ser de sólo lectura, no tiene que crear una función `SetVersion()`.

### Crear una biblioteca FLL básica

Como una biblioteca FLL es esencialmente una DLL que llama a la API de Visual FoxPro, puede crear una biblioteca FLL siguiendo los pasos siguientes en el entorno de programación para crear una DLL.

### Para crear un proyecto para la biblioteca FLL

1. Inicie Microsoft Visual C/C++.
2. En el menú **File**, elija **New**.
3. En el cuadro de diálogo **New**, elija **Project Workspace**.
4. En el cuadro de diálogo **New Project Workspace**, especifique un nombre de proyecto.
5. En la lista **Type**, elija **Dynamic-Link Library**.

Después de crear la estructura básica de la DLL, agregue las funciones que desee llamar desde Visual FoxPro. Las siguientes secciones proporcionan estructuras para crear funciones en C y C++.

### Establecer una plantilla de biblioteca

Cada biblioteca de funciones que cree tiene la misma estructura básica. Usando una plantilla para la estructura, todo lo que tiene que hacer es rellenar los trozos en blanco apropiados para su rutina de biblioteca específica.

Hay cinco elementos en un plantilla de biblioteca de Visual FoxPro:

1. Instrucción `#include`
2. Definición de función
3. Código de función
4. Estructura `FoxInfo`
5. Estructura `FoxTable`

### Una plantilla de ejemplo en C

Puede usar la siguiente plantilla para crear bibliotecas escritas en C:

```
#include <Pro_ext.h>

void Internal_Name(ParamBlk *parm)
{
    // Aquí va el código de la función.
}

FoxInfo myFoxInfo[] = {
    {"FUNC_NAME", (FPFI) Internal_Name, 0, ""},
};

FoxTable _FoxTable = {
    (FoxTable *)0, sizeof(myFoxInfo)/sizeof(FoxInfo), myFoxInfo
};
```

### Una plantilla de ejemplo en C++

Para rutinas en C++, puede usar la siguiente plantilla. Esta plantilla se diferencia de la plantilla en C porque declara la estructura `FoxTable` como externa:

```
#include <Pro_ext.h>

void Internal_Name(ParamBlk *parm)
{
    // Aquí va el código de función.
}
```

```

FoxInfo myFoxInfo[] = {
    {"FUNC_NAME", (FPFI) Internal_Name, 0, ""},
};

extern "C" {
    FoxTable _FoxTable = {
        (FoxTable *)0, sizeof(myFoxInfo)/sizeof(FoxInfo), myFoxInfo
    };
}

```

## Usar la plantilla

Para usar el archivo de encabezado y crear una biblioteca compilada, necesita:

- El archivo de encabezado Pro\_ext.h. Puede imprimir este archivo para ver las declaraciones de función, typedefs y structs utilizados en la API de Visual FoxPro.
- El archivo Winapims.lib.

Estos dos archivos se instalan en el subdirectorio API cuando instala Visual FoxPro.

La definición de función devuelve void y espera el siguiente parámetro: ParamBlk \*parm. La estructura ParamBlk se trata en [Transferir y recibir parámetros](#), más adelante en este capítulo.

Además de los archivos que se han mostrado, los únicos elementos necesarios de una biblioteca de Visual FoxPro son las estructuras FoxInfo y FoxTable.

## Usar las estructuras FoxInfo y FoxTable

Sus funciones de biblioteca se comunican con Visual FoxPro a través de la estructura FoxInfo. De esta estructura, Visual FoxPro determina el nombre de función y el número y tipo de parámetros. La estructura FoxTable es una lista vinculada que sigue las estructuras FoxInfo. Vea Pro\_ext.h en el directorio API de Visual FoxPro para las definiciones struct FoxInfo y FoxTable.

## Estructura FoxInfo

La estructura FoxInfo es el vehículo usado para comunicar nombres de funciones y descripciones de parámetros entre Visual FoxPro y su biblioteca. Una estructura FoxInfo genérica tiene la siguiente apariencia:

```

FoxInfo arrayname[ ] = {
    {funcName1, FPFI function1, parmCount1, parmTypes1}
    {funcName2, FPFI function2, parmCount2, parmTypes2}
    ...
    {funcNameN, FPFI functionN, parmCountN, parmTypesN}
};

```

Los marcadores se definen así:

*arrayname*

Variable de tipo FoxInfo. Observe que puede incluir varias líneas de estructura FoxInfo en esta

matriz.

*funcName*

Contiene el nombre (en mayúsculas y de longitud igual o inferior a 10 caracteres) que el usuario de Visual FoxPro llama para usar la función.

*function*

La dirección de la rutina de lenguaje C. Es el nombre exacto (que distingue mayúsculas y minúsculas) que usa para definir la función.

*parmCount*

Especifica el número de parámetros descrito en la cadena *parmTypes* o uno de los siguientes valores de indicadores.

Valor	Descripción
INTERNAL	Especifique que la función no se puede llamar directamente desde Visual FoxPro.
CALLONLOAD	Especifica que la rutina se llama cuando se carga la biblioteca. CALLONLOAD no puede llamar a ninguna rutina que devuelva resultados a Visual FoxPro.
CALLONUNLOAD	Especifica que la rutina se llama cuando la biblioteca se descarga o cuando se ejecuta el comando QUIT de Visual FoxPro. CALLONUNLOAD no puede llamar a ninguna rutina que devuelva resultados a Visual FoxPro.

*parmTypes*

Describe el tipo de datos de cada parámetro. La tabla siguiente muestra los valores válidos para *parmTypes*.

Valor	Descripción
" "	Ningún parámetro
" ? "	Se puede pasar cualquier tipo. En el cuerpo de la función, tendrá que comprobar el tipo del parámetro pasado
" C "	Parámetro de tipo Character
" D "	Parámetro de tipo Date
" I "	Parámetro de tipo Integer

"L"	Parámetro de tipo Logical
"N"	Parámetro de tipo Numeric
"R"	Referencia
"T"	Parámetro de tipo DateTime
"Y"	Parámetro de tipo Currency
"O"	Parámetro de tipo Object

Incluya un valor de tipo para cada parámetro transferido a la biblioteca. Por ejemplo, si crea una función que acepta un carácter y un parámetro numérico, sustituya "CN" por *parmType*.

**Nota** Para indicar que un parámetro es opcional, ponga como prefijo un punto. Sólo se pueden omitir los parámetros de la derecha.

La siguiente estructura `FoxInfo` define una biblioteca con una función (llamada internamente `dates` y la que se tiene acceso como `DATES`) que acepta un parámetro de tipo Character:

```
FoxInfo myFoxInfo[] = {
    { "DATES", (FPFI) dates, 1, "C" }
};
```

Cuando haya compilado la biblioteca con esta estructura `FoxInfo` y la haya cargado en Visual FoxPro con el comando [SET LIBRARY TO](#), puede llamar a esta función en Visual FoxPro con la siguiente línea de código:

```
=DATES( "01/01/95" )
```

### Estructura `FoxTable`

La estructura `FoxTable` es una lista vinculada que hace un seguimiento de todas las estructuras `FoxInfo` que tiene para una biblioteca determinada:

```
FoxTable _FoxTable = {nextLibrary, infoCount, infoPtr};
```

en donde los indicadores se definen así:

*nextLibrary*

Un puntero usado internamente por Visual FoxPro; se debe inicializar a 0.

*infoCount*

El número de rutinas externas de Visual FoxPro definidas en esta biblioteca.

*infoPtr*

La dirección del primer elemento de una matriz de estructuras FoxInfo. Este nombre tiene que coincidir con el nombre presentado en la instrucción FoxInfo.

A continuación se muestra un ejemplo de instrucción FoxTable. Si su nombre de *matriz* FoxInfo es myFoxInfo, nunca tendrá que cambiar esta instrucción:

```
FoxTable _FoxTable = {  
    (FoxTable *) 0,  
    sizeof( myFoxInfo ) / sizeof( FoxInfo ),  
    myFoxInfo  
};
```

## Agregar llamadas a la API de Visual FoxPro

Para integrar su programa con Visual FoxPro, puede llamar a las rutinas de la API de Visual FoxPro. Estas rutinas de la API son funciones que puede llamar desde cualquier programa C o C++, incluyendo un archivo .ocx o .fl, que le proporciona acceso a variables, administrar operaciones de bases de datos y realizar muchas otras tareas específicas de Visual FoxPro.

La tabla siguiente muestra las categorías generales de llamadas a la API disponibles en Visual FoxPro. Para obtener detalles sobre funciones de la API, consulte [Rutinas A-Z de la biblioteca API](#) o [Rutinas por categorías de la biblioteca API](#).

Para usar las rutinas de la API de Visual FoxPro, tiene que incluir el archivo Pro\_ext.h, disponible en el directorio API de Visual FoxPro. Este archivo incluye los prototipos para las funciones y estructuras que le permiten compartir información con Visual FoxPro.

Si está programando un control ActiveX, también tiene que agregar llamadas para inicializar y borrar la API.

### Para agregar rutinas de la API de Visual FoxPro a su objeto ActiveX

1. Use #INCLUDE para incluir el archivo Pro\_ext.h junto con otros archivos de encabezado necesarios.
2. En el Constructor (método Init) del control, llame a \_OCXAPI( ) para inicializar la interfaz a Visual FoxPro mediante este código:

```
_OCXAPI( AfxGetInstanceHandle( ), DLL_PROCESS_ATTACH );
```

3. Incluya llamadas a la API de Visual FoxPro como lo requiera su objeto.
4. En el Destructor (método Destroy) del objeto, vuelva a llamar a \_OCXAPI( ) para liberar el proceso creado en el Constructor, mediante este código:

```
_OCXAPI(AfxGetInstanceHandle(),DLL_PROCESS_DETACH);
```

Para ver un ejemplo de archivo .ocx que incluya llamadas a la API de Visual FoxPro, vea [Foxtlib.ocx](#). Para ver un ejemplo de biblioteca .fll que incluya llamadas a la API de Visual FoxPro, vea los programas de ejemplo que se encuentran en el directorio Vfp98\Api\Samples y que tengan la extensión C: EVENT.C, HELLO.C, etc.

Si usa llamadas a la API de Visual FoxPro en su control ActiveX, su objeto COM o su biblioteca .FLL, el código que contiene las llamadas es incompatible con otras aplicaciones. Es posible que entonces desee generar una o más pruebas en el programa para determinar si Visual FoxPro puede llamar al objeto.

Por ejemplo, cuando crea un control ActiveX con Microsoft Foundation Classes, puede cambiar el código del constructor del control para que incluya una prueba y avise al usuario si se ha llamado al control desde un programa distinto de Visual FoxPro:

```
if (!_OCXAPI(AfxGetInstanceHandle(),DLL_PROCESS_ATTACH))
{
    ::MessageBox(0,"Este OCX sólo se puede alojar en Visual Foxpro","",0);
    //Aquí puede hacer lo que desee cuando el host no es VFP:
    // es posible que no quiera cargarlo
    // o puede que desee establecer una propiedad
    // que diga que el host no es VFP y que el control utilizará
    // otros medios para lograr su objetivo.
}
```

Si crea un control ActiveX con Microsoft ActiveX Template Library, utilice el código siguiente:

```
if (!_OCXAPI(_Module.GetModuleInstance(),DLL_PROCESS_ATTACH))
{
    ::MessageBox(0,"Este OCX sólo se puede alojar en Visual Foxpro","",0);
    //Aquí puede hacer lo que desee cuando el host no es VFP:
    // es posible que no quiera cargarlo
    // o puede que desee establecer una propiedad
    // que diga que el host no es VFP y que el control utilizará
    // otros medios para lograr su objetivo.
}
```

En este ejemplo, el control no sale y seguirá ejecutándose después de que el usuario haya recibido el mensaje. La estrategia que elija dependerá de cómo crea que se va a usar el control. Por ejemplo, si detecta que se está usando el control fuera de Visual FoxPro, puede establecer un indicador que prueba en cada punto del control dónde llama a la API de Visual FoxPro. Si el indicador indica que el control está fuera de Visual FoxPro, puede ramificar la llamada de la API como forma alternativa de realizar la misma tarea.

## Transferir y recibir parámetros

Cuando Visual FoxPro llama a su programa, éste puede recibir parámetros. Por ejemplo, un control ActiveX puede recibir parámetros cuando se llama a uno de sus métodos. De forma similar, un programa de Visual FoxPro puede llamar a una función de la biblioteca FLL y pasarle parámetros.

Visual FoxPro puede transferir parámetros a su programa por valor o por referencia. De forma

predeterminada, los parámetros respetan la configuración realizada con SET UDFPARMS. Otras variables (como matrices o campos) y expresiones se pasan por valor.

Para obligar a que un parámetro se pase por referencia, tiene que usar como prefijo de la variable el operador @. Para forzar la transferencia por valor de un parámetro, entre paréntesis.

**Nota** En Visual FoxPro, los elementos de matriz individuales siempre se pasan por valor. Cuando SET UDFPARMS se establece a VALUE y no se especifica ningún elemento de matriz, el nombre de matriz se refiere al primer elemento de la matriz (a no ser que tenga el prefijo @).

Como los controles ActiveX y los objetos COM son programas estándar para Windows, no es necesario ningún mecanismo especial para transferir parámetros entre Visual FoxPro y su programa. Puede escribir el programa como si estuviera recibiendo programas desde cualquier programa escrito en C o C++.

En contraste, las funciones de una biblioteca FLL usan la estructura FoxInfo para recibir datos desde Visual FoxPro. La estructura FoxInfo muestra sus funciones de biblioteca y el número y tipo de parámetros que esperan. Por ejemplo, la siguiente estructura FoxInfo pertenece a una biblioteca con una función, internamente llamada `dates`, que acepta un parámetro Character:

```
FoxInfo myFoxInfo[] = {
    { "DATES", (FPFI) dates, 1, "C" }
};
```

Las funciones que defina en sus bibliotecas reciben actualmente un único parámetro, un puntero al bloque de parámetros. Este bloque de parámetros, definido en la estructura `ParamBlk`, guarda toda la información sobre los parámetros que se pasaron en la llamada de función de Visual FoxPro. La declaración de función sigue el siguiente formato:

```
void function_name(ParamBlk *parm)
```

Por ejemplo, la definición de función para `dates` es:

```
void dates(ParamBlk *parm)
```

La estructura de `ParamBlk` consiste en un entero que representa el número de parámetros, seguida inmediatamente por una matriz de uniones de parámetros. La definición de estructura se incluye en `Pro_ext.h`:

```
/* Lista de parámetros a una función de biblioteca.          */
typedef struct {
    short int pCount;          /* número de parámetros pasados */
    Parameter p[1];           /* parámetros pCount */
} ParamBlk;
```

El parámetro typedef incluido en la estructura `ParamBlk` es una unión de una estructura `Value` y una estructura `Locator`. La llamada por valor la controla la estructura `Value`; la llamada por referencia la controla la estructura `Locator`. Estas estructuras se usan para tener acceso a parámetros pasados a su función cuando Visual FoxPro la llama.

La siguiente información sale del archivo `Pro_ext.h` y muestra la definición del tipo `Parameter`:



```
/* Un parámetro a una biblioteca de funciones.          */
typedef union {
    Value val;
    Locator loc;
} Parameter;
```

### Definir la estructura Value

Si se pasa un parámetro a la función por valor, use la estructura Value para tener acceso al mismo. La siguiente definición de estructura Value se extrae del archivo Pro\_ext.h:

```
// El valor de una expresión.
typedef struct {
    char        ev_type;
    char        ev_padding;
    short       ev_width;
    unsigned    ev_length;
    long        ev_long;
    double      ev_real;
    CCY         ev_currency;
    MHANDLE     ev_handle;
    ULONG       ev_object;
} Value;
```

### Campos de la estructura Value

La siguiente tabla es una guía para los valores que puede pasar a y recibir en la estructura Value para distintos tipos de datos. Sólo los campos de la estructura mostrados para un tipo de datos se usan para ese tipo de datos.

#### Contenido de una estructura Value para distintos tipos de datos

Tipo de datos	Campo de estructura	Valor
Character	ev_type	'C'
	ev_length	longitud de cadena
	ev_handle	MHANDLE a cadena
Numeric	ev_type	'N'
	ev_width	Ancho de presentación
	ev_length	Posiciones decimales
	ev_real	Precisión doble
Integer	ev_type	'I'

	ev_width	Ancho de presentación
	ev_long	Entero largo
Date	ev_type	'D'
	ev_real	Date <sup>1</sup>
Date Time	ev_type	'T'
	ev_real	Date + (segundos/86400.0)
Currency	ev_type	'Y'
	ev_width	Ancho de presentación
	ev_currency	Value <sup>2</sup> de tipo Currency
Logical	ev_type	'L'
	ev_length	0 o 1
Memo	ev_type	'M'
	ev_wdith	FCHAN
	ev_long	Longitud de campo memo
	ev_real	Offset de campo memo
General	ev_type	'G'
	ev_wdith	FCHAN
	ev_long	Longitud de campo general
	ev_real	Offset de campo general
Object	ev_type	'O'

ev_object		Identificador de objeto
Null	ev_type	'0' (cero)
ev_long		Tipo de datos

1. La fecha está representada como un número de día Juliano de coma flotante de doble precisión calculado mediante el algoritmo 199 de Collected Algorithms de la ACM.
2. El valor de tipo currency es un entero largo, con una coma decimal implícita delante de los cuatro últimos dígitos.

**Nota** ev\_length es el único indicador de la longitud de una cadena. La cadena no puede tener un terminador nulo porque no puede contener caracteres nulos incrustados.

### Definir la estructura Locator

Use la estructura Locator para manipular parámetros pasados por referencia. La siguiente definición de estructura Locator se extrae del archivo Pro\_ext.h.

```
typedef struct {
    char l_type;
    short l_where,          /* Número de base de datos o -1 para memoria */
    l_NTI,                 /* Desplazamiento de tabla de nombre de variable */
    l_offset, /* Índice en la base de datos */
    l_subs, /* # subíndices especificados por 0 <= x <= 2 */
    l_sub1, l_sub2; /* valores enteros de subíndice */
} Locator;
```

### Campos de estructura Locator

La tabla siguiente es una guía a los campos de la estructura Locator.

Campo Locator	Uso del campo
l_type	'R'
l_where	Número de tabla que contiene este campo, o - 1 para una variable.
l_NTI	Name Table Index (Índice de tabla de nombres). Uso interno de Visual FoxPro.
l_offset	Número de campo de tabla. Uso interno de Visual FoxPro.
l_subs	Sólo para variables, el número de subíndices (0 - 2).
l_sub1	Sólo para variables, el primer subíndice si l_subs no es 0.

---



---

l_sub2	Sólo para variables, el segundo subíndice si l_subs es 2.
--------	---

---



---

**Nota** Es una buena práctica de programación comprobar el tipo de parámetro de `ev_type` para ayudarle a determinar a qué campos se tiene acceso desde la estructura `Value`.

### Un ejemplo de parámetros de acceso a una biblioteca FLL

El siguiente ejemplo usa `StrCpy( )` para devolver a Visual FoxPro un tipo `Character` que es la concatenación de sus dos parámetros `Character`. Observe que aunque el controlador de la estructura `Value` de cada parámetro se usa como memoria de trabajo para realizar la concatenación, los cambios a esta asignación de memoria no afectan al argumento de Visual FoxPro que se ha pasado por valor.

Para ver un ejemplo que usa la estructura `Locator` para administrar un parámetro pasado por referencia, consulte [Devolver un valor de una biblioteca FLL](#) más adelante en este mismo capítulo.

```
#include <Pro_ext.h>

Example(ParamBlk *parm)
{
    // simplificar la estructura paramBlk
    // para administrar mediante métodos abreviados #define
#define p0 (parm->p[0].val)
#define p1 (parm->p[1].val)

    // asegúrese de que hay suficiente memoria
    if (!_SetHandSize(p0.ev_handle, p0.ev_length + p1.ev_length))
        _Error(182); // "Memoria insuficiente"

    // bloquear los controladores
    _HLock(p0.ev_handle);
    _HLock(p1.ev_handle);

    // convertir controladores en punteros y asegurarse de
    // que las cadenas terminan en carácter nulo
    ((char *)_HandToPtr(p0.ev_handle))[p0.ev_length] = '\0';
    ((char *)_HandToPtr(p1.ev_handle))[p1.ev_length] = '\0';

    // concatenar cadenas mediante la función de la
    // API _StrCpy
    _StrCpy((char *)_HandToPtr(p0.ev_handle) + p0.ev_length,
            _HandToPtr(p1.ev_handle));

    // devuelve la cadena concatenada a Visual FoxPro
    _RetChar(_HandToPtr(p0.ev_handle));

    // desbloquear los controladores
    _HUnlock(p0.ev_handle);
    _HUnlock(p1.ev_handle);
}

FoxInfo myFoxInfo[] = {
    {"STRCAT", Example, 2, "CC"},
};

FoxTable _FoxTable = {
    (FoxTable *) 0, sizeof(myFoxInfo)/sizeof(FoxInfo), myFoxInfo
```

```
};
```

## Devolver un valor a Visual FoxPro

El método que se usa para devolver un valor desde un programa a Visual FoxPro depende de si está creando un control ActiveX o una biblioteca FLL.

### Devolver un valor desde un control ActiveX

Para devolver un valor desde un control ActiveX a Visual FoxPro, use la instrucción RETURN en el control, pasando un único valor, como en el siguiente ejemplo:

```
#define VERSION 101

// más código aquí

long CPyCtrl::GetVersion()
{
    // establecer el número de versión aquí en la
    // variable fVersion
    return VERSION;
}
```

### Devolver un valor desde una biblioteca FLL

Para devolver valores desde una biblioteca FLL, use funciones de la API, no comandos nativos de C o C++. Las siguientes funciones le permiten devolver valores a Visual FoxPro.

**Nota** No use la siguiente función de la API para devolver un valor desde un archivo .ocx; use la instrucción RETURN. Las funciones de devolución de la API sólo deben usarse en bibliotecas FLL.

Función	Descripción
<a href="#"><u>_RetChar(char *string)</u></a>	Establece el valor devuelto de la función a una cadena que termina en carácter nulo.
<a href="#"><u>_RetCurrency(CCY cval, int width)</u></a>	Establece el valor devuelto de la función a un valor de tipo currency.
<a href="#"><u>_RetDateStr(char *string)</u></a>	Establece el valor devuelto de la función a una fecha. La fecha se especifica en el formato dd/mm/aa[aa].
<a href="#"><u>_RetDateTimeStr(char *string)</u></a>	Establece el valor devuelto de la función a una fecha y una hora especificadas en el formato mm/dd/aa[aa] hh:mm:ss.
<a href="#"><u>_RetFloat(double flt, int width, int dec)</u></a>	Establece el valor devuelto de la función a un valor de tipo float.
<a href="#"><u>_RetInt(long ival, int width)</u></a>	Establece el valor devuelto de la función a un valor numérico.

<a href="#"><u>_RetLogical(int flag)</u></a>	Establece el valor devuelto de una función a un valor logical. Cero se considera FALSE. Cualquier valor distinto de cero se considera TRUE.
<a href="#"><u>_RetVal(Value *val)</u></a>	Pasa una estructura Value completa de Visual FoxPro; se puede devolver cualquier tipo de datos de Visual FoxPro excepto memo. Tiene que llamar a _RetVal( ) para devolver una cadena que contiene incrustados caracteres nulos o para devolver un valor .NULL.

**Nota** Para devolver el valor de un tipo de datos objeto, use la función \_RetVal(), rellenando el campo `ev_object` de la estructura Value.

El siguiente ejemplo, `Sum`, acepta una referencia a un campo numérico de una tabla y usa `_RetFloat` para devolver la suma de los valores del campo:

```
#include <Pro_ext.h>

Sum(ParamBlk *parm)
{
    // declarar variables
    double tot = 0, rec_cnt;
    int i = 0, workarea = -1; // -1 es el área de trabajo actual
    Value val;

    // GO TOP
    _DBRewind(workarea);

    // Get RECCOUNT( )
    rec_cnt = _DBRecCount(workarea);

    // Recorrer la tabla cíclicamente
    for(i = 0; i < rec_cnt; i++)
    {
        //Colocar valor del campo en la estructura Value
        _Load(&parm->p[0].loc, &val);

        // agregar el valor a la suma acumulada total
        tot += val.ev_real;

        // SKIP 1 en el área de trabajo
        _DBSkip(workarea, 1);
    }

    // Devolver el valor de la suma a Visual FoxPro
    _RetFloat(tot, 10, 4);
}

// La función Sum recibe un parámetro por referencia
FoxInfo myFoxInfo[] = {
    {"SUM", Sum, 1, "R"}
};

FoxTable _FoxTable = {
    (FoxTable *) 0, sizeof(myFoxInfo)/sizeof(FoxInfo), myFoxInfo
};
```

Suponiendo que hay un campo llamado `amount` en la tabla abierta actualmente, la siguiente línea de

código de un programa de Visual FoxPro llama a la función:

```
? SUM(@amount)
```

## Transferir parámetros a funciones de la API de Visual FoxPro

A menudo, las rutinas de la API de Visual FoxPro requieren parámetros de una estructura de datos concreta de Visual FoxPro. Las siguientes secciones proporcionan una lista de tipos de datos de Visual FoxPro y estructuras de datos adicionales. Para las definiciones de tipos y las definiciones de estructuras, consulte el archivo Pro\_ext.h.

### Tipos de datos de la API de Visual FoxPro

Los siguientes tipos de datos se usan en las rutinas de la API de Visual FoxPro.

Tipo de datos	Descripción
EDLINE	El número de una línea de un archivo abierto en una ventana de edición. La primera línea es 1.
EDPOS	La posición de desplazamiento de un carácter de un archivo abierto en una ventana de edición. La posición de desplazamiento del primer carácter o campo memo es 0.
FCHAN	Canal de archivo. A cada archivo abierto por Visual FoxPro, o mediante la API con <a href="#">FCreate()</a> y <a href="#">FOpen()</a> , se asigna un FCHAN.
FPFI	Un puntero de 32 bits a una función que devuelve un entero.
ITEMID	Un identificador único asignado a un comando individual de un menú.
MENUID	Un identificador único asignado a un menú.
MHANDLE	Un identificador único dado a cada bloque de memoria asignado por Visual FoxPro o asignado mediante la API con <a href="#">AllocHand()</a> . Se puede eliminar la referencia a su puntero con <a href="#">HandToPtr()</a> .
NTI	Name table index (Índice de tabla de nombres). Cada variable y nombre de campo de tabla tiene una entrada en esta tabla.
WHANDLE	Controlador de ventana. Identificador único asignado a cada ventana abierta por Visual FoxPro o abierta mediante la API con <a href="#">WOpen()</a> .

**Nota** Como los punteros FAR no son apropiados para compiladores de 32 bits, las instrucciones `#define` de Pro\_ext.h redefinen FAR, `_far` y `__far` como valores nulos.

### Estructuras de datos de la API de Visual FoxPro

Las estructuras de datos principales usadas en la biblioteca de la API de Visual FoxPro se muestran en la tabla siguiente.

<b>Estructura</b>	<b>Descripción</b>
EventRec	Estructura usada para describir qué está haciendo el sistema en un instante determinado.
FoxInfo	Se usa en las bibliotecas FLL para comunicarse entre Visual FoxPro y su programa; no se usa en los archivos .ocx. Se trata en <a href="#">Usar las estructuras FoxInfo y FoxTable</a> en una sección anterior de este capítulo.
FoxTable	Se usa en bibliotecas FLL para comunicarse entre Visual FoxPro y su programa; no se usa en archivos .ocx. Se trata en <a href="#">Usar las estructuras FoxInfo y FoxTable</a> en una sección anterior de este capítulo.
Locator	Una estructura usada para tener acceso a valores de parámetros (FLL) o variables o campos de Visual FoxPro (FLL y .ocx).
ParamBlk	Se usa en bibliotecas FLL para comunicarse entre Visual FoxPro y su programa; no se usa en archivos .ocx. Se trata en <a href="#">Usar las estructuras FoxInfo y FoxTable</a> en una sección anterior de este capítulo.
Parameter	Se usa en bibliotecas FLL para comunicarse entre Visual FoxPro y su programa; no se usa en archivos .ocx. Se trata en <a href="#">Usar las estructuras FoxInfo y FoxTable</a> en una sección anterior de este capítulo.
Point	Una estructura que define las coordenadas verticales y horizontales de un punto único de la pantalla. Las coordenadas se especifican en filas y columnas.
Rect	Una estructura que define las coordenadas de un rectángulo en la pantalla. La esquina superior izquierda del rectángulo se define por <i>(top, left)</i> y la esquina inferior derecha por <i>(bottom-1, right-1)</i> . Las coordenadas se especifican en filas y columnas.
Value	Una estructura usada para tener acceso a valores de parámetros (FLL) o variables o campos de Visual FoxPro (FLL y .ocx).

## Acceso a variables y campos de Visual FoxPro

Puede tener acceso a los valores de variables o campos de Visual FoxPro en el control ActiveX o función FLL, para leerlos o establecerlos. Además, puede crear nuevas variables a las que se puede tener acceso desde Visual FoxPro.

Las variables y los campos están disponibles en Visual FoxPro en una tabla de nombres, que es una matriz que contiene los nombres de todas las variables y todos los campos definidos actualmente. Puede tener acceso a un elemento individual de la matriz mediante un índice de tabla de nombres (Name Table Index, NTI). Una función especial de la API, [NameTableIndex\(\)](#), devuelve el índice de una variable o un campo existente en base al nombre proporcionado. Después de haber determinado



el NTI para una variable dada, puede leerla mediante la función [Load\(\)](#) de la API o establecerla mediante la función [Store\(\)](#) de la API. Para crear una variable nueva, puede llamar a la función [NewVar\(\)](#) de la API.

Para tener acceso a las variables o campos de Visual FoxPro, se usan las estructuras Value y Locator definidas en Pro\_ext.h. Si está creando una biblioteca FLL, puede usar la misma técnica que ha usado para tener acceso a parámetros transferidos a sus funciones. Para obtener detalles sobre las estructuras Value y Locator, consulte [Transferir y recibir parámetros](#) en una sección anterior de este capítulo.

El ejemplo siguiente, extraído del programa Foxtlibctl.cpp que se encuentra en el directorio Vfp98 \Api\Samples\Foxtlib de Visual FoxPro, ilustra cómo puede usar las estructuras Value y Locator de un control ActiveX para tener acceso a las variables de Visual FoxPro:

```
long CFoxtlibCtrl::TLGetTypeAttr(long pTypeInfo, LPCTSTR szArrName)
{
    int nResult = 1;
    TYPEATTR *lpTypeAttr;
    Locator loc;
    Value val;
    OLECHAR szGuid[128];
    char *szBuff;
    __try {
        if (_FindVar(_NameTableIndex(( char *)szArrName), -1, &loc)) {
            ((ITypeInfo *)pTypeInfo)->GetTypeAttr(&lpTypeAttr);
            if (_ALen(loc.l_NTI, AL_ELEMENTS) < 16) {
                _Error(631); //El argumento de matriz no tiene la dimensión apropiada.
            }

            //1 = Guid
            StringFromGUID2(lpTypeAttr->guid, (LPOLESTR )&szGuid, sizeof(szGuid));
            OLEOleToAnsiString(szGuid, &szBuff);
            val.ev_type = 'C';
            val.ev_length = strlen(szBuff);
            val.ev_handle = _AllocHand(val.ev_length);
            _HLock(val.ev_handle);
            _MemMove((char *) _HandToPtr( val.ev_handle ), szBuff, val.ev_length);
            OLEFreeString((void **)&szBuff);
            _HUnLock(val.ev_handle);
            loc.l_sub1 = 1;
            _Store(&loc, &val);
            _FreeHand(val.ev_handle);

            //2 = LCID
            loc.l_sub1 = 2;
            val.ev_type = 'I';
            val.ev_long = lpTypeAttr->lcid;
            _Store(&loc, &val);

            // código para los valores 3 - 16 aquí
            ((ITypeInfo *)pTypeInfo) -> ReleaseTypeAttr(lpTypeAttr);
        }
    } __except (EXCEPTION_EXECUTE_HANDLER) {
        nResult = 0;
    }
    return nResult;
}
```

## Administrar la memoria

La API de Visual FoxPro proporciona acceso directo al administrador de memoria dinámica de Visual FoxPro. Para las rutinas de la API que requieran asignaciones de memoria, se devuelve un identificador de memoria o controlador. La arquitectura de carga de segmentos de Visual FoxPro usa identificadores en lugar de punteros de forma que puede administrar la memoria de forma más eficaz.

**Nota** Las técnicas descritas en esta sección para administrar la memoria con la API de Visual FoxPro se aplican tanto a controles API ActiveX como a bibliotecas FLL.

## Usar identificadores

Un *identificador* se refiere a un identificador de memoria, que es esencialmente un índice a una matriz de puntero. Los punteros apuntan a bloques de memoria que Visual FoxPro conoce. Casi todas las referencias a la memoria en la API se hacen a través de identificadores en lugar de los punteros de C, más tradicionales.

### Para asignar y usar memoria en la biblioteca

1. Asigne un identificador con [\\_AllocHand\(\)](#).
2. Bloquee el identificador con [\\_HLock\(\)](#).
3. Convierta el identificador en un puntero con [\\_HandToPtr\(\)](#).
4. Haga referencia a la memoria mediante un puntero.
5. Desbloquee el identificador con [\\_HUnlock\(\)](#).

**Nota** Para evitar la corrupción de archivos memo, no escriba en un archivo memo antes de llamar a [\\_AllocMemo\(\)](#).

Para tratar la memoria asignada, las rutinas de la API deben convertir el identificador en un puntero llamando a la rutina [\\_HandToPtr\(\)](#). Incluso si el administrador de memoria de Visual FoxPro tiene que reorganizar la memoria para obtener más memoria contigua para sucesivas solicitudes de memoria, el identificador sigue siendo el mismo. También se proporcionan rutinas que aumentan, reducen, liberan y bloquean asignaciones de memoria.

Cuando esté creando rutinas externas, intente minimizar el uso de memoria. Si crea una rutina externa que asigna memoria de forma dinámica, intente usar la menor cantidad posible de memoria. Sea especialmente cuidadoso al bloquear grandes asignaciones de memoria durante largos períodos de tiempo. No se olvide de desbloquear los identificadores de memoria con [\\_HUnlock\(\)](#) cuando ya no tienen que estar bloqueados, porque el rendimiento de Visual FoxPro puede verse afectado negativamente por identificadores de memoria bloqueados.

**Precaución** El uso excesivo de memoria dinámica priva a Visual FoxPro de memoria para búferes, ventanas, menús, etc., y disminuye el rendimiento, porque la memoria dada para las solicitudes de la API la administra el administrador de memoria de Visual FoxPro. Asignar grandes identificadores y conservarlos puede causar que la memoria de Visual FoxPro se agote y el programa termine de forma anómala.

El entorno de Visual FoxPro no tiene protección de memoria. La rutina de la API externa no puede proporcionar toda la validación inherente a un programa estándar de Visual FoxPro. Si daña la memoria, recibirá mensajes como "Identificador transgredido", "Error de coherencia interna" y "Nodo transgredido durante la compactación".

La siguiente función de una biblioteca FLL ilustra la asignación de memoria. El ejemplo usa [\\_RetDateStr\(\)](#) para devolver un tipo Date de Visual FoxPro (asumiendo que el parámetro Character es una fecha adecuada):

```
#include <Pro_ext.h>

void dates(ParamBlk *parm)
{
    MHANDLE mh;
    char *instring;

    if ((mh = _AllocHand(parm->p[0].val.ev_length + 1)) == 0) {
        _Error(182); // "Memoria insuficiente "
    }
    _HLock(parm->p[0].val.ev_handle);
    instring = _HandToPtr(parm->p[0].val.ev_handle);
    instring[parm->p[0].val.ev_length] = '\0';
    _RetDateStr(instring);
    _HUnlock(parm->p[0].val.ev_handle);
}

FoxInfo myFoxInfo[] = {
    {"DATES", (FPFI) dates, 1, "C"}
};

FoxTable _FoxTable = {
    (FoxTable *) 0, sizeof(myFoxInfo)/sizeof(FoxInfo), myFoxInfo
};
```

## Descripción de las pilas

El control o biblioteca que crea no tiene pila propia. En lugar de ello, usa la pila del programa que lo llama, en este caso la pila de Visual FoxPro. No puede controlar el tamaño de la pila de Visual FoxPro o modificar la cantidad de espacio disponible para un control ActiveX o un archivo .fl.

En circunstancias normales, esta distinción no es importante. La pila de Visual FoxPro es generalmente suficientemente grande para guardar las variables automáticas que puede necesitar para asignar en un control o biblioteca. Si se agota el espacio de la pila, siempre puede asignar memoria adicional al montón de una manera dinámica.

## Seguir reglas de identificador

Las reglas siguientes se aplican a la propiedad de identificadores y la responsabilidad de liberarlos:

- Los usuarios pueden liberar todos los identificadores que asignen, incluyendo los identificadores asignados por funciones como [\\_Load\(\)](#).
- [\\_Load\(\)](#) sólo crea un identificador cuando la variable que está creando es una cadena de caracteres (es decir, `ev_type = 'C'`). Todos los demás tipos de datos almacenan sus valores en la misma estructura Value, mientras que al cargar una cadena de caracteres se coloca un MHANDLE en el `ev_handle` de la estructura Value.

- En la biblioteca FLL, Visual FoxPro asume la responsabilidad de liberar todos los identificadores devueltos por [\\_RetVal\(\)](#). Los usuarios no pueden liberar estos identificadores, incluso si los asignan.
- Los usuarios no deben liberar identificadores que se les han transferido a su ParamBlk.

**Precaución** Cuando escribe una rutina externa que llama a funciones, asegúrese de seguir todas las reglas y compruebe los resultados devueltos. Un puntero perdido o una referencia de identificador podría dañar las estructuras de datos internas de Visual FoxPro, provocando el final inmediato o problemas a posteriori, lo cual podría producir pérdida de datos.

## Generar y probar bibliotecas y controles ActiveX

Después de crear un proyecto, ya está preparado para generarlo y depurarlo.

### Generar el proyecto

Antes de generar, tiene que establecer las opciones del proyecto. Algunas de las opciones que elige dependen de si desea crear una versión de depuración o una versión final del control o biblioteca. Por norma, puede crear versiones de depuración del programa hasta que esté satisfecho de su funcionamiento y, a continuación, crear una versión final.

#### Parar especificar una versión de depuración o final

1. En el menú **Build**, elija **Set Default Configuration**.
2. Elija si va a crear una versión de depuración o una versión final del control.
3. Elija **OK**.

#### Para establecer las opciones del proyecto

1. En el menú **Build**, elija **Settings**.
2. En **Settings For**, elija si va a crear una versión de depuración o final del programa.
3. Haga clic en la ficha **C/C++** y, a continuación, elija estas opciones:
  - En la lista **Category**, elija **Code Generation**.
  - En la lista **Calling Convention**, elija **\_fastcall**.
  - En la lista **Use run-time library**, elija **Multithreaded DLL**.
4. Elija la ficha **Link** y, a continuación, en el cuadro de texto **Object/Library Modules**, agregue una de las bibliotecas siguientes:
  - Si está generando un .ocx, agregue `OCXAPI.LIB` del directorio API de Visual FoxPro.
  - Si está generando un .flx, agregue `WINAPIMS.LIB` del directorio API de Visual FoxPro.
5. Desactive **Ignore all default libraries**.

## 6. Elija **OK**.

### Para asegurarse de que el compilador puede encontrar los archivos necesarios

1. En el menú **Tools**, elija **Options**.
2. Haga clic en la ficha **Directories**.
3. En la lista **Show directories for**, elija **Include files**.
4. En la barra de herramientas **Directories**, haga clic en el botón **Add**.
5. Agregue el directorio que contiene 0Pro\_ext.h.
6. En la lista **Show directories for**, elija **Library files**.
7. En la barra de herramientas **Directories**, haga clic en el botón **Add**.
8. Agregue el directorio que contiene OCXAPI.LIB del directorio API de Visual FoxPro (al crear un control) o agregue WINAPIMS.LIB del directorio API de Visual FoxPro (al crear un FLL).
9. En el cuadro de diálogo **Options**, elija **OK**.

Después de especificar las opciones, puede compilar y enlazar el programa.

### Para compilar y enlazar un archivo .ocx

- En el menú **Build**, elija **Build *nombreproyecto.ocx***.

Cuando compile y enlace el archivo .OCX, Visual C++ registrará automáticamente el control en el equipo en el que se generó. Si por alguna razón tiene que registrar el control de otra forma, puede hacerlo mediante el siguiente procedimiento.

### Para registrar el control ActiveX

- En el menú **Tools** de Visual C++ Developer Studio, elija **Register Control**.  
–O bien–
- Declare y llame a `DLLRegisterServer()` desde su programa.

### Depurar un control ActiveX Control o una biblioteca FLL

Depurar el control o la biblioteca en el contexto de una aplicación de Visual FoxPro es más difícil que depurar de forma independiente de la aplicación. Es una buena idea crear un programa de prueba sencillo para probar el funcionamiento del control o la biblioteca.

### Depurar con Microsoft Visual C++

Microsoft Visual C++ versión 4.0 o posterior ofrece un entorno integrado de depuración que facilita el establecimiento de puntos de interrupción para ejecutar el código paso a paso. Puede incluso ejecutar Visual FoxPro desde Visual C++.

### Para iniciar la depuración con Microsoft Visual C++

1. En el menú **Build**, elija **Settings**.
2. En el cuadro de diálogo **Project Settings**, haga clic en la ficha **Debug**.
3. en el cuadro de texto **Executable for debug session**, escriba la ruta de acceso seguida por Vfp6.exe.

Por ejemplo, escriba **C:\Archivos de programa\Microsoft Visual Studio\Vfp98\Vfp6.exe**.

4. Elija **OK**.
5. Establezca un punto de interrupción en la biblioteca.
6. En el menú **Build**, elija **Debug**. A continuación, elija **Go** en el submenú.
7. Cuando Developer Studio muestra un mensaje que dice "Vfp6.exe does not contain debugging information", elija **Yes** para continuar.

Para obtener más información sobre depuración en Visual C++, consulte el conjunto de documentación de Visual C++.

### Depurar con otros depuradores

Debería poder depurar un control o biblioteca con cualquier depurador que controle correctamente un INT 3 ([\\_BreakPoint\(\)](#)) incrustado en su programa. Puede usar cualquier depurador para depuración simbólica siempre que pueda hacer lo siguiente:

- Haga una tabla de símbolos a partir de un archivo de asignaciones.
- Cargue la tabla de símbolos independiente del programa.
- Vuelva a colocar los símbolos en una nueva dirección.

