**INSTITUTO METROPOLITANO DE EDUCACIÓN**
**PROGRAMACIÓN DE COMPUTADORES**
**GUIA #7 DE VISUAL FOXPRO**
**DOCENTE: MAURICIO CANO**

# LOS REPORTES Y LAS CONSULTAS EN LAS APLICACIONES

La potencia de las aplicaciones, viene marcada por la cantidad de informes y consultas que permitan hacer sobre la información almacenada en las tablas que componen a un sistema de información.

La cantidad de informes queda limitada a la capacidad de cada programador y a los reportes que solicite la persona que nos ha contratado para desarrollar un proyecto. Lo cual significa, que un sistema por pequeño que sea puede incluir una serie de reportes que harían que el proyecto se viese como un "*gigante*".

Las consultas pueden ser utilizadas para enviar resultados de búsquedas basadas en criterios ofrecidos por los usuarios y su resultado se puede enviar a la pantalla o a la impresora.

## EL GENERADOR DE REPORTES

El FoxPro incluye un generador de reportes bastante potente. Dentro de un reporte se pueden incluir texto, campos, imágenes, líneas, cuadros, campos calculados, unión de campos, funciones, etc.

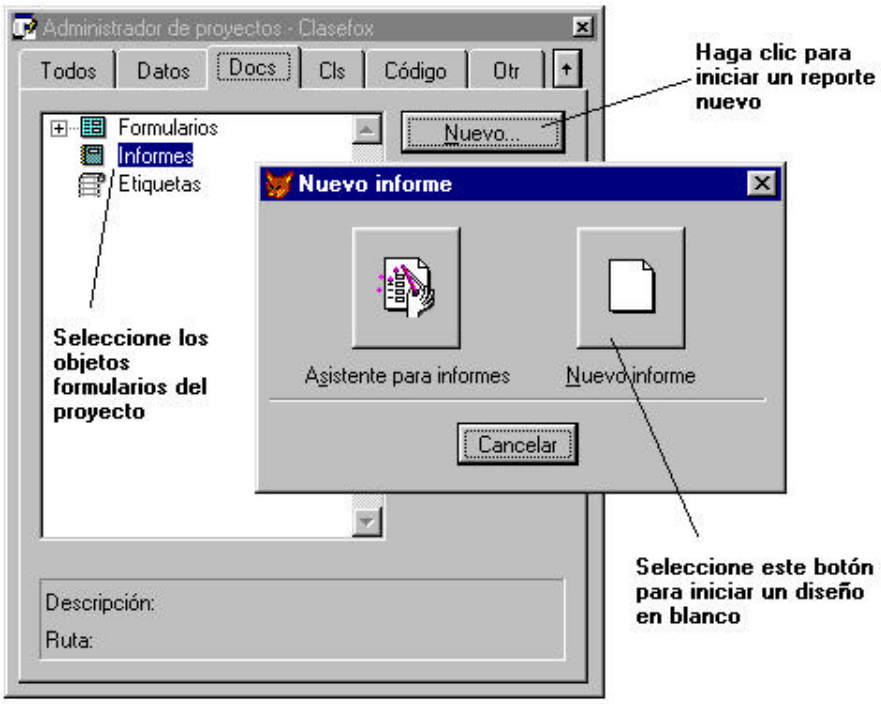
Si se domina (como programador) el tema de las funciones, se pueden ejecutar una serie de acciones que ahorrarían mucho tiempo a la hora de diseñar reportes. Las expresiones que conjuguen una serie de funciones, capos y cadenas de caracteres, permiten ampliar aún más las posibilidades del generador de reportes del FoxPro.

Tenga presente que el FoxPro "*amarra*" las tablas utilizadas para sacar la información, al reporte que las utiliza. Si analizamos la forma en que hemos venido trabajando hasta el momento, en ningún momento debemos permitir que tal cosa pase. Nuestra aplicación utiliza tablas ubicadas en directorios diferentes dependiendo del campo *DIRECTORIO* en la tabla *CIAS.DBF*.

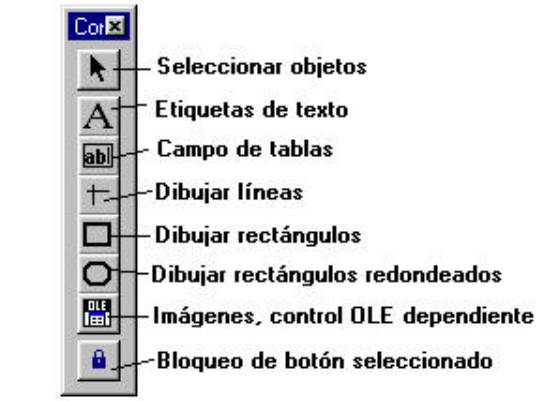
Como no podemos dejar que el FoxPro amarre las tablas a los reportes, iniciaremos el generador en blanco y diseñaremos el reporte teniendo a la mano los campos requeridos para el informe.