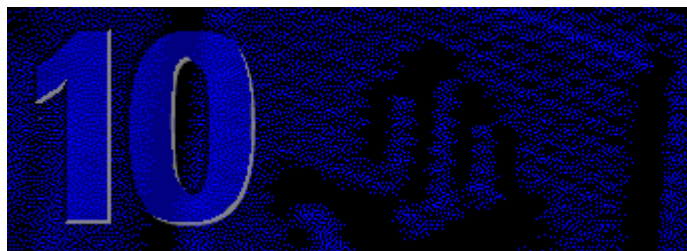
### Para depurar una biblioteca

1. Agregue una llamada [\\_BreakPoint\(\)](#) a la rutina en el punto en el que comienza la depuración.
2. Genere el control o biblioteca.
3. Llame al depurador.
4. Si el depurador admite símbolos, cargue la tabla de símbolos para la biblioteca.
5. Inicie Visual FoxPro.

6. Llame a la rutina de biblioteca desde Visual FoxPro.
7. Cuando se alcance el punto de interrupción, realice ajustes a la base de símbolos para alinear los símbolos con la ubicación actual en la que se cargó la biblioteca.
8. Incremente el registro del puntero de instrucción (IP) en una unidad para saltar la instrucción INT 3.
9. Siga depurando como si fuera un programa normal.

**Nota** Elimine siempre todos los puntos de interrupción especificados en el depurador antes de distribuir el producto.

# Manual del programador, Parte 10: Crear soluciones empresariales



Con las características de Visual FoxPro puede extender sus esfuerzos de programación para crear aplicaciones complejas orientadas a múltiples usos. Puede crear sus aplicaciones con un equipo de programadores, lo que le permite trabajar más rápidamente y programar aplicaciones que difícilmente puede crear un sólo programador. También puede combinar la eficacia de Visual FoxPro con la de otros programas para crear soluciones eficaces a nivel empresarial para los requisitos de su aplicación.

## Capítulo 29 [Programar en equipo](#)

Para trabajar en equipo con éxito, los programadores deben coordinar sus esfuerzos y evitar la duplicación del esfuerzo o sobrescribir el trabajo de otros. Para ayudarle a administrar la programación en equipo, Visual FoxPro le permite integrar software de control de código de origen en el Administrador de proyectos, de forma que pueda desproteger y proteger archivos de Visual FoxPro, combinar modificaciones, ver diferencias y más. También puede trabajar simultáneamente con otros programadores en la misma base de datos.

## Capítulo 30 [Soluciones empresariales de Visual FoxPro](#)

Además de crear completamente sus aplicaciones en Visual FoxPro, puede extender Visual FoxPro para usarlo como cliente de otros orígenes de datos y como origen de datos de otros programas para Windows. También puede usar Visual FoxPro de otras formas innovadoras, como motor de búsqueda para el World Wide Web o como herramienta de almacenamiento de datos.

# Capítulo 29: Programar en equipo

Puede crear aplicaciones complejas rápidamente combinando las habilidades de un equipo de programadores. Sin embargo, la programación en equipo requiere coordinación adicional para que funcione correctamente el esfuerzo de programación. Una estrategia es usar software de control de código fuente, como Microsoft Visual SourceSafe™, para administrar los archivos de un proyecto.

Este capítulo proporciona estrategias que puede seguir para hacer que la programación en equipo sea un éxito. Se supone que ya está familiarizado con la creación de aplicaciones de Visual FoxPro, como se ha tratado en capítulos anteriores de este libro.

Este capítulo incluye información sobre:



- [Descripción de la programación en equipo](#)
- [Trabajar con software de control de código fuente en Visual FoxPro](#)
- [Administrar proyectos de Visual FoxPro con control de código fuente](#)
- [Administrar archivos en un proyecto con control de código fuente](#)
- [Programar y modificar bases de datos en equipo](#)
- [Programar bibliotecas de clases en equipo](#)

## Descripción de la programación en equipo

Si trabaja con un equipo de programadores puede crear aplicaciones más rápido y puede programar aplicaciones más complejas. Puede combinar las habilidades de distintos programadores para crear aplicaciones que difícilmente podría crear un único programador.

Sin embargo, la programación en equipo requiere esfuerzo adicional en el proceso de programación. La programación en equipo exitosa depende de:

- Permitir a varios programadores trabajar con los mismos proyectos y bases de datos a la vez.
- Coordinar las modificaciones que se realizan en los mismos programas, formularios u otros elementos de la aplicación, de forma que las modificaciones de un programador no sobrescriban las de otro programador.
- Permitir a los programadores mejorar los elementos existentes de la aplicación (por ejemplo, programas o bibliotecas de clases) sin que afecte al trabajo de los otros programadores que usan actualmente estos elementos.

Por ejemplo, suponga que su equipo está programando una aplicación compleja. Como la aplicación es grande, Visual FoxPro debe permitir a varios programadores que trabajen simultáneamente en distintos componentes de la aplicación. Sin embargo, quiere estar seguro de que sólo trabaja un programador a la vez en un elemento individual, como un formulario, de forma que un programador no sobrescriba las modificaciones realizadas por otro programador.

Más aún, quiere que el programador pueda escribir código, probar y depurar un formulario sin que afecte a los otros programadores (y usuarios) que siguen trabajando con una versión anterior del formulario. Cuando el primer programador haya terminado el nuevo formulario, las mejoras se pueden integrar en la aplicación.

Puede seguir los métodos recomendados en este capítulo para coordinar el trabajo de varios programadores. Por ejemplo, este capítulo proporciona información sobre cómo trabajar con proyectos y bibliotecas de clases en un entorno de varios programadores. Para obtener detalles, consulte [Integrar control de código fuente en proyectos de Visual FoxPro](#) y [Programar bibliotecas de clases en equipo](#) más adelante en este capítulo.

## Descripción del control de código fuente

Visual FoxPro proporciona varias características que dan soporte a la programación en equipo. Una característica importante de la programación en equipo es el uso de un sistema de control de código fuente para coordinar quién puede tener acceso y modificar los archivos de un proyecto.

Control de código fuente es el término genérico para herramientas que administran archivos en un entorno de varios programadores. La mayor parte de las herramientas de control de código funcionan como una biblioteca pública tradicional, manteniendo un depósito central de archivos (documentos, programas u otros archivos) en una ubicación accesible a todos los programadores. Además, las herramientas de control de código fuente incluyen la capacidad de seguir las modificaciones que los programadores hacen en los archivos y volver a versiones anteriores si es necesario.

En general, las herramientas de control de código fuente proporcionan algunas o todas estas características:

- **Desproteger, proteger** Los programadores desprotegen un archivo transfiriendo una copia desde el depósito central a sus equipos locales antes de modificarlo. Como regla, mientras un archivo está desprotegido, otros programadores no pueden desprotegerlo o modificarlo, pero generalmente pueden verlo sincronizando u obteniendo una copia de sólo lectura del archivo. (Si el archivo es un archivo de texto, como el código fuente de un programa, es posible que varios programadores desprotejan el mismo archivo y, a continuación, combinen las modificaciones de otros con su copia local). Cuando los programadores hayan terminado con un archivo, pueden proteger sus modificaciones protegiendo o transfiriendo su copia local al depósito central. Como parte del proceso de protección de un archivo, la mayor parte de las herramientas de control de código fuente piden al programador que escriba comentarios sobre las modificaciones realizadas al archivo.
- **Combinación** Para permitir a varios programadores trabajar simultáneamente en el mismo archivo, el software de control de código fuente permite a varios programadores desproteger el archivo a la vez. (Esto sólo se puede hacer generalmente para archivos de texto como el código fuente de un programa). Si otro programador ha modificado el archivo, el sistema de control de código fuente puede integrar las modificaciones en su versión del archivo.
- **Control de proyectos** Los programadores pueden organizar archivos en sus proyectos u otras categorías específicas de trabajo. Si es necesario, varios proyectos pueden compartir los archivos.
- **Seguimiento de modificaciones** La mayor parte de los sistemas de control de código fuente hacen un seguimiento de las modificaciones realizadas en un archivo cuando se protege. Esto permite a los programadores reconstruir versiones anteriores del archivo, lo cual es útil para recuperar trabajo anterior.
- **Comprobación de diferencias** El software de control de código fuente permite a los programadores comparar versiones de un archivo y revisar las diferencias entre ellos.
- **Historial** Los programadores pueden examinar el historial de protección para cada archivo, incluyendo los comentarios realizados por cada programador al proteger el archivo.

**Sugerencia** Si el software de control de código fuente admite comentarios, aproveche esta característica. Los comentarios pueden ser de gran ayuda en el proceso de hacer un seguimiento y proporcionan un historial útil de la programación de la aplicación.

Para usar el control de código fuente, los usuarios deben combinar un proyecto con control de código fuente (a veces se llama "agregar un usuario" a un proyecto). Cuando los usuarios se han unido a un proyecto, pueden desproteger y proteger los archivos que pertenecen al proyecto.

**Nota** Tiene que activar el sistema de control de código fuente para desproteger varias veces el mismo archivo, para que varios programadores puedan trabajar simultáneamente en un proyecto. Para

obtener más detalles, consulte la documentación del software de control de código fuente.

## Trabajar con software de control de código fuente en Visual FoxPro

Uno de los aspectos más importantes de la programación en equipo es la capacidad de controlar quién puede modificar archivos. Por ejemplo, si no hay control sobre los archivos y si más de un programador está modificando un programa a la vez, hay una probabilidad muy alta de que un conjunto de modificaciones sea sobrescrito o descartado, desperdiciando tiempo y esfuerzo.

Visual FoxPro ayuda a su equipo a administrar los archivos de sus proyectos permitiéndole integrar un sistema de control de código fuente en el Administrador de proyectos de Visual FoxPro. Así, puede administrar archivos de proyecto en un entorno de programación en equipo y asegurarse de que los esfuerzos de programación se llevan a cabo correctamente.

### Integrar el control de código fuente en proyectos de Visual FoxPro

Visual FoxPro admite herramientas de control de código fuente permitiéndole integrar software de control de código fuente disponible comercialmente en sus proyectos. Puede usar muchos de los sistemas de control de código de versiones disponibles actualmente. (Póngase en contacto con el proveedor de software para averiguar si el software se puede integrar con las herramientas de programación de Microsoft). Por ejemplo, si su equipo de programadores ya usa Microsoft Visual SourceSafe, puede especificarlo como el software de control de código fuente que va a usar con Visual FoxPro.

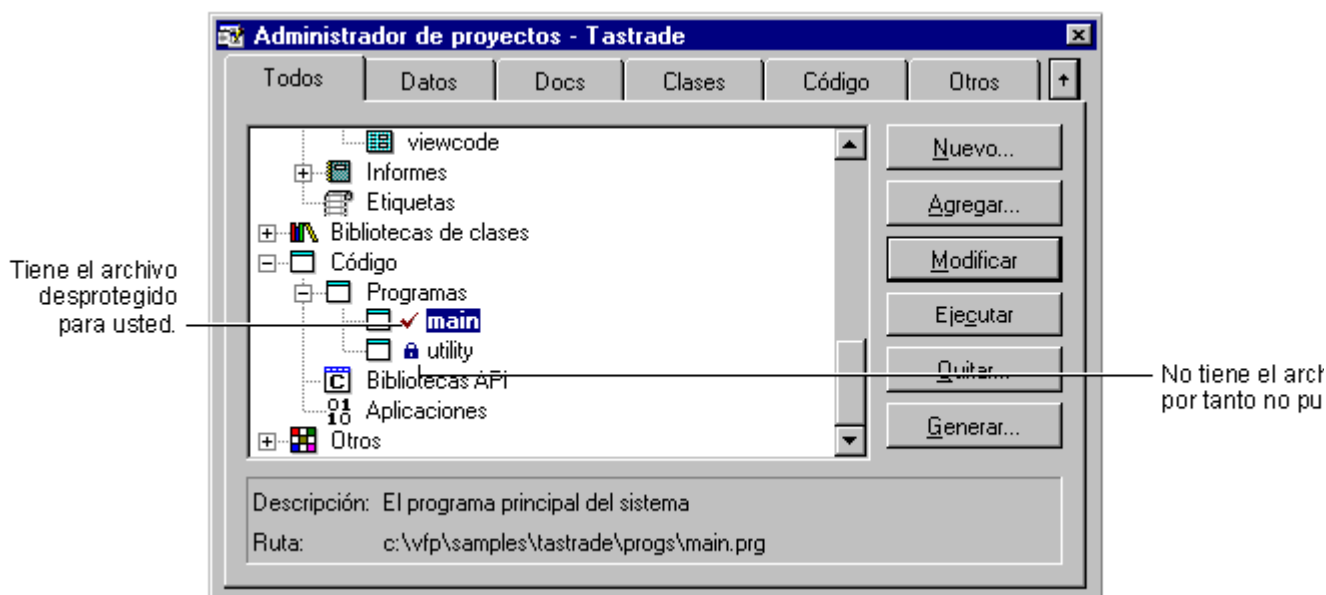
Todo el control de código fuente en Visual FoxPro se administra a través del Administrador de proyectos. Cuando configura un proyecto en Visual FoxPro, tiene la opción de crear un proyecto de control de código fuente, lo que se suele llamar "poner el proyecto bajo control de código fuente". Después de haber puesto un proyecto bajo control de código fuente, Visual FoxPro le ayuda a administrar los archivos del proyecto con control de código fuente. Cuando desee modificar un archivo (por ejemplo, si modifica un programa o un formulario) Visual FoxPro le pedirá que desproteja el archivo.

En Visual FoxPro, el control de código fuente se usa para administrar archivos de todos los tipos, no sólo archivos .prg, sino también archivos .scx, .frx, .lbx, .mnx y .vcx, entre otros. Aunque los archivos individuales se pueden compartir entre distintos proyectos de Visual FoxPro, todas las operaciones de control de código fuente se realizan en archivos dentro del contexto de un proyecto concreto.

**Nota** Visual FoxPro no le pide que coloque tablas de datos como archivos .dbf y .dbc bajo control de código fuente al crearlas, pero puede agregarlas manualmente al proyecto con control de código fuente.

Cuando trabaja en el Administrador de proyectos con un proyecto que tiene control de código fuente, Visual FoxPro muestra iconos junto a los archivos que están bajo control de código fuente para indicar su estado.

### Iconos de control de código fuente en el Administrador de programas



La tabla siguiente resume los iconos usados en el Administrador de programas para indicar el estado de control de código fuente.

Icono	Significado
✓	El archivo está desprotegido para usted.
✓	El archivo está desprotegido para usted y para uno o más programadores además de usted.
👤	El archivo está desprotegido para otro programador.
🔒	El archivo no está desprotegido; no puede modificarlo hasta que lo haya desprotegido.
⬆️	Se ha combinado el archivo. Después de examinar las modificaciones, puede proteger el archivo.
⬆️✖️	Se ha combinado el archivo y hay que resolver conflictos.
?	Visual FoxPro no puede determinar el estado de control de código fuente del archivo.

Si un archivo no está bajo control de código fuente, no aparece ningún icono asociado.

**Nota** Para obtener detalles sobre la combinación de archivos y conflictos de combinación, consulte [Proteger archivos de texto](#) más adelante en este capítulo.

## Activar el control de código fuente

Para activar el control de código fuente, instale en primer lugar el programa de control de código fuente de acuerdo con la documentación suministrada con él. Típicamente, se instalará una versión

administrador en un servidor en el que se guarda el código fuente y se instalan versiones cliente del producto en los equipos locales.

**Nota** Todos los programadores de un proyecto tienen que usar el mismo software de control de código fuente.

Después de instalar el software de control de código fuente, puede establecer opciones de modo que Visual FoxPro lo reconozca y especifique valores predeterminados para sus proyectos.

### Para activar el control de código fuente en Visual FoxPro

1. En el menú **Herramientas**, elija **Opciones**.
2. En el cuadro de diálogo [Opciones](#), elija la ficha **Proyectos**.
3. En el área **Opciones del control de código fuente**, seleccione el nombre del programa de control de código fuente en la lista **Activar el proveedor de control de código**.
4. Para que Visual FoxPro le pida que agregue nuevos proyectos al control de código fuente, seleccione **Agregar automáticamente nuevos proyectos al control de código fuente**.

Cada vez que inicie Visual FoxPro, éste comprobará si hay un proveedor de control de código fuente. Si encuentra uno, puede colocar o administrar proyectos bajo control de código fuente.

## Administrar proyectos de Visual FoxPro bajo control de código fuente

Para usar software de control de código fuente en Visual FoxPro, tiene que colocar los proyectos bajo control de código fuente, agregar archivos a los proyectos que tienen control de código fuente y actualizar la lista de proyecto para cada proyecto.

### Trabajar con el archivo de proyecto y el archivo lista de proyecto

En Visual FoxPro, la información de proyecto se guarda en un conjunto de archivos de tabla y archivos memo con las extensiones .pjx y .pjt. Por ejemplo, si ha creado un proyecto llamado "MiProy", la información sobre el proyecto, incluyendo la lista de archivos, su ubicación y si están compilados en el archivo de aplicación (archivo .app o .exe), se almacena en los archivos llamados Miproj.pjx y Miproj.pjt.

Al trabajar en un entorno de programación en equipo, los programadores no comparten los mismos archivos de proyecto (archivos .pjx y .pjt). En lugar de ello, guardan sus propias copias locales de los archivos .pjx y .pjt.

Para coordinar las modificaciones que los programadores individuales realizan a un proyecto bajo control de código fuente, Visual FoxPro guarda una lista de archivos de proyecto (o archivo .pjm, abreviatura de "metarchivo de proyecto"). El archivo que contiene la lista de archivos de proyecto es un archivo de texto que almacena la misma información que los archivos .pjx y .pjt, como los archivos que están incluidos actualmente en el proyecto.

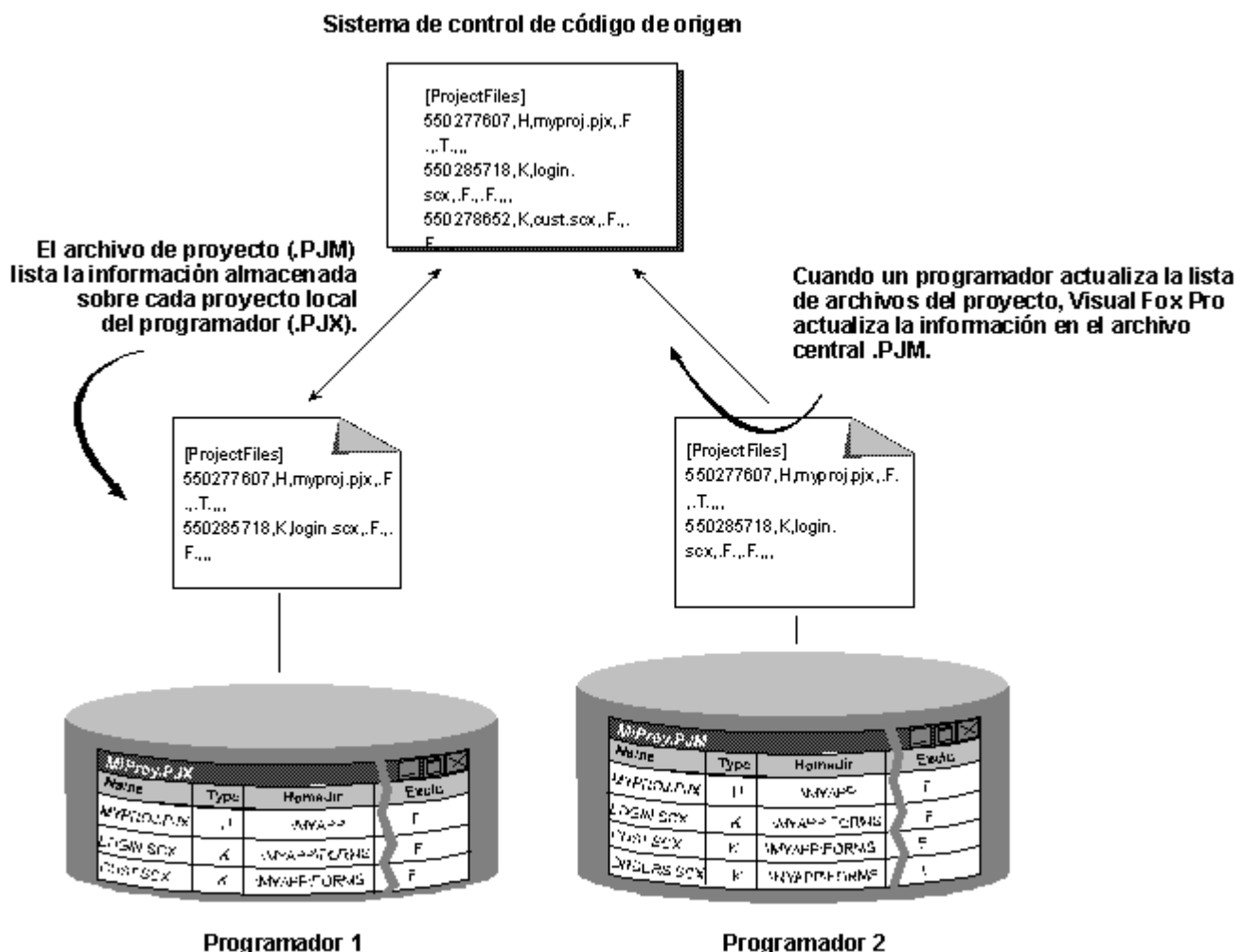
El software de control de código fuente guarda una lista de archivos de proyecto central almacenada con los otros archivos en el depósito central. Además, cada programador tiene una copia local de la lista de archivos de proyecto desprotegida que refleja su versión actual del proyecto.

Suponga que está trabajando en un proyecto y que va a agregar un programa nuevo (archivo .prg). Cuando agregue el nuevo archivo (y suponiendo que coloca este archivo bajo control de código fuente), Visual FoxPro actualizará la copia local del proyecto y mostrará el archivo cuando use el Administrador de proyectos en el equipo. Los otros programadores no conocen al principio la modificación, y sus copias locales del proyecto no muestran el archivo que ha agregado. Incluso si no ha actualizado la lista de archivos de proyecto, puede proteger el nuevo archivo por seguridad y desprotegerlo cuando sea necesario.

Cuando haya terminado de trabajar con el nuevo archivo, por ejemplo, cuando haya terminado de probar el nuevo programa, puede actualizar la lista de archivos de proyecto. Cuando lo haga, Visual FoxPro combina la información de la lista de archivos de proyecto local con la lista de archivos de proyecto central.

A cambio, Visual FoxPro actualiza su lista de archivos de proyecto local con las modificaciones que encuentre en la lista de archivos del proyecto central. Si otros programadores han agregado archivos al proyecto, se actualiza su lista de archivos de proyecto local, se colocan copias locales de los nuevos archivos en su equipo, Visual FoxPro vuelve a generar el proyecto (archivos .pjx y .pjt) y el Administrador de proyectos muestra los archivos agregados para que trabaje con ellos.

### **Administrar archivos de proyecto mediante la lista de proyecto**



**Nota** La lista de archivos de proyecto sólo hace un seguimiento de los archivos de proyecto que están explícitamente bajo control de código fuente. Si su proyecto incluye archivos que no están bajo control de código fuente, no aparecerán en la lista de archivos de proyecto y Visual FoxPro no agregará estos archivos a los proyectos de otros programadores cuando actualicen sus propias listas de proyecto.

## Colocar proyectos bajo control de código fuente

Tiene que especificar que un proyecto de Visual FoxPro va a estar bajo control de código fuente antes de poder usar el software de control de código fuente. Esto se hace agregando un proyecto al sistema de control de código fuente.

Si el software de control de código fuente está activado, puede especificar que cualquier proyecto nuevo que cree estará automáticamente bajo control de código fuente.

### Para crear un proyecto nuevo con control de código fuente

1. En el cuadro de diálogo [Opciones](#), elija la ficha **Proyectos** y, a continuación, seleccione un proveedor de control de código fuente si no lo ha hecho ya.

2. Asegúrese de que **Agregar automáticamente nuevos proyectos al control de código** está seleccionado y, a continuación, elija **Aceptar**. Para hacer que esta opción sea la predeterminada, elija **Establecer como predeterminado** y, a continuación, elija **Aceptar**.
3. En el menú **Archivo**, elija **Nuevo** y, a continuación, inicie un proyecto nuevo de Visual FoxPro.

Después de haber dado un nombre al proyecto nuevo, Visual FoxPro le pedirá que cree el nuevo proyecto con control de código fuente. El nombre predeterminado para el nuevo proyecto será el mismo que el nombre del proyecto de Visual FoxPro.

Después de haber creado el nuevo proyecto con control de código fuente, Visual FoxPro termina de crear el nuevo proyecto. Antes de que otros programadores puedan usar este archivo, tiene que agregarlos al proyecto. Para obtener detalles, consulte [Unirse a un proyecto existente con control de código fuente](#) más adelante en este capítulo.

Si está trabajando con un proyecto existente que no está aún bajo control de código fuente, puede crear un nuevo proyecto con control de código fuente y, a continuación, colocar sus archivos bajo control de código fuente.

### Para poner un proyecto existente bajo control de código fuente

1. Abra el proyecto de Visual FoxPro en el [Administrador de proyectos](#).
2. En el menú **Proyecto**, elija **Agregar proyecto a control de código**.

Visual FoxPro muestra el cuadro de diálogo para el sistema de control de código fuente que le permite crear un proyecto nuevo. De forma predeterminada, el nombre del proyecto con control de código fuente es el mismo que el del proyecto de Visual FoxPro.

3. Cree el proyecto con control de código fuente de la misma forma que lo hace normalmente con el software de control de código fuente.

Cuando agregue un proyecto existente al control de código fuente, Visual FoxPro le pedirá que agregue los archivos del proyecto al proyecto con control de código fuente. Para obtener detalles, consulte [Agregar archivos a un proyecto con control de código fuente](#) en la siguiente sección.

### Agregar archivos a un proyecto con control de código fuente

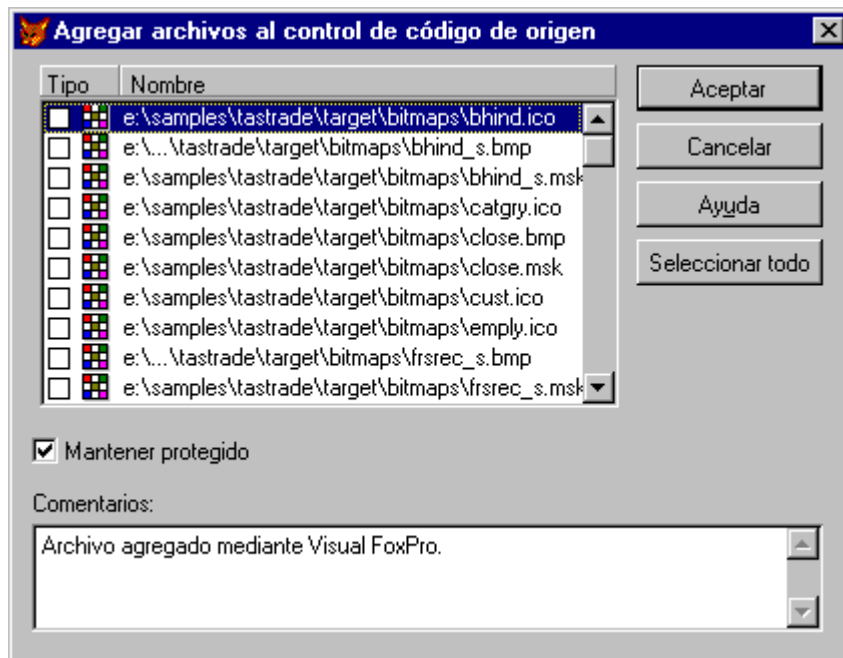
Después de poner un proyecto de Visual FoxPro bajo control de código fuente, puede agregar archivos individuales al proyecto bajo control de código fuente. Si el proyecto de Visual FoxPro ya contiene archivos cuando lo pone bajo control de código fuente, podrá agregarlos al proyecto bajo control de código fuente.

**Nota** Si el software de control de código fuente lo admite, Visual FoxPro le permite mantener el archivo desprotegido cuando lo agregue al proyecto con control de código fuente. Si no es así, se protege el archivo y tiene que volver a desprotegerlo para trabajar con él. Para obtener detalles sobre desprotección y protección de archivos cuando se agregan al proyecto, consulte [Administrar archivos en un proyecto con control de código fuente](#) más adelante en este mismo capítulo.



**Para agregar archivos existente a un proyecto controlado**

1. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación elija **Agregar archivos al control de código**.
2. En el cuadro de diálogo [Agregar archivos al control de código](#), seleccione los archivos que desea agregar.



**Nota** **Mantener desprotegido** y **Comentario** sólo aparecen si el software de control de código fuente admite estas opciones.

3. Elija **Aceptar**.

Visual FoxPro genera los archivos de control necesarios para el software de control de código fuente y, a continuación, agrega los archivos al proyecto. Si ha seleccionado muchos archivos, este proceso puede durar cierto tiempo.

Puede configurarlo de forma que cuando agregue un archivo a un proyecto, Visual FoxPro le pedirá que lo ponga bajo control de código fuente.

### Para especificar que Visual FoxPro le pida que coloque archivos nuevos bajo control de código fuente

- En la ficha **Proyectos** del cuadro de diálogo [Opciones](#), asegúrese de que **Agregar automáticamente nuevos proyectos al control de código** está seleccionado y, a continuación, elija **Aceptar**.

Para hacer que esta opción sea predeterminada, elija **Establecer como predeterminado** y, a continuación, elija **Aceptar**.

Tras agregar archivos a un proyecto, tiene que actualizar la lista de proyectos antes de que otros programadores puedan trabajar con los nuevos archivos. Para obtener detalles, consulte [Actualizar la lista de proyectos](#) más adelante en este capítulo.

### Unirse a un proyecto existente con control de código fuente

Si es un programador nuevo de un proyecto que ya está bajo control de código fuente, debe unirse al proyecto antes de poder desproteger y proteger archivos. Cuando se une a un proyecto, Visual FoxPro crea un archivo lista de proyecto local y genera un archivo de proyecto actual (archivo .PJX) para usted.

#### Para unirse a un proyecto existente

1. En el menú **Archivo**, elija **Unirse a proyecto de control de código fuente**.
2. En el cuadro de diálogo **Abrir proyecto**, seleccione el servidor y el directorio que contienen el archivo de proyecto de Visual FoxPro al que desea unirse.
3. Establezca el directorio de trabajo en su equipo local; especifique dónde colocará el sistema de control de código fuente los archivos cuando los proteja y dónde los buscará cuando los vuelva a proteger. Por ejemplo, si está usando Visual SourceSafe como proveedor de control de código fuente, elija **Examinar** en el área **Directorio** y seleccione el directorio existente, o escriba el nombre de un directorio nuevo.

**Sugerencia** Todos los programadores de un proyecto deben usar la misma estructura de directorios para los archivos de un proyecto, aunque los nombres de los subdirectorios individuales pueden variar.

### Actualizar la lista de proyecto

Incluso después de haber agregado los archivos al proyecto con control de código fuente, otros programadores no podrán trabajar con ellos. Los programadores podrán usar manualmente su sistema de control de código fuente para desproteger y proteger archivos si lo necesitan, pero los archivos agregados no se mostrarán en el Administrador de proyectos para cualquier programador excepto para el programador que ha agregado los archivos. Para poner los archivos a la disposición de otros

programadores, actualice la lista de proyecto.

Cuando actualice la lista de proyecto, Visual FoxPro:

- Generará una nueva lista de archivo de proyecto local (archivo .pjm).
- Protegerá la nueva lista de archivo de proyecto (con la opción establecida para conservar el archivo desprotegido).
- Combinará las lista de archivos de proyecto local y central si hay diferencias. Si ocurre un conflicto de combinación, Visual FoxPro mostrará un cuadro de diálogo para ayudarle a resolver los conflictos de combinación.
- Volverá a generar el archivo de proyecto local (.pjx) basado en la lista de archivos de proyecto combinada.
- Obtendrá copias locales de archivos agregados al proyecto por otros programadores.
- Le indicará que obtenga las versiones más recientes de los archivos de proyecto.
- Actualizará la presentación en el Administrador de proyectos para reflejar los cambios.

### Para actualizar la lista de proyecto

- En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Actualizar lista de proyecto**.

Como parte de los procedimientos de actualización, Visual FoxPro le pedirá que obtenga las versiones más recientes de los archivos. Si ya tiene un archivo desprotegido, como regla general no debe obtener la versión más reciente, porque su versión es con seguridad más actual que una de la red.

Si va a obtener la versión más reciente de un archivo de texto (como un programa), el software de control de código fuente puede intentar combinar las últimas modificaciones con su versión. Para obtener más información sobre la combinación de archivos de texto, consulte [Proteger archivos](#) más adelante en este capítulo.

Cuando haya terminado, los otros programadores también deberían actualizar su lista de proyecto (con el mismo procedimiento) para poder trabajar con los archivos que usted haya agregado.

### Quitar un proyecto de control de código fuente

Si ya no quiere controlar los archivos de un proyecto, puede quitar el proyecto del control de código fuente. Cuando lo haga, los archivos permanecerán en el proyecto con control de código fuente de forma que otros programadores puedan seguir usándolos y para que usted pueda examinar su historial o usarlos en otros proyectos.

Si tiene archivos de proyecto en su equipo etiquetados como de sólo lectura, es decir, si tiene copias de los archivos pero éstos no están desprotegidos, puede quitarles el atributo de sólo lectura cuando se quitan del proyecto del control de código fuente.

**Nota** Cuando usted quita un proyecto del control de código fuente, rompe el vínculo entre sus archivos de proyecto locales y el proyecto con control de código fuente, y sus archivos se convierten en archivos de lectura-escritura. Asegúrese de aplicar procedimientos de control manual de versiones

después de quitar un proyecto o correrá el riesgo inherente al trabajo con archivos que no están bajo control de código fuente.

### Para quitar un proyecto de control de código fuente

1. Proteja todos los archivos bajo control de código fuente.
2. En el menú Proyecto, elija **Quitar proyecto de control de código**.

### Quitar archivos de un proyecto con control de código fuente

Puede quitar archivos individuales del control de código fuente si ya no quiere que formen parte de su proyecto con control de código fuente. Podría hacer esto, por ejemplo, si un programa o formulario se convierte en innecesario y ya no forma parte del proyecto.

### Para quitar un archivo del control de código fuente

1. En el [Administrador de proyectos](#), seleccione el archivo que desea quitar.
2. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Quitar archivos de control de código**.
3. En el cuadro de diálogo [Quitar archivos de control de código fuente](#), seleccione los archivos que desea quitar y, a continuación, haga clic en **Aceptar**.

Si quita un archivo de un proyecto de Visual FoxPro que tiene control de código fuente, Visual FoxPro le pedirá como siempre si sólo desea quitar el archivo del proyecto o si desea eliminarlo del disco. Una opción del cuadro de diálogo Opciones determina si Visual FoxPro también le pide que quite el archivo del proyecto con control de código fuente.

- Si **Quitar archivos del control de código fuente al quitarlos del proyecto** está activada, Visual FoxPro también le pedirá que quite el archivo del proyecto con control de código fuente.
- Si **Quitar archivos del control de código fuente al quitarlos del proyecto** no está activada, no se le pide lo anterior y el archivo se deja bajo control de código fuente.

Después de quitar un archivo del control de código fuente, es posible que aún existan copias del mismo en los equipos de otros programadores. Si es así, el archivo se tratará como un archivo local sólo para estos programadores.

### Compartir archivos entre proyectos con control de código fuente

Puede configurar un archivo de forma que forme parte de dos o más proyectos con control de código fuente a la vez. Esto es útil si usa archivos comunes, como programas estándar, bibliotecas o cuadros de diálogo en más de un proyecto. Cuando comparte archivos entre proyectos, las modificaciones que proteja en un archivo se reflejan en todos los archivos que compartan el archivo.

El método específico para compartir archivos entre proyectos con control de código fuente depende del software de control de código fuente. Si las opciones para compartir no están incorporadas en su proveedor de control de código fuente, los comandos para compartir archivos no estarán disponibles

en el menú.

El primer paso del procedimiento siguiente se aplica a todos los sistemas de control de código fuente que admiten la posibilidad de compartir archivos. Los pasos sucesivos pueden variar, en función de su software de control de código fuente.

### Para compartir archivos entre proyectos controlados

1. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, **Compartir archivos**.
2. En el cuadro de diálogo que aparece, indique qué archivos desea compartir con el proyecto actual y a qué proyecto pertenecen actualmente.

Las opciones específicas disponibles en este comando de menú dependen del sistema de control de código fuente. Para obtener detalles, elija **Ayuda** en el cuadro de diálogo **Compartir** o consulte la documentación de su sistema de control de código fuente.

**Sugerencia** En Microsoft Visual SourceSafe puede ver a qué proyectos pertenece un archivo mediante el comando **Propiedades del proyecto** y al elegir la ficha **Vínculos**.

## Administrar archivos de un proyecto con control de código fuente

Después de poner un proyecto de Visual FoxPro bajo control de código fuente, puede trabajar con archivos individuales o administrar el proyecto como un todo.

### Trabajar con componentes multiarchivo

Algunos componentes de proyecto de Visual FoxPro están formados en realidad por varios archivos: un archivo principal y uno o más archivos implícitos. Por ejemplo, cuando cree un formulario, Visual FoxPro creará un archivo .scx (el archivo principal) y un archivo .sct (el archivo implícito). Los siguientes componentes tienen varios archivos:

Componente	Tipo de archivo principal	Tipos de archivos implícitos
Formulario	.scx	.sct
Informe	.frx	.frt
Etiqueta	.lbx	.lbt
Biblioteca de clases	.vcx	.vct
Menú	.mnx	.mnt
Tabla	.dbf	.fpt, .cdx, .idx
Base de datos	.dbc	.dct, .dcx

Cuando un programador desprotege un archivo de componente, como un formulario, Visual FoxPro también administra los correspondientes archivos implícitos. Asimismo, cuando se vuelve a proteger un archivo o se agrega un archivo nuevo, Visual FoxPro administra los archivos implícitos automáticamente.

**Nota** Si genera y compila un menú, también crea archivos .MPR y .MPX. Estos no están inicialmente bajo control de código fuente, pero puede agregarlos como archivos a su proyecto y, a continuación, ponerlos bajo control de código fuente igual que lo haría con otros archivos.

## Desproteger archivos

Cuando trabaja en un proyecto con control de código fuente, Visual FoxPro puede pedirle que desproteja archivos cuando los modifique abriendo el editor adecuado. Por ejemplo, si selecciona un formulario y elige Modificar para abrir el Diseñador de formularios, Visual FoxPro puede pedirle que desproteja los archivos del formulario. (Si no desprotege los archivos, el formulario se muestra en el Diseñador de formularios, pero es de sólo lectura).

Sin embargo, también puede desproteger los archivos manualmente, lo cuál es útil si desea acceso exclusivo al archivo, pero de momento no quiere abrir el editor para el archivo. Podría hacer esto, por ejemplo, si desea trabajar con un archivo fuera de la oficina.

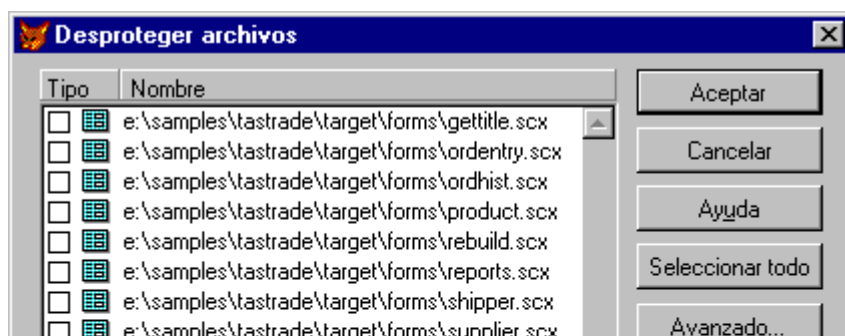
### Para especificar que Visual FoxPro le pida que desproteja archivos que se están modificando

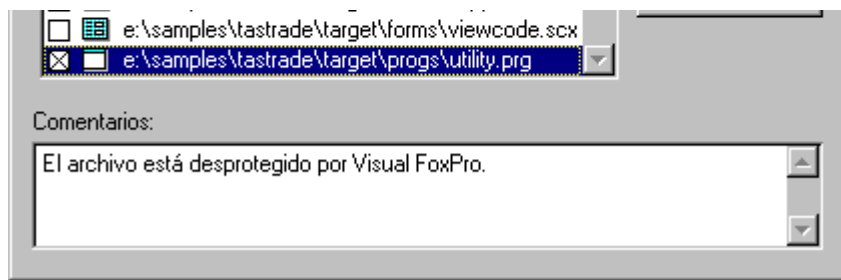
- En la ficha **Proyectos** del cuadro de diálogo [Opciones](#), asegúrese de que la opción **Desproteger los archivos al modificarlos** está activada y, a continuación, elija **Aceptar**.

Para hacer que este valor sea el predeterminado, elija **Establecer como predeterminado** y, a continuación, elija **Aceptar**.

### Para desproteger archivos manualmente

- En el **Administrador de proyectos**, seleccione el archivo con el que desea trabajar.
- En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Desproteger**.
- En el cuadro de diálogo [Desproteger archivos](#), seleccione los archivos con los que desea trabajar y, a continuación, haga clic en **Aceptar**.





## Proteger archivos

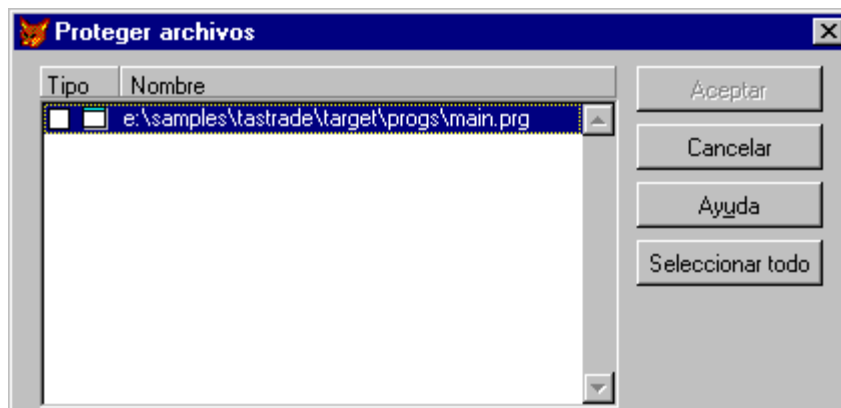
Siempre tiene que proteger los archivos manualmente. Visual FoxPro no protege automáticamente un archivo; por ejemplo, no protege un formulario cuando cierra el Diseñador de formularios. En lugar de ello, deja el archivo desprotegido de forma que pueda seguir modificándolo, llevárselo fuera de la oficina o trabajar con él de otra forma.