Seleccione la pestaña *Docs* en el administrador de proyectos y seleccione los objetos formularios para que FoxPro nos muestre los reportes diseñados. Si es un reporte a modificar, haga clic en la lista sobre el nombre del reporte, luego haga clic en el botón *Modificar*. Si lo que desea es agregar un reporte que hace falta dentro del proyecto, haga clic en *Agregar* y seleccione el nombre del reporte desde sus unidades de disco.

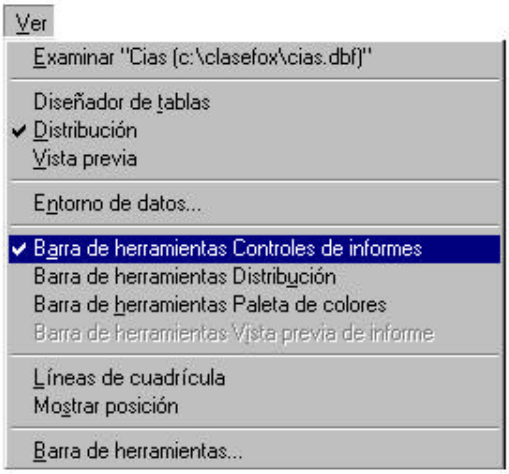
Si lo que se desea es crear un nuevo formulario, haga clic en *nuevo*, seleccione de la pantalla de diálogo que aparece el botón *Nuevo Informe*. Esta acción abrirá un formulario en blanco, sobre el cual iniciaremos las actividades que definirán el nuevo reporte. Recuerde que el reporte no es funcional en este punto debido a que no existe ninguna tabla abierta en memoria. Una técnica utilizada por muchos programadores, consiste en crear una rutina que abre las tablas necesarias para la funcionalidad del reporte, ejecutarla antes de entrara la creación de mismo, y verificar el correcto funcionamiento del reporte en el sistema. Si no se utiliza ninguna técnica de estas, nuestro reporte solo será funcional cuando lo llamemos desde el formulario que hace uso de las tablas requeridas por el reporte.




Para este ejemplo, crearemos el reporte *Programa.frt* para mostrar la información de la tabla *Programas*. Recuerde que el reporte no funcionará desde el modo diseño, así que no intente ejecutar un *vista preliminar* del reporte.



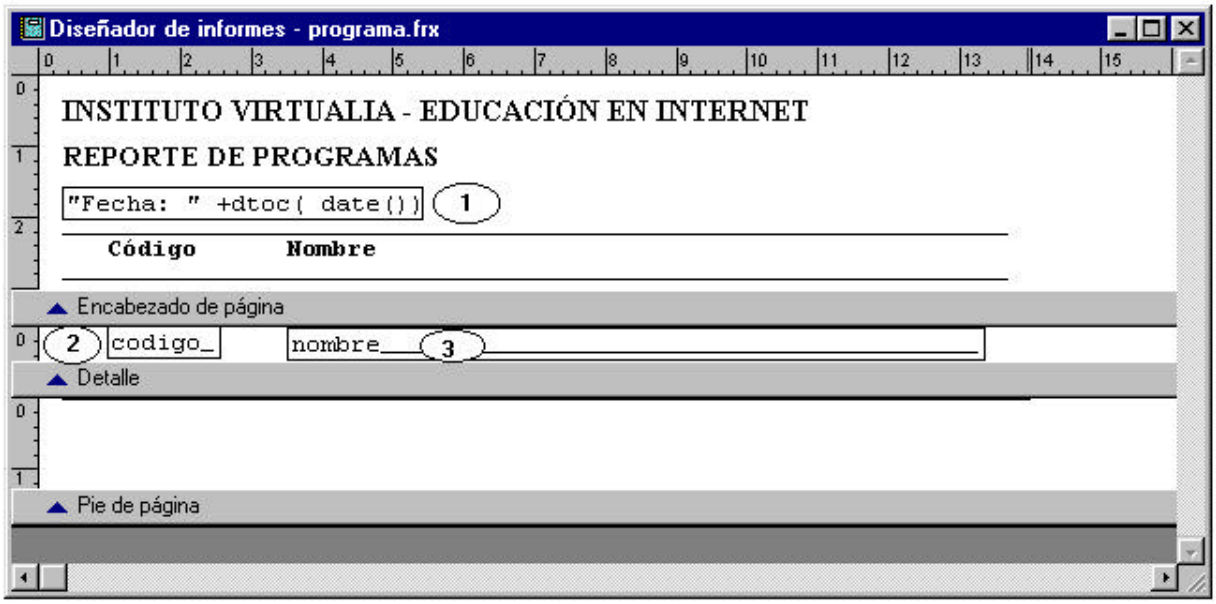

Barra controles de informe




Verifique que la barra de herramientas Controles de informes aparezca para poder trabajar en el diseñador de informes