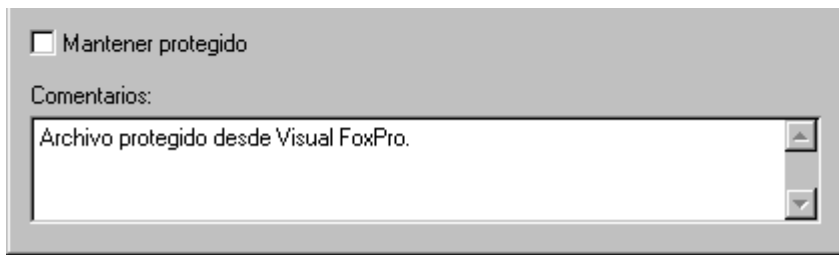
El resultado exacto del proceso de protección depende del archivo que está protegiendo y del software de control de código fuente. Para formularios, menús, etiquetas, bibliotecas de clases y otros tipos de archivos, el archivo se trata como un archivo binario y el software de control de código fuente convierte su nueva versión del archivo en la actual para que la desprotejan los otros programadores.

**Sugerencia** No olvide proteger siempre los archivos cuando termine de modificarlos. Si los deja desprotegidos durante largos períodos de tiempo, puede impedir que otros programadores trabajen con ellos e impedirá que se haga una copia de seguridad de la última versión del archivo cuando se realice la copia de seguridad a través de la red.

### Para proteger un archivo

1. En el [Administrador de proyectos](#), seleccione el archivo con el que desea trabajar.
2. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Proteger**.
3. Escriba un comentario que describa las modificaciones que ha realizado.
4. En el cuadro de diálogo [Proteger archivos](#), seleccione el archivo y, a continuación, elija **Aceptar**.





## Proteger archivos de texto

Cuando proteja un archivo de texto, como un archivo .prg, y si hay varias versiones del archivo desprotegidas, el software de control de código fuente no se limita a sobrescribir la versión central. En lugar de ello, comprueba si se han realizado modificaciones al archivo desde la última vez que lo desprotegió. Si es así, intenta combinar las modificaciones con su archivo. Para ello, agrega, elimina y modifica líneas de código en su copia del archivo.

Cuando haya terminado la combinación, el software de control de código fuente también podría darle la oportunidad de proteger el archivo. No proteja el archivo inmediatamente, pruebe la aplicación con la nueva versión del archivo que incorpore sus modificaciones y las de los otros programadores. Sólo cuando esté satisfecho con el funcionamiento de la aplicación deberá proteger el archivo. Si otros programadores han realizado modificaciones posteriores al archivo, es posible que tenga que combinar, probar y proteger de nuevo.

En algunos casos, el software de control de código fuente puede informar de un [conflicto de combinación](#), que indica que no puede resolver las modificaciones suyas y las realizadas por otros programadores. Esto puede ocurrir, por ejemplo, si usted y otro programador han actualizado las mismas líneas del mismo programa. Si el software de control de código fuente no puede combinar correctamente, crea una versión del archivo que contiene el texto original junto con sus modificaciones, marca los conflictos y escribe el archivo en su equipo. (La forma exacta en que se marcan los conflictos depende del software de control de código fuente que use). El archivo aparece entonces en el Administrador de proyectos con un icono de conflicto de combinación:



Para resolver el conflicto de combinación, tiene que volver a modificar el archivo, implementar sus cambios y quitar las marcas de conflicto de combinación. Cuando haya terminado las modificaciones, Visual FoxPro le pedirá que confirme que ha resuelto todos los conflictos. Entonces el archivo se marcará con el icono de combinación:



Pruebe su aplicación para asegurarse de que las modificaciones funcionan correctamente. Entonces podrá intentar proteger el archivo de nuevo. Si no ocurren más conflictos de combinación, su archivo se convierte en la versión actual.

## Descartar modificaciones

Si ha desprotegido un archivo, pero decide descartar las modificaciones realizadas, puede deshacer la desprotección. Esto hace que se vuelva a proteger el archivo (es decir, otros usuarios pueden desproteger el archivo), pero no actualiza sus modificaciones. Por ejemplo, si ha desprotegido por



error un archivo en lugar de simplemente obtener la última versión, deshaga la desprotección en lugar de volver a proteger el archivo. Esto impide que el sistema de control de código fuente tenga que crear otra versión del archivo, ahorrando tiempo y espacio.

**Sugerencia** Si quiere ver un archivo pero no tiene que desprotegerlo, puede obtener su última versión. Para obtener detalles, consulte [Obtener la última versión de los archivos](#) en la siguiente sección.

### Para deshacer una desprotección

1. En el [Administrador de proyectos](#), seleccione el archivo con el que va a trabajar.
2. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Deshacer proteger**.
3. En el cuadro de diálogo [Deshacer desproteger archivos](#), asegúrese de que el archivo que quiere está seleccionado y, a continuación, haga clic en **Aceptar**.

### Obtener las versiones más recientes de los archivos

Si desea ver la versión más reciente de un archivo, puede desprotegerlo. Sin embargo, si el archivo ya está desprotegido o si sólo desea ver el archivo (no modificarlo), puede obtener la última versión de un archivo. Cuando lo haga, Visual FoxPro copia la versión desprotegida más actual de un archivo de sólo lectura. Puede obtener la última versión de un archivo incluso si actualmente está desprotegido.

Si el archivo que obtiene es un archivo de texto, el software de control de código fuente combinará el más reciente con su versión en lugar de simplemente sobrescribirla.

**Nota** Para combinar archivos al obtener la última versión, es posible que tenga que activar esto como opción en el software de control de código fuente. Para obtener detalles, consulte la documentación de su software de control de código fuente.

### Para obtener la versión más reciente de un archivo

1. En el [Administrador de proyectos](#), seleccione el archivo del que desea la versión más reciente.
2. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Obtener la versión más reciente**. Si el archivo está actualmente desprotegido, se le pide que reemplace o combine su versión desprotegida con la versión actual del proyecto con control de código fuente.

**Importante** Si ya tiene desprotegido el archivo, Visual FoxPro le pide que lo desproteja. Si ha realizado modificaciones en el archivo desde la última vez que lo protegió, elija No cuando se le pregunte si desea reemplazar el archivo.

### Comparar archivos o proyectos

Cuando trabaja con archivos de un proyecto, es posible que tenga que comparar la copia local actual de su directorio de trabajo con la copia maestra actual del proyecto con control de código fuente. Esto

puede ayudarle a determinar si otro usuario ha modificado un archivo o puede ayudarle a precisar en dónde se han realizado las modificaciones desde que desprotegió el archivo.

La mayor parte de los sistemas de control de código fuente sólo pueden comparar y mostrar las diferencias entre archivos si están en formato de texto. Cuando Visual FoxPro compara formularios, informes, etiquetas y bibliotecas de clases, usa las representaciones de texto de estos archivos. Para obtener detalles, consulte [Comprobar diferencias en formularios, informes y otros archivos de tabla](#) en la siguiente sección.

**Nota** Si su proveedor de control de código fuente no admite las opciones de comparación de archivos, no estarán disponibles en el menú.

### Para ver diferencias entre proyectos

- En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Mostrar diferencias entre proyectos**.

El software de control de código fuente produce el informe resultante, de forma que la información específica proporcionada puede variar. Sin embargo, en general, el sistema de control de código fuente mostrará una ventana con dos paneles y resaltará o marcará las diferencias entre las copias local y maestra del archivo.

### Para ver diferencias entre archivos o la lista de proyecto

1. Si está viendo diferencias para un único archivo, seleccione en el **Administrador de proyectos** el archivo para el que desea ver las diferencias.
2. Para un único archivo, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Mostrar diferencias**. Para la lista de proyecto, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Mostrar diferencias de lista de proyectos**.

El software de control de código fuente produce el informe resultante, de forma que la información específica proporcionada puede variar. En general, sin embargo, el sistema de control de código fuente mostrará las dos versiones lado a lado, y resaltará o marcará las líneas nuevas, eliminadas y cambiadas.

### Comprobar diferencias en formularios, informes y otros archivos de tabla

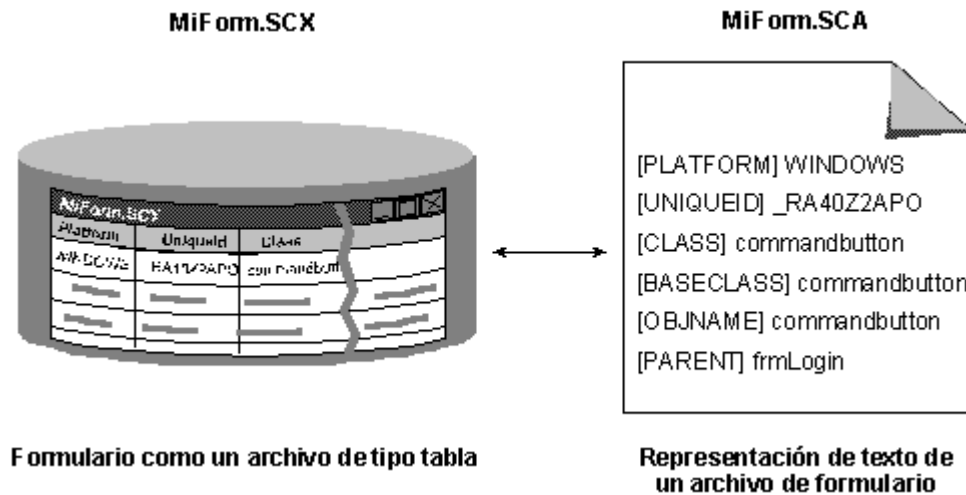
En Visual FoxPro, el software de control de código fuente sólo trata como archivos de texto algunos tipos de archivo. Entre estos están archivos de código fuente de programa (.prg) y la lista de archivos del proyecto (archivo .pjw). Los formularios, informes y otros tipos de archivos se almacenan realmente como tablas de información sobre sus componentes. Por ejemplo, un archivo .scx de un formulario es una tabla de los controles del formulario, junto con información sobre el mismo formulario. Los archivos de tipo tabla se usan para almacenar información sobre formulario (archivos .scx), informes (archivos .frx), menús (archivos .mnx), etiquetas (archivos .lbx) y bibliotecas de clases (archivos .vcx).

Como estos archivos se almacenan como tablas de Visual FoxPro, los sistemas de control de código fuente no pueden tratarlos como archivos de texto (el sistema de control de código fuente los trata

como archivos "binarios"). Como resultado, las herramientas para ver diferencias entre versiones de estos archivos no pueden precisar las diferencias, ni puede ver un historial de las modificaciones.

Para permitirle usar el control de código fuente para ver diferencias en formularios, informes y archivos similares, Visual FoxPro crea representaciones de los mismos. Entonces, cuando coloque uno de estos archivos bajo control de código fuente, Visual FoxPro creará una versión de texto del archivo, que mantendrá automáticamente mientras usted realiza modificaciones.

### Representación de texto de archivos de Visual FoxPro



Para admitir la posibilidad de generar representaciones de texto de archivos de tipo tabla, Visual FoxPro incluye el programa de utilidad Sctext.prg o puede usar un programa diferente que obtenga de otro origen o escriba usted mismo.

### Para especificar una utilidad de conversión de texto

1. En el cuadro de diálogo [Opciones](#), elija el archivo **Proyectos**.
2. En el cuadro **Generación de texto**, escriba el nombre del programa de conversión.
3. Elija **Establecer como predeterminado** y, a continuación, elija **Aceptar**.

Visual FoxPro llama automáticamente al programa de conversión de texto siempre que agregue un

Visual FoxPro llama automáticamente al programa de conversión de texto siempre que agregue un formulario, un informe, un menú, una etiqueta o un archivo a un proyecto con control de código fuente. La utilidad genera un archivo de texto que tiene el mismo nombre que el archivo principal, pero usa "A" como letra de la extensión. Por ejemplo, para un formulario llamado Miform.scx, la utilidad genera un archivo de texto llamado Miform.sca. Cuando proteja el formulario (u otro archivo) después de modificarlo, el software de control de código fuente crea automáticamente y protege el archivo de texto.

Si especifica una utilidad de conversión de texto cuando ya tenga formularios, informes y archivos similares en su proyecto con control de código fuente, tiene que quitarlos temporalmente del proyecto y, a continuación, volver a agregarlos con la generación de texto activada.

### Para generar representaciones de texto para archivos existentes en un proyecto

1. Haga una copia de seguridad de todos los archivos que se verán afectados: formularios, informes, menús, etiquetas y bibliotecas de clases.
2. Compruebe que los archivos no están ya desprotegidos.
3. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Quitar los archivos del control de código fuente**.
4. Seleccione los archivos que desea quitar del proyecto y, a continuación, elija **Aceptar**.
5. Active generación de texto, siguiendo los pasos descritos anteriormente.
6. En el menú **Proyecto**, elija **Control de código fuente** y, a continuación, elija **Agregar archivos al control de código fuente**.
7. Seleccione los archivos que desee agregar y, a continuación, elija **Aceptar**.

Al poner cada archivo bajo control de código fuente, Visual FoxPro también creará la correspondiente representación de texto para el archivo.

### Mostrar información de archivos y proyectos

Puede mostrar información sobre archivos individuales y sobre el proyecto como un todo. Por ejemplo, puede mostrar la historia de desprotecciones para un archivo individual o para el archivo lista de proyectos. La información disponible incluye normalmente:

- El número de versión, que indica cuántas veces se ha protegido una nueva versión del archivo o la lista de proyecto.
- Quién ha protegido el archivo o archivo de proyecto cada vez.
- La fecha y la hora en que se protegió.
- Los comentarios que el programador agregó al proteger el archivo o lista de proyecto.

### Para ver la historia de protección de un archivo o una lista de proyecto

1. Si está viendo el historial de un único archivo, seleccione en el **Administrador de proyectos**

el archivo para el que desea ver el historial.

2. Para un proyecto individual, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Mostrar historial**. Para la lista de proyecto, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Mostrar el historial de la lista de proyectos**.

Las opciones específicas disponibles en este comando de menú dependen del sistema de control de código fuente. Para obtener detalles, elija **Ayuda** en el cuadro de diálogo mostrado o consulte la documentación de su sistema de control de código fuente.

También puede ver la información sobre un usuario o sobre el proyecto que mantiene el sistema de control de código fuente. Normalmente incluye información sobre el estado de desprotección del archivo o la lista de proyecto, si el archivo es un archivo de texto o un archivo binario (que determina si puede combinar sus modificaciones con el archivo almacenado), etc.

### Para ver información de control de código fuente para un archivo o una lista de proyecto

1. Si esta viendo el historial de un único archivo, seleccione en el **Administrador de proyectos** el archivo para el que desea ver el historial.
2. Para un archivo individual, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Propiedades del control de código**. Para la lista de proyecto, elija **Control de código fuente** en el menú **Proyecto** y, a continuación, elija **Propiedades del proyecto**.

Las opciones específicas disponibles en este comando de menú dependen del sistema de control de código fuente. Para obtener detalles, elija **Ayuda** en el cuadro de diálogo mostrado o consulte la documentación de su sistema de control de código fuente.

## Programar y modificar bases de datos en equipo

Además de trabajar juntos con proyectos y archivos de proyecto, su equipo debe poder compartir información en bases de datos. Trabajar con bases de datos en equipo incluye no sólo los temas de control de simultaneidad para datos de tablas, sino también la necesidad de compartir información de control de bases de datos.

Para que varios programadores trabajen simultáneamente con una base de datos, tienen que poder compartir el archivo de la base de datos (.dbc). En Visual FoxPro, el archivo .dbc se puede compartir entre programadores como tabla de datos normal. El archivo .dbc debe almacenarse centralmente con las tablas que forman la base de datos. Los programadores no deben guardar copias locales de un archivo .dbc porque las modificaciones que realicen a la base de datos no se reflejarán en las versiones del archivo de los otros programadores.

Si no tiene que modificar el archivo .dbc, tenga en cuenta las siguientes restricciones:

- Los programadores no pueden modificar el mismo elemento de base de datos (como una estructura de tabla, una vista o una conexión) simultáneamente. Cuando el programador modifica un elemento de base de datos, Visual FoxPro bloquea su entrada en el archivo .dbc; otros usuarios pueden leer la entrada (es decir, pueden ejecutar un comando USE), pero no pueden modificarla (MODIFY STRUCTURE)

- pueden modificarla ([MODIFY STRUCTURE](#)).
- Si se está usando un elemento de base de datos, no puede modificar su estructura. Por ejemplo, si un programador tiene una tabla abierta, los otros programadores no pueden modificar su estructura.
- Si llama a la función [DBSETPROP\(\)](#) para modificar las propiedades de una base de datos, la función coloca un bloqueo de escritura en el objeto que se está actualizando. Si hay un conflicto de bloqueo, [DBSETPROP\(\)](#) sigue las reglas establecidas en [SET REPROCESS](#).

## Trabajar con vistas y conexiones

Las vistas y las conexiones funcionan de forma diferente que las tablas. Cuando define por primera vez la vista, Visual FoxPro usa las tablas de una base de datos, pero no las bloquea. Sin embargo, como las tablas están en uso, los otros programadores no pueden modificar sus estructuras.

Desde que guarda por primera vez una nueva vista o definición de conexión, Visual FoxPro la bloquea exclusivamente hasta que cierre el Diseñador de vistas o el Diseñador de conexiones. En otras palabras, mientras tenga la vista o conexión abierta en un diseñador, estará bloqueada de forma exclusiva. Mientras la vista esté bloqueada, nadie podrá modificarla.

Cuando usa una vista, su estructura se almacena localmente. Esto asegura que si la vista se modifica mientras la utiliza (por ejemplo, si llama a [REFRESH\(\)](#) o [REQUERY\(\)](#)) su formulario o informe seguirá ejecutándose correctamente.

## Programar bibliotecas de clases en equipo

Como las bibliotecas de clases (archivos .vcx) son una parte crucial de la mayor parte de las aplicaciones de Visual FoxPro, los equipos tienen que poder coordinar sus esfuerzos al crearlas. Trabajar con bibliotecas de clases en equipo incluye muchos de los mismos temas de coordinación inherentes a cualquier conjunto de componentes de aplicación, pero agrega algunos temas exclusivos de las clases:

- Las modificaciones realizadas a las clases se propagan no sólo a las aplicaciones que usan las clases, sino a todas las subclases que se deriven de ellas.
- Se suelen almacenar varias clases en un único archivo de biblioteca, la unidad básica que puede administrar un sistema de control de código fuente.

Como ocurre con formularios y programas complejos, es una buena práctica aislar la programación en una biblioteca de clases, de forma que un programador puede realizar modificaciones en la biblioteca sin que afecte a los otros programadores. De forma ideal, el equipo de programadores puede trabajar con una biblioteca de clases mientras la está mejorando otro programador, sin tener que preocuparse de si las modificaciones realizadas a la biblioteca afectarán a la aplicación.

Cuando se usa una clase, Visual FoxPro la almacena localmente en el equipo del usuario, incluso después de liberar un formulario que usa la clase. Debe liberar explícitamente la clase antes de que Visual FoxPro note que ya no se está utilizando. Si ha usado una clase en la sesión actual (y por lo tanto está almacenada localmente), pero desea cargar una versión nueva de la clase, asegúrese de liberar la clase para obligar a Visual FoxPro a volver a cargarla de la biblioteca modificada.

### Aplicar el control de código fuente a bibliotecas de clases

### **Aplicar el control de código fuente a bibliotecas de clases**

Al aplicar control de código fuente a una biblioteca de clases, sólo un programador puede desproteger la biblioteca cada vez. La biblioteca pasa a ser de sólo lectura para los otros programadores. Como regla general, esto no interfiere con la programación de la aplicación, porque los programadores pueden usar una biblioteca y crear subclases incluso si la biblioteca es de sólo lectura. Mientras los programadores de la aplicación trabajan con la versión de sólo lectura de la biblioteca, el programador de la biblioteca de clases puede modificar todas las clases de la biblioteca.

Si usa esta aproximación, el programador que está actualizando la biblioteca no debería proteger el archivo hasta que haya terminado y se haya probado. Si no es así, otros programadores obtendrán la versión incompleta del archivo cuando actualicen sus listas de archivos de proyecto u obtengan las últimas versiones de los archivos.

Si la biblioteca es muy compleja, también puede considerar la posibilidad de dividirla en bibliotecas más pequeñas para la programación. Otra ventaja de esta aproximación es que las bibliotecas se cargan más rápido. Sin embargo, esto significa que distintas clases podrían terminarse en plazos diferentes.

Como cada aproximación tiene sus ventajas, debería examinar los requisitos de su equipo de programadores y elegir la estrategia que mejor se aplique a su método de trabajo.

## **Capítulo 30: Soluciones empresariales de Visual FoxPro**

Además de usar Visual FoxPro para crear aplicaciones independientes, puede hacer que forme parte de una solución de negocios a nivel empresarial más amplia. De esta manera, puede integrar las características de Visual FoxPro con otras aplicaciones para Windows para crear una solución eficaz y amplia para las necesidades de su aplicación.

Este capítulo proporciona una introducción a dónde y cómo puede usar Visual FoxPro como parte de sus esfuerzos de programación a nivel empresarial. No incluye información sobre procedimientos para llevar a cabo tareas concretas; en lugar de ello, este capítulo es un recorrido que destaca las características de Visual FoxPro que lo convierten en único para soluciones empresariales.

Este capítulo incluye información sobre:

- [Programar para la empresa](#)
- [Usar Visual FoxPro como interfaz de usuario de aplicaciones](#)
- [Usar Visual FoxPro como origen de datos](#)

### **Programar para la empresa**

Muchas aplicaciones creadas con Visual FoxPro son soluciones independientes para requisitos de negocios específicos. Por ejemplo, puede crear una aplicación de Visual FoxPro para hacer un seguimiento de sus clientes, que puede incluir no sólo información de base de datos sobre los clientes sino también herramientas para recibir pedidos, crear facturas, etc. Puede crear todas las

clientes, sino también herramientas para recibir pedidos, crear facturas, etc. Puede crear todas las características necesarias para la aplicación con las herramientas disponibles en Visual FoxPro, incluyendo el motor de base de datos, las herramientas visuales de diseño y las posibilidades para crear informes.

Pero también puede usar Visual FoxPro como parte de una aplicación a mayor escala que incluya dos o más herramientas de programación. Usar Visual FoxPro de esta manera, que se llama "programación empresarial", le permite aprovechar las posibilidades únicas de cada producto. La programación empresarial puede ser tan sencilla como mantener una base de datos de clientes en Visual FoxPro y crear una carta de combinación de correspondencia en Microsoft Word, o crear una aplicación compleja con bases de datos cliente-servidor, servidores de Automatización, correo electrónico y más componentes.

Visual FoxPro es una herramienta ideal para crear soluciones de negocios a nivel empresarial por sus características:

- Herramientas de programación de aplicaciones eficaces y fáciles de usar, incluyendo un Diseñador de formularios y asistentes.
- Motor de base de datos rápido.
- Excelente conectividad con otros productos, incluyendo otros programas para Windows como Microsoft Excel y Word y sistemas cliente-servidor como Microsoft SQL Server.
- Control de código fuente integrado y otras herramientas de programación en equipo.

Estas características le permiten programar con Visual FoxPro en varias funciones de una aplicación a nivel empresarial. Puede usar Visual FoxPro:

- Como interfaz de usuario para otras aplicaciones. En este escenario, trabaja fundamentalmente con Visual FoxPro; por ejemplo, puede crear la interfaz de usuario para la aplicación en Visual FoxPro. Posteriormente puede tener acceso a otras aplicaciones que contienen datos necesarios para la aplicación o que pueden proporcionar servicios que mejoran los ya disponibles en Visual FoxPro. También puede hacer un [upsizing](#) de los datos de Visual FoxPro o moverlos a otra plataforma.
- Como origen de datos para otras aplicaciones. Para ello, debe crear la interfaz de usuario con otro programa y, a continuación, tener acceso a los datos de Visual FoxPro cuando sea necesario.

La estrategia que elija depende de los objetivos de su aplicación y de los programas que desee utilizar.

Las siguientes secciones proporcionan ideas y escenarios que ilustran cómo usar Visual FoxPro en cada una de las formas descritas anteriormente. Sin embargo, no piense que las aplicaciones presentadas aquí son los únicos tipos que puede crear; use las ideas que se presentan aquí para inventar y diseñar sus propias soluciones empresariales.

## Usar Visual FoxPro como interfaz de usuario de aplicaciones

Como programador de Visual FoxPro, probablemente encontrará natural diseñar sus aplicaciones en torno a las herramientas visuales de diseño del programa. Por ejemplo, probablemente concibe la interfaz de usuario de su aplicación en términos de formularios, menús e informes de Visual FoxPro. Además, cuando programa aplicaciones en Visual FoxPro, pensará almacenar los datos de la



Además, cuando programa aplicaciones en Visual FoxPro, piensa almacenar los datos de la aplicación en tablas de Visual FoxPro.

Una forma de integrar Visual FoxPro en una aplicación a nivel empresarial es usar sus herramientas visuales de diseño, pero mejorándolas con las posibilidades de otros productos. Otra forma consiste en crear la apariencia de la aplicación con Visual FoxPro, pero extendiendo las posibilidades de almacenamiento de datos aprovechando las posibilidades de otros programas o de opciones de almacenamiento de datos externas a Visual FoxPro. También puede hacer un upsizing de los datos de Visual FoxPro moviéndolos a un servidor de base de datos.

## **Extender las herramientas visuales de diseño de Visual FoxPro**

Las clases de base de controles de Visual FoxPro se diseñaron para incorporar la inmensa mayoría de necesidades de interfaz de aplicación. Visual FoxPro proporciona todos los controles básicos y los elementos de interfaz necesarios para crear una aplicación para Windows estándar. Sin embargo, a menudo verá que su aplicación requiere objetos o controles con otras funcionalidades que no poseen los proporcionados por las clases de Visual FoxPro. Si es así, puede extender las herramientas visuales de diseño creando subclases y usando controles ActiveX.

### **Crear subclases**

Una característica enormemente eficaz de Visual FoxPro es la capacidad de crear subclases de los controles de base. Creando una o más subclases puede personalizar los controles básicos de Visual FoxPro de casi cualquier manera requerida por su aplicación. Esta capacidad incluso permite crear nuevos objetos o controles que combinen las características de otros controles. Por ejemplo, el control cuadrícula de Visual FoxPro no sólo contiene sus propiedades, métodos y contenedor, sino también aquellos objetos que aparecen en la cuadrícula, como botones, cuadros de texto, etc.

Asimismo, creando subclases de controles de base, puede extender las posibilidades de Visual FoxPro creando objetos que agreguen nuevas características a clases de base existentes, o que combinen las posibilidades de varios objetos. Por ejemplo, puede agregar características visuales como marcos o efectos tridimensionales a un cuadro de texto. O puede combinar un control imagen, botones y un cuadro de texto para crear una control para ver mapas de bits con el que los usuarios pueden moverse por varios archivos .bmp. Crear clases personalizadas de esta forma puede ayudarle a administrar la programación a nivel de empresa permitiéndole crear controles estandarizados que aparecen en todas las aplicaciones. Para obtener más información sobre la creación de subclases, consulte el capítulo 3, [Programación orientada a objetos](#).

### **Usar controles ActiveX**

Una forma alternativa de crear un control nuevo con subclases de Visual FoxPro es usar un control ActiveX (archivo .ocx). Estos controles se crean independientemente de Visual FoxPro y se pueden integrar no sólo en Visual FoxPro, sino también en muchas otras aplicaciones para Windows.

En efecto, los controles ActiveX son componentes externos que puede integrar perfectamente en su aplicación. El uso de controles ActiveX proporciona varios beneficios:

- Ahorra el tiempo y esfuerzo necesarios para crear, probar y mantener un control específico de

Visual FoxPro para realizar las mismas tareas. Cuanto más eficaz sea el control ActiveX, más tiempo ahorrará.

- Muchos controles ActiveX ya están disponibles de terceros para responder a requisitos comunes de aplicaciones. Por ejemplo, si la aplicación pide que muestre un calendario y que permita a los usuarios elegir fechas en él, probablemente puede encontrar un control ActiveX (tal vez varios) que ya realiza esta tarea.
- Se puede usar el mismo control en varios programas. Por ejemplo, si tiene sentido, puede usar el mismo control ActiveX en Visual FoxPro y en Visual Basic. Se usan las mismas propiedades y métodos en cada caso para administrar el control, y el control tendrá la misma apariencia en todos los programas, facilitando su uso por parte de los usuarios.
- Los controles ActiveX proporcionan a menudo acceso a la funcionalidad de Windows que de otro modo puede ser difícil o requerir mucho tiempo para incluir usando exclusivamente las herramientas de Visual FoxPro. Por ejemplo, puede buscar controles ActiveX que proporcionen acceso a correo electrónico (usando funciones de la MAPI de Windows), funciones gráficas de Windows de bajo nivel, etc. Al incluir un control ActiveX, puede agregar estos tipos de características a su aplicación en una forma fácil de controlar con las propiedades, los métodos y los eventos de los controles ActiveX.

En resumen, usar los controles ActiveX le permite extender sus aplicaciones no sólo integrando la funcionalidad de Windows, sino también agregando una apariencia común entre sus aplicaciones y otras de la misma empresa. Para obtener más información sobre el uso de controles ActiveX, consulte el capítulo 16, [Agregar OLE](#). Para obtener información sobre la creación de sus propios controles ActiveX, consulte el capítulo 28, [Acceso a la API de Visual FoxPro](#).

## Integrar la funcionalidad de otros programas

Es posible que al programar una aplicación se de cuenta de que otros programas son apropiados para llevar a cabo ciertas tareas. Por ejemplo, Microsoft Word tiene posibilidades únicas para combinación de correspondencia, mientras que Microsoft Excel está optimizado para calcular fórmulas complejas y crear fácilmente gráficos a partir de ellas.

En lugar de emular estas posibilidades en Visual FoxPro, puede convertir su aplicación en una solución a nivel empresarial integrándolas en ella. Así puede resolver las necesidades de su aplicación usando la mejor aplicación para llevarlas a cabo.

Puede integrar la funcionalidad de otras aplicaciones en Visual FoxPro de las formas siguientes:

- Ejecute un asistente de Visual FoxPro que ponga los datos de Visual FoxPro a la disposición de otra aplicación.
- Escriba programas de Visual FoxPro que usen Automatización para comunicarse y compartir datos con otros programas para Windows y controlarlos.

Las siguientes secciones proporcionan detalles sobre estos métodos de extender las posibilidades de Visual FoxPro.

### Usar asistentes

Varios asistentes de Visual FoxPro le permiten integrar datos de Visual FoxPro con la funcionalidad de otros programas para Windows. Por ejemplo, puede enviar cartas modelo a sus clientes con el

[Asistente para combinar correspondencia](#). Cuando ejecute el asistente, puede especificar una tabla o vista que contenga datos de Visual FoxPro para usar y, a continuación, exportar los datos a un formato de archivo apropiado (como delimitado por comas, por ejemplo) o especificar que el programa procesador de textos use el controlador ODBC de Visual FoxPro para tener acceso a los datos. Si usa Microsoft Word, el asistente incluso iniciará el programa procesador de textos, creará el documento de combinación en blanco y mostrará la barra de herramientas Combinar correspondencia para que vincule campos con los datos de Visual FoxPro.

Asimismo, con Microsoft Excel y Microsoft Query, puede analizar sus datos mediante una tabla dinámica, que resume datos en columnas y le permite reorganizarlos para mostrarlos de distintas maneras. Con el [Asistente para tablas dinámicas](#) de Visual FoxPro, puede usar los datos de su aplicación como origen de datos para Microsoft Excel y generar la tabla dinámica en Microsoft Excel.

## Usar la Automatización

Una forma más eficaz de interactuar con otras aplicaciones es usar la Automatización. Con programas de Visual FoxPro, puede tener acceso a los objetos expuestos por otras aplicaciones y después controlarlos estableciendo sus propiedades y llamando a sus métodos. Por ejemplo, Microsoft Excel expone un objeto Application así como hojas, columnas, filas y celdas del objeto Application. Puede manipular directamente cualquiera de estos objetos, obteniendo datos de ellos o estableciendo datos en ellos. Además, normalmente puede controlar el objeto Application con todos los comandos disponibles en el programa. Por ejemplo, administrando el objeto Application en Microsoft Excel puede abrir, guardar o imprimir hojas, llamar al Asistente para gráficos de Microsoft Excel, etc.

La Automatización es una forma particularmente atractiva y eficaz de unir programas para Windows por varias razones:

- Tiene acceso directo al otro programa, incluyendo todos sus objetos y comandos.
- Puede compartir datos directamente con el otro programa sin tener que exportarlos o convertirlos a otro formato.
- Puede controlar el otro programa con el modelo familiar de propiedades y métodos.
- El otro programa no tiene que estar visible necesariamente cuando lo llame. Por ejemplo, puede llamar a Microsoft Excel, colocar algunos datos en celdas, ejecutar un cálculo complejo en los datos, leer el resultado y, a continuación, mostrarlo en Visual FoxPro, todo sin mostrar Microsoft Excel. El usuario seguiría viendo exclusivamente Visual FoxPro, a menos que quisiera mostrar Microsoft Excel explícitamente.
- Los comandos (métodos y propiedades) para controlar el otro programa están incrustados en programas de Visual FoxPro familiares. No tiene que aprender un lenguaje de programación diferente para poder controlar el otro programa.

La Automatización es particularmente eficaz porque es un método ilimitado para trabajar con otros programas. En esencia, la Automatización simplemente pone a su disposición los datos y comandos de otras aplicaciones, para que las use de la forma más apropiada para su aplicación.

Un simple escenario ilustra cómo puede integrar varios programas para Windows. Suponga que almacena los datos de clientes y ventas en Visual FoxPro. Le gustaría crear un informe de ventas que resuma las ventas trimestrales.

Una solución sería usar la Automatización para copiar los datos de ventas de Visual FoxPro a celdas de una hoja de Microsoft Excel. Entonces puede llamar al asistente para gráficos de Microsoft Excel para crear un gráfico de los datos y copiarlos al Portapapeles de Windows. Aún usando Automatización, puede llamar a Microsoft Word y crear o abrir un documento de informe de ventas (si lo crea como un documento nuevo, puede insertar texto estándar almacenado en Visual FoxPro) y, a continuación, pegarlo en el gráfico creado en Microsoft Excel.

Esto es sólo una de las formas de usar la Automatización para convertir Visual FoxPro en parte de una solución a nivel empresarial. Familiarizándose con los objetos y métodos disponibles en programas que usa típicamente, se le pueden ocurrir muchas más formas de hacer que cada programa mejore las posibilidades de los otros. Para obtener detalles sobre la Automatización, consulte "Manipular objetos con la Automatización" en el capítulo 16, [Agregar OLE](#).

## **Extender las posibilidades de almacenamiento de datos en Visual FoxPro**

Las posibilidades de tablas de datos e indexado de Visual FoxPro son generalmente más apropiadas para los requisitos de una aplicación si le importan la velocidad y el tamaño de bases de datos. Sin embargo, a veces deseará extender Visual FoxPro con datos almacenados en otros formatos. Esto puede ocurrir si:

- La aplicación tiene que tener acceso a datos heredados que crea y mantiene una aplicación existente. Por ejemplo, suponga que, como parte de su aplicación de ventas, necesita acceso a datos que mantiene una aplicación de contabilidad que se programó con un lenguaje diferente, tal vez incluso en una plataforma diferente.
- Puede optimizar el acceso a datos con un servidor de base de datos, que puede acelerar el acceso a datos, particularmente para bases de datos muy grandes.
- Desea compartir datos con otros programas y, por lo tanto, desea almacenar los datos en un formato accesible a todos los programas.
- Los datos se ajustan mejor al formato de un programa concreto (como una hoja de cálculo). Esto puede ser verdad, por ejemplo, si su aplicación sólo requiere accesos ocasionales a datos que mantenidos por el otro programa.

Si los datos que necesita están en formato de hoja de cálculo, documento de procesador de textos u otro programa para Windows, puede tener acceso a los mismos a través de la Automatización. Por ejemplo, puede hacer esto si su aplicación requiere un conjunto de cartas modelo. En ese caso, las cartas pueden estar almacenadas como documentos de Microsoft Word y su aplicación usaría Automatización para llamar a Word, abrir la carta apropiada e insertar o reemplazar texto cuando sea necesario.

Una aproximación más común al uso de datos externos a Visual FoxPro es usar ODBC para tener acceso a los mismos. Los controladores ODBC le permiten conectarse a los datos en el formato de otros programas, típicamente otros programas de base de datos y consultarlos o modificarlos mediante comandos SQL estándar.

Por ejemplo, podría decidir que las posibilidades de seguridad y procesamiento de transacciones son una parte vital de su aplicación, por lo que desea almacenar los datos con Microsoft SQL Server. Para tener acceso a los datos, defina una conexión a SQL Server mediante el controlador ODBC. Entonces

podrá ejecutar consultas normales (y otros comandos SQL) como si los datos estuvieran en formato de Visual FoxPro.

Otras aplicaciones pueden tener acceso a los mismos datos y aprovechar las mismas características. Por ejemplo, una hoja de Microsoft Excel puede obtener sus datos de la misma base de datos de SQL Server. La hoja no sólo se beneficiará de las mismas ventajas de rendimiento que su aplicación, también puede aprovechar las características de seguridad y procesamiento de transacciones del servidor, que no están disponibles de otro modo en una hoja Microsoft Excel.

En algunos casos, es posible que desee llegar más lejos y usar comandos SQL específicos del origen de datos al que tiene acceso con ODBC. Por ejemplo, Microsoft SQL Server le permite crear y ejecutar procedimientos almacenados, que pueden manipular datos en el servidor (en lugar de en su aplicación). Para beneficiarse de los procedimientos almacenados, puede enviar instrucciones SQL "nativas" al servidor de base de datos. los comandos de paso a través de SQL también le permiten realizar tareas de administración del sistema en el servidor y en algunos casos se ejecutarán más rápido que comandos SQL similares ejecutados en Visual FoxPro.

Para obtener más detalles sobre cómo extender las posibilidades de almacenamiento de datos de Visual FoxPro, consulte la documentación indicada en la tabla siguiente.

Para obtener detalles sobre	Consulte
Automatización	"Manipular objetos mediante Automatización" en el capítulo 16, <a href="#">Agregar OLE</a>
Uso de ODBC para tener acceso a datos	"Acceso a datos remotos" en el capítulo 8, <a href="#">Crear vistas</a>
Uso de Visual FoxPro en un entorno cliente-servidor	Capítulo 19, <a href="#">Diseñar aplicaciones cliente-servidor</a>

## Upsizing de datos de Visual FoxPro

Puede elegir guardar sus datos en tablas de Visual FoxPro o en otra plataforma, como un servidor de base de datos. O puede hacer ambas cosas: guardar los datos en tablas de Visual FoxPro mientras programa o hasta que la base de datos se haga muy grande y después mover los datos (hacer un *upsizing*) a otra plataforma.

Por ejemplo, puede modelar su aplicación conservando todos los datos en tablas locales de Visual FoxPro. Esto le proporciona la flexibilidad de modificar sus tablas, vistas e índices cuando programa la aplicación sin la complejidad de administrar tablas en un servidor de base de datos. Puede guardar datos de ejemplo en las tablas locales para probar los formularios, informes y otros programas. Cuando la estructura de la base de datos esté terminada, puede hacer un upsizing de los datos a un servidor de base de datos y enviar a producción la aplicación.

Otra forma de trabajar es guardar los datos en tablas de Visual FoxPro sólo mientras sea práctico. Cuando la base de datos se hace grande, puede hacer un upsizing y aprovechar el rendimiento

optimizado proporcionado por un servidor de base de datos. El punto en el que tiene sentido hacer un upsizing de la base de datos depende de muchos factores, incluyendo la complejidad de la base de datos, el rendimiento de su equipo local o de la red y las exigencias de la aplicación.

Finalmente puede modelar la base de datos en Visual FoxPro y, a continuación, hacer un upsizing para compartir los datos con otras aplicaciones que también pueden tener acceso a un servidor de base de datos. De forma similar, puede hacer un upsizing de la base de datos para aprovechar la seguridad y las posibilidades de procesamiento de transacciones por parte del servidor de base de datos.

Para obtener más detalles sobre el upsizing de bases de datos, consulte el capítulo 20, [Upsizing de bases de datos de Visual FoxPro](#).

## Usar Visual FoxPro como origen de datos

Una forma diferente de integrar Visual FoxPro en una solución empresarial es usarlo como un componente, pero no necesariamente como la aplicación principal. En efecto, puede usarlo como servicio de fondo para una aplicación escrita con otro producto. En este caso, el usuario no vería directamente a Visual FoxPro. En lugar de ello, la interfaz de usuario de la aplicación estaría programada con herramientas de la otra aplicación y se comunicaría con Visual FoxPro para obtener o manipular datos.

Visual FoxPro funciona bien en este papel porque puede ofrecer su motor de base de datos, que proporciona acceso rápido a datos a otras aplicaciones. Además, puede ofrecer sus objetos y comandos a otros programas, incluyendo objetos personalizados que puede crear.

### Poner los datos de Visual FoxPro a la disposición de otros programas

Una forma de que una aplicación a nivel empresarial aproveche Visual FoxPro es usar el motor de base de datos de Visual FoxPro para almacenar y administrar datos. Esto proporciona almacenamiento de alto rendimiento y posibilidad de consulta para otros programas.

Los programas pueden conectarse a datos de Visual FoxPro mediante el controlador ODBC de Visual FoxPro. Este controlador ofrece el motor de bases de datos de Visual FoxPro para comandos SQL estándar.

Por ejemplo, una aplicación podría usar Microsoft Excel como herramienta de cálculo para análisis de datos complejos. Si los datos que hay que manipular son muy flexibles, es posible que haga más sentido almacenarlos en una base de datos en lugar de en una hoja de cálculo. Entonces la hoja se podría crear de modo que use el controlador ODBC de Visual FoxPro para conectarse a la base de datos, extraer la información relevante y mostrarla en una hoja para su procesamiento posterior.

Otro ejemplo podría ser una aplicación quiosco, como un stand de información en un aeropuerto o un centro de convenciones. Podría crear la presentación de información mediante un programa de creación multimedia. Pero si algunos de los datos de la aplicación cambian a menudo, sería incómodo cambiar páginas de la presentación. En lugar de ello, el programa de presentación podría conectarse a una base de datos de Visual FoxPro mediante el controlador ODBC y extraer los datos en tiempo de ejecución.