La barra controles de informes nos facilita el manejo de los objetos que pueden aparecer dentro de un informes. Si no parece, haga clic en el menú *Ver* y *Barra de herramientas Controles de informes.*

Utilice el siguiente esquema para diseñar el reporte para la tabla programas:

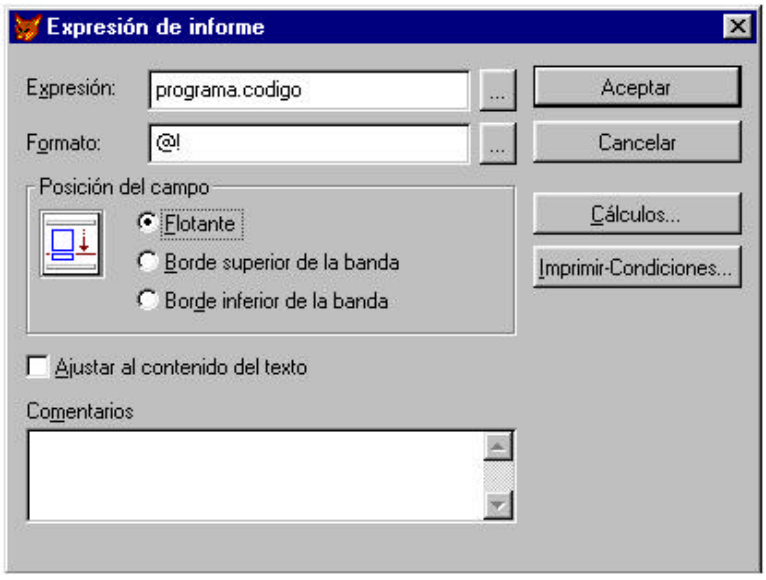



Como aspecto importante a considerar a la hora de diseñar reportes, es la inclusión del nombre de la empresa para la cual se diseña la aplicación. En algún sitio del encabezado del reporte debe aparecer el nombre del reporte. Incluir la fecha del sistema es también una buena idea para el usuario que tome uno de los reportes en un futuro posterior a la impresión.

Los objetos 1, 2 y 3 se diseñan con el botón *Campos de Tablas*. Utilice estos parámetros para indicarle al generador de reportes de donde proviene la información:

| #Objeto | Expresión origen de datos | Formato |
|---|---|---|
| 1 | "Fecha: " +dtoc( date()) | |
| 2 | programa.codigo | @! |
| 3 | Programa.nombre | @! |

En la pantalla de *Propiedades* del campo, coloque el parámetro de Posición a *Flotante*.



## EL FORMULARIO PARA SELECCIONAR EL DESTINO DE UN REPORTE

El siguiente paso después de diseñar el reporte, es construir un formulario de uso global que nos permita llamarla desde cualquier formulario, permitirle al usuario seleccionar el destino de un reporte (pantalla o impresora), recoger la selección hecha por el usuario y ejecutar el reporte seleccionado.

En cada formulario diseñado hasta el momento, se venia dejando un botón denominado *Reporte*. En este botón insertaremos el código necesario para activar el formulario que se diseñará a continuación. Vea más adelante en esta guía, la serie de ordenes que utilizan el formulario *Reportes*. El formulario se diseñará una sola vez, en los demás formularios, se invocará a este formulario para permitirle al usuario que seleccione el destino de un reporte.

Construya el siguiente formulario y guárdelo con el nombre de *reportes*.


| | | |
|---|---|---|
| 1: btnPantalla | 2: btnImpresora | 3: btnCerrar |

## CÓDIGO EN LOS OBJETOS DEL FORMULARIO REPORTES

En las propiedades del formulario, active la pestaña *Otras*, busque la propiedad WindowType y colóquela al valor 1-Modal. Si no hace esto, el formulario generará un error al tratar de ejecutarlo. Esto sucede debido a que se retornarán datos que indicarán la selección hecha por el usuario.

**EVENTO INIT DEL FORMULARIO REPORTES (REPORTES.INIT)**

```
public _porPantalla, _porImpresora, _btnCerrar
```

**EVENTO UNLOAD DEL FORMUARIO (REPORTE.UNLOAD)**

```
if _porPantalla
        _resultado = 1
endif

if _porImpresora
        _resultado = 2
endif

if _btnCerrar
        _resultado = 0
endif

return( _resultado)
```

Este evento se ejecuta cuando se cierra el formulario. Observe que cada vez que el usuario hace clic en cada botón, se le asignan vabres a unas variables lógicas, indicadoras al final de sobre cual botón se hizo clic.

En esta parte del formulario, se recogen los valores de esas variables y se asigna un valor para retornar a la aplicación que llamó al formulario.