Para obtener más información, vea la Ayuda Controlador ODBC de Visual FoxPro (Drvvpfp.hlp) instalada en el directorio ...\\Vfp98\\Distrib\\Src\\System. También está disponible en el grupo de programas ODBC si instaló ODBC durante la instalación de Visual FoxPro.

## **Poner los objetos y comandos de Visual FoxPro a la disposición de otros programas**

Además de poner los datos de Visual FoxPro a la disposición de otros programas como parte de una solución empresarial, puede ofrecer los objetos y comandos de Visual FoxPro. Otras aplicaciones pueden llamar a los métodos y propiedades de conjunto de los objetos de Visual FoxPro; incluyendo no sólo los objetos de base, sino también los objetos definidos en clases personalizadas.

Por ejemplo, puede crear una aplicación en Microsoft Excel que almacene datos en una base de datos de Visual FoxPro. Además, para simplificar la lectura y escritura de datos, Microsoft Excel puede llamar a los comandos de Visual FoxPro para mostrar un formulario como un cuadro de diálogo. Una posible utilización es reunir datos para una vista parametrizada.

Otra forma de ofrecer objetos Visual FoxPro es crear un servidor de Automatización. Esto le permite crear objetos específicos de la aplicación que pueden realizar casi cualquier función que pueda programar en Visual FoxPro, con la ventaja de que puede distribuir el servidor.

Un uso para un servidor personalizado es crear un objeto que incluya un conjunto de reglas de negocios que aseguren la integridad de los datos que le pase otra aplicación. Por ejemplo, puede crear un objeto en Visual FoxPro para almacenar información de empleados que no sólo valide que la aplicación ha pasado información de empleados válida, sino que compruebe el nivel de acceso del usuario para asegurar que el usuario tiene acceso de seguridad para hacer los cambios de empleado.

Un servidor personalizado también puede ofrecer un objeto que incorpore lógica compleja para actualizar o leer información. Por ejemplo, es posible que un objeto de entrada de pedidos pueda no sólo almacenar el pedido, sino también mantener un registro de transacciones de pedidos, un inventario de actualizaciones, calcular una comisión de ventas, etc.

Este tipo de servidor de Automatización es ideal para crear la "capa intermedia" de una aplicación empresarial de tres niveles. En este modelo, los datos forman el nivel más bajo y la aplicación forma el más alto. La funcionalidad está en el medio, y proporciona una vista específica independiente de la aplicación de los datos que incorpora reglas de negocios (u otras posibilidades de proceso de datos) que no pertenecen exactamente a los datos ni a la aplicación.

Para obtener información sobre la creación de servidores de Automatización, consulte [Crear servidores de Automatización](#) en el Capítulo 16, "Agregar OLE."

## **Crear un almacén de datos con Visual FoxPro**

Además de crear la aplicación en Visual FoxPro, puede usar el programa para crear y mantener un [almacén de datos](#) o una versión de sus datos optimizada para informes. Para crear un almacén de datos hace una copia de los datos necesarios para realizar informes y, a continuación, los pone a disposición de los usuarios que los necesiten. Manteniendo estos datos separados de los datos actuales puede:

- Estructurarlos para hacer que la elaboración de informes sea más fácil y más rápida que si los

usuarios crearan informes a partir de los datos actuales.

- Colocar datos para informes en una ubicación distinta que los datos actuales, lo cual reduce el contenido de datos, mejora el rendimiento y pone los datos a disposición de usuarios que no deberían ver los datos actuales por motivos de seguridad.

Un almacén de datos es un "snapshot" de los datos obtenido cuando los crea. Usted actualiza los datos del almacén periódicamente, programando la actualización de acuerdo con las necesidades de informe de su aplicación.

Por ejemplo, suponga que está creando una aplicación para administrar una biblioteca, incluyendo un inventario de materiales. Durante el día, el sistema se usa constantemente a medida que los clientes sacan e introducen material, y consultan el sistema para buscar o reservar libros. Además de administrar estas transacciones individuales, los bibliotecarios desean poder analizar su biblioteca para determinar hechos como qué libros son más populares, que libros están con atraso, etc.

Para ayudarle en el análisis, la aplicación puede crear un almacén de datos de la información de transacciones. Puede almacenar los datos periódicamente (por ejemplo, cada noche) y los bibliotecarios pueden crear consultas sin afectar al rendimiento del sistema durante el día. Además, el almacén de datos puede excluir detalles sobre los clientes que usan la biblioteca, porque esta información no es necesaria para el análisis y se podría considerar información confidencial.

Para obtener el mayor beneficio de un almacén de datos, lo crea en un servidor distinto de los datos actuales. Si los datos actuales y el almacén de datos están en el mismo servidor, aún puede beneficiarse de tener los datos optimizados en el almacén. Sin embargo, a medida que los usuarios hacen consultas al almacén, pueden generar una gran cantidad de tráfico de red que podría afectar al rendimiento del sistema actual.

Cuando crea el almacén de datos, puede simplemente copiar los archivos actuales sobre los archivos paralelos del almacén de datos. De forma alternativa, puede reestructurar los datos del almacén para optimizarlos con el fin de hacer informes. Por ejemplo, puede querer crear índices para el almacén que reducen la sobrecarga de informes.

Como otro ejemplo, los datos de una aplicación deberían normalizarse para evitar la duplicación de datos. Sin embargo, podría ser útil combinar tablas en el almacén de datos que de otro modo serían independientes; esto puede eliminar la necesidad de combinar tablas, facilitando la creación de informes para usuarios menos experimentados.

También puede ajustar el nivel de detalle del almacén de datos a los requisitos de creación de informes de su aplicación. Para mayor flexibilidad, debería almacenar el mismo nivel de detalle en el almacén de datos que el que tiene en los datos actuales. Sin embargo, si los usuarios quisieran crear sólo informes resumen (como hojas electrónicas o gráficos), podría eliminar los datos detallados de la aplicación y almacenar únicamente datos resumidos en el almacén de datos.

## **Usar Visual FoxPro como motor de búsqueda del World Wide Web**

Si su solución empresarial incluye la creación de un servidor World Wide Web para Internet, puede incorporar Visual FoxPro a la aplicación como motor de búsqueda. Esto le permite poner la eficacia de su base de datos de Visual FoxPro a la disposición de cualquiera que pueda tener acceso a su



servidor Web a través de Internet o a través de intranet en la empresa.

Por ejemplo, suponga que como parte de su intranet a nivel de empresa quiere hacer que esté disponible un directorio de empleados. Los empleados podrían apuntar con sus exploradores a una página "Buscar el empleado", que mostraría una página con la apariencia de un formulario de Visual FoxPro, con cuadros de texto para introducir criterios. Para realizar una búsqueda, los usuarios escribirían el nombre del empleado, la extensión de teléfono, el departamento, el cargo o cualquier otra información disponible y, a continuación, elegirían un botón Buscar ahora. En unos instantes verían un listado de los empleados que cumplen los criterios de búsqueda. Podrían guardar la lista como archivo de texto que podría importar otro programa, como un procesador de textos.

### **Descripción de Visual FoxPro como motor de búsqueda del Web**

En general, para usar Visual FoxPro como servidor de información para el Web, necesitará estos componentes:

- Un servidor Web con servicio HTTP, que ejecute el sistema operativo Microsoft Windows NT.
- Una aplicación de Visual FoxPro que se pueda llamar como servidor de automatización. Esta aplicación se puede ejecutar en cualquier servidor al que tenga acceso el servidor Web.
- Un medio que muestre resultados de búsqueda, que normalmente consiste en una plantilla de página Web en la que puede insertar datos.

La secuencia usual de eventos que implica una búsqueda de Visual FoxPro en el Web es:

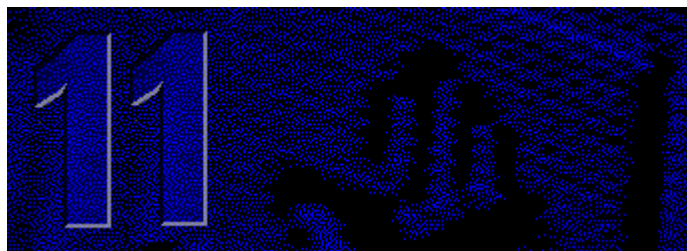
1. El usuario muestra la página de búsqueda de su aplicación señalando con un explorador de Web. La página de búsqueda incluye cualquier gráfico o texto que desee, más cuadros de texto en los que el usuario puede escribir texto de búsqueda.
2. El usuario elige un botón "Buscar ahora". Los datos de un formulario cumplimentado se envían a un servidor Web para ser procesados, junto con el nombre de la aplicación de búsqueda de páginas Web.
3. El servidor Web llama a su aplicación mediante el protocolo ISAPI (Internet Server API), pasándole un parámetro que contiene la información de búsqueda.
4. La aplicación busca la base de datos. Cuando obtiene resultados, los inserta en una plantilla de página Web y, a continuación, envía la página Web al servidor otra vez como una secuencia de caracteres.
5. El servidor Web envía la página de resultados al explorador que inició la búsqueda.
6. El explorador muestra al usuario la página resultados.

Si ha creado páginas Web, la mayor parte de los pasos de este proceso le resultarán familiares. Por ejemplo, es posible que ya sepa cómo crear páginas Web. Incluso si aún no está familiarizado con el diseño de páginas Web, probablemente encontrará bastante fácil el proceso de crear estas páginas.

Para ver un ejemplo de cómo usar Visual FoxPro como motor de búsqueda de Web, vea el ejemplo Foxisapi.dll del directorio ...\\Samples\\Vfp98\\Servers\\Foxisapi de Visual Studio. Lea el archivo

README.TXT de ese directorio para obtener detalles sobre cómo ejecutar el ejemplo.

# Manual del programador, Parte 11: Lo nuevo en Visual FoxPro



Los capítulos siguientes describen las características nuevas de Visual FoxPro 6.0. Estas características aceleran y facilitan más que nunca la creación de aplicaciones de Visual FoxPro y le permiten crear aplicaciones para Internet e intranets.

## **Capítulo 31** [Interoperabilidad e Internet](#)

Use arrastrar y colocar de OLE para programar aplicaciones que le permitan mover datos entre aplicaciones para Windows y dentro de una aplicación de Visual FoxPro. Cree aplicaciones y servidores de Visual FoxPro para su uso con Internet.

## **Capítulo 32** [Programación de aplicaciones y productividad del programador](#)

La Galería de componentes y las Foundation Classes, la aplicación Analizador de trayecto, Enganches del Administrador de proyectos y Asistentes

## **Capítulo 33** [Mejoras para la programación](#)

Nuevas características de programación diseñadas para mejorar la productividad del programador, entre las que se cuentan los métodos Access y Assign, la compatibilidad con más formatos gráficos y nuevas incorporaciones al lenguaje para simplificar las tareas de programación. Además, se han agregado a Visual FoxPro muchas de las funciones de manipulación de nombres de archivos disponibles en Foxtools.fll, una biblioteca API de Visual FoxPro.

# Capítulo 31: Interoperabilidad e Internet

Microsoft Visual FoxPro 6.0 admite la técnica arrastrar y colocar OLE, que permite mover datos entre Visual FoxPro y otras aplicaciones, y también dentro de las propias aplicaciones de Visual FoxPro.

Además, Visual FoxPro 6.0 facilita la creación de aplicaciones para su uso con Internet y otros programas para Windows, como Microsoft Excel y Microsoft Visual Basic. Visual FoxPro 6.0 permite crear [documentos activos](#) que pueden alojarse en contenedores de documentos activos, como exploradores de Internet.

Visual FoxPro 6.0 proporciona servidores de Automatización mejorados para trabajar con Internet, Microsoft Transaction Server y Active Desktop.

En este capítulo se tratan los temas siguientes:

- [Técnica arrastrar y colocar de OLE](#)
- [Documentos activos](#)
- [Mejoras de servidor](#)

## Técnica arrastrar y colocar de OLE

Visual FoxPro ya es compatible con la técnica arrastrar y colocar de OLE, una eficaz y útil herramienta que permite mover datos entre las aplicaciones que admiten dicha técnica (como Visual FoxPro, Visual Basic, el Explorador de Windows, Microsoft Word y Excel, etc.). En una aplicación de Visual FoxPro distribuida, puede mover datos de unos controles a otros o entre controles y otras aplicaciones para Windows compatibles con la técnica arrastrar y colocar de OLE.

Tenga en cuenta que las versiones anteriores de Visual FoxPro admitían la funcionalidad de arrastrar y colocar para los controles mediante programación, lo que permitía mover los controles en un formulario. Esta forma de arrastrar y colocar se mantiene en Visual FoxPro 6.0. Sin embargo, si elige implementar la técnica de arrastrar y colocar en sus aplicaciones, debe usar sólo una de las dos técnicas, arrastrar y colocar controles mediante programación o la técnica arrastrar y colocar de OLE, sin mezclar ambos tipos.

El conocimiento de los fundamentos de la técnica arrastrar y colocar de OLE facilita el aprovechamiento total de sus posibilidades.

### Arrastrar y colocar datos

Para arrastrar y colocar datos entre aplicaciones y controles se utiliza el *mouse* (ratón). Por ejemplo, puede seleccionar un conjunto de archivos en el Explorador de Windows. A continuación, puede mantener presionado el botón del *mouse* mientras los arrastra y después liberarlo para colocar los archivos en el Administrador de proyectos de Visual FoxPro; o bien, puede seleccionar texto en un documento de Word y colocarlo en un cuadro de texto de un formulario de Visual FoxPro. Durante la operación arrastrar y colocar de OLE, el cursor del *mouse* cambia de forma para indicar que la operación está en curso.

### Origen de arrastre

La aplicación o control desde el que se mueven los datos se denomina *origen de arrastre*.

### Propiedades, eventos y métodos del origen de arrastre

En la siguiente tabla se indican las propiedades, eventos y métodos disponibles para un origen de arrastre OLE.

Propiedad, evento o método	Descripción
<a href="#">Evento OLECompleteDrag</a>	Ocurre cuando se colocan los datos en el destino de colocación o cuando se cancela la operación OLE de arrastrar y colocar.
<a href="#">Método OLEDrag</a>	Inicia una operación OLE de arrastrar y colocar.
<a href="#">Propiedad OLEDragPicture</a>	Especifica la imagen que aparece debajo del puntero del <i>mouse</i> durante una operación arrastrar y colocar de OLE. Puede especificar un archivo de imagen de tipo .bmp, .dib, .jpg, .gif, .ani, .cur o .ico.
<a href="#">Propiedad OLEDragMode</a>	Especifica la forma en que un origen de arrastre administra las operaciones de arrastre OLE.
<a href="#">Evento OLEGiveFeedBack</a>	Ocurre después de cada evento OLEDragOver. Permite al origen de arrastre especificar el tipo de operación arrastrar y colocar de OLE, así como el resultado visual.
<a href="#">Evento OLESetData</a>	Ocurre cuando un destino para colocar llama al método GetData y no hay datos con un formato especificado en el objeto DataObject al que se refiere la operación OLE de arrastrar y colocar.
<a href="#">Evento OLEStartDrag</a>	Ocurre cuando se llama al método OLEDrag.

## Destino para colocar

La aplicación o control al que se mueven los datos se denomina *destino para colocar*.

### Propiedades y eventos del destino para colocar

En la tabla siguiente se indican las propiedades, eventos y métodos disponibles para un destino para colocar OLE.

Propiedad o evento	Descripción
<a href="#">Evento OLEDragDrop</a>	Ocurre cuando se colocan datos en un destino para colocar y la propiedad OLEDropMode de éste tiene el valor 1 – Activado.
<a href="#">Evento OLEDragOver</a>	Ocurre cuando se arrastran datos a un destino para colocar y la propiedad OLEDropMode de éste tiene el valor 1 – Activado.
<a href="#">Propiedad OLEDropEffects</a>	Especifica el tipo de operaciones de colocación que admite un destino para colocar OLE.
<a href="#">Propiedad OLEDropHasData</a>	Especifica la forma de administrar una operación de

colocación.

---

[Propiedad OLEDropMode](#)

---

Especifica la forma en que un destino para colocar administra las operaciones colocar de OLE.

---

## Mover datos

Para realizar una operación de arrastrar y colocar con el fin de mover datos con el botón predeterminado del *mouse* (principal), seleccione los datos que desee mover en el origen de arrastre. Una vez seleccionados los datos, mantenga presionado el botón del *mouse* mientras desplaza el puntero hasta el destino de colocación. Suelte el botón del *mouse* para colocar los datos en el destino. Durante la operación arrastrar y colocar de OLE, el cursor del *mouse* cambia de forma para indicar que la operación está en curso.

También puede hacer clic con el botón no predeterminado del *mouse* (secundario) en los datos del origen de arrastre y arrastrarlos hasta un destino de colocación. En función del destino, al colocar los datos puede aparecer un menú contextual con un conjunto de opciones que permiten elegir la forma en que se van a procesar los datos en el destino de colocación.

## Copiar datos

También puede copiar datos desde un origen de arrastre y pegarlos en un destino para colocar. Presione la tecla Ctrl mientras hace clic con el *mouse* en los datos seleccionados en el origen de arrastre. En el cursor del *mouse* aparecerá un signo más (+) mientras se arrastran los datos, para indicar que se está realizando una copia.

## Destinos y orígenes que no admiten la técnica arrastrar y colocar de OLE

Sólo se puede mover o copiar datos desde un origen de arrastre compatible con la técnica arrastrar y colocar de OLE hasta un destino de colocación que también admita dicha característica. Tenga en cuenta que aunque un destino de colocación admita la técnica arrastrar y colocar de OLE, ello no implica que acepte los datos que desee colocar en él. Por ejemplo, es posible que los datos que desea mover o copiar tengan un formato que no es compatible con el destino de colocación. En las operaciones de arrastrar y colocar, el cursor se transforma en el símbolo No colocar (un círculo tachado) para indicar que el *mouse* se encuentra en una área de una aplicación o de un control en el que no se pueden colocar los datos.

## Cancelar una operación

Para cancelar una operación arrastrar y colocar de OLE, presione ESC mientras la efectúa.

## Técnica arrastrar y colocar de OLE en tiempo de diseño

La técnica arrastrar y colocar de OLE en tiempo de diseño de Visual FoxPro hace que la programación de aplicaciones sea aún más rápida que en las versiones anteriores. Esta técnica permite colocar de forma sencilla en el Administrador de proyectos y en los diseñadores de Visual FoxPro archivos del Explorador de Windows. También se permite mover o copiar fácilmente texto desde otras aplicaciones a la ventana Comandos, a los editores de texto de Visual FoxPro y a la ventana

## Propiedades.

En la tabla siguiente se indican las características de tiempo de diseño de Visual FoxPro que admiten la técnica arrastrar y colocar de OLE, junto con una descripción de su uso.

Elemento de la interfaz	Descripción
Ventana Comandos	<p data-bbox="605 405 1422 472">Destino para colocar archivos y origen de arrastre y destino para colocar texto.</p> <p data-bbox="605 516 1438 873">Si se coloca un archivo creado con Visual FoxPro en la ventana Comandos, se abrirá el archivo con el comando de Visual FoxPro correspondiente. Por ejemplo, si se coloca en la ventana Comandos una base de datos, Visual FoxPro ejecutará los comandos OPEN DATABASE y MODIFY DATABASE para abrir el archivo y poder modificarlo. Si se coloca en la ventana Comandos una tabla, ésta se abrirá con los comandos USE ... AGAIN y BROWSE. Si SET EXCLUSIVE tiene el valor ON, la tabla se abrirá para uso exclusivo. Si SET EXCLUSIVE tiene el valor OFF, se abrirá para uso compartido.</p> <p data-bbox="605 917 1417 1056">Otros archivos de Visual FoxPro se abren con el comando MODIFY correspondiente: los formularios se abren con MODIFY FORM, las consultas con MODIFY QUERY, los archivos de texto y de encabezado (.H) con MODIFY FILE, etc.</p> <p data-bbox="605 1100 1422 1272">Si se coloca en la ventana Comandos un archivo creado con otra aplicación, se abrirá en la aplicación a la que esté asociado. Por ejemplo, al colocar una hoja de cálculo de Microsoft Excel en la ventana Comandos se iniciará Excel y abrirá dicha hoja de cálculo.</p>
Administrador de proyectos	<p data-bbox="605 1306 990 1335">Destino para colocar archivos.</p> <p data-bbox="605 1379 1438 1554">Los archivos se agregan a las correspondientes categorías del Administrador de proyectos, en función de sus extensiones de archivo. Si Visual FoxPro no reconoce la extensión de un archivo que se coloca en el Administrador de proyectos, el archivo se agregará a la categoría Otros.</p> <p data-bbox="605 1598 1443 1881">Si se coloca en el Administrador de proyectos una tabla contenida en una base de datos, ésta se agregará a la categoría Bases de datos del elemento Datos y se marcará como Excluida. Si se coloca en el Administrador de proyectos una tabla libre, ésta se agregará a la categoría Tablas libres del elemento Datos y se marcará como Excluida. Si se coloca en el Administrador de proyectos una base de datos, ésta se agregará a la categoría Bases de datos del elemento Datos y se marcará como Excluida.</p>

Aunque con la técnica arrastrar y colocar de OLE es fácil agregar archivos al Administrador de proyectos, recuerde que el Administrador de proyectos agrega automáticamente al proyecto todos los archivos a los que se hace referencia cuando se genera. Por ejemplo, si un programa agregado al proyecto ejecuta otro programa, éste se agregará automáticamente al generar el proyecto. No es necesario agregar el segundo programa manualmente.

Editores de texto	<p>Origen de arrastre y destino de colocación para texto.</p> <p>Se consideran editores de texto las ventanas de edición abiertas con MODIFY COMMAND, MODIFY FILE y MODIFY MEMO; la ventana Comandos; las ventanas de edición de miniprogramas de los Diseñadores de formularios, de clases, de menús y de entornos de datos; y el editor de procedimientos almacenados del Diseñador de bases de datos.</p>
Depurador	<p>Origen de arrastre y destino para colocar texto.</p> <p>El cuadro de texto de la ventana Inspección y la lista Nombre son orígenes de arrastre y destinos de colocación de texto. Las ventanas Seguimiento y Resultados del depurador son sólo orígenes de arrastre para texto.</p>
Diseñador de bases de datos	<p>Destino para colocar archivos.</p> <p>Al colocar una tabla en el Diseñador de bases de datos, la tabla se agrega a la base de datos actual.</p>
Diseñador de clases	<p>Destino para colocar texto y archivos.</p> <p>De forma predeterminada, al colocar texto en un objeto contenedor del Diseñador de clases se crea una etiqueta con ese texto como valor de la propiedad Caption. Puede cambiar el control predeterminado creado al colocar texto en el Diseñador de formularios. Para ello, utilice la ficha Asignación de campos del cuadro de diálogo Opciones.</p> <p>Si coloca texto en un control no contenedor (CheckBox, CommandButton, Header, Label u OptionButton), el texto se asignará a la propiedad Caption del control.</p> <p>Al colocar un archivo gráfico (.ani, .bmp, .cur, .gif, .ico o .jpg) en el Diseñador de clases se crea un control Image cuya propiedad Picture tiene como valor el nombre de ese archivo gráfico.</p>
Diseñador de entornos de datos	Destino para colocar archivos.



Al colocar una tabla en el Diseñador de entornos de datos, se agrega la tabla al entorno de datos. Al colocar una base de datos en el Diseñador de entornos de datos aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista al entorno de datos.

---

---

Diseñador de consultas

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de consultas, se agrega la tabla a la consulta. Al colocar una base de datos en el Diseñador de consultas aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista a la consulta.

---

---

Diseñador de vistas

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de vistas, se agrega la tabla a la vista. Al colocar una base de datos en el Diseñador de vistas aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista a la vista.

---

---

Ventana Propiedades

Destino para colocar texto.

Puede arrastrar texto al cuadro de texto que aparece en la parte superior de la ventana Propiedades al seleccionar una propiedad en tiempo de diseño.

---

---

Galería de componentes

Origen de arrastre y destino para colocar archivos.

Puede arrastrar objetos desde la Galería de componentes y colocarlos en el Diseñador de formularios. También puede arrastrar archivos desde la Galería de componentes y colocarlos en el Administrador de proyectos.

Además, puede colocar archivos en la Galería de componentes.

---

---

## Técnica arrastrar y colocar de OLE en tiempo de ejecución

La técnica arrastrar y colocar de OLE está disponible en tiempo de ejecución para los controles de Visual FoxPro y para el editor de texto. En tiempo de ejecución, los controles y el editor de texto admiten esta técnica de forma interactiva; además, los controles admiten la técnica mediante programación. El [objeto DataObject](#) permite el uso de la técnica arrastrar y colocar de OLE mediante programación para los controles.

Existen dos técnicas arrastrar y colocar de OLE disponibles para los controles de Visual FoxPro: el modo intrínseco y el modo manual. En el primero, Visual FoxPro controla intrínsecamente la operación arrastrar y colocar de OLE. En el modo manual, las operaciones arrastrar y colocar de OLE se controlan mediante programación. Los eventos que se producen están determinados por el modo utilizado. Si desea obtener más información, vea la sección "Modos intrínseco y manual Arrastrar y

colocar de OLE".

## Arrastrar y colocar en versiones anteriores de Visual FoxPro

En las versiones anteriores de Visual FoxPro se utilizaba la técnica de arrastrar y colocar mediante programación para los controles, lo que permitía mover los controles de un formulario. Este modo de arrastrar y colocar sigue siendo compatible. Si utiliza los valores predeterminados de las propiedades `OLEDragMode` y `OLEDropMode`, podrá ejecutar las aplicaciones existentes como antes, sin ningún cambio.

## El objeto DataObject

El [objeto DataObject](#) es un contenedor para los datos que se transfieren desde un origen de arrastre OLE hasta un destino para colocar OLE y sólo existe mientras se realiza la operación arrastrar y colocar de OLE. El objeto `DataObject` no se puede crear mediante programación y las referencias al mismo dejan de ser válidas al finalizar la operación de arrastrar y colocar. El objeto `DataObject` se pasa como parámetro `oDataObject` en los eventos `OLEStartDrag`, `OLEDragOver`, `OLEDragDrop` y `OLESetData`.

El objeto `DataObject` puede almacenar varios conjuntos de datos, cada uno con un formato distinto. Puede usar el método `GetFormat` para determinar si existe un formato específico en el objeto `DataObject`. Consulte [Método GetFormat](#) para ver una lista con los formatos que admite el objeto `DataObject`.

## Métodos del objeto DataObject

El objeto `DataObject` tiene métodos que permiten manipular mediante programación los datos que se arrastran y colocan. En la tabla siguiente se indican los métodos disponibles en tiempo de ejecución para el objeto `DataObject`.

Método	Descripción
<a href="#">ClearData</a>	Borra todos los datos y formatos del objeto <code>DataObject</code> de la operación arrastrar y colocar de OLE.
<a href="#">GetData</a>	Recupera los datos del objeto <code>DataObject</code> de la operación arrastrar y colocar de OLE.
<a href="#">GetFormat</a>	Determina si hay datos con el formato especificado disponibles en el objeto <code>DataObject</code> de la operación arrastrar y colocar de OLE.
<a href="#">SetData</a>	Establece los datos y el formato de los mismos en el objeto <code>DataObject</code> de la operación arrastrar y colocar de OLE.
<a href="#">SetFormat</a>	Establece un formato de datos, sin los datos, en el objeto <code>DataObject</code> de la operación arrastrar y colocar de OLE.

## Modos intrínseco y manual arrastrar y colocar de OLE

Visual FoxPro admite dos modos de arrastrar y colocar de OLE para los controles: el modo intrínseco

y el modo manual. En el primero, las operaciones arrastrar y colocar de OLE las controla Visual FoxPro. En el modo manual, las operaciones se controlan mediante programación.

### Modo intrínseco de arrastrar y colocar de OLE

El modo intrínseco arrastrar y colocar de OLE se puede implementar en una aplicación para proporcionar compatibilidad con la técnica arrastrar y colocar de OLE sin necesidad de programación adicional.

#### Para implementar la técnica intrínseca de arrastrar y colocar de OLE para un control

1. Asigne el valor **1** – Automático a la propiedad `OLEDragMode` del control para permitir que éste actúe como origen de arrastre OLE.
2. Asigne el valor **1** – Activado a la propiedad `OLEDropMode` del control para permitir que éste actúe como destino para colocar OLE.

En las operaciones intrínsecas de arrastrar y colocar de OLE, Visual FoxPro determina si el destino para colocar admite el formato de los datos que se intentan colocar en él. Sólo se colocarán los datos si el destino es compatible.

En la tabla siguiente se indican los controles de Visual FoxPro con los formatos de datos que admiten como orígenes de arrastre en el modo intrínseco. Tenga en cuenta que `CF_TEXT` es texto, como el que escribe en un cuadro de texto, y `CFSTR_VFPSOURCEOBJECT` es una referencia a tipo de objeto para un control u objeto de Visual FoxPro. En el caso de los controles que admiten el formato de datos `CF_TEXT`, puede arrastrar texto desde la parte de texto del control.

#### Formatos de datos de los orígenes de arrastre

Control	Formato de datos (definido en Foxpro.h)
Container, Image, Line, PageFrame y Shape	CFSTR_VFPSOURCEOBJECT
CommandButton y Label	CFSTR_VFPSOURCEOBJECT y CF_TEXT
CheckBox, ComboBox, EditBox, ListBox, Spinner y TextBox	CFSTR_VFPSOURCEOBJECT, CF_TEXT y CFSTR_OLEVARIANT

En la tabla siguiente se indican los controles de Visual FoxPro con los formatos de datos que admiten como destinos para colocar en el modo intrínseco. En el caso de los controles, puede colocar texto en la parte de texto del control. El texto se sitúa en el punto de inserción.

#### Formatos de datos de los destinos para colocar

Control	Formato de datos
EditBox y ComboBox (cuando la propiedad Style del ComboBox es 0 - Cuadro desplegable)	CF_TEXT
Spinner y TextBox	CFSTR_OLEVARIANT

### Modo manual de arrastrar y colocar de OLE

Es posible que en algunos casos desee controlar el tipo de datos que se pueden colocar en un destino para colocar o proporcionar una funcionalidad adicional a una operación de arrastrar y colocar. Por ejemplo, puede convertir los datos a un formato que el destino para colocar admita o puede mostrar un cuadro de diálogo que pregunte al usuario si desea colocar los datos en el destino. En estos casos, puede prescindir el modo intrínseco de arrastrar y colocar de OLE para lograr un mayor control sobre las operaciones de arrastrar y colocar.

Para implementar el modo manual de arrastrar y colocar de OLE para un control, anule los eventos o métodos de arrastrar y colocar que desee controlar; para ello, escriba su propio código para cada evento o método. Incluya la palabra clave NODEFAULT en el código del evento o método para pasar por alto el comportamiento intrínseco de Visual FoxPro al arrastrar y colocar.

La compatibilidad con aplicaciones existentes de versiones anteriores (sin arrastre OLE) se consigue cuando el valor de OLEDragMode es 0 (valor predeterminado) y no se incluye código adicional para operaciones arrastrar y colocar de OLE.

## Documentos activos

Visual FoxPro 6.0 permite crear documentos activos. Los documentos activos permiten ver documentos con formato distinto de HTML en un contenedor explorador de Web, como Microsoft Internet Explorer. La tecnología de documentos activos permite ver varios tipos de documentos de diversos orígenes en un mismo contenedor de documentos activos.

Un documento activo es un tipo específico de documento OLE que se puede incrustar. Se muestra en todo el área cliente de un contenedor de documentos activos y sus menús se combinan con los del contenedor. El documento activo ocupa todo el marco y está siempre activo en contexto.

Las siguientes son algunas de las características de los documentos activos:

- Los documentos activos siempre están activos en el contexto.
- Los comandos de menús y de barras de herramientas del documento activo se pueden dirigir al contenedor de documentos activos.
- Los documentos activos proporcionan integración directa con otras páginas Web cuando se ven en Internet Explorer.
- Los documentos activos suponen un paso adelante en la evolución desde las aplicaciones

clientes puras de Visual FoxPro a las aplicaciones Active Platform que utilizan una interfaz de cliente basada en HTML.

## Crear un documento activo

Los documentos activos de Visual FoxPro se crean fácilmente. Un documento activo de Visual FoxPro, como cualquier otra aplicación de Visual FoxPro, puede manipular datos, ejecutar formularios, informes y etiquetas, crear instancias de clases y ejecutar código.

Un documento activo de Visual FoxPro es una aplicación (.app) creada desde un proyecto de Visual FoxPro. Las versiones anteriores de Visual FoxPro ya permitían crear aplicaciones, por lo que es posible que esté familiarizado con el proceso. Si desea obtener más información sobre la creación de aplicaciones, consulte el capítulo 13, [Compilar una aplicación](#), en el *Manual del programador*.

Puede ejecutar cualquier aplicación en Internet Explorer. Sin embargo, sólo las aplicaciones basadas en la clase de base ActiveDoc, descrita más adelante, admiten las propiedades, eventos y métodos que permiten la comunicación con el contenedor de documentos activos.

## La clase de base ActiveDoc

Los documentos activos de Visual FoxPro son ligeramente diferentes de otras aplicaciones (.app). La diferencia más destacada es que el “archivo principal” de un documento activo debe ser una clase basada en la clase base ActiveDoc. Otros tipos de aplicaciones requieren que el archivo principal sea un programa o un formulario.

Una clase basada en la clase de base ActiveDoc se crea con el Diseñador de clases y sirve como base para todos los documentos activos de Visual FoxPro. La clase de base ActiveDoc proporciona las propiedades, eventos y métodos para un documento activo, así como la comunicación con el contenedor de documentos activos. Por ejemplo, el evento ContainerRelease se produce cuando un contenedor libera un documento activo. Puede incluir código en este evento para cerrar archivos, completar transacciones y realizar otras tareas de limpieza antes de liberar el documento activo.

## Para establecer como archivo principal una clase basada en la clase de base ActiveDoc

1. Agregue al proyecto la biblioteca de clases visuales (.vcx) que contiene la clase basada en la clase de base ActiveDoc.
2. Expanda la jerarquía de la biblioteca de clases visuales (.vcx); para ello, haga clic en el cuadro más (+) situado a la izquierda del nombre de la biblioteca o haga clic con el botón secundario del *mouse* en la biblioteca y elija **Expandir todo** en el menú contextual.
3. Seleccione la clase basada en la clase de base ActiveDoc. Haga clic con el botón secundario del *mouse* en la clase y elija **Establecer principal** en el menú contextual.

## El objeto ActiveDoc

Cuando se ejecuta un documento activo de Visual FoxPro Active en Internet Explorer, se crea un objeto ActiveDoc a partir de la clase de base ActiveDoc. Este objeto responde a los eventos y llamadas a métodos de la clase de base ActiveDoc.

## Propiedades, eventos y métodos del objeto ActiveDoc

En las tablas siguientes se indican las propiedades, eventos y métodos que admite el objeto ActiveDoc.

### Propiedades

<a href="#">BaseClass</a>	<a href="#">Caption</a>	<a href="#">Class</a>
<a href="#">ClassLibrary</a>	<a href="#">Comment</a>	<a href="#">ContainerReleaseType</a>
<a href="#">Name</a>	<a href="#">Parent</a>	<a href="#">ParentClass</a>
<a href="#">Tag</a>		

### Eventos

<a href="#">CommandTargetExec</a>	<a href="#">CommandTargetQuery</a>	<a href="#">ContainerRelease</a>
<a href="#">Destroy</a>	<a href="#">Error</a>	<a href="#">HideDoc</a>
<a href="#">Init</a>	<a href="#">Run</a>	<a href="#">ShowDoc</a>

### Métodos

<a href="#">AddProperty</a>	<a href="#">ReadExpression</a>	<a href="#">ReadMethod</a>
<a href="#">ResetToDefault</a>	<a href="#">SaveAsClass</a>	<a href="#">WriteExpression</a>

### Secuencia de eventos en los documentos activos

Cuando se abre una aplicación de tipo documento activo en Internet Explorer, se ejecuta el documento activo y se produce su evento Init. A continuación, se produce el evento ShowDoc del documento activo. Cuando Internet Explorer aloja correctamente el documento activo, se produce el evento Run del mismo. En general, el código del programa de documento activo debe situarse en este evento. Normalmente, el evento Run contiene código que ejecuta los menús, ejecuta el formulario principal de la aplicación y contiene READ EVENTS para iniciar el procesamiento de los eventos, como en una aplicación Visual FoxPro estándar.

Puede incluir código de configuración en el evento Init del documento activo, pero si tarda demasiado tiempo en ejecutarse, es posible que el contenedor del documento activo genere un error de tiempo de espera. Si incluye código de configuración en el evento Init, no debe requerir interacción del usuario ni crear una interfaz de usuario.

El evento HideDoc ocurre al desplazarse desde un documento activo y el evento ShowDoc se produce al volver al mismo.

Si se cierra Internet Explorer cuando aloja el documento activo, se producirá el evento HideDoc y, a continuación, el evento ContainerRelease. Este último evento también se produce si el documento activo queda fuera de la memoria caché de Internet Explorer 3.0.

Cuando se produce el evento ContainerRelease, el código de programa del evento puede realizar las siguientes acciones:

- Cerrar archivos, limpiar su rastro y ejecutar QUIT para cerrar el documento activo.
- Asignar el valor 0 (predeterminado) a la propiedad [ContainerReleaseType](#), con lo que se abre el documento activo en el entorno de tiempo de ejecución de Visual FoxPro. El documento activo continuará ejecutándose en la ventana principal de tiempo de ejecución de Visual FoxPro.

**Nota** El evento CommandTargetExec se produce cuando Internet Explorer 4.0 va a cerrar el documento activo o a explorar desde el mismo. En este caso, se asigna el valor 37 al parámetro *nIdComando* de CommandTargetExec y puede asignar el valor falso (.F.) al parámetro *eSalArg* para evitar que Internet Explorer cierre el documento activo. Internet Explorer 3.0 no admite el evento CommandTargetExec.

## Nuevas funciones de documentos activos

Se han agregado a Visual FoxPro dos nuevas funciones, [GETHOST\(\)](#) e [ISHOSTED\(\)](#), para proporcionar información sobre el contenedor de un documento activo. GETHOST() devuelve la referencia de un objeto al contenedor de un documento activo. ISHOSTED() devuelve un valor lógico que indica si un documento activo se encuentra en un contenedor.

## Cambios en el objeto Form

La interfaz de usuario de un documento activo de Visual FoxPro está definida por su código de programa. En general, un formulario de Visual FoxPro debe mostrarse como interfaz de usuario inicial. Las siguientes propiedades, eventos y métodos de formulario se han agregado a Visual FoxPro para que los formularios funcionen correctamente con documentos activos.

### Propiedades

<a href="#">AlwaysOnBottom</a>	<a href="#">ContinuousScroll</a>	<a href="#">HscrollSmallChange</a>
<a href="#">Scrollbars</a>	<a href="#">TitleBar</a>	<a href="#">ViewPortHeight</a>
<a href="#">ViewPortLeft</a>	<a href="#">ViewPortTop</a>	<a href="#">ViewPortWidth</a>
<a href="#">VscrollSmallChange</a>		

### Eventos

<a href="#">Scrolled</a>
--------------------------

## Métodos

---

### [SetViewPort](#)

---

## Formularios de documentos activos

Los formularios de un documento activo se muestran en el área cliente que ofrece Internet Explorer. Para hacer que un formulario se muestre completamente en el área cliente de Internet Explorer, asigne los siguientes valores a las propiedades de formulario que se indican a continuación:

BorderStyle = 0 (Sin borde)  
 TitleBar = 0 (Desactivado)  
 WindowState = 2 (Maximizado)

Además, si se van a mostrar barras de desplazamiento cuando el área de cliente de Internet Explorer sea más pequeña que el área de visualización del documento activo (el área determinada por un rectángulo que incluye todos los controles del formulario), debe asignar el siguiente valor a la propiedad Scrollbars:

ScrollBars = 3 (Barras de desplazamiento horizontal y vertical)

## Menús en documentos activos

Si se ejecuta código de menús en un documento activo de Visual FoxPro, los menús se combinan con los menús de Internet Explorer, conforme a unas reglas de combinación de menús específicas. Cuando se hayan combinado los menús del documento activo con los de Internet Explorer, los primeros se verán como en una aplicación normal de Visual FoxPro.

### Negociar menús

En Visual FoxPro 6.0 y en las versiones anteriores, puede especificar el comportamiento de la negociación de menús cuando un control ActiveX contenido en un formulario de Visual FoxPro se modifica de forma visual mediante OLE. En Visual FoxPro 6.0, se ha mejorado la negociación de menús para permitir controlar el lugar de Internet Explorer en el que deben aparecer los menús del documento activo.

Cuando se abre un documento activo en Internet Explorer, el espacio para menús de Internet Explorer se comparte y los menús se combinan. Los menús combinados se dividen en seis grupos y cada uno de ellos pertenece a Internet Explorer, al documento activo, o a ambos.

Grupo	Propietario
Grupo Archivo	Internet Explorer
Grupo Edición	Documento activo
Grupo Contenedor	Internet Explorer
Grupo Objeto	Documento activo



Grupo Ventana	Internet Explorer
Grupo Ayuda	Documento activo o Internet Explorer

### Combinar el menú Ayuda

El documento activo comparte su menú Ayuda con Internet Explorer. Si Internet Explorer tiene un menú Ayuda, el documento activo puede agregar el suyo al final del menú Ayuda de Internet Explorer.

### Mejoras del lenguaje para la negociación de menús

Se ha mejorado la cláusula `DEFINE PAD NEGOTIATE` para permitir especificar la forma en que se produce la negociación de los menús en un documento activo. Una nueva segunda opción, *cPosiciónObjeto*, especifica la ubicación del título de un menú en la barra de menús de Internet Explorer.

Para obtener más información, vea [DEFINE PAD](#) en la *Referencia del lenguaje*.

### Negociación de menús y el Diseñador de menús

Se ha mejorado el cuadro de diálogo [Opciones de la acción](#) del Diseñador de menús para permitir especificar la negociación de los menús creados en el Diseñador de menús e incluidos en documentos activos. Se ha agregado un cuadro desplegable **Objeto**, que especifica la forma de negociar el título del menú cuando Internet Explorer actúa como contenedor de un documento activo de Visual FoxPro.

### Información de negociación de menús

La información sobre la negociación de menús se almacena en el campo `Location` del archivo `.mnx` de cada menú. En la tabla siguiente se indican los valores de este campo y el tipo de negociación que representa cada uno. Para obtener más información sobre *cPosiciónContenedor* y *cPosiciónObjeto*, vea [DEFINE PAD](#).

Valor	<i>cPosiciónContenedor</i>	<i>cPosiciónObjeto</i>
0	Ninguna	Ninguna
1	Izquierda	Ninguna
2	Centro	Ninguna
3	Derecha	Ninguna
4	Ninguna	Izquierda
5	Izquierda	Izquierda
6	Centro	Izquierda
7	Derecha	Izquierda

8	Ninguna	Centro
9	Izquierda	Centro
10	Centro	Centro
11	Derecha	Centro
12	Ninguna	Derecha
13	Izquierda	Derecha
14	Centro	Derecha
15	Derecha	Derecha

Tenga en cuenta que el tamaño del campo Location se ha aumentado de 1 a 2 dígitos en Visual FoxPro 6.0. Éste es el único cambio en Visual FoxPro 6.0 en las [estructuras de las tablas](#), incluidas las tablas de bases de datos (.dbc), de formularios (.scx), de etiquetas (.lbx), de proyectos (.pjx), de informes (.frx) y bibliotecas de clases visuales (.vcx).

### Eventos CommandTargetExec y CommandTargetQuery

Dos eventos de documentos activos, CommandTargetExec y CommandTargetQuery, permiten administrar las selecciones de los menús de Internet Explorer (y otros eventos de Internet Explorer) desde un documento activo. El evento CommandTargetExec se produce cuando Internet Explorer notifica a un documento activo que se va a ejecutar un comando (que puede ser un comando de menú). El evento CommandTargetQuery ocurre cuando Internet Explorer actualiza su interfaz de usuario. Si desea obtener más información sobre estos eventos, consulte [CommandTargetExec \(Evento\)](#) y [CommandTargetQuery \(Evento\)](#) en la *Referencia del lenguaje*.

### Ejecutar documentos activos

Para ejecutar los documentos activos de Visual FoxPro son necesarios los archivos Vfp6.exe y Vfp6run.exe, o Vfp6r.exe, Vfp6r.dll y Vfp6res.dll (es indica que se trata de la versión en español). Estos archivos deben estar instalados y registrados en el equipo en el que está instalado Internet Explorer. Cuando se instala Visual FoxPro, Vfp6.exe se instala en el directorio de Visual FoxPro y los archivos restantes en el directorio Windows\System de Windows 95 o en el directorio WinNT\System32 de Windows NT.

### Ejecutar documentos activos desde el menú Herramientas

El menú Herramientas de Visual FoxPro contiene el comando **Ejecutar documento activo** que muestra el cuadro de diálogo **Ejecutar documento activo**, en el que puede especificar cómo se va a ejecutar un documento activo. Están disponibles las siguientes opciones:

Opción	Descripción
En el explorador (Predeterminada)	El documento activo se ejecuta en Internet Explorer en el entorno de tiempo de ejecución de Visual FoxPro.
Independiente	El documento activo se ejecuta como una aplicación independiente con el entorno de tiempo de ejecución de Visual FoxPro.
En el explorador (Depuración)	El documento activo se ejecuta en Internet Explorer con el ejecutable de Visual FoxPro (Vfp6.exe). Estarán disponibles las herramientas de depuración, la ventana Comandos y todas las características del entorno de programación de Visual FoxPro.
Independiente (Depuración)	El documento activo se ejecuta como una aplicación independiente con el ejecutable de Visual FoxPro (Vfp6.exe). Estarán disponibles las herramientas de depuración, la ventana Comandos y todas las características del entorno de programación de Visual FoxPro.  Esta opción equivale a ejecutar DO <Nombre de Active Doc> en la ventana Comandos.

También puede ejecutar un documento activo si lo abre en el cuadro de diálogo Abrir archivo de Internet Explorer o si se desplaza hasta el documento activo desde otra página Web que contenga un vínculo al mismo.

### El entorno de tiempo de ejecución de Visual FoxPro y los documentos activos

Desde Visual FoxPro puede ejecutar un documento activo si hace doble clic en el icono del mismo en el Explorador de Windows. También puede ejecutar un documento activo desde una aplicación en el entorno de tiempo de ejecución de Visual FoxPro. El entorno de tiempo de ejecución de Visual FoxPro consta de dos archivos, Vfp6run.exe y Vfp6r.dll. Ambos deben estar instalados y registrados para poder ejecutar documentos activos. También se puede utilizar el entorno de tiempo de ejecución para ejecutar otros archivos Visual FoxPro distribuibles, como programas de Visual FoxPro compilados (archivos .fxp), por ejemplo.

Una vez registrado, puede utilizar Vfp6run.exe para ejecutar directamente documentos activos (y otros archivos Visual FoxPro distribuibles).

### Sintaxis de Vfp6run.exe

```
VFP6RUN [/embedding] [/regserver] [/unregserver] [/security]
[/s] [/version] [NombreArchivo]
```

### Argumentos

/embedding

Carga Vfp6run.exe como un servidor de documentos activos. En este modo, Vfp6run.exe se registra como servidor COM capaz de crear un objeto de tipo documento activo de Visual FoxPro ("Visual.FoxPro.Application.6"). Sin este argumento, Vfp6run.exe no actúa como servidor COM.

/regserver

Registra Vfp6run.exe.

/unregserver

Elimina Vfp6run.exe del registro.

/security

Muestra el cuadro de diálogo **Configuración de seguridad de aplicaciones**, que permite especificar las opciones de seguridad para los documentos activos y para otros archivos de aplicación (.app). Si desea obtener más información, consulte la siguiente sección, "Seguridad de documentos activos".

/s

Silenciosa. Especifica que se generará un error si Vfp6run.exe no puede cargar el componente de tiempo de ejecución Vfp6r.dll.

/version

Muestra la información de versión de Vfp6run.exe y Vfp6r.dll.

*NombreArchivo*

Especifica el archivo Visual FoxPro que se va a ejecutar.

Vfp6run.exe requiere que la biblioteca de vínculos dinámicos de soporte en tiempo de ejecución, Vfp6r.dll, esté instalada y registrada. Para registrar Vfp6r.dll, ejecute Regsvr32 con el nombre del entorno de tiempo de ejecución:

```
Regsvr32 Vfp6r.dll
```

## Seguridad de documentos activos

La opción /security del entorno de tiempo de ejecución de Visual FoxPro Vfp6run.exe permite establecer niveles de seguridad para los documentos activos y para otros archivos de aplicación (.app). Al ejecutar Vfp6run.exe /security aparece el cuadro de diálogo **Configuración de seguridad de aplicaciones**, en el que puede establecer niveles de seguridad para los documentos activos y para otros archivos .app.

En el cuadro de diálogo **Configuración de seguridad de aplicaciones** están disponibles las siguientes opciones:

**Alojado**

Elija esta opción de modo de aplicación para especificar el nivel de seguridad de un documento activo o de una aplicación (.app) que se ejecute desde un contenedor de documentos activos, como Internet Explorer.

**No alojado**

Elija esta opción de modo de aplicación para especificar el nivel de seguridad de un documento activo o de una aplicación (.app) que se ejecute desde el Explorador de Windows al hacer doble clic en su icono o que se ejecute con el entorno de tiempo de ejecución de Visual FoxPro, Vfp6run.exe.

**Alta (seguridad máxima)**

Elija esta opción para impedir que se ejecute un documento activo o una aplicación (.app).

**Media (más seguro)**

Elija esta opción para que aparezca una advertencia antes de que se ejecute un documento activo o una aplicación (.app). Ésta es la opción predeterminada para los documentos activos y aplicaciones que no se ejecutan en un contenedor.

**Baja (ninguna seguridad)**

Elija esta opción para ejecutar un documento activo o una aplicación (.app) sin que aparezca ninguna advertencia. Ésta es la opción predeterminada para los documentos activos y aplicaciones que se ejecutan en un contenedor.

**Restablecer**

Restaura el nivel de seguridad predeterminado para el modo de aplicación seleccionado (con contenedor o sin contenedor).

**Aceptar**

Guarda las opciones elegidas en el cuadro de diálogo.

**Notas sobre Internet Explorer**

Con el fin de aumentar el rendimiento, Internet Explorer 3.0 guarda en memoria caché las cuatro últimas páginas visitadas. Esto significa que un documento activo puede quedar fuera de la memoria caché de Internet Explorer 3.0 y se producirá el evento ContainerRelease. Internet Explorer 4.0 no mantiene una memoria caché de páginas, de forma que el evento ContainerRelease se produce en cuanto abandona el documento activo.

**Ejemplo de documento activo**

La aplicación de ejemplo Solutions de Visual FoxPro incluye un ejemplo llamado “Crear documentos activos para el Web” que ilustra muchas de las características de los documentos activos.

### Para ejecutar la aplicación de ejemplo Solutions

- Escriba la siguiente línea en la ventana **Comandos**:

```
DO (HOME(2) + 'solution\solution')
```

– o bien –

1. En el menú **Programa**, elija **Ejecutar**.
2. Elija la carpeta ...\\Samples\\Vfp98\\Solution.
3. Haga doble clic en **Solution.app**.

### Para ejecutar el ejemplo “Crear documentos activos para el Web”

1. Después de iniciar Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Crear documentos activos para el Web** y, a continuación, haga clic en el botón **Ejecutar ejemplo**.

El ejemplo “Crear documentos activos para el Web” permite abrir un proyecto que contiene todos los archivos necesarios para crear un documento activo desde un proyecto. Cuando el proyecto está abierto, puede examinar el código de la clase Actdoc para ver cómo se administran los eventos de documentos activos y cómo se ejecutan los formularios. Tenga en cuenta que Actdoc, una clase basada en la clase de base ActiveDoc, es el archivo principal del proyecto. Un documento activo debe tener como archivo principal una clase basada en la clase de base ActiveDoc.

También puede generar un documento activo desde el proyecto; para ello, elija **Generar** en el Administrador de proyectos. Una vez generado el documento activo, elija **Ejecutar documento activo** en el menú **Herramientas** para ejecutarlo.

## Mejoras en servidores de Automatización

En este tema se describen las mejoras realizadas en los servidores de Automatización de Visual FoxPro 6.0 y se incluyen explicaciones sobre la forma en que los servidores de Automatización de Visual FoxPro pueden funcionar con productos y tecnologías como Microsoft Transaction Server y Microsoft Visual Basic.

Visual FoxPro permite crear servidores de Automatización: aplicaciones de componentes que ofrecen funcionalidad que otras aplicaciones pueden usar y reutilizar a través de Automatización. Por ejemplo, con Visual FoxPro puede crear un servidor de Automatización que muestre formularios reutilizables (en un ejecutable fuera de proceso) o que empaquete una rutina compleja en un

componente simple disponible para otros programadores. Además, puede crear una o más clases para controlar las normas de negocios de toda una compañía. Una aplicación cliente que utilice los objetos de normas de negocios pasará los parámetros de entrada de una llamada a un método y el servidor de Automatización realizará numerosas operaciones para recuperar o almacenar datos de diversas fuentes y realizar cálculos complejos antes de devolver los resultados.

En Visual FoxPro puede crear servidores de Automatización [fuera de proceso](#) o [en proceso](#). Un componente *fuera de proceso* es un archivo ejecutable (.exe) que se ejecuta en su propio proceso. La comunicación entre una aplicación cliente y un servidor fuera de proceso se denomina, por lo tanto, comunicación *entre procesos*. Un componente *en proceso* es un archivo de biblioteca de vínculos dinámicos (.dll) que se ejecuta en el mismo espacio de direcciones que el proceso del cliente que lo llama o en un proceso de Microsoft Transaction Server.

Para obtener más información sobre la creación de servidores de Automatización en Visual FoxPro, vea [Crear servidores de Automatización](#), en el capítulo 16 del *Manual del programador*.

## Mejoras de los servidores de Automatización en Visual FoxPro 6.0

En los temas siguientes se describen las características nuevas y mejoradas de los servidores de Automatización en Visual FoxPro 6.0.

### Modelo de subprocesamiento controlado

Los servidores de Automatización de Visual FoxPro son compatibles ahora con el modelo de subprocesamiento controlado. Microsoft Transaction Server aprovecha la funcionalidad de los servidores marcados como servidores de subprocesamiento controlado y ofrece una mejor protección y escalabilidad de subprocesos.

En cada objeto del modelo controlado (por ejemplo, un servidor de Automatización de Visual FoxPro) sólo puede entrar el subproceso que lo creó (por ejemplo, con la llamada a CoCreateInstance en Microsoft Visual C++). Sin embargo, un servidor de objetos, como Microsoft Transaction Server, puede admitir múltiples objetos, en cada uno de los cuales entran simultáneamente subprocesos diferentes. Los datos comunes que mantiene el servidor de objetos se deben proteger contra colisiones entre subprocesos. El servidor de objetos crea un objeto del modelo de subprocesamiento controlado en el mismo subproceso que llamó a CoCreateInstance. Las llamadas al objeto desde el subproceso controlado no están resueltas.

Para obtener más información sobre el modelo de subprocesamiento controlado, busque “Apartment-Model Threading in Visual Basic” en la biblioteca MSDN.

### Interfaces de usuario y servidores en proceso

La nueva característica del modelo de subprocesamiento controlado requiere que los servidores de Automatización .dll en proceso no tengan interfaces de usuario. En Visual FoxPro 5.0 era posible (aunque no recomendable) crear un servidor de Automatización .dll en proceso con una interfaz de usuario, como un formulario. El formulario sólo se podía mostrar ya que los eventos del formulario no eran compatibles. En Visual FoxPro 6.0, cualquier intento de crear una interfaz de usuario en un servidor de Automatización .dll en proceso producirá un error.

Por el contrario, un servidor de Automatización .exe fuera de proceso sí puede tener una interfaz de usuario. Se ha agregado una nueva función a Visual FoxPro 6.0, [SYS\(2335\)](#), que permite desactivar los eventos modales de un servidor de Automatización .exe fuera de proceso de forma que se pueda ejecutar de forma remota sin intervención del usuario. Los eventos modales se crean mediante formularios modales definidos por el usuario, cuadros de diálogo del sistema, la función MESSAGEBOX( ), el comando WAIT, etc.

## Enlace en tiempo de compilación (vtable)

Visual FoxPro 6.0 permite ahora el enlace en tiempo de compilación (vtable) además de la interfaz IDispatch existente (lo que se conoce conjuntamente como interfaz dual). El enlace en tiempo de ejecución (vtable) mejora el rendimiento de los controladores de Automatización que admiten esta técnica, como Visual Basic y Microsoft Transaction Server.

## Entorno de tiempo de ejecución de Visual FoxPro Vfp6r.dll

Ya no hay un único entorno de tiempo de ejecución de Visual FoxPro 6.0, Vfp6r.dll, que dé servicio a múltiples servidores de Automatización .dll en proceso. Cada .dll en proceso utiliza ahora una instancia independiente del entorno de tiempo de ejecución Vfp6r.dll. Las reglas siguientes determinan la forma en que las .dll en proceso utilizan el entorno de tiempo de ejecución Vfp6r.dll:

- La .dll en proceso que se llamó en primer lugar tiene el uso exclusivo de la biblioteca del entorno de tiempo de ejecución Vfp6r.dll (normalmente instalada en la carpeta System de Windows 95 o en la carpeta System32 de Windows NT).
- Si una .dll en proceso ya tiene el uso exclusivo de Vfp6r.dll, por cada nueva .dll en proceso que se llame se creará en el disco y se cargará en memoria una copia de Vfp6r.dll con un nombre diferente. El nombre asignado al entorno de tiempo de ejecución Vfp6r.dll se basa en el nombre de la biblioteca .dll en proceso. Por ejemplo, si se llama a una .dll en proceso denominada Miservidor.dll, se asignará a la copia de Vfp6r.dll el nombre Miservidorr.dll (observe la “r” anexada al nombre), y se cargará en memoria para dar servicio a la .dll en proceso.
- Los entornos de tiempo de ejecución de Visual FoxPro sólo cambian de nombre para las .dll en proceso que se ejecutan en el mismo proceso. Esto significa que dos clientes independientes, cada uno ejecutándose en su propio proceso, pueden cargar dos .dll en proceso diferentes de Visual FoxPro sin cambiar el nombre del entorno de tiempo de ejecución. En este caso, ambas .dll en proceso de Visual FoxPro utilizarán Vfp6r.dll ya que los clientes se cargan en procesos separados.
- Cuando hay múltiples servidores de Automatización (creados con OLEPUBLIC en DEFINE CLASS) en una misma .dll en proceso, todos compartirán el mismo entorno de tiempo de ejecución Vfp6r.dll. En este caso, es posible que los servidores de Automatización se afecten mutuamente al compartir variables de memoria públicas, al establecer los mismos comandos SET, etc. Evite que los servidores de Automatización de una misma .dll en proceso interfieran entre sí.

## Bibliotecas de tipos



Visual FoxPro 6.0 admite ahora propiedades, eventos y métodos intrínsecos (de Visual FoxPro) en bibliotecas de tipos de servidores de Automatización. Solamente las propiedades declaradas como Public se incluyen en la biblioteca de tipos. Las propiedades protegidas y ocultas no aparecen en la biblioteca. Tenga en cuenta que el método Release de Visual FoxPro no está incluido en la biblioteca de tipos, pues ya existe como método COM.

Tanto los métodos como las propiedades PUBLIC personalizadas definidas por el usuario aparecen en las bibliotecas de tipos de Visual FoxPro, siempre que estén marcadas como Public. Para los métodos, Visual FoxPro incluye también un tipo de valor de retorno (de tipo variant) y una lista de parámetros (de tipo variant) que se analizan a partir de la definición del método original.

Tenga en cuenta que en Visual FoxPro 6.0 sólo puede designarse un archivo de Ayuda como biblioteca de tipos.

## Tratamiento de excepciones

Los servidores de Automatización de Visual FoxPro son ahora más robustos, por lo que pueden terminar una operación con menos problemas cuando se produce una excepción. Cuando ocurre una excepción en un servidor de Automatización de Visual FoxPro 6.0, el servidor establece el objeto COM ErrorInfo (a través de IErrorInfo) y cancela el método actual. El cliente de Automatización tiene la opción de liberar el servidor de Automatización de Visual FoxPro o tratar la excepción, dependiendo de la información del objeto COM ErrorInfo (y el cliente tiene acceso al objeto COM ErrorInfo).

Una nueva función agregada a Visual FoxPro 6.0, COMRETURNERROR( ), permite tratar los errores que se producen en un servidor de Automatización. COMRETURNERROR( ) se puede usar en el método Error; rellena la estructura de la excepción COM con información que los clientes de Automatización pueden utilizar para determinar el origen de los errores de servidor de Automatización. Para obtener más información, consulte [COMRETURNERROR\(\)](#) en la *Referencia del lenguaje*.

## Pasar matrices

Visual FoxPro 5.0 pasa por valor las matrices a los objetos COM (por ejemplo, los servidores de Automatización creados con Visual FoxPro, Visual Basic o Visual C); los elementos de las matrices son los mismos después de una llamada a un método y los cambios en el objeto COM no se propagan a los elementos del cliente. Esta restricción evita el paso de grandes cantidades de datos entre Visual FoxPro 5.0 y los objetos COM.

Además, se asume que la matriz pasada al objeto COM hace referencia al primer elemento, fila y columna con el número uno (por ejemplo, Mimatriz[1]). Sin embargo, algunos objetos COM requieren que la matriz pasada haga referencia al primer elemento, fila y columna con el número cero, como Mimatriz [0].

Una nueva función de Visual FoxPro 6.0, COMARRAY( ), permite especificar la forma de pasar una matriz a un objeto COM y si está basada en cero o en uno. Para obtener más información, consulte [COMARRAY\(\)](#) en la *Referencia el lenguaje*.

Tenga en cuenta que COMARRAY( ) sólo se utiliza cuando se pasan matrices a objetos COM con la sintaxis siguiente:

```
oObjetoCom.Metodo(@MiMatriz)
```

Si se omite el símbolo @, sólo se pasará al objeto COM el primer elemento de la matriz y COMARRAY( ) no tendrá ningún efecto. Éste es el comportamiento en las versiones anteriores de Visual FoxPro.

## Generar archivos .dll y .exe a partir de proyectos

Debido a que los servidores de Automatización .dll en proceso y .exe fuera de proceso se invocan a través de la creación de instancias de clases, no es necesario especificar un [archivo principal](#) para los mismos. En Visual FoxPro 6.0 puede generar un servidor de Automatización .dll en proceso o .exe fuera de proceso sin tener que especificar primero un archivo principal en el Administrador de proyectos.

## Lenguaje

En la tabla siguiente se muestran las propiedades y funciones que se han agregado a Visual FoxPro 6.0 para facilitar la administración de clientes y servidores de Automatización. Para obtener más información, consulte los temas indicados.

<b>Nuevos elementos del lenguaje para mejoras en el servidor</b>	<b>Descripción</b>
<a href="#">Función COMARRAY( )</a>	Especifica cómo se pasan las matrices a los objetos COM.
<a href="#">Función COMCLASSINFO( )</a>	Devuelve información del registro sobre un objeto COM, como un servidor de Automatización de Visual FoxPro.
<a href="#">Función CREATEOBJECTEX( )</a>	Crea una instancia de un objeto COM registrado (por ejemplo, de un servidor de Automatización de Visual FoxPro) en un equipo remoto. Puede utilizar Microsoft Transaction Server para crear una instancia de una .dll en proceso de Visual FoxPro en un equipo remoto.
<a href="#">Función COMRETURNERROR( )</a>	Rellena la estructura de excepción COM con información que los clientes de Automatización pueden utilizar para determinar el origen de los errores de un servidor de Automatización.
<a href="#">Propiedad ServerName</a>	Contiene la ruta completa y el nombre de archivo de un servidor de Automatización. La propiedad ServerName corresponde al objeto Application.
<a href="#">Propiedad StartMode</a>	Contiene un valor numérico que indica cómo se inició la instancia de Visual FoxPro.
<a href="#">SYS(2334) – Modo de invocación de</a>	Devuelve un valor que indica cómo se ha invocado un

<a href="#">servidor de Automatización</a>	método de un servidor Automation de Visual FoxPro.
<a href="#">SYS(2335) – Modo de servidor sin supervisión</a>	Activa o desactiva el uso de estados modales en los servidores de Automatización .exe distribuibles de Visual FoxPro.

## Notas sobre la programación de servidores de Automatización

En esta sección se ofrece información adicional sobre la programación de servidores de Automatización.

### El objeto Application

El [objeto Application](#) no se expone en una biblioteca de tipos de un servidor de Automatización. De este modo se impide el acceso a los métodos [DoCmd](#) y [Eval](#) del objeto Application, que potencialmente podrían dar acceso al lenguaje completo de Visual FoxPro. Puede exponer el objeto Application si crea una propiedad personalizada y le asigna el valor de objeto Application. También puede proporcionar un método con acceso al objeto Application.

### Ejemplos de servidores de Automatización

Visual FoxPro 6.0 incluye dos servidores de Automatización ISAPI de ejemplo, FoxWeb y FoxIS. Estos ejemplos administran el envío de registros seleccionados de datos de Visual FoxPro con formato HTML a un explorador de Internet. Para obtener más información sobre estos ejemplos, vea [FoxISAPI: ejemplo de servidor OLE](#).

## Capítulo 32: Programación de aplicaciones y productividad del programador

Microsoft FoxPro siempre ha proporcionado las herramientas necesarias para la programación de aplicaciones con la aplicación FoxPro y el lenguaje XBase. Visual FoxPro ha agregado el lenguaje y los comportamientos para la orientación a objetos. Esta versión de Visual FoxPro incluye un marco de trabajo mejorado, así como herramientas de creación y mantenimiento de objetos diseñadas para ayudar a una programación de aplicaciones más rápida y facilitar las tareas de mantenimiento.

En este capítulo se tratan los temas siguientes:

[Galería de componentes](#)

[Aplicación Analizador de trayecto](#)

[Enganches del Administrador de proyectos](#)

[Asistentes nuevos y mejorados](#)

[Marco de trabajo de aplicación mejorado](#)

## Galería de componentes

La Galería de componentes es un contenedor de catálogos de objetos de software tales como bibliotecas de clases, formularios, botones, etcétera. También contiene nuevas clases de Visual FoxPro. Puede utilizar la Galería de componentes para organizar los componentes por objetos, proyectos, aplicaciones u otros agrupamientos. Estos agrupamientos visuales pueden personalizarse dinámicamente de forma que pueda utilizar, duplicar o reorganizar los componentes de varias clasificaciones en la Galería de componentes. Puede tener acceso a un elemento específico desde cualquier punto de la Galería de componentes en el que coloque una referencia al mismo. También puede tener varias referencias a un mismo objeto en distintos catálogos o carpetas. Por ejemplo, un botón puede aparecer en una o más categorías de proyecto de la Galería de componentes (representadas como carpetas), pero también puede ser visible en una categoría llamada "Herramientas" que contenga referencias a todos los botones que utilice.

Puede utilizar la Galería de componentes con todas las funciones que proporcionan los componentes independientes [Administrador de proyectos](#), [Examinador de clases](#) y la [barra de herramientas](#) [Controles de formularios](#). Cada uno de los restantes componentes de Visual FoxPro proporciona un enfoque muy específico de los proyectos o clases desde el entorno especial del archivo de proyecto o de la biblioteca de clases. La Galería de componentes permite administrar las relaciones entre los componentes y muchos de sus comportamientos, desde un nivel abstracto de diseño y también desde una perspectiva de programación más concreta.

Puede arrastrar y colocar componentes dentro de la Galería de componentes y también desde la Galería de componentes a proyectos o formularios. También puede cambiar las propiedades de los objetos o clases desde la Galería de componentes.

La Galería de componentes puede contener cualquier elemento visual de Visual FoxPro, como documentos locales y remotos, archivos o carpetas, servidores de Automatización del tipo Microsoft Excel y Word, y ubicaciones y archivos HTML. También puede incluir archivos .prg con código de miniprogramas, clases, asistentes, generadores o elementos gráficos.

### Para abrir la Galería de componentes

- En el menú **Herramientas**, haga clic en la **Galería de componentes**.
- O bien -
- Escriba **DO** ([\\_GALLERY](#)) en la ventana **Comandos**.

### Administrar proyectos con la Galería de componentes

Puede utilizar la Galería de componentes para crear proyectos y aplicaciones, y también para administrar su desarrollo. La Galería de componentes permite organizar los componentes que contiene y puede utilizar sus plantillas, generadores y asistentes para crear el proyecto o la aplicación deseados.

### Para crear un proyecto o una aplicación desde la Galería de componentes

- Utilice el **Asistente para aplicaciones** o la **plantilla Nueva aplicación** de la carpeta **Aplicaciones** del catálogo de **Visual FoxPro**.

En el caso de los catálogos y carpetas, seleccione las fichas y opciones para realizar el cambio que desee. Si desea más información, vea el [cuadro de diálogo Opciones de la Galería de componentes](#).

### Mover y ver elementos en la Galería de componentes

Puede mover los elementos del panel derecho (Objeto) de la ventana de la Galería de componentes hasta el escritorio o hasta un proyecto o formulario abierto. El Administrador de proyectos reconoce el elemento al que se refiere la Galería de componentes y lo coloca en el lugar apropiado. Los elementos de la Galería de componentes colocados en el escritorio no son funcionales. No hay ninguna representación en el escritorio para las bases de datos, carpetas y elementos de la Galería de componentes que representan archivos no visuales.

#### Para mover elementos desde la Galería de componentes

1. En el panel derecho, haga clic en el elemento que desee mover.

El [icono Mover](#), situado en la esquina superior izquierda de la ventana **Galería de componentes**, cambia de según el elemento seleccionado.

2. Arrastre y coloque el icono **Mover** en el escritorio o en un proyecto o formulario abierto.

Cuando la Galería de componentes no encuentra el elemento original representado por el elemento de galería, se abre un cuadro de diálogo Buscar en el que puede ayudarle a encontrarlo.

En la tabla siguiente se identifican los elementos de galería incluidos en Visual FoxPro, así como sus comportamientos predeterminados.

#### Tipo de elemento de la Galería de componentes   Destinos de arrastrar y colocar

	Proyecto	Formulario	Pantalla	Controles
Clase (_ClassItem)			6	
Archivo (_FileItem)				
Dirección URL (_UrlItem)	1			
Formulario (_FormItem)	9		11	
Informe (_ReportItem)	9		11	
Programa (_ProgramItem)			11	
Menú (_MenuItem)	10		11	
Imagen (_ImageItem)	2		7	2
Sonido (_SoundItem)	3			

Vídeo (_VideoItem)	3
ActiveX (_ActiveXItem)	
Datos (_DataItem)	4
Plantilla (_TemplateItem)	5
Catálogo (_CatalogItem)	8
Ejemplo (_SampleItem)	
Proyecto (_ProjectItem)	11

- 1 – Agrega una clase de hipervínculo
- 2 – Agrega una clase de imagen o establece una propiedad Picture
- 3 – Agrega una clase multimedia
- 4 – Agrega una clase cuadrícula
- 5 – Según el tipo (por ejemplo, en un formulario) crea un nuevo archivo y lo agrega al proyecto
- 6 – Crea una instancia en Pantalla
- 7 – Establece el papel tapiz de Visual FoxPro
- 8 – Inicia una nueva ventana de la Galería con ese catálogo
- 9 – Agrega una clase de botones para iniciar un formulario o informe
- 10 – Agrega un menú contextual a un formulario
- 11 – Se abre en un diseñador (para modificación)

### Usar menús contextuales en la Galería de componentes

Puede hacer clic con el botón secundario en un elemento seleccionado del panel derecho (Objeto) para que aparezca un [menú contextual de elemento](#) con todas las acciones correspondientes, como **Agregar al proyecto** o **Agregar al formulario**. Puede utilizar el menú contextual para modificar o, en algunos casos, ejecutar el elemento de galería. Los menús contextuales son característicos de cada tipo de elemento de la galería. Puede cambiar algunas de las propiedades del elemento seleccionado con la opción **Propiedades** del menú contextual, que abrirá el [cuadro de diálogo Propiedades de elemento](#).

### Organizar los componentes de Visual FoxPro o Windows en grupos definidos por el usuario

Las carpetas de la Galería de componentes representan un agrupamiento arbitrario de los elementos de galería. Puede reorganizarlas con la técnica de arrastrar y colocar, o bien puede duplicarlos en otras carpetas. También puede copiar y cambiar el nombre de un catálogo o carpeta, o reordenar los elementos que contiene. Hay pocos límites, si es que hay alguno, en el modo de utilizar, modificar o crear catálogos o carpetas.

### Revisar y modificar clases

Al representar los elementos de la Galería de componentes elementos reales que pueden ser objetos o clases, para revisarlos o modificarlos puede lograr el acceso al objeto original a través de la Galería de componentes.

### Para revisar una clase

1. En la Galería de componentes, haga clic con el botón secundario en una clase.
2. En el menú contextual, haga clic en **Ver en el examinador**.

Se abrirá el [Examinador de clases](#), en el que puede ver las propiedades y métodos de la clase seleccionada.

### Para modificar una clase

1. En la Galería de componentes, haga clic con el botón secundario en una clase.
2. En el menú contextual, haga clic en **Modificar**.

Se abrirá la clase en el **Diseñador de clases**.

### Crear y modificar formularios

Puede utilizar la Galería de componentes para duplicar o modificar formularios y para agregar formularios y otros elementos de galería a un proyecto.

#### Para crear un formulario desde la Galería de componentes

- Haga doble clic en cualquier plantilla, o bien seleccione **Nuevo formulario** en el menú contextual de cualquier plantilla de la carpeta Formularios de la Galería de componentes.  
- O bien -
- Haga doble clic en el **Asistente para formularios** en la carpeta Formularios de la Galería de componentes.  
- O bien -
- Seleccione **Crear formulario** en el menú contextual de los elementos de la carpeta Formularios de la Galería de componentes.

### Características de edición avanzadas de la Galería de componentes

La configuración predeterminada de los catálogos y formularios permite realizar tareas básicas de revisión y administración de elementos de la galería. Si desea modificar las características de los catálogos o carpetas, o si desea tener un mayor acceso a las propiedades de galería, seleccione **Modificación avanzada habilitada** en el [cuadro de diálogo Opciones de la Galería de componentes](#).

### Catálogos de la Galería de componentes

Al abrir la Galería de componentes, el panel izquierdo (Catálogo) muestra el catálogo predeterminado



incluido en la Galería de componentes. Un catálogo es una representación visual de los elementos que pertenecen a un grupo de Visual FoxPro o a un grupo definido por el usuario. Dentro de cada catálogo, puede crear carpetas para organizar con más detalle subgrupos de elementos. Los elementos pueden ser formularios, consultas, programas, plantillas, archivos gráficos, archivos de sonido u otros objetos. El catálogo predeterminado de Visual FoxPro en la Galería de componentes incluye elementos agrupados en varias categorías, como formularios y controles, entre otros. También incluye una carpeta vacía llamada Favoritos, que puede utilizar para crear o copiar en ella elementos de la galería. Puede copiar y cambiar el nombre del catálogo predeterminado, o crear los suyos propios.

Las opciones de catálogo de la Galería de componentes determinan el contenido del catálogo y su comportamiento al abrirlo. Los catálogos *globales* pueden contener cualquier tipo de elemento de la Galería de componentes. Los *predeterminados* se abren automáticamente al iniciar la Galería de componentes. Si desea más información, vea la [ficha Catálogos](#) del cuadro de diálogo Opciones de la Galería de componentes .

La Galería de componentes incluye los catálogos siguientes.

Catálogo	Descripción
VFPGLRY	Contiene componentes que utilizan otros catálogos de la galería. Contiene todos los catálogos incluidos con Visual FoxPro. Catálogo predeterminado y global.
Visual FoxPro	Contiene las clases Visual FoxPro Foundation Classes. Catálogo predeterminado.
Favoritos	Carpeta vacía. Catálogo global.
Mis clases de base	Contiene clases derivadas de las clases de base de Visual FoxPro. Catálogo predeterminado.
ActiveX	Catálogo dinámico que contiene una lista de todos los controles ActiveX registrados, o bien una lista de todos los controles ActiveX de Visual FoxPro. Catálogo predeterminado.
World Wide Web	Colección de direcciones URL de sitios Web.
Multimedia	Diversos elementos con imágenes, sonidos y vídeos que puede utilizar en sus aplicaciones.
Ejemplos	Referencias a los ejemplos Solutions, Tastrade, servidores ActiveX y cliente-servidor.

Al hacer clic en un catálogo en la vista de lista, el panel derecho (Objeto) muestra su contenido. Puede abrir otros catálogos si hace doble clic en ellos en uno de los dos paneles. En la carpeta Galería se incluyen varios catálogos.

## Personalizar la Galería de componentes



Si desea personalizar la Galería de componentes, puede cambiar el comportamiento predeterminado de los elementos de los catálogos, carpetas y galerías a través de los cuadros de diálogo Propiedades correspondientes.

### Para crear un catálogo de la Galería de componentes

1. Seleccione el botón **Opciones** de la [barra de herramientas de la Galería de componentes](#).
2. Haga clic en la [ficha Catálogos](#) del cuadro de diálogo **Opciones de la Galería de componentes**.
3. Haga clic en **Nuevo** y dé nombre al nuevo catálogo en el cuadro de diálogo **Abrir**.
4. Haga clic en **Aceptar**.
5. La **Galería de componentes** agregará el catálogo al árbol para que pueda empezar a utilizarlo como lo haría con cualquier otro catálogo existente.

### Para cambiar la configuración de un catálogo o carpeta

1. Haga clic con el botón secundario en el catálogo o carpeta.
2. En el menú contextual, haga clic en **Propiedades**.
3. En el [cuadro de diálogo Propiedades de catálogo](#) o en el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha que contenga las opciones que desea configurar.

Los catálogos y las carpetas mostrados en el panel **Catálogo** de la [ventana Galería de componentes](#) pueden representar direcciones URL, carpetas o archivos del disco duro. Puede ver una carpeta como **vista de Web** o como **vista a nivel de explorador**, dependiendo de la forma en que especifique su nombre en la ficha General del cuadro de diálogo Propiedades de carpeta.

### Vistas de Web

Puede especificar direcciones URL o archivos como catálogos o elementos de galería. Al configurar un elemento como carpeta de galería, se abrirá automáticamente como vista de Web en el panel **Objeto** (derecho) al seleccionarlo en el panel **Catálogo**.

### Para configurar un catálogo o carpeta de galería como vista de Web

1. En el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha **Nodo**.
2. En el campo **Carpeta dinámica**, especifique el nombre de la página Web o del archivo como se hace en los ejemplos siguientes:

`http:\\www.microsoft.com\\`

archivo:\\c:\mis documentos\paginatest.htm

archivo:\\c:\mis documenotss\archword.doc

Al resaltar el icono de vista Web en el **panel Catálogo**, la barra de herramientas se modifica para incluir los botones de navegación por el Web. La vista de Web reflejará la configuración del Explorador de Windows.

### Vistas a nivel de explorador

Puede especificar un directorio como carpeta o catálogo de galería que tiene las características del Explorador de Windows.

#### Para configurar un catálogo o carpeta de galería como vista a nivel de explorador

1. En el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha **Nodo**.
2. En el campo **Carpeta dinámica**, especifique el nombre de una carpeta o de un archivo y una barra inversa (\), como en el ejemplo siguiente:

C:\Mis documentos\

**Nota** Esta especificación crea una vista de archivos reales, a diferencia de otras vistas de la Galería de componentes. En esta vista *sí* puede eliminar archivos del disco.

Para crear una vista a nivel de explorador que mantenga la protección de los archivos mostrados, especifique el destino con una designación de comodín, como en el ejemplo siguiente:

C:\Mis documentos\\*.\*

Evite el uso de comodines para crear carpetas dinámicas cuando prevea encontrar más de 512 elementos, a no ser que disponga de un equipo rápido con gran cantidad de RAM.