La instrucción RETURN es la encargada de devolver el parámetro a la aplicación que llame a esta rutina.

| **btnPantalla.Click** | **btnImpresora.Click** |
|---|---|
| *botón Pantalla<br><br>_porPantalla = .t.<br>_porImpresora = .f.<br>_btnCerrar = .f.<br><br>thisform.release | *botón Impresora<br><br>_porPantalla = .f.<br>_porImpresora = .t.<br>_btnCerrar = .f.<br><br>thisform.release |

| **BtnCerrar.Click** |
|---|
| *botón Cerrar<br><br>_porPantalla = .f.<br>_porImpresora = .f.<br>_btnCerrar = .t.<br><br>thisform.release |

Observe que cada botón libera al formulario y a su vez, asigna unos valores lógicos equivalentes a los tres únicos botones en la pantalla. Existe una variable por cada botón y cuando se hace clic en cada botón, este coloca el valor de verdadero sobre la variable equivalente al botón pulsado y coloca el valor de falso a las otras variables. En la rutina del evento Unload del formulario, se chequea el estado de esas tres variables y se asigna un nuevo valor al parámetro que se debe retornar.

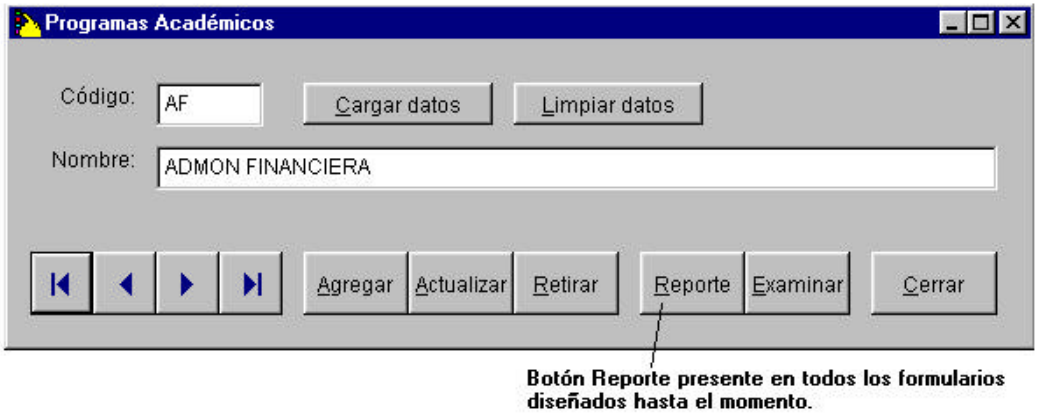
Se ha determinado que la rutina devolverá 0 cuando el usuario no escoja nada o haga clic en el *botón Cerrar*. Retorna 1 cuando se hace clic sobre el *botón Pantalla* y retorna un 2 cuando se hace clic sobre el *botón Impresora.*

**FORMA DE UTILIZAR EL FORMULARIO REPORTE**

Hasta el momento tenemos un reporte diseñado (*Programas.frt*), un formulario llamado *Reportes* que permite seleccionar el destino de un reporte y un formulario, diseñado en clases anteriores llamado *Programas*. En ese formulario, aparece un botón que hasta el momento no se había utilizado, denominado *btnReporte*.

Cuando el usuario haga clic sobre este botón, el sistema debe mostrar una pantalla donde el usuario pueda seleccionar el destino del reporte. Eso es lo programaremos a continuación:

Abra el formulario Programa en modo de diseño, ubique el botón *btnReporte* y haga doble clic sobre este. Busque el evento *Clic* y escriba el código que se especifica a continuación




**BtnReporte.Click**

```
*generar reporte de programas

*consultar el destino para el reporte
do form forms\reportes to mDestino

do case
case mDestino = 1		&& pantalla
	report form informes\programa preview
case mDestino = 2		&& impresora
	*verificar estado de la impresora
	if printstatus()	&& verificar el estado de la impresora
		report form informes\programa to print noeject noconsole
	else
		mMens1 = "La impresora no esta lista. Preparela e intente de nuevo..." +chr(13)
		mOpc = messagebox( mMens1, 0+48, "Atención!")
	endif
endcase

*regresar a la pantalla
go top
scatter memvar memo
thisform.refresh
return
```

Observe que el código incluye la instrucción:

*do form forms\reportes to mDestino*

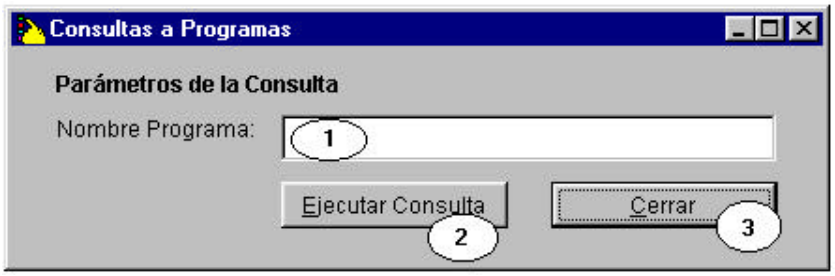
Con ésta orden lo que se hace es invocar al formulario Reportes e indicarle al FoxPro que deseamos almacenar el valor que retorne el formulario en la variable *mDestino*. El formulario Reportes se ejecuta, permite que el usuario haga su selección, retorna el valor en la variable *mDestino*, y lo único que resta es verificar el valor retornado. Recuerde que un valor de 1 indica que seleccionaron *Pantalla*, un valor de 2, *Impresora*, y un valor de 0, indica que el usuario no seleccionó nada.