### Objetos miembros de la Galería de componentes

La Galería de componentes consta de una interfaz, cuyas clases están contenidas en Vfpglry.vcx y de elementos que hacen referencia a las clases Visual FoxPro Foundation Classes siguientes:

Objeto	Descripción	Biblioteca de clases
<a href="#">About Dialog</a>	Proporciona un sencillo cuadro de diálogo Acerca de... para aplicaciones personalizadas.	_dialogs.vcx
<a href="#">ActiveX Calendar</a>	Control calendario que puede asociarse a un campo de fecha.	_datetime.vcx
<a href="#">Array Handler</a>	Proporciona métodos para tratar	_utility.vcx

	operaciones con matrices que no realizan las funciones de matrices nativas.	
<a href="#">Cancel Button</a>	Libera un formulario y descarta los datos que queden almacenados en búfer.	_miscbtns.vcx
<a href="#">Clock</a>	Control de reloj simple para un formulario o contenedor.	_datetime.vcx
<a href="#">Conflict Catcher</a>	Cuadro de diálogo para resolver los conflictos de filas que aparezcan al modificar con almacenamiento optimista en búfer.	_dataquery.vcx
<a href="#">Cookies Class</a>	Clase simple de Web para el tratamiento de cookies entre páginas Web.	_internet.vcx
<a href="#">Cross Tab</a>	Genera una tabla de referencias cruzadas.	_utility.vcx
<a href="#">Data Edit Buttons</a>	Conjunto completo de botones de modificación (como los que utilizan los Asistentes para formularios).	Wizbtns.vcx
<a href="#">Data Navigation Buttons</a>	Grupo de botones de exploración Top, Next, Prev, Bottom y clase DataChecker para comprobar si hay conflictos al mover registros.	_datanav.vcx
<a href="#">Data Navigation Object</a>	Objeto de exploración no visual que otras clases pueden utilizar.	_table.vcx
<a href="#">Data Session Manager</a>	Administra sesiones de datos y se ocupa de las actualizaciones.	_app.vcx
<a href="#">Data Validation</a>	Intercepta conflictos de datos almacenados en búfer.	_datanav.vcx
<a href="#">DBF -&gt; HTML</a>	Convierte un cursor de Visual FoxPro (.dbf) a HTML.	_internet.vcx
<a href="#">Distinct Values Combo</a>	Realiza una búsqueda de valores únicos en el campo origen del control para rellenar un cuadro combinado.	_dataquery.vcx
<a href="#">Error Object</a>	Tratamiento genérico de errores que funciona con código de objeto y también con código procedimental.	_app.vcx
<a href="#">Field Mover</a>	Cuadro de lista supermover que carga automáticamente campos del origen de datos actual.	_movers.vcx
<a href="#">File Registry</a>	Proporciona un conjunto de funciones de	Registry.vcx

	registro que devuelven información específica de la aplicación.	
<a href="#">File Version</a>	Recupera información de los datos de versión de un archivo.	_utility.vcx
<a href="#">Filter Button</a>	Muestra un cuadro de diálogo en el que especificar un filtro de datos para un campo determinado.	_table2.vcx
<a href="#">Filter Dialog</a>	Cuadro de diálogo que permite especificar condiciones de filtrado de los datos.	_table.vcx
<a href="#">Filter Expression Dialog</a>	Crea un cuadro de diálogo para expresiones de filtro avanzadas.	_table.vcx
<a href="#">Find (Findnext) Buttons</a>	Conjunto de botones Buscar/Buscar siguiente genérico.	_table.vcx
<a href="#">Find Button</a>	Busca un registro que satisfaga criterios específicos.	_table.vcx
<a href="#">Find Dialog</a>	Cuadro de diálogo Buscar con opciones simples, tales como elección de campos.	_table.vcx
<a href="#">Find Files/Text</a>	Utiliza el objeto COM Filer.DLL para buscar archivos.	_utility.vcx
<a href="#">Find Object</a>	Crea un objeto genérico que busca un registro con criterios específicos.	_table.vcx
<a href="#">Font Combobox</a>	Cuadro combinado que contiene las fuentes disponibles. También lo utilizan las clases tbrEditing y rtfControls.	_format.vcx
<a href="#">Fontsize Combobox</a>	Cuadro combinado que contiene los tamaños de fuente disponibles. También lo utilizan las clases tbrEditing y rtfControls.	_format.vcx
<a href="#">Format Toolbar</a>	Proporciona una barra de herramientas para aplicar formato de fuente al texto del control activo.	_format.vcx
<a href="#">FRX -&gt; HTML</a>	Convierte el resultado de un informe de Visual FoxPro (.frx) al formato HTML.	_internet.vcx
<a href="#">GetFile and Directory</a>	Recupera un nombre de archivo y de carpeta.	_controls.vcx
<a href="#">Goto Dialog Button</a>	Crea un botón que muestra el cuadro de diálogo Ir a.	_table2.vcx
<a href="#">Goto Dialog</a>	Crea un cuadro de diálogo Ir a registro.	_table.vcx

<a href="#">Graph By Record Object</a>	Grupo de botones de exploración que permite actualizar un nuevo gráfico por cada registro instantáneamente.	_utility.vcx
<a href="#">Graph Object</a>	Genera un gráfico con el motor del Asistente para gráficos.	Autgraph.vcx
<a href="#">Help Button</a>	Muestra el archivo de Ayuda mientras comienza a buscar el HelpContextID especificado.	_miscbtns.vcx
<a href="#">Hyperlink Button</a>	Inicia un explorador de Web desde un botón.	_hyperlink.vcx
<a href="#">Hyperlink Image</a>	Inicia un explorador de Web desde una imagen.	_hyperlink.vcx
<a href="#">Hyperlink Label</a>	Inicia un explorador de Web desde una etiqueta.	_hyperlink.vcx
<a href="#">INI Access</a>	Conjunto de funciones de registro que permiten el acceso a las configuraciones de archivo del antiguo tipo INI.	Registry.vcx
<a href="#">Item Locator</a>	Este botón abre un cuadro de diálogo con el que puede buscar un registro.	_dialogs.vcx
<a href="#">Keywords Dialog</a>	Crea un cuadro de diálogo similar al cuadro de palabras clave de la Galería de componentes.	_dialogs.vcx
<a href="#">Launch Button</a>	Inicia una aplicación con un documento opcional.	_miscbtns.vcx
<a href="#">Locate Button</a>	Muestra un cuadro de diálogo con el que buscar un registro.	_table2.vcx
<a href="#">Lookup Combobox</a>	Realiza una búsqueda de valores en un campo para rellenar un cuadro combinado.	_dataquery.vcx
<a href="#">Mail Merge Object</a>	Genera una combinación de Word Mail con el motor del Asistente para combinar correspondencia.	Mailmerge.vcx
<a href="#">MessageBox Handler</a>	Envoltura simple de la función MessageBox.	_dialogs.vcx
<a href="#">MouseOver Effects</a>	Resalta un control cuando pasa el mouse sobre él.	_ui.vcx
<a href="#">Mover</a>	Proporciona una clase sencilla de cuadro de lista con movimiento y botones mover/quitar.	_movers.vcx

<a href="#">Navigation Shortcut Menu</a>	Menú contextual que puede colocarse en un formulario.	_table2.vcx
<a href="#">Navigation Toolbar</a>	Conjunto de botones de navegación en una barra de herramientas.	_table2.vcx
<a href="#">Object State</a>	Determina el estado de un objeto y guarda o restablece la configuración de sus propiedades.	_app.vcx
<a href="#">ODBC Registry</a>	Conjunto de funciones de registro que devuelven información específica de ODBC.	Registry.vcx
<a href="#">Offline Switch</a>	Proporciona una vista de datos con conexión para su uso sin conexión.	_dataquery.vcx
<a href="#">OK Button</a>	Realiza una liberación simple de formulario.	_miscbtns.vcx
<a href="#">Output Control</a>	Muestra un cuadro de diálogo complejo que solicita al usuario una opción de resultado de informe.	_reports.vcx
<a href="#">Output Dialog</a>	Muestra un cuadro de diálogo que solicita al usuario una opción de resultado de informe.	_reports.vcx
<a href="#">Output Object</a>	Diversas opciones de resultado de informe.	_reports.vcx
<a href="#">Password Dialog</a>	Sencillo cuadro de diálogo Contraseña para aplicaciones personalizadas.	_dialogs.vcx
<a href="#">Pivot Table</a>	Genera una tabla dinámica de Microsoft Excel con el motor del Asistente para tablas dinámicas.	Pivtable.vcx
<a href="#">Preview Report</a>	Botón genérico para ejecutar un informe.	_miscbtns.vcx
<a href="#">QBF</a>	Proporciona un conjunto de botones para consultas de tipo Consulta por formulario.	_dataquery.vcx
<a href="#">Registry Access</a>	Proporciona acceso a la información del Registro de Windows.	registry.vcx
<a href="#">Resize Object</a>	Hace que los objetos de un formulario cambien de tamaño y posición cuando se produce el evento Resize del objeto Form.	_controls.vcx
<a href="#">RTF Controls</a>	Proporciona un conjunto de botones para aplicar formato al texto del control activo.	_format.vcx
<a href="#">Run Form Button</a>	Botón que ejecuta un formulario.	_miscbtns.vcx

<a href="#">SCX -&gt; HTML</a>	Convierte un formulario .scx al formato HTML.	_internet.vcx
<a href="#">SendMail Buttons</a>	Utiliza el control ActiveX de MAPI ActiveX para enviar un mensaje de correo desde un formulario.	_miscbtns.vcx
<a href="#">Shell Execute</a>	Proporciona el comportamiento de doble clic del Explorador de Windows.	_environ.vcx
<a href="#">Shortcut Menu Class</a>	Esta clase de envoltura crea dinámicamente menús contextuales emergentes.	_menu.vcx
<a href="#">Simple Edit Buttons</a>	Proporciona sencillos botones Agregar, Modificar, Eliminar, Duplicar, Guardar y Cancelar (como los de los Asistentes para formularios).	Wizbtns.vcx
<a href="#">Simple Navigation Buttons</a>	Proporciona un conjunto de botones de exploración Siguiente y Anterior.	_table.vcx
<a href="#">Simple Picture Navigation Buttons</a>	Conjunto de botones de exploración con imágenes sencillas.	_table2.vcx
<a href="#">Sort Button</a>	Muestra un cuadro de diálogo que permite ordenar los datos de un campo determinado de forma ascendente o descendente.	_table2.vcx
<a href="#">Sort Dialog</a>	Permite realizar una ordenación ascendente o descendente de los datos de un campo determinado.	_table2.vcx
<a href="#">Sort Mover</a>	Esta subclase de la clase cuadro de lista supermover se ocupa de la ordenación de los datos.	_movers.vcx
<a href="#">Sort Object</a>	Realiza una ordenación de un origen de datos.	_table.vcx
<a href="#">Sort Selector</a>	Realiza una ordenación ascendente o descendente, basada en el control actual.	_table2.vcx
<a href="#">Sound Player</a>	Esta clase carga y reproduce un archivo de sonido.	_multimedia.vcx
<a href="#">Splash Screen</a>	Proporciona una sencilla pantalla de inicio para aplicaciones personalizadas.	_dialogs.vcx
<a href="#">SQL Pass Through</a>	Proporciona paso a través de SQL y permite ejecutar procedimientos almacenados en la base de datos host.	_dataquery.vcx

<a href="#">Stop Watch</a>	Proporciona un control de detención de inspección para un formulario o contenedor.	_datetime.vcx
<a href="#">String Library</a>	Realiza diversas conversiones de cadenas.	_utility.vcx
<a href="#">Super Mover</a>	Proporciona los botones Mover, Quitar, Mover todos y Quitar todos.	_movers.vcx
<a href="#">System Toolbars</a>	Clase administrativa que maneja y hace un seguimiento de las barras de herramientas del sistema.	_app.vcx
<a href="#">Table Mover</a>	Esta subclase de la clase cuadro de lista supermover carga automáticamente tablas y campos desde el origen de datos actual.	_movers.vcx
<a href="#">Text Preview</a>	Proporciona un visor del texto resultante.	_reports.vcx
<a href="#">Thermometer</a>	Proporciona una clase termómetro estándar.	_controls.vcx
<a href="#">Trace Aware Timer</a>	Utilidad de aplicación que determina si la ventana de seguimiento está abierta.	_app.vcx
<a href="#">Type Library</a>	La rutina principal ExportTypeLib crea un archivo de texto con el resultado Typelib.	_utility.vcx
<a href="#">URL Combo</a>	Crea un cuadro combinado para escribir en una dirección URL de Web. Inicia Microsoft Internet Explorer y se sitúa en el sitio correspondiente.	_internet.vcx
<a href="#">URL Open Dialog</a>	Proporciona un cuadro de diálogo que crea una lista desplegable con el historial de direcciones URL.	_internet.vcx
<a href="#">VCR Buttons</a>	Grupo de botones de exploración Top, Next, Prev y Bottom.	_table.vcx
<a href="#">VCR Picture Navigation Buttons</a>	Conjunto de botones de exploración de imagen VCR.	_table2.vcx
<a href="#">Video Player</a>	Carga y reproduce un archivo de vídeo mediante comandos MCI.	_multimedia.vcx
<a href="#">Web Browser control</a>	Subclase de control Browser de Internet Explorer 4.0, que proporciona enganches con el código de Visual FoxPro.	_webview.vcx
<a href="#">Window Handler</a>	Realiza diversas operaciones comunes de ventanas habituales en las aplicaciones.	_ui.vcx

Si desea más detalles sobre estas bibliotecas de clases, vea el tema [Visual FoxPro Foundation Classes](#)



. Puede obtener información sobre cómo utilizar estas clases en el tema [Directivas para usar las Visual FoxPro Foundation Classes](#).

## Biblioteca de clases de la Galería de componentes (Vpfgallery.vcx)

La biblioteca de clases de la Galería de componentes, Vpfgallery.vcx, proporciona los tipos de elemento como clases.

Tipo de elemento	Descripción
<b>Clase</b> (_ClassItem)	Tipo de elemento genérico de cualquier clase de Visual FoxPro. Puede provenir de archivos .vcx o .prg.
<b>Archivo</b> (_FileItem)	Cualquier archivo. Visual FoxPro busca en el Registro funciones de shell y las agrega al menú. La galería incluye una rutina de búsqueda que comprueba si hay extensiones específicas y redirige el tipo de elemento.  La galería admite las convenciones UNC para la programación en grupo (compartir catálogos entre redes).
<b>ActiveX</b> (_ActiveXItem)	Control o servidor ActiveX, como por ejemplo, un .ocx creado por Visual Basic CCE, o un archivo .exe o .dll creado por Visual FoxPro.
<b>Datos</b> (_DataItem)	Origen de datos de Visual FoxPro (.dbc, .dbf, Vista, etc.).
<b>Imagen</b> (_ImageItem)	Tipo de elemento de archivo cuya extensión corresponde a una imagen, como por ejemplo .bmp, .jpg, .gif, .ico, .cur, .ani, etcétera.
<b>Sonido</b> (_SoundItem)	Tipo de elemento de archivo cuya extensión es .wav o .rmi.
<b>Vídeo</b> (_VideoItem)	Tipo de elemento de archivo cuya extensión es .avi.
<b>Dirección URL</b> (_UrlItem)	Tipo de elemento de Web que incluye documentos de Web y locales, como por ejemplo archivos HTML o documentos activos de Visual FoxPro.
<b>Ejemplo</b> (_SampleItem)	Tipo de elemento de archivo para archivos que se ejecutan en Visual FoxPro y que pueden ser archivos ejecutables de Visual FoxPro, como por ejemplo .app, .exe, .prg, .scx o .frx.
<b>Plantilla</b> (_TemplateItem)	Tipo de elemento de archivo de comandos que abre un generador para el elemento de Visual FoxPro representado por el tipo del elemento resaltado, como un formulario o un informe.
<b>Catálogo</b> (_CatalogItem)	Tipo de Galería de componentes que permite agregar y abrir catálogos de Visual FoxPro.
<b>Formulario</b> (_FormItem)	Tipo para formularios de Visual FoxPro (.scx).

<b>Informe</b> (_ReportItem)	Tipo para informes de Visual FoxPro (.frx).
<b>Menú</b> (_MenuItem)	Tipo para menús de Visual FoxPro (.mnx).
<b>Programa</b> (_ProgramItem)	Tipo para programas de Visual FoxPro (.prg).
<b>Proyecto</b> (_ProjectItem)	Tipo para proyectos de Visual FoxPro (.pjx).

Puede utilizar el Examinador de clases para examinar los detalles de cualquiera de las clases anteriores.

Si desea más información sobre otras clases utilizadas en la Galería de componentes, vea el tema [Visual FoxPro Foundation Classes](#) o bien utilice el [Examinador de clases](#) para examinar las bibliotecas de la carpeta Ffc.

## Tabla de estructura de la Galería de componentes

La estructura de la Galería de componentes se describe en la siguiente tabla.

<b>Campo</b>	<b>Nombre de campo</b>	<b>Tipo</b>	<b>Ancho</b>	<b>Índice</b>
1	TYPE	Character	12	No
2	ID	Character	12	No
3	PARENT	Memo	4	No
4	LINK	Memo	4	No
5	TEXT	Memo	4	No
6	TYPEDESC	Memo	4	No
7	DESC	Memo	4	No
8	PROPERTIES	Memo	4	No
9	FILENAME	Memo	4	No
10	CLASS	Memo	4	No
11	PICTURE	Memo	4	No
12	FOLDERPICT	Memo	4	No
13	SCRIPT	Memo	4	No
14	CLASSLIB	Memo	4	No
15	CLASSNAME	Memo	4	No
16	ITEMCLASS	Memo	4	No
17	ITEMTPDESC	Memo	4	No

18	VIEWS	Memo	4	No
19	KEYWORDS	Memo	4	No
20	SRCALIAS	Memo	4	No
21	SRCRECNO	Numeric	6	No
22	UPDATED	DateTime	8	No
23	COMMENT	Memo	4	No
24	USER	Memo	4	No

## Aplicación Analizador de trayecto

Una aplicación de trayecto escribe información sobre las líneas de código de un archivo que se han ejecutado. Una aplicación analizador de trayecto proporciona información sobre las líneas que se han ejecutado, cuántas veces se ha ejecutado cada una, su duración, etcétera. El [trayecto](#) y el [analizador de trayecto](#) permiten al programador identificar áreas problemáticas de una aplicación, especialmente código que se pasa por alto y cuellos de botella para el rendimiento.

El analizador de trayecto de Visual FoxPro se divide en dos partes: un motor de trayecto que puede utilizar o personalizar y una aplicación multiventana que puede utilizar para analizar los programas y proyectos.

La aplicación Analizador de trayecto ofrece distintas formas de ver los datos que proporciona el motor de trayecto. Coverage.app es una subclase del motor de trayecto. Puede automatizar el trayecto, modificar la interfaz de usuario para adaptarla a sus necesidades, ejecutar el Analizador de trayecto en modo no supervisado y no mostrar la ventana de la aplicación o utilizar las características del motor sin usar la interfaz.

Al iniciarse, la aplicación de trayecto suspende el registro de trayecto activado con un comando SET COVERAGE TO. Al liberar el objeto trayecto, la aplicación permite elegir si se desea restablecer el valor de SET COVERAGE.

### Archivo de registro del Analizador de trayecto

El Analizador de trayecto utiliza un archivo de registro que genera Visual FoxPro al utilizar la opción Trayecto del menú Herramientas del depurador o al utilizar [SET COVERAGE TO](#) como en el comando siguiente:

```
SET COVERAGE TO cTrayecto.log
```

Al utilizar el comando, la cláusula ADDITIVE permite no sobrescribir un registro ya existente. Este comando inicia el flujo de datos y abre *cTrayecto.log*, un archivo de texto que almacena el flujo de datos sobre el archivo o la aplicación examinados.

Un archivo de registro de trayecto consta de registros dispuestos en líneas delimitadas por comas. En la lista siguiente se describe la estructura de cada registro.

Elemento	Descripción
1	tiempo de ejecución
2	clase que ejecuta el código
3	objeto, método o procedimiento en el que se encuentra o se invoca el código
4	número de línea dentro del método o procedimiento
5	archivo definido completamente
6	nivel de pila de llamada (sólo Visual FoxPro 6.0)

Una vez especificado el nombre del archivo de registro, ejecute el programa o aplicación que desee examinar. Cuando termine el programa, puede utilizar el comando SET COVERAGE TO para detener el flujo de datos hacia el registro de trayecto.

Para visualizar el registro de trayecto puede iniciar el Analizador de trayecto en el menú Herramientas o utilizar [DO](#) como en el comando siguiente:

```
DO (_COVERAGE) [WITH cCoverage]
```

Si no especifica un archivo de registro, Visual FoxPro le pedirá el nombre de uno. La variable de sistema [\\_COVERAGE](#) tiene como valor predeterminado en Visual FoxPro 6.0 la aplicación Analizador de trayecto, Coverage.app.

## Examinar el trayecto y el perfil de una aplicación

Para utilizar eficazmente el Analizador de trayecto, debe preparar cuidadosamente la aplicación y el entorno. Si sigue las directrices indicadas a continuación, el Analizador de trayecto le proporcionará información útil y precisa sobre su proyecto o aplicación.

### Para utilizar el Analizador de trayecto con el fin de examinar el trayecto de una aplicación

1. Utilice la opción **Registro de trayecto** del menú **Herramientas** del depurador, o bien ejecute el [comando SET COVERAGE](#) para iniciar el flujo de datos de trayecto y abrir el archivo para registrar los datos.
2. Ejecute el programa o aplicación cuyo trayecto desea examinar.
3. Ejecute la aplicación de trayecto en el menú **Herramientas** o bien utilice DO (\_COVERAGE) en la ventana de comandos.

La aplicación Analizador de trayecto se iniciará de forma predeterminada en **modo Trayecto**.

### Para utilizar el Analizador de trayecto con el fin de examinar el perfil de una aplicación

1. Utilice el [comando SET COVERAGE](#) para iniciar el flujo de datos de trayecto y abrir el

archivo en el que va a registrar los datos.

2. Ejecute el programa o aplicación que desea analizar.
3. Ejecute la aplicación de trayecto desde el menú **Herramientas**, o bien utilice DO ([\\_COVERAGE](#)) en la ventana de comandos.
4. Haga clic en el botón **Modo perfil** del cuadro de diálogo **Analizador de trayecto**.

Si le interesa un análisis con más frecuencia, puede cambiar la opción predeterminada por el modo perfil en el [cuadro de diálogo Opciones del Analizador de trayector](#).

### Para utilizar el Analizador de trayecto con un archivo de trayecto específico

Ejecute la aplicación de trayecto con la opción WITH y el nombre del archivo de registro como en el ejemplo siguiente:

```
DO (_COVERAGE) WITH "Mireg.LOG"
```

En este ejemplo se utiliza el archivo de registro Mireg.log y se abre la ventana de la aplicación Analizador de trayecto para mostrar el resultado. Si no especifica ningún nombre de archivo, el Analizador de trayecto utilizará el especificado en el comando SET COVERAGE TO actual o mostrará el cuadro de diálogo Abrir archivo si el registro de trayecto está desactivado (OFF).

### Para utilizar el Analizador de trayecto sin la interfaz de usuario

Ejecute la aplicación de trayecto con la opción WITH y especifique "verdadero" (.T.) para utilizar el modo no supervisado, como en el ejemplo siguiente.

```
DO (_COVERAGE) WITH "Mireg.LOG",.T.
```

En este ejemplo, la aplicación Analizador de trayecto utiliza el archivo de registro Mireg.log, y se ejecuta sin mostrar su ventana de aplicación.

### Para utilizar el Analizador de trayecto con un archivo de complemento específico

Ejecute la aplicación de trayecto con la opción WITH y el nombre del archivo de complemento, como en el ejemplo siguiente:

```
DO (_COVERAGE) WITH "Mireg.LOG",, "add_ui.prg"
```

En este ejemplo se utiliza el archivo de registro Mireg.log y se abre la ventana de la aplicación Analizador de trayecto para mostrar el resultado; y a continuación se ejecuta el programa complemento ADD\_UI.PRG. El segundo parámetro, no especificado, es un valor lógico que indica si el motor de trayecto funciona en modo no supervisado. Con la opción predeterminada, "falso" (.F.), aparece la ventana del Analizador de trayecto.

Además de ver la información de perfil, puede insertar en ella comentarios o marcas, y guardarla en un archivo para utilizarla posteriormente.

## Modificar el Analizador de trayecto

De forma predeterminada, la aplicación Analizador de trayecto se ejecuta en una ventana independiente. Si lo desea, puede cambiar la opción Entorno para que se ejecute dentro de la ventana principal de Visual FoxPro. En el [cuadro de diálogo Opciones del Analizador de trayecto](#), cambie la selección de **Entorno** de **Marco de trayecto** a **Marco de FoxPro** y a continuación reinicie el Analizador de trayecto.

También puede utilizar el cuadro de diálogo **Opciones del Analizador de trayecto** para modificar las siguientes características del Analizador de trayecto:

Característica	Descripción
Complementos	Especifica si se deben registrar los complementos en el Analizador de trayecto cuando se utilizan. Si desea más información, vea la sección "Complementos del Analizador de trayecto".
Marcas de trayecto	Especifica si el Analizador de trayecto debe marcar el código que se ejecuta o el que no se ejecuta. Especifica los caracteres empleados para marcar el código. Especifica cuándo se debe marcar el código.
Fuentes	Especifica las fuentes que utiliza el Analizador de trayecto para el código y las presentaciones.
Búsqueda inteligente	Especifica si el Analizador de trayecto debe buscar automáticamente archivos en las ubicaciones especificadas anteriormente.
Modo inicial	Especifica si el Analizador de trayecto se debe abrir en el modo trayecto o en el modo perfil.

## Asegurar la corrección de los datos del Analizador de trayecto

Para ayudar a asegurar que los archivos que procesa el Analizador de trayecto son los correctos:

- Establezca el directorio del proyecto como predeterminado antes de iniciar el registro del trayecto, de forma que los archivos a los que haga referencia sean relativos a ese directorio.
- Evite cambiar dinámicamente el nombre de los objetos. El Analizador de trayecto no encontrará los objetos cuyo nombre haya cambiado en tiempo de ejecución.
- Evite el uso de archivos de origen cuyos nombres tengan exactamente la misma raíz, incluso cuando las extensiones sean distintas. Internamente, el Analizador de trayecto no puede distinguirlos.
- Asegúrese de que el proyecto sólo contiene las versiones correctas de los archivos modificados con frecuencia.
- Asegúrese de que el proyecto no contiene varias copias de un mismo archivo en subdirectorios

distintos.

- Realice una compilación para ejecutar el trayecto:
  - Asegúrese de que la aplicación contiene información de depuración.
  - Desactive el cifrado (OFF).
  - Utilice RECOMPILE o Generar todo para obligar a compilar todo el código fuente.
- Realice la compilación inmediatamente antes de la ejecución del trayecto, para asegurar que el código fuente concuerde exactamente con el código objeto.

Algunas líneas del código, como los comentarios, las instrucciones `DEFINE CLASS` y `ELSE`, así como las líneas comprendidas entre `TEXT` y `ENDTEXT` no aparecen en los registros de trayecto, ya que no son ejecutables ni siquiera potencialmente. Además, las líneas interrumpidas por símbolos de continuación (caracteres punto y coma) se consideran como una sola línea de código y sólo se marca la última línea que las forma.

## Complementos del Analizador de trayecto

Los complementos son archivos de código (normalmente .prg o .scx) que proporcionan un método sencillo para reajustar el Analizador de trayecto. La subclase `cov_standard` del motor de trayecto que engloba la interfaz de usuario de `Coverage.app` sólo muestra una pequeña parte de lo que puede hacer con el motor. El motor analiza el registro de trayecto; mientras que `cov_standard` simplemente muestra el resultado en una de las numerosas formas en que puede obtenerse.

Puede crear una subclase de `cov_engine` distinta con una presentación muy diferente. Por ejemplo, esa subclase podría mostrar un cuadro de diálogo que ejecutase consultas sobre las estadísticas de trayecto reunidas por el motor. Las opciones de presentación podrían ofrecer una vista del código marcado para un conjunto filtrado de entradas del registro o únicamente un gráfico con los resultados del perfil.

Es posible que no encuentre conveniente usar otra subclase de `cov_engine` para crear una nueva interfaz desde cero, ya que la clase `cov_engine` permite un proceso más sencillo: puede agregar funcionalidad a `cov_standard` o a cualquier otra subclase de `cov_engine` mediante complementos. `Cov_standard` expone esta característica a través de un botón del cuadro de diálogo principal del Analizador de trayecto. Al ejecutar un complemento en una instancia de `cov_standard`, como por ejemplo el Analizador de trayecto, el complemento puede manipular la funcionalidad de `cov_engine`, las tablas de trayecto y también `cov_standard`. Los complementos podrían además agregar nuevos cuadros de diálogo y otras características a la interfaz visual de `cov_standard`.

## Escribir complementos

Puede escribir complementos para mejorar la interfaz estándar o bien puede usar una subclase de `cov_standard` para crear su propia interfaz, completamente nueva.

## Mejorar la aplicación estándar

En la lista siguiente se señalan mejoras que podrían conseguirse mediante complementos:

- Agregar una característica visible al cuadro de diálogo principal.
- Agregar un cuadro de diálogo al conjunto de formularios del motor de trayecto (observe más abajo la limitación sobre la forma de asegurarse de que el cuadro de diálogo aparezca en el lugar correcto).
- Mostrar un cuadro de diálogo aparte para tener acceso a una característica del motor de trayecto (observe más abajo la limitación sobre la forma de asegurarse de que el cuadro de diálogo aparezca en el lugar correcto).
- Proporcionar una interfaz de consultas que utilice la tabla de origen y presente una lista con todas las líneas que satisfagan los criterios, y que filtre y ordene el resultado.

**Nota** Puede utilizar los métodos `Adjust...` del motor (`AdjustCoverageFilenameCursor()`, `AdjustSourceCursor()` y `AdjustTargetCursor()`) para agregar campos a las tablas de origen y destino cuando el motor las crea, y utilizar esos campos en los complementos.

- Agregar nombres de archivo al cursor `IgnoredFiles` para eliminarlos del análisis, lo que puede ahorrar tiempo.
- Utilizar en enlace especial `Init` para los complementos.
- Registrar los complementos para recuperarlos y tener acceso a los mismos fácilmente desde una lista.

La clase de diálogo modal `cov_AddInDialog` de la subclase del motor de trayecto estándar presenta en una lista desplegable los cuadros de diálogo registrados previamente. Al establecer `ON` en la opción `IRegisterAdd-In` del motor de trayecto, se agrega al Registro de Windows la ruta completa de los complementos ejecutados con éxito, lo que permite volver a ejecutarlos fácilmente. La clase `Standard UI` también permite establecer esta propiedad en el [cuadro de diálogo Opciones del Analizador de trayecto](#).

El objeto Motor de trayecto mantiene en la propiedad `aAddIns` una lista con todos los complementos registrados.

- Utilizar la información de los campos del archivo final `coverage.log`, la pila de llamadas, para diseñar su propia interfaz o su propia vista del registro de trayecto.

Al escribir complementos, considere lo siguiente:

- Puede utilizar cualquiera de los tipos de archivo admitidos como complementos. Los tipos de archivo admitidos son `.qpr`, `.qpx`, `.mpr`, `.mpx`, `.app`, `.exe`, `.scx`, `.fxp`, `.prg` y los procedimientos (si ya están disponibles en una biblioteca de procedimientos abierta).
- El conjunto de formularios del Motor de trayecto tiene una barra de herramientas "invisible". Si un complemento no es visual, puede utilizar esta barra de herramientas para que lo contenga. Si el complemento es un control visual, probablemente el cuadro de diálogo principal de la subclase estándar es el lugar más adecuado para ubicarlo. De este modo su posición y tamaño



se sincronizará automáticamente con el resto del cuadro de diálogo al cambiar el tamaño de éste.

- Todos los métodos del motor que utilizan las tablas de origen y destino tienen argumentos opcionales que permiten apuntarlos desde los alias correspondientes al trabajar con ellos. También puede cambiar el contenido actual de las propiedades `cSourceAlias` y `cTargetAlias` para hacerlo coincidir con el par de cursores que le interesen. Así podrá comparar diversas ejecuciones del registro de trayecto entre sí desde la misma interfaz.
- Limitaciones:
  - Los complementos deben aceptar un parámetro (el Motor de trayecto pasa una referencia a sí mismo).
  - Los complementos deben ser de uno de los tipos de archivo permitidos indicados anteriormente.
  - Los procedimientos que utilice como complementos deben estar disponibles en una biblioteca de procedimientos que se encuentre cargada (vea [SET PROCEDURE](#)). El Motor no utiliza la sintaxis `IN NombreArchivo` y no llama a los procedimientos ni a los archivos `.prg` como funciones, ni obtiene sus valores con `RETURN`. Tampoco utiliza las palabras clave `NAME` o `LINK` en el comando `DO FORM`, por lo que puede administrar usted mismo la referencia o hacer que el formulario sea miembro del conjunto de formularios del Motor para que éste lo alcance automáticamente.
  - Si ejecuta un complemento al inicio deberá utilizar una referencia, ya que la variable pública `_oCoverage` aún no estará disponible. En las demás ocasiones, puede utilizar la referencia a la variable pública en el código, si lo prefiere.
  - Al escribir un complemento como un formulario, si crea el formulario con `ShowWindow = 1` y ejecuta el trayecto en su propio marco, los complementos formulario deberán aparecer en el marco del trayecto.
  - Si utiliza `.RunAddIn` desde la ventana Comandos, asegúrese de que el marco del trayecto sea el marco MDI activo antes de instanciar los formularios.

### Crear una subclase de la clase `Cov_Standard`

Puede crear una subclase del motor de trayecto, su subclase estándar. En la lista siguiente se describe la estructura del conjunto de archivos de origen del proyecto `COVERAGE`.

Archivo	Descripción
Coverage.prg	"Envoltura" del objeto trayecto, que instancia el objeto.
Coverage.vcx Coverage.vct	Todas las clases del motor y su subclase estándar.
Cov_short.mnx Cov_short.mnt	Menú contextual.

Cov_pjx.frx Cov_pjx.frt	Mecanismo predeterminado para obtener resultados a nivel de proyecto.
Coverage.h	<p>Archivo de encabezado para todo el código de trayecto, que incorpora los elementos siguientes:</p> <p>*— Constantes de caracteres de trayecto para registro y análisis:</p> <pre>#INCLUDE COV_CHAR.H</pre> <p>*— Cadenas de trayecto localizadas (puede utilizar algunas constantes de registro y análisis):</p> <pre>#INCLUDE COV_LOCS.H</pre> <p>*— Constantes de componentes de cuadros de diálogo comunes del trayecto:</p> <pre>#INCLUDE COV_DLGS.H</pre> <p>*— Especificaciones y requisitos del trayecto:</p> <pre>#INCLUDE COV_SPEC.H</pre> <p>*— Constantes de objetos del registro de trayecto:</p> <pre>#INCLUDE COV_REGS.H</pre> <p>*— Opciones de ajuste de trayecto:</p> <pre>#INCLUDE COV_TUNE.H</pre>

El conjunto de archivos de origen del proyecto COVERAGE también incluye otros archivos .ico .bmp y .msk.

Puede utilizar el archivo COV\_TUNE.H (que contiene los comentarios y explicaciones correspondientes) para familiarizarse con las opciones disponibles sin tener que rescribir el código.

Como el uso de complementos está gobernado por la superclase del motor de trayecto, cualquier otra subclase de trayecto que cree podrá utilizar complementos de la misma forma en que lo hace la subclase estándar.

La subclase del motor de trayecto instanciada por la aplicación predeterminada Coverage.app no aumenta el método RunAddIn( ) del motor de trayecto de ningún modo. El cuadro de diálogo modal recibe una referencia al objeto Trayecto y establece la propiedad cAddIn del motor de trayecto.

Si escribe su propia subclase del motor de trayecto, asegúrese de que pueda utilizar la misma clase de cuadro de diálogo modal (cov\_AddInDialog) para tratar los complementos como lo hace la aplicación trayecto estándar. El cuadro de diálogo no se apoya en ninguna característica de la subclase estándar.

Puede llamar a un cuadro de diálogo modal distinto, establecer el nombre de archivo directamente en la propiedad `cAddIn` o anular el contenido de esta propiedad al pasar el nombre del archivo de complemento que desea ejecutar al método `RunAddIn()`.

Independientemente del modo en que logre el acceso a un complemento para ejecutarlo en una subclase, puede observar la lista de complementos registrados en `Coverage.app` si busca los nombres de los archivos en la propiedad `aAddIns` del motor de trayecto.

Si desea información sobre las propiedades, eventos y métodos del motor de trayecto, vea [Objeto Motor de trayecto](#).

## Enganches del Administrador de proyectos

En versiones anteriores de Visual FoxPro, el único acceso a un proyecto se lograba mediante la manipulación directa de las tablas del archivo `.pjx` del proyecto. En Visual FoxPro 6.0, el acceso a un proyecto puede realizarse mediante programa, lo que permite manipular el proyecto como un objeto. Un proyecto puede manipularse en tiempo de diseño, mientras está abierto en el Administrador de proyectos, o en tiempo de diseño y en tiempo de depuración sin que el Administrador de proyectos esté visible.

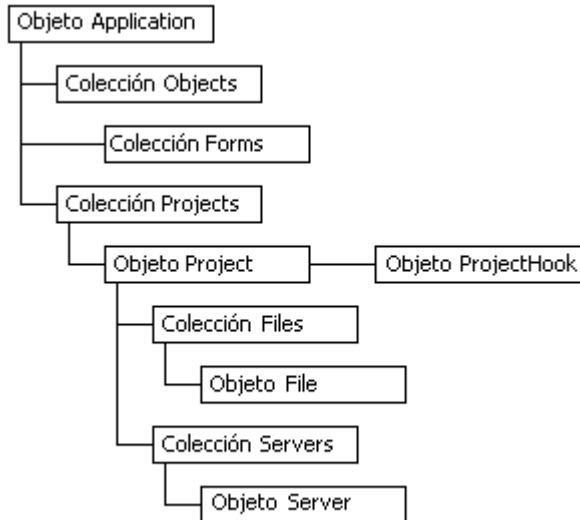
Las siguientes son algunas de las acciones que pueden realizarse sobre un proyecto mediante programación:

- Agregar o eliminar archivos de un proyecto.
- Agregar archivos del proyecto a aplicaciones de control de código fuente (como por ejemplo Microsoft Visual SourceSafe) y proteger o desproteger archivos con control de código fuente.
- Determinar el número de archivos de un proyecto y sus tipos.
- Abrir y modificar los archivos de un proyecto.
- Cambiar las propiedades de un proyecto.
- Cambiar las propiedades de los archivos de un proyecto.
- Cambiar las propiedades de los servidores Automation (bibliotecas de vínculo dinámico, `.dll` o archivos ejecutables `.exe`) generados desde el proyecto.
- Ejecutar código cuando se producen eventos en el proyecto.
- Volver a generar el proyecto o generar archivos `.app`, `.exe` o `.dll` a partir del mismo.

Con los nuevos enganches del Administrador de proyectos, los programadores avanzados pueden crear sus propios administradores de proyectos con interfaces de usuario únicas y personalizadas.

## Jerarquía de objetos de un proyecto

La jerarquía de objetos de un proyecto consta del proyecto, un objeto proyecto y su objeto ProjectHook asociado. El objeto Project contiene una colección de archivos, formada por los archivos del proyecto y una colección de servidores, formada por los servidores de Automatización creados desde el proyecto. En los diagramas siguientes se ilustra la jerarquía de objetos de un proyecto dentro del modelo de objetos de Visual FoxPro:



## Colección Proyectos

La colección Proyectos proporciona acceso directo al objeto proyecto y así permite manipular el proyecto, y también los archivos y servidores que contiene. A la colección Proyectos se le agrega un objeto Project siempre que se crea o se abre un proyecto, o cuando se genera un .app, .dll o .exe desde él.

Al igual que ocurre con otras colecciones OLE, es posible obtener información sobre un proyecto a partir de la colección Proyectos. Por ejemplo, el código siguiente utiliza las propiedades Count e Item de la colección Proyectos para mostrar los nombres de todos los proyectos de la colección y a continuación utiliza el comando FOR EACH para mostrar la misma información:

```

nProjectCount = Application.Projects.Count

FOR nCount = 1 TO nProjectCount
    ? Application.Projectos.Item(nCount).Name
NEXT

FOR EACH oProj IN Application.Projectos
    ? oProj.Name
ENDFOR
  
```

Esta línea de código utiliza la propiedad ActiveProject para agregar un programa, Main.prg, al proyecto activo actualmente:

```
Application.ActiveProject.Archivos.Add('Main.prg')
```

Esta línea de código agrega Main.prg al primer proyecto agregado a la colección Proyectos:

```
Application.Proyectos[1].Archivos.Add( 'Main.prg' )
```

La colección Proyectos tiene la propiedad y el método siguientes:

### Propiedades

---

---

[Count](#)

---

---

### Métodos

---

---

[Item](#)

---

---

## Objeto Project

El objeto Project se instancia siempre que se abre un proyecto desde el menú Archivo o con los comandos CREATE PROJECT, MODIFY PROJECT, BUILD APP, BUILD DLL, BUILD EXE o BUILD PROJECT. El objeto Project permite manipular el proyecto mediante programa y es posible el acceso al mismo a través del [objeto Application](#) de Visual FoxPro. Observe que el objeto Application cuenta con la nueva propiedad [ActiveProject](#), que proporciona una referencia de objeto al proyecto abierto en el Administrador de proyectos activo actualmente.

El objeto Project tiene las propiedades y métodos siguientes:

### Propiedades

<a href="#">Application</a>	<a href="#">AutoIncrement</a>
<a href="#">BaseClass</a>	<a href="#">BuildDateTime</a>
<a href="#">Debug</a>	<a href="#">Encrypted</a>
<a href="#">HomeDir</a>	<a href="#">Icon</a>
<a href="#">MainClass</a>	<a href="#">MainFile</a>
<a href="#">Name</a>	<a href="#">Parent</a>
<a href="#">ProjectHook</a>	<a href="#">ProjectHookClass</a>
<a href="#">ProjectHookLibrary</a>	<a href="#">SCCProvider</a>
<a href="#">ServerHelpFile</a>	<a href="#">ServerProject</a>
<a href="#">TypeLibCLSID</a>	<a href="#">TypeLibDesc</a>
<a href="#">TypeLibName</a>	<a href="#">VersionComments</a>
<a href="#">VersionCompany</a>	<a href="#">VersionCopyright</a>
<a href="#">VersionDescription</a>	<a href="#">VersionLanguage</a>
<a href="#">VersionNumber</a>	<a href="#">VersionProduct</a>

[VersionTrademarks](#)[Visible](#)**Métodos**[Build](#)[CleanUp](#)[Refresh](#)[SetMain](#)**Objeto ProjectHook**

Un objeto ProjectHook es una clase de base de Visual FoxPro que se instancia de forma predeterminada siempre que se abre un proyecto asignado al objeto ProjectHook. (Puede incluir la cláusula NOPROJECTHOOK en CREATE PROJECT y MODIFY PROJECT para evitar que se instancie un objeto ProjectHook para el proyecto.)

El objeto ProjectHook permite el acceso mediante programación a los eventos que ocurren en un proyecto. Por ejemplo, puede ejecutar código siempre que agregue un archivo al proyecto.

Puede especificar una clase ProjectHook predeterminada para los nuevos proyectos en la [ficha Proyectos](#) del cuadro de diálogo Opciones. Si no se especifica una clase ProjectHook predeterminada en la ficha Proyectos, no se asignará ninguna a los proyectos nuevos. Puede especificar una clase ProjectHook para un proyecto individual (anulando la predeterminada) en el [cuadro de diálogo Información del proyecto](#). En tiempo de ejecución, puede utilizar la propiedad ProjectHook para especificar una clase ProjectHook para un proyecto. Si cambia la clase ProjectHook de un proyecto, la nueva no tendrá efecto hasta que cierre el proyecto y lo abra de nuevo.

El objeto ProjectHook tiene las propiedades, eventos y métodos siguientes:

**Propiedades**[BaseClass](#)[Class](#)[ClassLibrary](#)[Comment](#)[Name](#)[OLEDropEffects](#)[OLEDropHasData](#)[OLEDropMode](#)[Parent](#)[ParentClass](#)[Tag](#)**Eventos**[AfterBuild](#)[BeforeBuild](#)[Destroy](#)[Error](#)[Init](#)[OLEDragDrop](#)

[OLEDragOver](#)[OLEGiveFeedBack](#)[QueryAddFile](#)[QueryModifyFile](#)[QueryRemoveFile](#)[QueryRunFile](#)

## Métodos

[AddProperty](#)[ReadExpression](#)[ReadMethod](#)[ResetToDefault](#)[SaveAsClass](#)[WriteExpression](#)

## El objeto Project y su interacción con el objeto ProjectHook

Al abrir el Administrador de proyectos desde el menú Archivo o con los comandos [CREATE PROJECT](#) o [MODIFY PROJECT](#), aparecerá la ventana del Administrador de proyectos y se instanciará un objeto Project con su objeto ProjectHook asociado. Los comandos de generación de proyectos ([BUILD PROJECT](#), [BUILD APP](#), [BUILD DLL](#) y [BUILD EXE](#)) también instancian los objetos Project y ProjectHook.

Cuando se produce un evento en un proyecto, el objeto Project lo envía al objeto ProjectHook. Entonces se ejecuta el código de usuario del evento en el objeto ProjectHook y el control se envuelve al objeto Project. El valor devuelto al objeto Project desde el objeto ProjectHook determina si el objeto Project termina la operación. Si incluye NODEFAULT en el código del evento, evitará que se realice la acción predeterminada. Por ejemplo, al agregar NODEFAULT al evento QueryAddFile se impide que se agregue un archivo con éxito al proyecto.

## Colección Archivos

La colección Archivos proporciona acceso directo al objeto archivo, lo que permite manipular los objetos archivo de un proyecto mientras éste está abierto. Al igual que ocurre con otras colecciones OLE, es posible obtener información sobre un archivo de un proyecto a partir de la colección Archivos. Por ejemplo, en el código siguiente se utilizan las propiedades Count e Item de la colección Archivos para mostrar los nombres de todos los archivos de la colección y, a continuación, se utiliza el comando FOR EACH para mostrar la misma información:

```
nFileCount = Application.ActiveProject.Archivos.Count

FOR nCount = 1 TO nFileCount
    ? Application.ActiveProject.Archivos.Item(nCount).Name
NEXT

FOR EACH oProj IN Application.ActiveProject.Archivos
    ? oProj.Name
ENDFOR
```

Esta línea de código utiliza la propiedad ActiveProject para agregar un archivo, Main.prg, al proyecto activo actualmente:

```
Application.ActiveProject.Archivos.Add( 'Main.prg' )
```

Esta línea de código agrega Main.prg al primer proyecto agregado a la colección Proyectos:

```
Application.Proyectos[1].Archivos.Add( 'Main.prg' )
```

La colección Archivos tiene las propiedades y métodos siguientes:

### Propiedades

---

---

[Count](#)

---

---

### Métodos

---

---

[Add](#)

---

---

[Item](#)

---

---

### Objeto archivo

El objeto archivo permite manipular los archivos individuales de un proyecto.