# LAS CONSULTAS EN LAS APLICACIONES

Una de las características potentes en un gestor de bases de datos, lo constituye la capacidad de generar consultas a la información almacenada en las tablas de un sistema de información determinado.

La forma en que se puede generar consultas a las tablas son muchas, pero una de las más sencillas y que permiten gran flexibilidad durante la ejecución, son las ofrecidas por la instrucción SELECT-SQL. Esta orden es verdaderamente potente y permite programar instrucciones de consultas en forma fácil. Por ejemplo, se puede diseñar un formulario, unos datos en los cuales se recogen los parámetros de la consulta, y basado en esos datos entrados por el usuario, ejecutar una consulta en tiempo real.

Abra el diseñador de formulario con un formulario en blanco y diseñe el siguiente formulario. Cuando termine guárdelo en la carpeta de los formularios con el nombre de *SQLProg*. El anteponer SQL a los formularios de consulta, hará que estos se agrupen visualmente en el administrador de proyectos, permitiéndonos en un futuro, administrar de forma fácil las pantallas de consultas.




| | | |
|---|---|---|
| 1: mNombreSQL | 2: btnEjecutarSQL | 3: btnCerrar |

El usuario deberá entrar en el campo 1 un texto. Cuando haga clic en el botón Ejecutar Consulta, el sistema ejecutará una instrucción SELECT-SQL y traerá de la tabla PROGRAMA los programas que incluyan en su nombre el texto entrado.

**FORMULARIO.INIT**

```
*abrir entorno utilizando las variables públicas

close data

mFile1 = _dircia +"programa.dbf"
if !file( mFile1)
        do cPrograma in prg\creartbl
endif

sele 1
use &mFile1 order tag codigo alias programa

return
```

En *botón ejecutar* lanza la consulta en el sistema. Observe el uso que se hace del parámetro entrado por el usuario. Si entra algo, se usa como parámetro, si deja el campo en blanco, simplemente se traen todos los registros. El resultado de la consulta es mostrado en una ventana examinar, que debe ser cerrada con *ctrl.+w.*

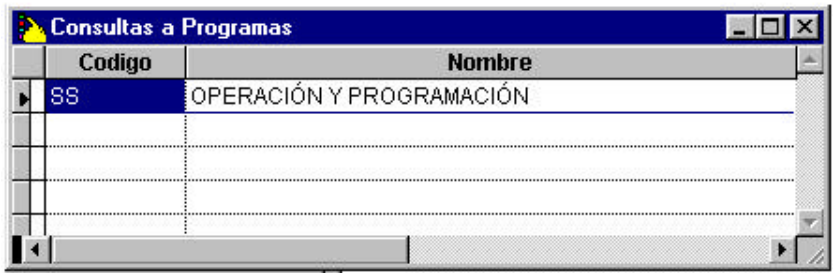
**BtnEjecutarSQL.Click**

```
*ejecutar consulta copn los parámetros entrados
mNombreSQL = alltrim( thisform.mNombreSQL.value)

if empty( mNombreSQL)
	select * from programa
else
	select * from programa where mNombreSQL $ programa.nombre
endif

return
```

**BtnCerrar.Click**

```
thisform.release
```

Por ejemplo, si introducimos como parámetro la palabra "PROG" obtendremos como resultado la siguiente pantalla:




Para cerrar esta pantalla, pulse *Ctrl.+W.*

Aunque en este ejemplo se hizo una búsqueda basado una clase datos, no indica que siempre se tenga que hacer de tal forma. Las posibilidades de programar parámetros son muchas y la capacidad de programar tales ordenes dentro de una instrucción SELECT-SQL hacen de esta orden lo máximo para controlar consultas.

Para mayor información acerca de la orden SELECT-SQL, consulte el manual de comandos y funciones del VisualFoxPro.

**ANEXO A LA GUÍA DE CLASES – LA INSTRUCCIÓN SELECT – SQL**

El siguiente texto fue tomado de la ayuda del FoxPro.

Recupera datos de una o más tablas.