El objeto archivo tiene las propiedades y métodos siguientes:

### Propiedades

---

---

[CodePage](#)

---

---

[Description](#)

---

---

[Exclude](#)

---

---

[FileClass](#)

---

---

[FileClassLibrary](#)

---

---

[LastModified](#)

---

---

[Name](#)

---

---

[ReadOnly](#)

---

---

[SCCStatus](#)

---

---

[Type](#)

---

---

### Métodos

---

---

[AddToSCC](#)

---

---

[CheckIn](#)

---

---

[CheckOut](#)

---

---

[GetLatestVersion](#)

---

---

[Modify](#)

---

---

[Remove](#)

---

---

[RemoveFromSCC](#)

---

---

[Run](#)

---

---

[UndoCheckOut](#)

---

---

### Colección Servidores



La colección Servidores ofrece acceso directo al objeto servidor, lo que permite manipular los servidores que contiene un proyecto. A la colección Servidores se le agrega un objeto servidor siempre que se genera una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) desde el proyecto. Si desea más información sobre la creación de servidores de Automatización, vea [Crear servidores de Automatización](#) en el Capítulo 16, “Agregar OLE”, del *Manual del programador*.

La colección Servidores tiene la propiedad y el método siguientes:

### Propiedades

---

---

[Count](#)

---

---

### Métodos

---

---

[Item](#)

---

---

### Objeto servidor

El objeto servidor permite determinar información (incluida información de biblioteca de tipos) sobre los servidores de Automatización contenidos en un proyecto. Esta información también está disponible en la [ficha Servidores](#) del cuadro de diálogo Información del proyecto. Observe que el objeto servidor no se crea hasta que se genera el proyecto que contiene la clase OLEPUBLIC (especificada en el comando [DEFINE CLASS](#)).

El objeto servidor tiene las propiedades y métodos siguientes:

### Propiedades

---

---

<a href="#">CLSID</a>	<a href="#">Description</a>
<a href="#">HelpContextID</a>	<a href="#">Instancing</a>
<a href="#">ProgID</a>	<a href="#">ServerClass</a>
<a href="#">ServerClassLibrary</a>	

---

---

### Arquitectura del objeto Project

El objeto Project de Visual FoxPro expone una interfaz IDispatch para que los clientes de Automatización, los controles ActiveX y otros objetos COM tengan acceso a él a través de interfaces OLE estándar. Al exponer el objeto Project una interfaz IDispatch, los errores que puedan generarse al manipular proyectos serán errores OLE.

### Mejoras del lenguaje

Existen dos nuevas cláusulas para los comandos CREATE PROJECT y MODIFY PROJECT. La primera, NOPROJECTHOOK, evita que se instancie el objeto ProjectHook de un proyecto. La

segunda, NOSHOW, abre el proyecto sin mostrarlo en el Administrador de proyectos, lo que permite manipularlo mediante programación sin mostrarlo. Puede utilizar la propiedad Visible para mostrar posteriormente el Administrador de proyectos. Si desea más información sobre estas nuevas cláusulas, vea [CREATE PROJECT](#) y [MODIFY PROJECT](#).

## Eventos de proyecto

En las secciones siguientes se describen los eventos que ocurren cuando se crean proyectos, se modifican, cierran, generan, etcétera; y también el orden en que se producen.

### Crear un nuevo proyecto

Los eventos siguientes se producen al ejecutar CREATE PROJECT, al crear un nuevo proyecto desde el menú **Archivo** o al hacer clic en el botón **Nuevo** de la barra de herramientas y especificar la creación de un proyecto nuevo:

1. Se crea el objeto Project.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, el proyecto se crea y se muestra en el Administrador de proyectos.

Si el evento Init devuelve "falso" (.F.), el proyecto no se crea, se liberan los objetos Project y ProjectHook y no se muestra el Administrador de proyectos.

### Modificar un proyecto existente

Los eventos siguientes se producen al ejecutar MODIFY PROJECT, al modificar un proyecto existente desde el menú **Archivo** o al hacer clic en el botón **Abrir** de la barra de herramientas y especificar un proyecto nuevo o ya existente:

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.) (el valor predeterminado), el proyecto se abre para su modificación en el Administrador de proyectos.

Si el evento Init devuelve "falso" (.F.), el proyecto no se abre para su modificación, se liberan los objetos Project y ProjectHook, y no se muestra el Administrador de proyectos.

### Cerrar un proyecto

Los eventos siguientes se producen al cerrar un proyecto abierto:

1. Se produce el evento Destroy del objeto ProjectHook y se libera el objeto ProjectHook.

2. Se libera el objeto Project.

### **Ejecutar BUILD APP, BUILD DLL o BUILD EXE**

Los eventos siguientes se producen cuando se ejecuta BUILD APP, BUILD DLL o BUILD EXE:

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init para el objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, se produce el evento BeforeBuild del objeto ProjectHook. Si se ha especificado NODEFAULT en el evento BeforeBuild, el archivo .app, .dll o .exe no se genera. En caso contrario, el proceso de generación continúa.

Si se agrega algún archivo al proyecto durante el proceso de generación, se producirá el evento QueryAddFile del objeto ProjectHook antes de agregar efectivamente el archivo. Si se ha especificado NODEFAULT en el evento QueryAddFile, no se agregará el archivo al proyecto. En caso contrario, el archivo se agrega. Cuando se ha generado con éxito el archivo .app, .dll o .exe, se produce el evento AfterBuild del objeto ProjectHook y, a continuación, su evento Destroy.

Si el evento Init devuelve "falso" (.F.), el archivo .app, .dll o .exe se genera, y se liberan los objetos Project y ProjectHook.

### **Ejecutar BUILD PROJECT**

Los eventos siguientes se producen al ejecutar BUILD PROJECT con la cláusula FROM. Si se omite la cláusula FROM, los eventos se producen en el orden descrito anteriormente para la ejecución de BUILD APP, BUILD DLL o BUILD EXE.

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, los archivos especificados en la cláusula FROM se agregan individualmente al proyecto. El evento QueryAddFile del objeto ProjectHook se produce antes de agregar cada uno de los archivos. Si se ha especificado NODEFAULT en el evento QueryAddFile, los archivos no se agregarán al proyecto. En caso contrario, sí se agregan.

A continuación se produce el evento BeforeBuild del objeto ProjectHook. Si se ha especificado NODEFAULT en el evento BeforeBuild, el proyecto no se generará. En caso contrario, el proyecto se genera. Cuando termina la generación, se produce el evento AfterBuild del objeto ProjectHook y, a continuación, tiene lugar su evento Destroy.

Si el evento Init del objeto ProjectHook devuelve "falso" (.F.), el proyecto no se genera. En tal caso se liberan los objetos Project y ProjectHook, y no se crea un nuevo archivo .pjx.

## Realizar una operación arrastrar y colocar

Los eventos siguientes se producen al arrastrar un archivo o un conjunto de archivos sobre la sección de esquema (árbol) del Administrador de proyectos:

1. Cuando el puntero del *mouse* se sitúa sobre la sección de esquema del Administrador de proyectos, se produce el evento `OLEDragOver` del objeto `ProjectHook`, con el valor 0 (`DRAG_ENTER` en `Foxpro.h`) en el parámetro *nEstado*. A continuación se produce el evento `OLEDragOver` repetidamente con el valor 2 (`DRAG_OVER` en `Foxpro.h`) en el parámetro *nEstado*. Si el puntero del *mouse* sale de la sección de esquema del Administrador de proyectos, se produce el evento `OLEDragOver` con el valor 1 (`DRAG_LEAVE` en `Foxpro.h`) en el parámetro *nEstado*.
2. El evento `OLEDragDrop` del objeto `ProjectHook` se produce al soltar el botón del *mouse* cuando el puntero se encuentra sobre la sección de esquema del Administrador de proyectos. De forma predeterminada, Visual FoxPro agrega al proyecto todos los archivos colocados en el Administrador de proyectos. Antes de agregar cada archivo, se produce el evento `QueryAddFile` del objeto `ProjectHook`.

## Agregar un archivo con el botón Agregar

Los eventos siguientes se producen al agregar un archivo a un proyecto con el botón **Agregar** del Administrador de proyectos:

1. Aparece el cuadro de diálogo **Abrir**.
2. Si selecciona un archivo y elige **Aceptar**, se creará un objeto archivo para el archivo seleccionado.
3. Se produce el evento `QueryAddFile` del objeto `ProjectHook` y se pasa al evento el nombre del objeto archivo. Si se ha especificado `NODEFAULT` en el evento `QueryAddFile`, el archivo no se agrega. En caso contrario, el archivo se agrega al proyecto.

## Agregar un archivo con el botón Nuevo

Los eventos siguientes se producen al agregar un archivo nuevo a un proyecto con el botón **Nuevo** del Administrador de proyectos:

1. Aparece el diseñador o editor correspondiente al archivo.
2. Cuando se guarda el nuevo archivo, aparece el cuadro de diálogo **Guardar como**. Al hacer clic en **Guardar** se crea un objeto archivo para el nuevo archivo.
3. Se produce el evento `QueryAddFile` del objeto `ProjectHook` y se le pasa el nombre del objeto archivo. Si se ha incluido `NODEFAULT` en el evento `QueryAddFile`, el archivo no se agrega. En caso contrario, el archivo se agrega al proyecto.

## Modificar un archivo con el botón Modificar

Los eventos siguientes se producen al modificar un archivo de un proyecto con el botón **Modificar** del Administrador de proyectos:

1. Se produce el evento QueryModifyFile del objeto ProjectHook antes de que se muestre el diseñador o editor correspondiente al archivo.
2. Se pasa como parámetro al evento QueryModifyFile el objeto archivo del archivo a modificar. Si se ha incluido NODEFAULT en el evento QueryModifyFile, no se muestra el diseñador o editor correspondiente al archivo y éste no se modifica. En caso contrario, el archivo se abre con el diseñador o editor correspondiente para su modificación.

### Quitar un archivo con el botón **Quitar**

Los eventos siguientes se producen al quitar un archivo de un proyecto con el botón **Quitar** del Administrador de proyectos:

1. Se produce el evento QueryRemoveFile del objeto ProjectHook.
2. El objeto archivo correspondiente al archivo que se va a quitar se pasa como parámetro al evento QueryRemoveFile. Si se ha especificado NODEFAULT en el evento QueryRemoveFile, el archivo no se quita. En caso contrario, el archivo se quita del proyecto.

### Ejecutar un archivo con el botón **Ejecutar**

Los eventos siguientes se producen al ejecutar un archivo de un proyecto con el botón **Ejecutar** del Administrador de proyectos:

1. Se produce el evento QueryRunFile del objeto ProjectHook.
2. El objeto archivo correspondiente al archivo que se va a ejecutar se pasa como parámetro al evento QueryRunFile. Si se ha especificado NODEFAULT en el evento QueryRunFile, el archivo no se ejecuta. En caso contrario, se ejecuta.

### Volver a generar un proyecto o generar un archivo con el botón **Generar**

Los eventos siguientes se producen al volver a generar un proyecto, o al generar un archivo .app, .dll o .exe desde un proyecto con el botón **Generar** del Administrador de proyectos:

1. Aparece el cuadro de diálogo **Opciones de generación**.
2. Puede elegir **Volver a generar el proyecto**, **Generar aplicación**, **Generar ejecutable** o **Generar DLL COM** y luego especificar opciones de generación adicionales. Si hace clic en **Cancelar**, la generación no se produce.
3. El evento BeforeBuild del objeto ProjectHook se produce al elegir **Aceptar**, y entonces comienza el proceso de generación.
4. Al completarse la generación, se produce el evento AfterBuild del objeto ProjectHook.

## Ejemplo de enlaces del Administrador de proyectos

La aplicación de ejemplo Solutions de Visual FoxPro incluye un ejemplo llamado “Seguir actividades de un proyecto” que ilustra muchos de los nuevos enlaces del Administrador de proyectos.

### Para ejecutar la aplicación de ejemplo Solutions

- Escriba lo siguiente en la ventana **Comandos**:

```
DO (HOME(2) + 'solution\solution')
```

– O bien –

1. En el menú **Programa**, elija **Ejecutar**.
2. Elija la carpeta ...\**Samples\Vfp98\Solution**.
3. Haga doble clic en **Solution.app**.

### Para ejecutar el ejemplo “Seguir actividades de un proyecto”

1. Una vez iniciado Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Seguir actividades de un proyecto** y luego en el botón **Ejecutar ejemplo**.

El ejemplo “Seguir actividades de un proyecto” permite abrir un proyecto para después manipularlo de diversas formas. Los cambios realizados en el proyecto se almacenan en una tabla. Al cerrar el proyecto, podrá ver los cambios realizados en él en una ventana Examinar.

Si desea más información sobre el funcionamiento del ejemplo “Seguir actividades de un proyecto” y desea observar más de cerca el código que tiene detrás, puede abrir el formulario utilizado para crearlo.

### Para abrir el formulario “Seguir actividades de un proyecto”

1. Una vez iniciado Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Seguir actividades de un proyecto** y luego en el botón **Ver código**.

Acttrack.scx, el formulario empleado para crear el ejemplo “Seguir actividades de un proyecto”, se abrirá en el Diseñador de formularios.

También puede estimar conveniente observar más de cerca la biblioteca de clases de ProjectHook, Project\_hook.vcx, que está asignada al proyecto abierto en el ejemplo “Seguir actividades de un proyecto”. La mayoría del código que se ejecuta cuando se producen eventos en el proyecto se encuentra en los procedimientos de evento de esta biblioteca de clases. Project\_hook.vcx se encuentra

en el directorio ...\\Samples\\Vfp98\\Solution\\Tahoe.

## Asistentes y generadores nuevos y mejorados

Los asistentes y generadores siguientes son nuevos o se han mejorados.

### Asistente para aplicaciones *Nuevo*

El Asistente para aplicaciones de Visual FoxPro 6.0 admite el [Marco de trabajo de aplicación](#) mejorado y el nuevo [Generador de aplicaciones](#). Puede ejecutar el Asistente para aplicaciones desde la [Galería de componentes](#) o desde el menú **Herramientas** de Visual FoxPro, al hacer clic en **Asistentes** y luego en **Aplicación**.

**Nota** El [Asistente para aplicaciones \(5.0\)](#) de Visual FoxPro 5.0 está disponible en el [cuadro de diálogo Selección de asistente](#) por motivos de compatibilidad con versiones anteriores.

### Asistentes para conexión *Nuevos*

Los Asistentes para conexión incluyen el **Asistente para generación de código** y el **Asistente para ingeniería inversa**. Estos asistentes permiten administrar fácilmente las transferencias entre las bibliotecas de clases de Visual FoxPro y los modelos de Microsoft Visual Modeler.

### Asistente para bases de datos *Nuevo*

El Asistente para bases de datos de Visual FoxPro utiliza plantillas para crear bases de datos y tablas. También puede usar este asistente para crear índices y relaciones entre las tablas de una nueva base de datos.

### Asistente para documentación *Mejorado*

El Asistente para documentación de Visual FoxPro ofrece ahora una versión para utilizar los [Analizadores de código](#) mientras se crea la documentación.

### Asistente para formularios *Mejorado*

El Asistente mejorado para formularios de Visual FoxPro proporciona máscaras de entrada, formatos y una clase de asignación de campos para campos específicos almacenados en una base de datos. Este asistente incluye también más opciones de estilo de formularios, como por ejemplo, si son desplazables.

### Asistente para gráficos *Mejorado*

El Asistente para gráficos de Visual FoxPro crea un gráfico a partir de una tabla de Visual FoxPro mediante Microsoft Graph. Este asistente actualizado admite el componente Graph 8.0 de Microsoft Office 97, que incluye la automatización de la hoja de datos y la serie por la opción fila/columna.

### Asistente para importar *Mejorado*

El Asistente mejorado para importar de Visual FoxPro admite Office 97 y el tratamiento de múltiples hojas de Microsoft Excel y proporciona la opción de importar una tabla en una base de datos.

#### Asistente para etiquetas    *Mejorado*

El Asistente para etiquetas de Visual FoxPro permite ahora un mayor control de las fuentes de las etiquetas y proporciona acceso directo al Asistente para agregar etiquetas.

#### Asistente para combinar correspondencia    *Mejorado*

El Asistente para Combinar correspondencia de Visual FoxPro crea un origen de datos con un documento combinado de Microsoft Word o con un archivo de texto que puede abrir cualquier procesador de texto. Este asistente admite el componente Microsoft Word 8.0 de Office 97, permite una auténtica automatización VBA con el [objeto Application](#) y utilizar colecciones.

#### Asistente para tablas dinámicas    *Mejorado*

El Asistente para tablas dinámicas de Visual FoxPro ayuda a crear tablas de hoja de cálculo interactivas que resumen y analizan los datos de dos o más campos. Este asistente actualizado admite el componente Microsoft Excel 8.0 de Office 97. Puede elegir guardar una tabla dinámica directamente en Excel o agregar una tabla dinámica como un objeto a un formulario.

#### Asistente para informes    *Mejorado*

El Asistente para informes de Visual FoxPro incluye ahora funcionalidad avanzada de agrupamiento y resumen que permite personalizar más fácilmente los informes dentro del propio asistente. También incorpora más estilos de informes entre los que elegir.

#### Asistente para vistas remotas    *Mejorado*

El Asistente para vistas remotas de Visual FoxPro proporciona acceso a las tablas de sistema, lo que le permite utilizar la funcionalidad de los controladores ODBC que las admitan.

#### Asistente para ejemplos    *Nuevo*

El Asistente para ejemplos de Visual FoxPro muestra en pasos sencillos cómo crear su propio asistente. El resultado es un archivo HTML creado a partir de los registros del origen de datos especificado.

#### Asistente para instalación    *Mejorado*

El Asistente para instalación de Visual FoxPro permite un mejor uso de los controles ActiveX y también anular el límite en el número de archivos que puede utilizar Windows para las instalaciones en NT. También permite agregar .DLL externas a la aplicación a través de la instalación y crear instalaciones basadas en Web.

#### Asistente para tablas    *Mejorado*



El Asistente para tablas de Visual FoxPro ofrece ahora nuevas plantillas de tablas, configuraciones de estilo opcionales, uso de los tipos de datos binarios Character y Memo y acceso a las bases de datos. Puede agregar una tabla a una base de datos, y también utilizar la configuración de la base de datos para determinar los formatos de los campos que se van a agregar a la tabla. También puede establecer relaciones entre las tablas de la base de datos.

### [Asistente para publicación en Web](#) *Nuevo*

El Asistente para publicación en Web de Visual FoxPro genera un archivo HTML creado a partir de los registros del origen de datos que especifique.

## Marco de trabajo de aplicación mejorado

El Marco de trabajo de aplicación de Visual FoxPro 6.0 está diseñado para hacer más fácil la programación de aplicaciones de Visual FoxPro. Para tener acceso al mismo puede utilizar el [Asistente para aplicaciones](#) o el elemento **Nueva aplicación** de la [Galería de componentes](#). Este marco de trabajo mejorado admite el marco disponible en Visual FoxPro 5.0, que incluye:

- Un archivo de proyecto (.pjx).
- Un archivo de programa principal (Main.prg) para la configuración global y de entorno, que inicia la pantalla de inicio u otras llamadas específicas y que inicia los formularios Tutorial.
- Un menú principal.
- El [objeto Marco de trabajo de Visual FoxPro](#) para ejecutar el menú principal, las barras de herramientas de los formularios y la administración de informes, el control de errores y la administración de las sesiones de datos.

El marco de trabajo de Visual FoxPro 6 utiliza un objeto Application mejorado, y proporciona los elementos adicionales siguientes:

- Archivo de inclusión principal que contiene el valor APP\_GLOBAL para facilitar su localización y su uso por parte de componentes con configuraciones y cadenas.
- Archivo de configuración opcional (Config.fpw) para determinados tipos de aplicaciones.
- Uso de la [clase ProjectHook](#) para el control de los eventos relacionados con el proyecto.
- Metatabla de aplicación para mantener la información que utilizan la clase ProjectHook y los generadores de aplicaciones para crear formularios en el nuevo proyecto.
- Uso del [Generador de aplicaciones](#) para facilitar la adición de componentes al proyecto.

### Iniciar el Generador de aplicaciones

Puede iniciar el [Generador de aplicaciones](#) desde el menú **Herramientas** de Visual FoxPro o desde la

## Galería de componentes.

### Para iniciar el Generador de aplicaciones en el menú Herramientas

1. Haga clic en **Asistentes** y, a continuación, haga clic en **Todos**.
2. Haga clic en **Generador de aplicaciones** en el cuadro de diálogo **Selección de Asistente**.

### Para iniciar el Generador de aplicaciones desde la Galería de componentes

- Haga doble clic en el elemento **Nueva aplicación**.

Al elegir **Aceptar** se cerrará el generador y se aplicará la configuración de las propiedades de todas las fichas.

También puede iniciar el Generador de aplicaciones con el botón secundario del *mouse* en la ventana Administrador de proyectos, pero al hacerlo de este modo el Generador de aplicaciones sólo crea metatablas para la aplicación y sólo verá tres fichas en él. La única forma de obtener el marco de trabajo de aplicación mejorado completo es a través del [Asistente para aplicaciones](#) o con el elemento Nueva aplicación de la [Galería de componentes](#).

Para obtener detalles sobre el contenido y el uso del marco de trabajo mejorado y el Generador de aplicaciones, vea [Programar aplicaciones con el marco de trabajo de aplicaciones](#) en la Ayuda.

## Archivos

### Archivo de inclusión principal

Este archivo #INCLUDE común lo utilizan los componentes con configuraciones y cadenas. También incluye el valor APP\_GLOBAL, el nombre que utilizan los componentes como referencia.

### Archivo de configuración

Archivo Config.fpw opcional utilizado en aplicaciones tales como formularios de alto nivel, para implementar configuraciones tales como SCREEN=OFF.

### Clase ProjectHook

Controla los eventos relacionados con el proyecto, tales como agregar nuevos archivos. También tiene acceso al Generador de aplicaciones para establecer acciones y propiedades de la interacción de los archivos con la aplicación.

### Metatabla de aplicación

Contiene información tal como la configuración de proyecto, establecida o utilizada por el Generador de aplicaciones y los objetos ProjectHook.

### Generador de aplicaciones

Facilita la adición de componentes al proyecto y el establecimiento de propiedades tales como las opciones desplazamiento.

Un marco de trabajo de aplicación incluye el archivo de proyecto como biblioteca de clases inicial, una subclase de las clases de base de Visual FoxPro lista para rellenarse con tablas y documentos nuevos o ya existentes.

El marco de trabajo le permite especificar si desea crear una aplicación completa o sólo un marco de trabajo de aplicación. Si elige crear una aplicación completa, puede incluir en ella una base de datos y formularios o informes ya creados. O bien puede crear una nueva aplicación desde cero con una plantilla de base de datos. Si elige crear un marco de trabajo, podrá volver atrás posteriormente y agregarle componentes.

## Crear un marco de trabajo

Puede crear un marco de trabajo de aplicación con el [Asistente para aplicaciones](#) o con el elemento Nueva aplicación de la [Galería de componentes](#). Al utilizar la Galería de componentes, se agrega un nuevo elemento carpeta de proyecto a la carpeta Favoritos.

Independientemente del método que utilice, Visual FoxPro mostrará un [Generador de aplicaciones](#) en el que podrá agregar la información a almacenar en una metatabla.

### Para crear una aplicación

1. En el menú **Herramientas**, haga clic en **Asistentes**, y luego haga clic en **Aplicaciones**.  
  
- o bien -
  1. En la carpeta Catálogos de la **Galería de componentes**, haga doble clic en el elemento **Nueva aplicación**.
  2. En el cuadro de diálogo **Escriba el nombre del proyecto**,
    - Especifique el nombre del proyecto.
    - Acepte o busque el archivo de proyecto.
    - Elija las opciones para **Crear estructura de directorios del proyecto** (predeterminado) y **Agregar al catálogo Favoritos** (predeterminado)

Para obtener detalles sobre el contenido y el uso del marco de trabajo mejorado, y el Generador de aplicaciones, vea [Programar aplicaciones con el marco de trabajo de aplicaciones](#) en la Ayuda.

Puede utilizar también la [Galería de componentes](#) para agregar formularios, informes, datos y objetos de servicio al marco de trabajo de la nueva aplicación, y para agregar controles a los formularios.

Al utilizar la Galería de componentes para agregar un formulario a una aplicación, puede crear un nuevo formulario o crear una subclase a partir de una clase existente.

# Capítulo 33: Mejoras para la programación

Microsoft Visual FoxPro incluye ahora nuevas características de programación para aumentar la productividad del programador. Entre dichas características se incluyen los métodos Access y Assign (que permiten ejecutar código cuando se consulta o se modifica el valor de una propiedad), la compatibilidad con más formatos de archivos gráficos y nuevos elementos del lenguaje para simplificar la programación. Además, se han incluido en Visual FoxPro muchas de las funciones de manipulación de nombres de archivo disponibles en Foxtools.fll, una biblioteca API de Visual FoxPro.

En este capítulo se trata:

- [Métodos Access y Assign](#)
- [Compatibilidad con gráficos GIF y JPEG](#)
- [Elementos del lenguaje nuevos y mejorados](#)
- [Compatibilidad con el milenio](#)

## Métodos Access y Assign

Se ha mejorado Visual FoxPro para que admita los métodos Access y Assign. Estos métodos definidos por el usuario permiten ejecutar código cuando se consulta el valor de una propiedad o cuando se intenta modificar el valor de una propiedad.

El código del método Access se ejecuta cuando se consulta el valor de una propiedad, normalmente con la propiedad en una referencia de objeto, al almacenar el valor de la propiedad en una variable o al mostrar el valor de la propiedad con un signo de interrogación (?).

El código del método Assign se ejecuta cuando se intenta modificar el valor de una propiedad, normalmente mediante los comandos STORE o = para asignar un nuevo valor a la propiedad.

Los métodos Access y Assign sólo se ejecutan cuando se consultan o modifican los valores de las propiedades en tiempo de ejecución. La consulta o modificación de los valores de las propiedades en tiempo de diseño no hace que se ejecuten los métodos Access y Assign.

**Nota** Como el valor que intenta asignar a la propiedad se pasa al método Assign, debe incluir una instrucción PARAMETERS o LPARAMETERS en el método Assign para aceptar el valor.

Puede crear independientemente los métodos Access y Assign (puede crear un método Access sin un método Assign o un método Assign sin un método Access).

Puede crear métodos Access y Assign para propiedades creadas mediante programación en una instrucción DEFINE CLASS o de forma interactiva para un formulario o una clase con el Diseñador de formularios y el Diseñador de clases.

**Nota** También se pueden crear métodos Access y Assign para todas las propiedades nativas de Visual FoxPro. Por ejemplo, puede crear un método Access para la propiedad Left de un formulario,

lo que le permitirá ejecutar código siempre que se consulte la propiedad Left del formulario. Puede crear un método Assign para una propiedad nativa de sólo lectura de Visual FoxPro (por ejemplo, la propiedad ParentClass), pero el método nunca se ejecutará.

## Ventajas de los métodos Access y Assign

Los métodos Access y Assign proporcionan las ventajas siguientes:

- Puede crear una interfaz pública para una clase o un objeto que separe la interfaz de la implementación.
- Puede implementar fácilmente la validación de las propiedades.
- Puede proteger fácilmente las propiedades en controles ActiveX que derivan de clases.

## Crear métodos Access y Assign

Las mejoras del comando DEFINE CLASS y de los Diseñadores de formularios y de clases le permiten crear métodos Access y Assign mediante programación y de forma interactiva.

## Nuevos sufijos para DEFINE CLASS

Se han agregado dos sufijos, \_ACCESS y \_ASSIGN, al comando DEFINE CLASS para crear métodos Access y Assign. Si anexa una de estas palabras clave al nombre de una función o un procedimiento, se creará un método Access o Assign para una propiedad que tenga el mismo nombre que la función o el procedimiento.

Por ejemplo, el siguiente ejemplo de código utiliza DEFINE CLASS para crear una clase personalizada llamada MiClase. Se crea una propiedad definida por el usuario, MiPropiedad, para la clase. A continuación se crea un método Access para MiPropiedad con la instrucción PROCEDURE.

Cuando se consulta el valor de la propiedad, se ejecuta el código del procedimiento (WAIT WINDOW 'Éste es el método Access'). También se crea un método Assign para MiPropiedad, de nuevo con una instrucción PROCEDURE. Cuando se intente modificar el valor de la propiedad, se ejecutará el código del procedimiento (WAIT WINDOW 'Éste es el método Assign').

Observe el uso de la instrucción LPARAMETERS para aceptar el valor pasado al método Assign. Este ejemplo también muestra cómo puede crear propiedades de sólo lectura.

```
DEFINE CLASS MiClase AS Custom
    MiPropiedad = 100 && Propiedad definida por el usuario

    PROCEDURE MiPropiedad_ACCESS && Método Access
        WAIT WINDOW 'Éste es el método Access';
        + ' ' + PROGRAM( )
        RETURN THIS.MiPropiedad
    ENDPROC

    PROCEDURE MiPropiedad_ASSIGN && Método Assign
        LPARAMETERS tAssign && Necesario para aceptar el valor
        WAIT WINDOW 'Éste es el método Assign';
        + ' ' + PROGRAM( )
    ENDPROC
ENDDEFINE
```

El ejemplo siguiente muestra cómo puede agregar un método Assign a una propiedad nativa de Visual FoxPro y realizar una sencilla validación del valor de la propiedad que intenta establecer. Observe que en este ejemplo se crea un método Assign sin un método Access correspondiente.

Se usa DEFINE CLASS para crear una clase Form llamada frmMiForm. Se crea un método Assign llamado Left\_ASSIGN con una instrucción PROCEDURE. El código del método Assign se ejecuta siempre que se intente asignar un valor a la propiedad Left del formulario.

Si intenta asignar un valor negativo a la propiedad Left, se muestra un mensaje y no se modifica el valor de la propiedad Left. Si intenta asignar un valor no negativo a la propiedad Left, la propiedad Left del formulario queda establecida a dicho valor.

```
DEFINE CLASS frmMiForm AS Form

    PROCEDURE Left_ASSIGN && Método Assign
        LPARAMETERS tAssign && Necesario para aceptar el valor

        DO CASE
            CASE tAssign < 0 && valor de Left negativo
                WAIT WINDOW 'El valor tiene que ser mayor que 0'
            OTHERWISE && valor de Left no negativo
                THIS.Left = tAssign
        ENDCASE
    ENDPROC
ENDDEFINE
```

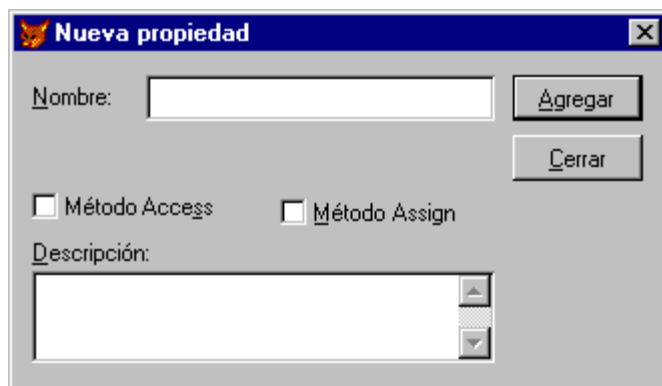
Para obtener más información sobre la sintaxis utilizada para crear métodos Access y Assign, vea [DEFINE CLASS](#).

## Los Diseñadores de clases y de formularios

### Para crear un método Access o Assign en el Diseñador de formularios

1. Elija **Nueva propiedad** en el menú **Formulario**.

Se muestra el cuadro de diálogo **Nueva propiedad**.



2. Escriba el nombre de la propiedad que va a crear en el cuadro de texto **Nombre** y, a continuación, seleccione la casilla de verificación **Método Access** o la casilla de verificación

**Método Assign** (o ambas).

3. Elija **Agregar** para crear la propiedad del formulario y para crear los métodos Access y Assign de la propiedad.

### **Para crear un método Access o Assign para una propiedad intrínseca de Visual FoxPro en el Diseñador de formularios**

1. Elija **Nuevo método** en el menú **Formulario**.

Se muestra el cuadro de diálogo **Nuevo método**.

2. Escriba el nombre de la propiedad intrínseca seguido de `_Access` o `_Assign` en el cuadro de texto **Nombre**. Por ejemplo, para crear un método Access para la propiedad Left, escriba `Left_Access` en el cuadro de texto **Nombre**.
3. Elija **Agregar** para crear métodos Access o Assign para la propiedad intrínseca.

**Nota** En el Diseñador de formularios sólo puede crear propiedades con métodos Access y Assign para un formulario o un conjunto de formularios. Para crear propiedades con métodos Access y Assign para un control o un objeto, utilice el Diseñador de clases para crear la clase de control o de objeto. En el Diseñador de clases, agregue propiedades con métodos Access y Assign al control o al objeto y después agregue la clase de control u objeto al formulario en el Diseñador de formularios.

### **Para crear un método Access o Assign para una clase en el Diseñador de clases**

1. Elija **Nueva propiedad** en el menú **Clase**.

Se muestra el cuadro de diálogo **Nueva propiedad**.

2. Escriba el nombre de la propiedad que va a crear en el cuadro de texto **Nombre** y, a continuación, seleccione la casilla de verificación **Método Access** o la casilla de verificación **Método Assign** (o ambas).
3. Elija **Agregar** para crear una propiedad para la clase y crear los métodos Access o Assign de la propiedad.

Para obtener más información acerca de la creación de métodos Access y Assign, vea el tema [Nueva propiedad \(Nueva propiedad\)](#).

### **Método THIS\_ACCESS**

Se ha agregado a Visual FoxPro 6.0 un nuevo método global de clase, `THIS_ACCESS`. El código de un método `THIS_ACCESS` se ejecuta siempre que se intente modificar el valor de un miembro de un objeto o siempre que se intente consultar un miembro de un objeto.

El método `THIS_ACCESS` se crea en el código en un comando `DEFINE CLASS` o en los cuadros de diálogo **Nuevo método** o **Modificar propiedades** de las bibliotecas de clases visuales `.vcx`. Un método `THIS_ACCESS` siempre debe devolver una referencia de objeto; si no es así, se generará un error.

Normalmente se devuelve la referencia de objeto THIS. El método THIS\_ACCESS también debe incluir un parámetro para aceptar el nombre del miembro del objeto que se modifica o consulta.

El siguiente ejemplo muestra cómo crear un método THIS\_ACCESS en el código de un comando DEFINE CLASS. Cuando este ejemplo se ejecuta como programa, 'Caption' se muestra dos veces, la primera cuando se le asigna un valor a la propiedad Caption y la segunda cuando se consulta el valor de la propiedad Caption. Después se muestra el valor de la propiedad Caption ('abc').

```
CLEAR
oTempObj = CREATEOBJECT('MiForm')  && Crea una instancia del formulario
oTempObj.Caption = 'abc'  && Asigna un valor y desencadena THIS_ACCESS
? oTempObj.Caption  && Consulta un valor y desencadena THIS_ACCESS

DEFINE CLASS MiForm AS Form
    PROCEDURE THIS_ACCESS
        LPARAMETER cMemberName  && Nombre del miembro del objeto

        IF cMemberName = 'caption'
            ? cMemberName  && Muestra el nombre del miembro del objeto
        ENDIF
        RETURN THIS
    ENDPROC
ENDDEFINE
```

Observe que THIS\_ACCESS no pretende ser un sustituto global de los métodos Access y Assign (sólo proporciona información acerca del miembro del objeto al que se tiene acceso o se consulta). A diferencia de los métodos Access y Assign, THIS\_ACCESS no proporciona control sobre los valores devueltos a miembros de objeto específicos.

## Notas de programación de Access y Assign

Las secciones siguientes describen la información de programación para métodos Access y Assign.

### Alcance

Los métodos Access y Assign están protegidos de forma predeterminada (no puede tener acceso a un método Access o Assign ni modificarlo desde fuera de la clase en la que se cree el método Access o Assign).

Incluya la palabra clave **HIDDEN** cuando cree un método Access o Assign para impedir el acceso y las modificaciones a las propiedades desde fuera de la definición de clase. Sólo los métodos y los eventos de la definición de la clase pueden tener acceso a las propiedades ocultas. Mientras que las subclases de la definición de clase pueden tener acceso a las propiedades protegidas, sólo la definición de la clase puede tener acceso a las propiedades ocultas.

**Nota** Si no incluye la palabra clave **HIDDEN**, puede crear subclases con los métodos Access y Assign.

### Depuración

Puede ver el código de los métodos Access y Assign en la ventana Seguimiento de la ventana Depurador. Sin embargo, los métodos Access y Assign no se pueden ejecutar en las ventanas



Inspección y Locales de la ventana Depurador.

### **Pasar matrices a los métodos Assign**

Se pasan matrices a los métodos Access y Assign de la misma forma en que se pasan a procedimientos estándar de Visual FoxPro.

Si ejecuta SET UDFPARMS TO REFERENCE o se antepone @ al nombre de la matriz, se pasa la matriz completa al método Access o Assign. Si ejecuta SET UDFPARMS TO VALUE o escribe el nombre de la matriz entre paréntesis, se pasa por valor el primer elemento de la matriz. Los elementos de matriz siempre se pasan por valor. Para obtener más información acerca de cómo pasar valores y matrices, vea [SET UDFPARMS](#).

### **Controles ActiveX**

Las propiedades, los eventos o los métodos nativos de los controles ActiveX no admiten las propiedades Access y Assign. Sin embargo, las propiedades, los eventos y los métodos del Contenedor OLE que contiene al control ActiveX sí admite los métodos Access y Assign.

### **Método ResetToDefault**

Si ejecuta el método ResetToDefault para un método Access o Assign se modifica el código del método Access o Assign al miniprograma predeterminado. El resultado es que el código heredado del método, si lo hubiera, no se ejecuta. La técnica utilizada para asegurar que el código heredado de la clase primaria se ejecuta varía según el tipo de método.

Coloque el código siguiente en la subclase de un método Access para ejecutar el código en la clase primaria:

```
RETURN DODEFAULT( )
```

Coloque el código siguiente en la subclase de un método Access para ejecutar el código de la clase primaria:

```
LPARAMETERS vnewval  
DODEFAULT(vnewval)  
THIS.<propiedad> = vnewval
```

Coloque el código siguiente en la subclase de un método THIS\_ACCESS para ejecutar el código de la clase primaria:

```
LPARAMETERS cmember  
RETURN DODEFAULT(cmember)
```

## **Compatibilidad con gráficos GIF y JPEG**

Se ha mejorado Visual FoxPro para que admita los formatos de archivos gráficos GIF (Graphics Interchange Format) y JPEG (Joint Photographic Electronic Group), ampliamente utilizados en Internet.

En general, las áreas que aceptaban el formato .bmp (mapa de bits) en versiones anteriores de Visual FoxPro aceptan ahora los formatos de archivos gráficos.

Formato gráfico	Extensión de archivo
Mapa de bits	.bmp
Device Independent Bitmap	.dib
Graphics Interchange Format	.gif
Joint Photographic Electronic Group	.jpg
Cursor	.cur
Cursor animado	.ani
Icono	.ico

**Nota** En Visual FoxPro se pueden usar los archivos de cursores, cursores animados e iconos como archivos gráficos. Por ejemplo, puede especificar un cursor animado para la propiedad Picture del control Imagen (sin embargo, el control Imagen muestra la representación estática del cursor).

Visual FoxPro proporciona la compatibilidad con gráficos en tres áreas: lenguaje, controles y objetos, y la interfaz.

## El lenguaje de Visual FoxPro

Se han mejorado los siguientes comandos y funciones para admitir los nuevos formatos de archivos gráficos.

### GETPICT()

Se ha mejorado el cuadro de diálogo **Abrir imagen** que se muestra al ejecutar la [función GETPICT\(\)](#) en la ventana **Comandos**, lo que le permitirá encontrar rápidamente todos los archivos gráficos admitidos por Visual FoxPro. Ahora el cuadro de lista desplegable **Archivos de tipo** incluye los siguientes elementos.

Elemento	Especificaciones de archivo
Todos los archivos	*.*
Todos los archivos gráficos	*.bmp, *.dib, *.jpg, *.gif, *.ani, *.cur, *.ico
Mapa de bits	*.bmp, *.dib
Cursor	*.cur
Cursor animado	*.ani

Icono	*.ico
JPEG	*.jpg
GIF	*.gif

Active la casilla de verificación **Vista previa** para mostrar el archivo gráfico seleccionado actualmente. En versiones anteriores de Visual FoxPro era necesario elegir el botón **Vista previa** cada vez que se seleccionaba un nuevo archivo gráfico. También se ha aumentado el tamaño del área **Imagen**.

## CLEAR RESOURCES

Ahora el [comando CLEAR RESOURCES](#) de Visual FoxPro borra todos los archivos gráficos almacenados localmente, como los archivos .gif y .jpg.

## Controles y objetos de Visual FoxPro

La tabla siguiente muestra los controles y objetos de Visual FoxPro que tienen propiedades para las que puede especificar archivos gráficos. Ahora puede especificar archivos gráficos de tipo .gif, .jpg, cursores, cursores animados e iconos para estas propiedades, además de los archivos gráficos de tipo .bmp y .dib admitidos en las versiones anteriores de Visual FoxPro.

Control u objeto	Propiedades
Control CheckBox	DisabledPicture DownPicture Picture
Contenedor de objetos CommandButton	DisabledPicture DownPicture Picture
Objeto Container	Picture
Objeto Control	Picture
Objeto Custom	Picture
Objeto Form	Picture
Control Image	Picture
Control OptionButton	DisabledPicture DownPicture Picture
Objeto Page	Picture
Objeto _Screen	Picture

## La interfaz de Visual FoxPro

Varios de los diseñadores de Visual FoxPro le permiten especificar archivos gráficos con el cuadro de diálogo **Abrir**. Los cuadros de diálogo **Abrir** de los siguientes diseñadores se han mejorado para incluir los nuevos formatos de archivos gráficos.

### Diseñador de formularios y Diseñador de clases

En la ventana **Propiedades**, si hace doble clic en la propiedad o elige el botón del cuadro de diálogo de la propiedad, puede mostrar el cuadro de diálogo **Abrir** para las propiedades que admiten archivos gráficos.

### Administrador de proyectos

Puede agregar archivos gráficos a un proyecto desde las fichas **Todos** y **Otros** del Administrador de proyectos. Cuando está seleccionada la ficha **Todos** o la ficha **Otros**, seleccione el elemento **Otros archivos** y, a continuación, elija **Agregar**. Se muestra el cuadro de diálogo **Abrir**, que le permitirá agregar un archivo de gráficos al proyecto.

### Diseñador de informes

La barra de herramientas **Controles de informes** contiene el botón **Imagen/Control ActiveX dependiente**. Haga clic en dicho botón y arrastre el cursor sobre una banda del Diseñador de informes para mostrar el cuadro de diálogo **Imagen para informe**. Para mostrar el cuadro de diálogo **Abrir**, elija el botón **Archivo**.