Sintaxis

```
SELECT [ALL | DISTINCT] [TOP nExpresión [PERCENT]]
        [Alias.] Elemento_Selección [AS Nombre_Columna]
        [, [Alias.] Elemento_Selección [AS Nombre_Columna] ...]
FROM [FORCE]
[NombreBaseDatos!]Tabla [Local_Alias]
        [[INNER | LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER] JOIN
                NombreBaseDatos!]Tabla [Alias_Local]
                [ON CondiciónCombinación …]
[[INTO Destino]
        | [TO FILE NombreArchivo [ADDITIVE] | TO PRINTER [PROMPT]
        | TO SCREEN]]

[PREFERENCE NombrePreferencia]
[NOCONSOLE]
[PLAIN]
[NOWAIT]
[WHERE CondiciónCombinación [AND CondiciónCombinación ...]
        [AND | OR CondiciónFiltro [AND | OR CondiciónFiltro ...]]]
[GROUP BY ColumnaGrupo [, ColumnaGrupo ...]]
[HAVING CondiciónFiltro]
[UNION [ALL] SELECTCommand]
[ORDER BY Elemento_Orden [ASC | DESC] [, Elemento_Orden [ASC | DESC] ...]]
```

Argumentos

SELECT Especifica los campos, constantes y expresiones que se mostrarán en el resultado de la consulta.

ALL De forma predeterminada, se muestran todas la filas del resultado de la consulta.

DISTINCT Excluye duplicados de cualquier fila del resultado de la consulta.

Nota Puede utilizar DISTINCT únicamente una vez por cláusula SELECT.

TOP nExpresión [PERCENT] Especifica que el resultado de la consulta contenga un número determinado de filas o un porcentaje de filas en el resultado de la consulta. Es necesario incluir una cláusula ORDER BY si incluye la cláusula TOP. La cláusula ORDER BY especifica las columnas en las que la cláusula TOP determinará el número de filas que se va a incluir en el resultado de la consulta.

Puede especificar desde 1 a 32,767 filas. Las filas de valores idénticos para las columnas especificadas en la cláusula ORDER BY se incluyen en el resultado de la consulta. A partir de entonces, si especifica 10 para nExpr, el resultado de la consulta podrá obtener más de 10 filas si hay más de 10 filas con valores idénticos para las columnas especificadas en la cláusula ORDER BY.

Si se incluye la palabra clave PERCENT, se redondeará al número entero más alto el número de columnas devuelto en el resultado. Los valores permitidos para nExpr cuando se incluye la palabra clave PERCENT son 0.01 a 99.99.

Alias. Califica nombres de elementos coincidentes. Cada elemento que especifique con Elemento_Selección genera una columna de los resultados de la consulta. Si dos o más elementos tienen el mismo nombre, incluya el alias de la tabla y un punto antes del nombre del elemento para impedir la duplicación de las columnas.

Elemento_Selección especifica un elemento a incluir en el resultado de la consulta. Un elemento puede ser uno de los siguientes:

· El nombre de un campo de una tabla de la cláusula FROM.
· Una constante especificando que el mismo valor constante ha de aparecer en cada fila del resultado de la consulta.
· Una expresión que puede ser el nombre de una función definida por el usuario (FDU).

AS Nombre_Columna Especifica el título de una columna en el resultado de la consulta. Esta opción resulta muy útil cuando Elemento_Selección es una expresión o contiene una función de campo y desea dar un nombre significativo a la columna. Nombre_Columna
puede ser una expresión pero no puede contener caracteres (por ejemplo, espacios) que no estén permitidos para nombres de campos de tablas.

FROM Enumera las tablas que contienen los datos que obtuvo la consulta. Si no hay ninguna tabla abierta, Visual FoxPro mostrará el cuadro de diálogo Abrir para permitirle especificar la ubicación del archivo. Una vez abierta, la tabla permanecerá abierta cuando la consulta se haya terminado.

FORCE Especifica que las tablas se combinarán en el orden de aparición en la cláusula FROM. Si se omite FORCE, Visual FoxPro intentará optimizar la consulta. Sin embargo, es posible que la consulta se ejecute más rápido si se incluye la palabra clave FORCE para desactivar la optimización de consultas de Visual FoxPro.

NombreBaseDatos! Especifica el nombre de una base de datos inactiva que contiene la tabla. Es necesario incluir el nombre de la base de datos que contiene la tabla en caso de que no sea la base de datos activa. Incluya el delimitador de signo de exclamación (!) después del nombre de la base de datos y antes del nombre de la tabla.

Alias_Local Especifica un nombre temporal para la tabla indicada en Tabla. Si especifica un alias local, debe utilizar el alias local en lugar de la tabla a través de todo el SELECT.

INNER JOIN Especifica que el resultado de la consulta contenga sólo filas para una tabla con la que coincidan una o varias filas en otra tabla.

LEFT [OUTER] JOIN Especifica que el resultado de la consulta contenga todas las filas de la tabla a la izquierda de la palabra clave JOIN y sólo las filas que concuerden procedentes de la tabla a la derecha de la palabra clave JOIN. La palabra clave OUTER es opcional; se puede incluir para resaltar que se ha creado una combinación externa.

RIGHT [OUTER] JOIN Especifica que el resultado de la consulta contenga todas las filas desde la tabla hasta la derecha de la palabra clave JOIN y sólo las filas que concuerden desde la tabla hasta la izquierda de la palabra clave JOIN. La palabra clave OUTER es opcional; puede incluirse para resaltar la creación de una combinación externa.

FULL [OUTER] JOIN Especifica que el resultado de la consulta contenga todas las filas, concuerden o no, de ambas tablas. La palabra clave OUTER es opcional; se puede incluir para resaltar que se ha creado una combinación externa.

ON CondiciónCombinación Especifica las columnas según las cuales se combinan las tablas.

INTO Destino Determina donde se almacenan los resultados de la consulta. Si incluye una cláusula INTO y una cláusula TO en la misma consulta, la cláusula TO se pasará por alto. Si no incluye la cláusula INTO, los resultados de la consulta se mostrarán en la ventana Examinar. Los resultados de la consulta pueden dirigirse también a la impresora o a un archivo mediante la cláusula TO.

Destino puede ser uno de los siguientes:

· ARRAY NombreMatriz, que almacena los resultados de la consulta en una matriz de variable de memoria. Si la consulta selecciona 0 registros, la matriz no se creará.
· CURSOR NombreCursor [NOFILTER], que almacena los resultados de la consulta en un cursor. Si especifica el nombre de una tabla abierta, Visual FoxPro generará un mensaje de error. Después de que se ejecute SELECT, el cursor temporal permanecerá abierto y estará activo pero solamente para lectura. Una vez que cierre este cursor temporal, se borrará. Los cursores pueden existir como un archivo temporal en la unidad SORTWORK.

Incluya NOFILTER para crear un cursor que se pueda usar en consultas posteriores. En versiones anteriores de Visual FoxPro, era necesario incluir una expresión o una constante adicional como un filtro para crear un cursor utilizable en consultas posteriores. Por ejemplo, la adición de un Logical verdadero como una expresión de filtro creaba una consulta utilizable en consultas posteriores:

SELECT *, .T. FROM customers INTO CURSOR myquery

Si se incluye NOFILTER es posible que disminuya el rendimiento de la consulta, puesto que se creará una consulta temporal en el disco. Cuando se cierre el cursor se eliminará del disco la consulta temporal.

· DBF | TABLE NombreTabla
[DATABASE NombreBaseDatos [NAME NombreLargoTabla]] que almacena el resultado de la consulta en una tabla. Si especifica una tabla que ya esté abierta y SET SAFETY está en OFF, Visual FoxPro sobrescribirá la tabla sin previo aviso. Si no ha especificado ninguna extensión, Visual FoxPro dará una extensión .DBF a la tabla. La tabla permanecerá abierta y activa después de ejecutar SELECT.

Incluya DATABASE NombreBaseDatos para especificar una base de datos a la que se agregará la tabla. Incluya NAME NombreLargoTabla para especificar un nombre largo para la tabla. Los nombres largos pueden contener un máximo de 128 caracteres y pueden utilizarse en lugar de nombres cortos en la base de datos.

TO FILE NombreArchivo Si incluye una cláusula TO pero no una cláusula INTO, podrá dirigir el resultado de la consulta a un archivo de texto ASCII llamado NombreArchivo, o a la impresora además de al escritorio o la ventana principal de Visual FoxPro.

ADDITIVE añade la salida de la consulta al contenido existente del archivo de texto especificado en TO FILE NombreArchivo.

TO PRINTER [PROMPT] Dirige la salida de la consulta a una impresora. Utilice la cláusula PROMPT opcional para que aparezca en pantalla un cuadro de diálogo antes de que empiece la impresión. En este cuadro de diálogo podrá modificar la configuración de la impresora. Los valores de la impresora modificables dependen del controlador de impresora instalado en este momento. Sitúe la palabra clave PROMPT inmediatamente después de TO PRINTER.

TO SCREEN Dirige la salida de la consulta a la ventana principal de Visual FoxPro o a una ventana definida por el usuario que esté activa.

PREFERENCE NombrePreferencia Guarda los atributos y opciones de la ventana Examinar para uso posterior, si se envía el resultado de la consulta a una ventana Examinar.

Emitiendo SELECT con un NombrePreferencia de PREFERENCE, la primera vez se crea la preferencia. Emitiendo posteriormente SELECT con el mismo nombre de preferencia, se restaurará la ventana Examinar con el mismo estado de preferencia. Cuando se cierra la ventana Examinar, se actualiza la preferencia.

Si sale de una ventana Examinar presionando CTRL+Q+W, no se guardarán los cambios de la ventana Examinar en el archivo de recurso.