## Elementos del lenguaje nuevos y mejorados

Se han agregado muchos elementos nuevos al lenguaje de Visual FoxPro y se han mejorado otros. Los nuevos elementos del lenguaje mostrados en esta sección incluyen [documentos activos](#), [enganches del Administrador de proyectos](#), [arrastrar y colocar de OLE](#), [mejoras en el servidor](#) y [otros elementos nuevos](#).

También se muestran los elementos [mejorados](#).

Además, se han agregado a Visual FoxPro muchas de las funciones de manipulación de nombres de archivo disponibles en Foxtools.fll, una biblioteca API de Visual FoxPro. Ya no es necesario utilizar SET LIBRARY TO FOXTOOLS.FLL para llamar a estas funciones de [Foxtools](#); puede llamarlas directamente desde sus programas de Visual FoxPro.

En esta sección también se describen las mejoras de rendimiento, solidez y facilidad de uso de Visual FoxPro.

## Nuevos elementos del lenguaje para documentos activos Descripción

<a href="#"><u>Objeto ActiveDoc</u></a>	Crea un documento activo que se puede alojar en un contenedor de documentos activos como Microsoft Internet Explorer.
<a href="#"><u>Propiedad AlwaysOnBottom</u></a>	Evita que una ventana de formulario cubra otras ventanas.
<a href="#"><u>Evento CommandTargetExec</u></a>	Se produce cuando el usuario hace clic en un elemento de menú o en un elemento de una barra de herramientas que pertenece al contenedor de documento activo.
<a href="#"><u>Evento CommandTargetQuery</u></a>	Se produce cuando el contenedor de documentos activos tiene que saber si el documento activo admite varios comandos de menú o de barra de herramientas del contenedor de forma que pueda habilitar o deshabilitar los correspondientes elementos de menú o botones de la barra de herramientas.
<a href="#"><u>Evento ContainerRelease</u></a>	Se produce cuando un contenedor libera un documento activo.
<a href="#"><u>Propiedad ContainerReleaseType</u></a>	Especifica si se abre un documento activo en el entorno de tiempo de ejecución de Visual FoxPro cuando lo libera su contenedor.
<a href="#"><u>Propiedad ContinuousScroll</u></a>	Especifica si el desplazamiento en un formulario es continuo o si el desplazamiento sólo tiene lugar cuando se libera un cuadro de desplazamiento.
<a href="#"><u>Comando DEFINE PAD</u></a>	Admite nuevas opciones NEGOTIATE para especificar la ubicación del título del menú en documentos activos.
<a href="#"><u>Función GETHOST( )</u></a>	Devuelve una referencia de objeto al contenedor de un documento activo.
<a href="#"><u>Método GoBack</u></a>	Explora hacia atrás en la lista del historial de un contenedor de documentos activos.
<a href="#"><u>Método GoFoward</u></a>	Explora hacia adelante en la lista del historial de un contenedor de documentos activos.
<a href="#"><u>Evento HideDoc</u></a>	Se produce cuando se explora desde un documento activo.
<a href="#"><u>Propiedad HscrollSmallChange</u></a>	Especifica la cantidad que se desplaza un formulario en la dirección horizontal cuando hace clic en una flecha de desplazamiento horizontal.
<a href="#"><u>Objeto Hyperlink</u></a>	Crea un objeto Hyperlink.
<a href="#"><u>Función ISHOSTED( )</u></a>	Devuelve un valor de tipo Logical que indica si un

	documento activo está alojado en un contenedor de documentos activos.
<a href="#">Método NavigateTo</a>	Explora en un contenedor de documentos activos a una ubicación especificada.
<a href="#">Evento Run</a>	Se produce cuando un documento activo termina de coordinarse con su contenedor y con COM y está listo para ejecutar el código de usuario.
<a href="#">Variable del sistema RUNACTIVEDOC</a>	Especifica la aplicación que inicia un documento activo.
<a href="#">Propiedad ScrollBars</a>	Disponible ahora para formularios. Si un formulario está en un documento activo, las barras de desplazamiento se muestran automáticamente cuando el tamaño del contenedor de documentos activos sea menor que el tamaño del formulario.
<a href="#">Evento Scrolled</a>	Disponible ahora para formularios. Le permite determinar si se hace clic en las barras de desplazamiento horizontal o vertical o si se mueve una barra de desplazamiento.
<a href="#">Método SetViewport</a>	Establece los valores de las propiedades ViewPortLeft y ViewPortTop de un formulario.
<a href="#">Evento ShowDoc</a>	Se produce al explorar hasta un documento activo.
<a href="#">SYS(4204) – Depuración de documentos activos</a>	Activa o desactiva el soporte para la depuración de documentos activos en el depurador de Visual FoxPro.
<a href="#">Propiedad ViewPortHeight</a>	Contiene el alto de la vista de un formulario.
<a href="#">Propiedad ViewPortLeft</a>	Contiene la coordenada izquierda del formulario visible en la vista.
<a href="#">Propiedad ViewPortTop</a>	Contiene la coordenada superior del formulario visible en la vista.
<a href="#">Propiedad ViewPortWidth</a>	Contiene el ancho de la vista de un formulario.
<a href="#">Propiedad VscrollSmallChange</a>	Especifica la cantidad de desplazamiento vertical de un formulario cuando hace clic en una flecha de desplazamiento.

<b>Nuevos elementos del lenguaje para enganches del Administrador de proyectos</b>	<b>Descripción</b>
--	--------------------

<a href="#">Propiedad ActiveProject</a>	Contiene una referencia de objeto al objeto Project de la ventana del Administrador de proyectos.
<a href="#">Método Add</a>	Agrega un archivo a un proyecto.

<a href="#">Método AddToSCC</a>	Agrega un archivo de un proyecto al control de código fuente.
<a href="#">Evento AfterBuild</a>	Se produce después de que se vuelva a generar un proyecto o de que se cree en un proyecto un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe).
<a href="#">Propiedad AutoIncrement</a>	Especifica si la versión generada de un proyecto se incrementa automáticamente cada vez que se genere un archivo .exe o .dll en proceso distribuible.
<a href="#">Evento BeforeBuild</a>	Se produce antes de que se vuelva a generar un proyecto o de que se cree en un proyecto un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe).
<a href="#">Método Build</a>	Vuelve a generar un proyecto o crea un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) a partir de un proyecto.
<a href="#">Propiedad BuildDateTime</a>	Contiene la fecha y la hora en que se generó por última vez un proyecto.
<a href="#">Método CheckIn</a>	Protege las modificaciones realizadas a un archivo de proyecto bajo control de código fuente.
<a href="#">Método CheckOut</a>	Desprotege un archivo que está bajo control de código fuente y le permite hacer modificaciones en el archivo.
<a href="#">Método CleanUp</a>	Limpia una tabla del proyecto; para ello, quita los registros marcados para eliminación y empaqueta los campos memo.
<a href="#">Método Close</a>	Cierra un proyecto y libera el objeto ProjectHook del proyecto y los demás objetos del proyecto.
<a href="#">Propiedad CLSID</a>	Contiene el CLSID registrado (Identificador de clase) para un servidor de un proyecto.
<a href="#">Propiedad CodePage</a>	Contiene la página de códigos de un archivo de un proyecto.
<a href="#">Propiedad Count</a>	La cuenta del número de objetos proyecto, archivo o servidor de una colección de proyectos, archivos o servidores.
<a href="#">Comando CREATE PROJECT</a>	Mejorado en Visual FoxPro 6.0. Admite dos nuevas opciones, NOSHOW y NOPROJECTHOOK, para utilizarlas con los nuevos enganches del Administrador de proyectos.
<a href="#">Propiedad Debug</a>	Especifica si la información de depuración se incluye en el

	código fuente compilado de un proyecto.
<a href="#">Propiedad Description</a>	En un objeto archivo es la descripción del archivo. En un objeto servidor es la descripción de la clase del servidor.
<a href="#">Propiedad Encrypted</a>	Especifica si se va a cifrar el código fuente compilado de un proyecto.
<a href="#">Propiedad Exclude</a>	Especifica si se va a excluir un archivo de una aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) cuando se genere a partir de un proyecto.
<a href="#">Propiedad FileClass</a>	Contiene el nombre de la clase de formulario en la que se basa un formulario de un proyecto.
<a href="#">Propiedad FileClassLibrary</a>	Contiene el nombre de la biblioteca de clases que contiene la clase en la que se basa un formulario de un proyecto.
<a href="#">Objeto File</a>	Proporciona referencias a archivos específicos de un proyecto.
<a href="#">Colección Files</a>	Una colección de objetos archivo.
<a href="#">Método GetLatestVersion</a>	Obtiene la versión más reciente de un archivo de un proyecto que está bajo control de código fuente y copia una versión de sólo lectura en su unidad local.
<a href="#">Propiedad HomeDir</a>	Especifica el directorio de inicio de un proyecto.
<a href="#">Propiedad Instancing</a>	Especifica cómo se puede crear una instancia de un servidor de un proyecto.
<a href="#">Método Item</a>	Devuelve una referencia de objeto al elemento especificado en una colección de proyectos.
<a href="#">Propiedad LastModified</a>	Contiene la fecha y la hora de la última modificación realizada a un archivo de un proyecto.
<a href="#">Propiedad MainClass</a>	Contiene el nombre de una clase ActiveDoc establecida como programa principal de un proyecto.
<a href="#">Propiedad MainFile</a>	Contiene el nombre y la ruta del archivo establecido como programa principal de un proyecto.
<a href="#">Método Modify</a>	Abre un archivo de un proyecto para su modificación en el editor o diseñador apropiado.
<a href="#">Comando MODIFY PROJECT</a>	Mejorado en Visual FoxPro 6.0. Admite dos nuevas opciones, NOSHOW y NOPROJECTHOOK, para su utilización con los nuevos enganches del Administrador de proyectos.
<a href="#">Propiedad ProgID</a>	Contiene el PROGID registrado (Identificador programático) de un servidor de un proyecto.



<a href="#"><u>Objeto Project</u></a>	Se crea una instancia cuando se crea o se abre un proyecto.
<a href="#"><u>Objeto ProjectHook</u></a>	Se crea una instancia cuando se abre un proyecto y proporciona acceso mediante programación a eventos de proyecto.
<a href="#"><u>Propiedad ProjectHook</u></a>	Una referencia de objeto al objeto ProjectHook que se crea para un proyecto.
<a href="#"><u>Propiedad ProjectHookClass</u></a>	La clase ProjectHook predeterminada de un proyecto.
<a href="#"><u>Propiedad ProjectHookLibrary</u></a>	La biblioteca de clases visuales .vcx que contiene la clase ProjectHook predeterminada de un proyecto.
<a href="#"><u>Colección Projects</u></a>	Una colección de objetos proyecto.
<a href="#"><u>Evento QueryAddFile</u></a>	Se produce justo antes de que se agregue un archivo a un proyecto.
<a href="#"><u>Evento QueryModifyFile</u></a>	Se produce justo antes de que se modifique un archivo en un proyecto.
<a href="#"><u>Evento QueryRemoveFile</u></a>	Se produce justo antes de que se elimine un archivo de un proyecto.
<a href="#"><u>Evento QueryRunFile</u></a>	Se produce justo antes de que se ejecute un archivo o de que se realice una vista previa de un informe o una etiqueta en un proyecto.
<a href="#"><u>Método Remove</u></a>	Quita un archivo de su colección de archivos y del proyecto.
<a href="#"><u>Método RemoveFromSCC</u></a>	Elimina un archivo del proyecto del control de código fuente.
<a href="#"><u>Método Run</u></a>	Ejecuta o muestra una vista previa de un archivo de un proyecto.
<a href="#"><u>Propiedad SCCProvider</u></a>	El nombre del proveedor de control de código fuente para un proyecto.
<a href="#"><u>Propiedad SCCStatus</u></a>	Contiene un valor numérico que indica el estado de control de código fuente de un archivo de un proyecto.
<a href="#"><u>Objeto Server</u></a>	Una referencia de objeto a un servidor del proyecto.
<a href="#"><u>Colección Servers</u></a>	Una colección de objetos servidor.
<a href="#"><u>Propiedad ServerClass</u></a>	Contiene el nombre de una clase servidor de un proyecto.
<a href="#"><u>Propiedad ServerClassLibrary</u></a>	Contiene el nombre de la biblioteca de clases o de programa que contiene una clase de servidor.
<a href="#"><u>Propiedad ServerHelpFile</u></a>	El archivo de Ayuda de la biblioteca de tipos creada para las clases de servidor de un proyecto.

<a href="#">Propiedad ServerProject</a>	El nombre del proyecto que contiene las clases de servidor.
<a href="#">Método SetMain</a>	Establece el archivo principal de un proyecto.
<a href="#">Propiedad Type</a>	El tipo de archivo de un archivo del proyecto.
<a href="#">Propiedad TypeLibCLSID</a>	El CLSID de registro (Identificador de clase) para una biblioteca de tipos creada para las clases de servidor de un proyecto.
<a href="#">Propiedad TypeLibDesc</a>	La descripción de una biblioteca de tipos creada para las clases de servidor de un proyecto.
<a href="#">Propiedad TypeLibName</a>	El nombre de una biblioteca de tipos creada para las clases de servidor de un proyecto.
<a href="#">Método UndoCheckOut</a>	Descarta las modificaciones realizadas en un archivo y vuelve a protegerlo.
<a href="#">Propiedad VersionComments</a>	Los comentarios del proyecto.
<a href="#">Propiedad VersionCompany</a>	El nombre de la compañía del proyecto.
<a href="#">Propiedad VersionCopyright</a>	La información de copyright del proyecto.
<a href="#">Propiedad VersionDescription</a>	La descripción del proyecto.
<a href="#">Propiedad VersionLanguage</a>	La información de idioma del proyecto.
<a href="#">Propiedad VersionNumber</a>	El número de versión del proyecto.
<a href="#">Propiedad VersionProduct</a>	El nombre del producto del proyecto.
<a href="#">Propiedad VersionTrademarks</a>	La información de marcas registradas del proyecto.

#### **Nuevos elementos del lenguaje para arrastrar y colocar de OLE**

<a href="#">Método ClearData</a>	Borra todos los datos y los formatos de datos del objeto DataObject con arrastrar y colocar de OLE.
<a href="#">Objeto DataObject</a>	Contenedor para los datos que se transfieren desde un origen de arrastre OLE a un destino para colocar OLE.
<a href="#">Método GetData</a>	Obtiene datos desde el objeto DataObject con arrastrar y colocar de OLE.
<a href="#">Método GetFormat</a>	Determina si los datos de un formato especificado están disponibles en el objeto DataObject con arrastrar y colocar de OLE.
<a href="#">Evento OLECompleteDrag</a>	Se produce cuando se colocan datos en el destino o cuando

	se cancela la operación arrastrar y colocar de OLE.
<a href="#">Método OLEDrag</a>	Inicia una operación arrastrar y colocar de OLE.
<a href="#">Evento OLEDragDrop</a>	Se produce cuando se colocan datos en el destino y la propiedad OLEDropMode del destino es 1 - Activado.
<a href="#">Propiedad OLEDragMode</a>	Especifica cómo se inicia una operación de arrastre.
<a href="#">Evento OLEDragOver</a>	Se produce cuando se arrastran datos a su destino y la propiedad OLEDropMode del destino es 1 – Activado.
<a href="#">Propiedad OLEDragPicture</a>	Especifica la imagen que se muestra bajo el puntero del <i>mouse</i> durante una operación arrastrar y colocar de OLE. Puede especificar un archivo de imagen del tipo .bmp, .dib, .jpg, .gif, .ani, .cur e .ico.
<a href="#">Propiedad OLEDropEffects</a>	Especifica el tipo de operaciones arrastrar y colocar de OLE que admite un destino.
<a href="#">Propiedad OLEDropHasData</a>	Especifica cómo se controla una operación colocar.
<a href="#">Propiedad OLEDropMode</a>	Especifica cómo controla un destino de colocar las operaciones colocar de OLE.
<a href="#">Propiedad OLEDropTextInsertion</a>	Especifica si puede colocar texto en medio de una palabra en la parte de cuadro de texto de un control.
<a href="#">Evento OLEGiveFeedBack</a>	Se produce después de cada evento OLEDragOver. Permite que el origen de arrastre especifique el tipo de operación arrastrar y colocar de OLE y la información visual.
<a href="#">Evento OLESetData</a>	Se produce en un origen de arrastre cuando un destino para colocar llama al método GetData y no hay datos con el formato especificado en el objeto DataObject con arrastrar y colocar de OLE.
<a href="#">Evento OLEStartDrag</a>	Se produce cuando se llama al método OLEDrag.
<a href="#">Método SetData</a>	Coloca datos en el objeto DataObject con arrastrar y colocar de OLE.
<a href="#">Método SetFormat</a>	Coloca un formato de datos en el objeto DataObject con arrastrar y colocar de OLE.

## Nuevos elementos del lenguaje para mejoras de servidor

<a href="#">Función COMARRAY()</a>	Especifica cómo se pasan matrices a objetos COM.
<a href="#">Función COMCLASSINFO()</a>	Devuelve información de registro acerca de un objeto

	COM como un servidor de automatización de Visual FoxPro.
<a href="#">Función COMRETURNERROR()</a>	Llena la estructura de excepciones de COM con información que los clientes de COM pueden utilizar para determinar el origen de los errores de Automatización.
<a href="#">Función CREATEOBJECTEX()</a>	Crea una instancia de un objeto COM registrado (como un servidor de automatización de Visual FoxPro) en un equipo remoto.
<a href="#">Propiedad ServerName</a>	Contiene la ruta completa y el nombre de archivo de un servidor de automatización.
<a href="#">Propiedad StartMode</a>	Contiene un valor numérico que indica cómo se inició la instancia de Visual FoxPro.
<a href="#">SYS(2335) – Modo de servidor desatendido</a>	Activa o desactiva los estados modales en los servidores de automatización .exe distribuibles de Visual FoxPro.
<a href="#">SYS(2334) – Modo de invocación de servidor de automatización</a>	Devuelve un valor que indica cómo se invocó el servidor de automatización de Visual FoxPro o si se está ejecutando una aplicación independiente (.exe).
<b>Otros elementos nuevos del lenguaje Descripción</b>	
<a href="#">Método AddProperty</a>	Agrega una nueva propiedad a un objeto.
<a href="#">Función AGETFILEVERSION()</a>	Crea una matriz que contiene información acerca de archivos con recursos de versión de Microsoft Windows como los archivos .exe, .dll y .fl, o de servidores de automatización creados en Visual FoxPro. Corresponde a la función GetFileVersion( ) de Foxtools.
<a href="#">Función AGETCLASS()</a>	Muestra bibliotecas de clases en el cuadro de diálogo Abrir y crea una matriz que contiene el nombre de la biblioteca de clases y la clase elegidas.
<a href="#">Función ALINES()</a>	Copia cada línea de una expresión de cadena de caracteres o un campo memo en una fila correspondiente de una matriz.
<a href="#">Función AMOUSEOBJ()</a>	Devuelve los datos de posición del puntero del <i>mouse</i> y referencias de objeto para el objeto y el contenedor del objeto sobre los que se encuentra el puntero del <i>mouse</i> .
<a href="#">Función ANETRESOURCES()</a>	Coloca los nombres de los recursos compartidos de red e impresoras en una matriz y después devuelve el número de recursos.
<a href="#">Función AVCXCLASSES()</a>	Coloca la información acerca de las clases de una biblioteca de clases en una matriz.

<a href="#"><u>Propiedad DisplayCount</u></a>	Especifica el número de elementos mostrados en la parte de lista de un control ComboBox.
<a href="#"><u>Función FILETOSTR()</u></a>	Devuelve el contenido de un archivo como una cadena de caracteres.
<a href="#"><u>Variable del sistema _GALLERY</u></a>	Especifica el programa que se ejecuta cuando elige Galería de componentes en el menú Herramientas.
<a href="#"><u>Variable del sistema _GENHTML</u></a>	Especifica un programa de generación de HTML (Lenguaje de marcado de hipertexto) que crea un archivo de texto que contiene la versión HTML de un formulario, menú, informe o tabla.
<a href="#"><u>Variable del sistema _GETEXPR</u></a>	Especifica el programa que se ejecuta cuando se ejecute el comando GETEXPR o cuando se muestre el cuadro de diálogo Generador de expresiones.
<a href="#"><u>Método GridHitTest</u></a>	Devuelve, como parámetros de resultados, los componentes de un control cuadrícula correspondientes a las coordenadas horizontal (X) y vertical (Y) especificadas.
<a href="#"><u>Variable del sistema _INCLUDE</u></a>	Especifica un archivo de encabezado predeterminado incluido en las clases, formularios o conjuntos de formularios definidos por el usuario.
<a href="#"><u>Función INDEXSEEK()</u></a>	Sin mover el puntero de los registros, busca en una tabla indexada la primera ocurrencia de un registro cuya clave de índice coincida con una expresión especificada.
<a href="#"><u>Función NEWOBJECT()</u></a>	Crea una nueva clase u objeto directamente desde una biblioteca de clases visuales .vcx o desde un programa.
<a href="#"><u>Método NewObject</u></a>	Agrega una nueva clase u objeto a un objeto directamente desde una biblioteca de clases visuales .vcx o desde un programa.
<a href="#"><u>Variable del sistema _SAMPLES</u></a>	Contiene la ruta del directorio en el que se instalan los ejemplos de Visual FoxPro.
<a href="#"><u>Comando SET BROWSEIME</u></a>	Especifica si se abre el Editor de métodos de entrada cuando se llega a un cuadro de texto en una ventana Examinar.
<a href="#"><u>Comando SET STRICTDATE</u></a>	Especifica si constantes ambiguas de tipo Date y DateTime generan errores.
<a href="#"><u>Función STRTOFILE()</u></a>	Escribe el contenido de una cadena de caracteres en un archivo.
<a href="#"><u>SYS(3055) – Complejidad de las cláusulas FOR y WHERE</u></a>	Establece el nivel de complejidad de las cláusulas FOR y WHERE en los comandos y las funciones que las admiten.

<a href="#">SYS(3056) – Leer la configuración del Registro</a>	Hace que Visual FoxPro vuelva a leer su configuración y la actualice con la configuración del Registro del sistema.
<a href="#">Propiedad TitleBar</a>	Especifica si se muestra una barra de título en la parte superior de un formulario.
<a href="#">Función VARTYPE()</a>	Devuelve el tipo de datos de una expresión.
Elementos del lenguaje mejorados	Descripción
<a href="#">Operador =</a>	Se puede utilizar en Visual FoxPro 6.0 para determinar si dos referencias de objeto hacen referencia al mismo objeto.
<a href="#">Comando ALTER TABLE – SQL</a>	Acepta una nueva cláusula FOR para las cláusulas ADD PRIMARY KEY y ADD FOREIGN KEY. FOR le permite crear índices principales y externos filtrados.
<a href="#">Comando APPEND FROM</a>	Admite una nueva opción XL8 para importar datos desde hojas de cálculo de Microsoft Excel 97, y una nueva opción CSV para importar datos desde archivos con valores separados por comas.
<a href="#">Propiedad Century</a>	Ahora el valor predeterminado es 1 – Activado. La parte de siglo de la fecha se muestra en un cuadro de texto para proporcionar compatibilidad con el milenio.
<a href="#">Control CheckBox</a>	Ahora admite la propiedad <a href="#">ReadOnly</a> .
<a href="#">Objeto Column</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .
<a href="#">Comando COMPILE DATABASE</a>	Ahora COMPILE DATABASE empaqueta los campos memo en el archivo memo .dct de la base de datos para eliminar el espacio no utilizado del archivo de memos.
<a href="#">Objeto Container</a>	Ahora admite la propiedad <a href="#">Tag</a> .
<a href="#">Objeto Control</a>	Ahora admite la propiedad <a href="#">Tag</a> .
<a href="#">Comando COPY TO</a>	Admite una nueva opción CSV para exportar datos como archivo de valores separados por comas.
<a href="#">Comando CREATE FORM</a>	Admite una nueva cláusula AS que le permite crear un nuevo formulario o conjunto de formularios a partir de un formulario o de un conjunto de formularios de una biblioteca de clases visuales .vcx.
<a href="#">Objeto Cursor</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> .
<a href="#">Objeto Custom</a>	Ahora admite la propiedad <a href="#">Tag</a> .

<a href="#">Objeto DataEnvironment</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> .
<a href="#">Función DATE()</a>	Ahora admite argumentos numéricos opcionales que le permiten crear valores de fechas compatibles con el milenio.
<a href="#">Función DATETIME()</a>	Ahora admite argumentos numéricos opcionales que le permiten crear valores de fechas compatibles con el milenio.
<a href="#">Comando DEFINE CLASS</a>	Admite nuevos métodos Access y Assign, que le permiten ejecutar código siempre que consulte una propiedad o que intente cambiar el valor de una propiedad.
<a href="#">Función FDATE()</a>	Ahora admite un argumento opcional que le permite determinar la hora de la última modificación de un archivo sin tener que utilizar funciones de manipulación de caracteres.
<a href="#">Objeto Form</a>	Ahora admite la propiedad <a href="#">Scrollbars</a> y el evento <a href="#">Scrolled</a> .
<a href="#">Objeto FormSet</a>	Ahora admite las propiedades <a href="#">Parent</a> y <a href="#">Tag</a> .
<a href="#">Función GETDIR()</a>	Se ha mejorado el cuadro de diálogo Seleccionar directorio para que pueda mostrar más información.
<a href="#">Función GETFILE()</a>	Admite una nueva opción <i>cTítuloBarraTítulo</i> que le permite especificar el título del cuadro de diálogo Abrir.
<a href="#">Función GETFONT()</a>	Le permite especificar la fuente, el tamaño y el estilo de la fuente seleccionados inicialmente cuando se muestra el cuadro de diálogo Fuente.
<a href="#">Objeto Header</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .
<a href="#">Función HOME()</a>	Ahora le permite determinar los directorios de los ejemplos, las herramientas, los gráficos y los directorios comunes de Visual FoxPro y Visual Studio.
<a href="#">Control Image</a>	Ahora admite la propiedad <a href="#">ToolTipText</a> .
<a href="#">Comando IMPORT</a>	Admite una nueva opción XL8 para importar datos desde una hoja de cálculo de Microsoft Excel 97.
<a href="#">Control Label</a>	Ahora admite la propiedad <a href="#">ToolTipText</a> .
<a href="#">Comando MODIFY MEMO</a>	Ahora la sintaxis con colores en las ventanas de edición de campos memo está desactivada en las aplicaciones de tiempo de ejecución distribuidas.
<a href="#">Función OS()</a>	Ahora admite una opción que le permite determinar si el



	sistema operativo acepta DBCS (juegos de caracteres de dos bytes).
<a href="#">Objeto Page</a>	Ahora acepta la propiedad <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .
<a href="#">Control PageFrame</a>	Ahora admite la propiedad <a href="#">Tag</a> .
<a href="#">Función PEMSTATUS( )</a>	PEMSTATUS( ) admite una nueva opción 6 para <i>nAtributo</i> que le permite determinar si se ha heredado de otro objeto o clase una propiedad, evento o método.
<a href="#">Función PROGRAM( )</a>	Ahora admite -1 como argumento y le permite determinar el nivel del programa actual.
<a href="#">Método Refresh</a>	Ahora le permite actualizar la presentación visual del Administrador de proyectos y admite un nuevo parámetro para actualizar el estado de control de código fuente de los archivos de un proyecto.
<a href="#">Objeto Relation</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> , los eventos <a href="#">Destroy</a> , <a href="#">Error</a> e <a href="#">Init</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> y <a href="#">WriteExpression</a> .
<a href="#">Comando REPORT</a>	Ahora admite una cláusula PREVIEW IN SCREEN, que le permite colocar una ventana de vista previa en la ventana principal de Visual FoxPro.
<a href="#">Objeto Separator</a>	Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> .
<a href="#">SET BELL</a>	Ya no es necesaria la duración del sonido.
<a href="#">SET('PRINTER')</a>	Acepta una nueva opción 3 que le permite determinar la impresora predeterminada actual de Visual FoxPro establecida en los cuadros de diálogo Impresora o Configurar impresión de Visual FoxPro.
<a href="#">SET('BELL')</a>	Ahora se puede utilizar para determinar el tipo de sonido que se va a reproducir.
<a href="#">Función STRCONV( )</a>	Admite un nuevo argumento <i>nIdLocale</i> que le permite especificar el Id. de configuración regional que se va a utilizar en la conversión.
<a href="#">SYS(2333) – Compatibilidad con interfaz ActiveX dual</a>	Ahora le permite determinar el valor actual y el valor de inicio predeterminado de la compatibilidad para interfaz ActiveX dual se ha cambiado de habilitada en Visual FoxPro 5.0 a deshabilitada en Visual FoxPro 6.0.
<a href="#">Función TABLEUPDATE( )</a>	Si al actualizar registros se produce un error distinto a un



simple error de confirmación, el primer elemento de la matriz de errores contendrá -1 y después podrá utilizar AERROR( ) para determinar por qué no se han podido confirmar las modificaciones.

<a href="#">Objeto ToolBar</a>	Ahora admite la propiedad <a href="#">Tag</a> y el método <a href="#">Release</a> .
<a href="#">Función TRANSFORM( )</a>	El código de formato <i>cCódigosFormato</i> ahora es opcional. Se utiliza una transformación predeterminada si se omite el código de formato <i>cCódigosFormato</i> .
<a href="#">Función VERSION( )</a>	Admite dos nuevas opciones <i>de nExpresión</i> , 4 y 5, para devolver el número de versión de Visual FoxPro en formatos que se puedan analizar con facilidad.

Funciones de Foxtools	Descripción
Se han agregado las siguientes funciones a Visual FoxPro 6.0 desde Foxtools; ahora se pueden utilizar sin ejecutar SET LIBRARY TO FOXTOOLS.	Tenga en cuenta que tiene que volver a compilar los programas, bibliotecas de clases, etiquetas o informes creados en versiones anteriores de Visual FoxPro si contenían alguna de estas funciones.
<a href="#">Función ADDBS( )</a>	Agrega una barra invertida (si fuera necesario) a una expresión de ruta.
<a href="#">Función AGETFILEVERSION( )</a>	Crea una matriz que contiene información acerca de archivos con recursos de versión de Windows como los archivos .exe, .dll y .flf, o de servidores de automatización creados en Visual FoxPro. Corresponde a la función GetFileVersion( ) de Foxtools.
<a href="#">Función DEFAULTTEXT( )</a>	Devuelve un nombre de archivo con una nueva extensión si no la tuviera.
<a href="#">Función DRIVETYPE( )</a>	Devuelve el tipo de la unidad especificada.
<a href="#">Función FORCEEXT( )</a>	Devuelve una cadena con la extensión de archivo anterior reemplazada por una nueva extensión.
<a href="#">Función FORCEPATH( )</a>	Devuelve un nombre de archivo con una nueva ruta en lugar de la ruta anterior.
<a href="#">Función JUSTDRIVE( )</a>	Devuelve la letra de unidad de una ruta completa.
<a href="#">Función JUSTEXT( )</a>	Devuelve la extensión de tres letras de una ruta completa.
<a href="#">Función JUSTFNAME( )</a>	Devuelve la parte del nombre de archivo de una ruta completa.
<a href="#">Función JUSTPATH( )</a>	Devuelve la parte de ruta de una ruta completa.
<a href="#">Función JUSTSTEM( )</a>	Devuelve el nombre (el nombre del archivo sin la

---

extensión) de una ruta completa y un nombre de archivo.

---

## Mejoras en el rendimiento de Visual FoxPro

Se ha mejorado significativamente el rendimiento de la concatenación de cadenas en Visual FoxPro 6.0. La concatenación de cadenas suele utilizarse para crear páginas Web con código como el siguiente:

```
cMiCadena = cMiCadena + <etiquetas html>
cMiCadena = cMiCadena + <más etiquetas html>
cMiCadena = cMiCadena + <aún más etiquetas html>
```

Además, también se ha mejorado el rendimiento de la creación de objetos y la creación de instancias; normalmente es 10 o más veces más rápida que en versiones anteriores.

## Mejoras en la robustez de Visual FoxPro

Ahora Visual FoxPro 6.0 detecta errores de protección general (General Protection Faults, GPF) en los controles ActiveX colocados en un formulario o en instancias de objetos COM creadas en Visual FoxPro. Ahora se trata un GPF de control ActiveX o de objeto COM como errores detectables de Visual FoxPro ([Error 1440](#) - El objeto OLE puede estar dañado).

## Mejoras en la facilidad de uso de Visual FoxPro

Puede especificar la cadena de comentario del editor de Visual FoxPro en el Registro de Windows. Abra la carpeta Options de Visual FoxPro 6.0 con el Editor del Registro de Windows (RegEdit) y haga clic con el botón secundario del *mouse* en la carpeta. Elija **Nuevo** y, a continuación, **Valor de la cadena**. Escriba el nombre "EditorCommandString" para el nuevo valor de la cadena. Haga clic con el botón secundario del *mouse* en la cadena y elija **Modificar**. Escriba la cadena de comentario del editor (\*!\* es el valor predeterminado que se utiliza cuando esta entrada no existe en el Registro).

Ahora puede tener acceso al menú Formulario desde la ventana de código del formulario. Además, puede ejecutar un formulario con el método abreviado de teclado CTRL+E, incluso desde una ventana de código de un formulario.

## Compatibilidad con el milenio

Se ha mejorado Visual FoxPro 6.0 para proporcionar mejor compatibilidad con el milenio. Esta sección describe las mejoras de Visual FoxPro que facilitan la creación de aplicaciones compatibles con el milenio.

### SET CENTURY TO

La documentación de Visual FoxPro 5.0 indica que si ejecuta el comando SET CENTURY TO sin argumentos adicionales establece el siglo al siglo actual. Esto sólo es cierto en el siglo 20, porque el siglo se establece a 19 independientemente de cuál sea el siglo actual. En Visual FoxPro 6.0, SET CENTURY TO establece el siglo al siglo actual. Además, el valor de SET CENTURY TO en nuevas sesiones de datos se inicializa al siglo actual.

Además, en Visual FoxPro 6.0, se ha modificado el valor predeterminado de ROLLOVER para SET CENTURY con los dos dígitos del año actual más 50 años (si el año actual es 1998, *nAño* es 48, los dos últimos dígitos de 2048 (1998 + 50). En Visual FoxPro 5.0, el valor predeterminado es 0.

Vea [SET CENTURY](#) para obtener más información.

## Formatos estrictos de fecha

Normalmente, las constantes o expresiones de tipo Date y DateTime se interpretan en función de los valores actuales de [SET DATE](#) y [SET CENTURY](#) en el momento en el que las constantes o las expresiones se compilan o evalúan. Esto significa que muchas constantes de fecha son ambiguas puesto que se pueden evaluar como valores diferentes en función de cuándo se compilaron y el valor de fecha del momento de la compilación.

Por ejemplo, ¿es la constante de fecha {10/11/12} 11 de octubre de 1912, 11 de octubre de 2012, 10 de noviembre de 1912, 12 de noviembre de 1910 o 12 de noviembre de 2010?

Todo depende de los valores actuales de SET DATE y SET CENTURY TO. Esto puede introducir errores en el código existente de Visual FoxPro siempre que se compilen o evalúen en tiempo de ejecución constantes o expresiones de tipo Date o DateTime, como expresiones de informe u objeto. Esto puede producir incompatibilidad con el milenio cuando el valor de SET CENTURY pase al año 2000 y no se especifiquen fechas de cuatro dígitos.

Para evitar esta incompatibilidad, ahora se dispone de un formato de fecha estricto en Visual FoxPro 6.0 (y en Visual FoxPro 5.0). Las fechas estrictas siempre se evalúan con el mismo valor Date o DateTime independientemente de los valores de las fechas. El formato de fecha estricto es:

`^aaaa-mm-dd[,][hh[:mm[:ss]]][a[p]]`

El carácter (^) denota siempre el formato de fecha estricto y hace que los valores de tipo Date y DateTime se interpreten en formato AMD. Los separadores válidos son los guiones, las barras, los puntos y los espacios.

Los valores de tipo Date y DateTime no son ambiguos y siempre son válidos. Los formatos vacíos de Date y DateTime incluyen {}, {--} y {--:}.

Con los formatos de fecha estrictos tiene a su disposición un rango más grande de valores de tipo Date y DateTime. En Visual FoxPro 5.0, el menor valor de fecha que se puede expresar es {^0100/1/1}, 1 de Enero de 100 A.C. Esto es así porque los valores de año inferiores a 100 siempre se redondeaban hasta el siglo siguiente basándose en el valor de SET CENTURY.

La menor fecha válida en Visual FoxPro 6.0 es {^0001-01-01}, 1 de enero del 1 A.C. La mayor fecha válida en Visual FoxPro 6.0 es {^9999-12-31}, 31 de diciembre de 9999 D.C.

Observe que el formato estricto de fecha ignora el valor TAIWAN en SET DATE, de forma que el año en el formato Date o DateTime estricto siempre hace referencia al calendario occidental. (Recuerde que esto no es así en Visual FoxPro 5.0).

## SET STRICTDATE

Puede utilizar un nuevo comando, [SET STRICTDATE](#), para forzar la compatibilidad de las constantes y cadenas de fechas con el milenio.

### SET STRICTDATE TO 0

Establecer STRICTDATE a 0 significa que la comprobación del formato estricto de fecha está desactivada. Esto es compatible con Visual FoxPro 5.0. 0 es el valor predeterminado para el entorno de tiempo de ejecución de Visual FoxPro y el controlador ODBC. Cuando STRICTDATE está establecido a 0, los valores Date y DateTime no válidos se evalúan como cadenas vacías.

### SET STRICTDATE TO 1

Establecer STRICTDATE a 1 requiere que todas las constantes de tipo Date y DateTime estén en el formato estricto. Cualquier constante de tipo Date o DateTime que no esté en el formato estricto o que se evalúe como valor no válido generará un error, durante la compilación, en tiempo de ejecución o durante una sesión interactiva en Visual FoxPro. 1 es el valor predeterminado para las sesiones interactivas en Visual FoxPro.

### SET STRICTDATE TO 2

Es idéntico a establecer STRICTDATE a 1, pero además genera un error de compilación (2033 – CTOD y CTOT pueden producir resultados incorrectos) siempre que las funciones [CTOD\(\)](#) y [CTOT\(\)](#) aparezcan en el código.

Como los valores devueltos por CTOD() y CTOT() se basan en SET DATE y SET CENTURY para interpretar la fecha que contienen pueden producir errores de incompatibilidad con el milenio. Utilice DATE() y DATETIME() con los argumentos numéricos opcionales para crear constantes y expresiones de tipo Date y DateTime.

Este valor es útil en las sesiones de depuración para detectar el código que pueda contener errores de incompatibilidad con el milenio.

## Errores de formato estricto de fecha

Se han agregado los siguientes errores nuevos a Visual FoxPro 6.0 y se pueden generar cuando SET STRICTDATE está establecido a 1 ó 2.

### Error 2032: constante Date/DateTime ambigua.

Este error se produce cuando un valor de tipo Date o DateTime no cumple el formato estricto. Las siguientes condiciones producirán este error:

- Falta el signo ^.
- Los separadores de fecha no son guiones, ni barras, ni puntos, ni espacios.

- El campo año contiene menos de cuatro caracteres ({^98-02-16}).
- Los campos mes o día están vacíos ({^1998-02}).

**Error 2033: CTOD y CTOT pueden producir resultados incorrectos.**

Este error se produce por las mismas razones que el error 2032, pero CTOD( ) y CTOT( ) pueden ser no compatibles o ambiguas. Utilice en su lugar las funciones [DATE\(\)](#) o [DATETIME\(\)](#).

**Error 2034: valor Date/DateTime evaluado a un valor no válido.**

Un valor de tipo Date o DateTime no tiene el formato válido o está fuera del intervalo válido para Date y DateTime.

Cuando SET STRICTDATE está establecido a 0, las constantes de tipo Date o DateTime no válidas se evalúan como constantes Date y DateTime vacías. Cuando se establece SET STRICTDATE a 1 ó 2, las constantes de fecha no válidas como {^2000-02-31}, 31 de febrero o {^2000-01-01,25:00}, 25 en punto, generarán este error.

Algunos ejemplos de valores de tipo Date y DateTime no válidos son:

- {^2000-02-31}, 31 de febrero, 2000.
- {^2000-01-01,25:00} 25 en punto.
- {^2000-01-01, 14a}, 14 A.M.

**Error 2035: un valor Date/DateTime contiene caracteres no válidos.**

La constante Date o DateTime contiene caracteres no admitidos por las constantes de tipo Date y DateTime.

Cuando se establece SET STRICTDATE a 0, las constantes de tipo Date o DateTime que contengan caracteres ilegales se evalúan como valores Date o DateTime vacíos. Cuando se establece SET STRICTDATE a 1 ó 2, la constante de tipo Date o DateTime que contenga los caracteres ilegales generará este error.

Tenga en cuenta que la propiedad [StrictDateEntry](#) no se ve afectada por el valor de SET STRICTDATE. Esta propiedad no cambia en Visual FoxPro 6.0.

**Cuadro de diálogo Opciones**

La [ficha General](#) del cuadro de diálogo **Opciones** incluye ahora un cuadro de lista desplegable **Compatibilidad con el milenio**, que especifica el valor de SET STRICTDATE. Como los demás elementos del cuadro de diálogo Opciones, el valor se establece para la sesión actual de Visual FoxPro y la elección de **Establecer como predeterminado** guarda el valor en el Registro de Windows para la siguiente sesión de Visual FoxPro.

**Funciones DATE( ) y DATETIME( )**

Ahora las funciones [DATE\(\)](#) y [DATETIME\(\)](#) admiten argumentos numéricos opcionales que le permiten crear valores de tipo Date o DateTime compatibles con el milenio. Las mejoras de estas funciones proporcionan ahora un método preferible para crear valores de tipo Date y DateTime; ya no es necesario utilizar funciones de manipulación de caracteres para crearlos.

## Función FDATE()

Ahora la función [FDATE\(\)](#) acepta un argumento opcional que le permite determinar la hora de la última modificación de un archivo sin utilizar funciones de manipulación de caracteres. Por ejemplo, en versiones anteriores de Visual FoxPro era necesario escribir código como el siguiente para determinar cuándo se modificó por última vez el archivo de recursos de Visual FoxPro:

```
tLastModified = CTOT(DTOC(FDATE('Foxuser.dbf')) + ' ' ;  
    + FTIME('Foxuser.dbf'))
```

Ahora puede reemplazar este código con el siguiente:

```
tLastModified = FDATE('Foxuser.dbf', P)
```

## Propiedad Century

El valor predeterminado de la propiedad [Century](#) en Visual FoxPro 6.0 es 1 – Activado. La parte de siglo de la fecha se muestra en un cuadro de texto. En las versiones anteriores de Visual FoxPro, el valor predeterminado es 2 (el valor de SET CENTURY determina si se muestra la parte de siglo de las fechas).