NOCONSOLE Impide que el resultado de la consulta se envíe a un archivo, a la impresora o a la ventana principal de Visual FoxPro.

PLAIN Impide que aparezcan las cabeceras de las columnas al mostrar la salida de la consulta. PLAIN puede utilizarse tanto si está presente una cláusula TO como si no. Si se incluye una cláusula TO, se pasará por alto PLAIN.

NOWAIT Continúa la ejecución del programa después de abrir la ventana Examinar y de dirigir a ella los resultados de la consulta. El programa no esperará a que la ventana Examinar se cierre, sino que continuará con la ejecución de la línea de programa inmediatamente siguiente a la instrucción SELECT.

Cuando se incluye TO SCREEN para dirigir la salida hacia la ventana principal de Visual FoxPro o una ventana definida por el usuario, la salida se detiene cuando la ventana principal de Visual FoxPro se llena con resultados de la consulta. Presione una tecla para ver el siguiente conjunto de resultados de la consulta. Si se incluye NOWAIT, los resultados de la consulta se desplazarán fuera del escritorio, la ventana principal de Visual FoxPro o una ventana definida por el usuario sin esperar a que se presione una tecla. NOWAIT se pasa por alto si se incluye con la cláusula INTO.

WHERE Indica a Visual FoxPro que incluya únicamente ciertos registros en el resultado de la consulta. WHERE es necesario para recuperar datos de varias tablas.

CondiciónCombinación especifica los campos que vinculan las tablas de la cláusula FROM. Si incluye más de una tabla en una consulta, deberá especificar una condición de combinación para cada tabla después de la primera.

Las condiciones de combinación múltiple deben conectarse mediante el operador AND. Cada condición de combinación tiene la forma siguiente:

NombreCampo1 Comparación NombreCampo2

NombreCampo1 es el nombre de un campo de una tabla, NombreCampo2 es el nombre de un campo de otra tabla y Comparación es uno de los operadores siguientes:

| Operador | Comparación |
|---|---|
| = | Igual |
| == | Exactamente igual |
| LIKE | SQL LIKE |
| <>, !=, # | Distinto de |
| > | Mayor que |
| >= | Mayor o igual que |
| < | Menor que |
| <= | Menor o igual que |

Cuando utiliza el operador = con cadenas, actúa de forma distinta dependiendo del ajuste de SET ANSI. Cuando SET ANSI está OFF, Visual FoxPro trata las comparaciones de cadenas en la forma habitual en Xbase. Cuando SET ANSI está a ON, Visual FoxPro sigue las normas ANSI para comparaciones de cadenas. Vea SET ANSI y SET EXACT para obtener información adicional sobre la forma en que Visual FoxPro realiza las comparaciones de cadenas.

La cláusula WHERE acepta el operador ESCAPE para la CondiciónCombinación, lo que le permite realizar consultas significativas sobre datos que contengan caracteres comodín _ y % de SELECT - SQL.

La cláusula ESCAPE le permite especificar que se traten los caracteres comodín de SELECT - SQL como si fueran caracteres literales. En la cláusula ESCAPE se especifica un carácter, el cual, cuando se sitúa inmediatamente antes del carácter comodín, indica que se tratará al carácter comodín como a un carácter literal.

CondiciónFiltro Especifica los criterios que deben cumplir los registros para que se incluyan en el resultado de la consulta. Una consulta puede incluir tantas condiciones de filtro como se deseen, conectadas con el operador AND y OR. También puede utilizar el operador NOT para invertir el valor de una expresión lógica o utilizar EMPTY( ) para comprobar si un campo está vacío.

CondiciónFiltro puede presentar una de estas formas:
Ejemplo 1
En el Ejemplo 1 se muestra la CondiciónFiltro en el formulario de NombreCampo1 Comparación NombreCampo2

customer.cust_id = orders.cust_id

Ejemplo 2

En el Ejemplo 2 se muestra CondiciónFiltro en el formulario de NombreCampo Comparación Expresión

payments.amount >= 1000

Ejemplo 3

En el Ejemplo 3 se muestra CondiciónFiltro en el formulario de NombreCampo Comparación ALL (Subconsulta)

Cuando la condición de filtro incluye ALL, el campo debe cumplir la condición de comparación para todos los valores generados por la subconsulta antes de que se incluya el registro en el resultado de la consulta.

```
company < ALL ;
(SELECT company FROM customer WHERE country = "Reino Unido")
```

Ejemplo 4

En el Ejemplo 4 se muestra CondiciónFiltro en el formulario de NombreCampo Comparación ANY | SOME (Subconsulta)

Cuando la condición de filtro incluye ANY o SOME, el campo debe cumplir la condición de comparación en al menos uno de los valores generados por la subconsulta.

```
company < ANY ;
(SELECT company FROM customer WHERE country = "Reino Unido")
```

Ejemplo 5

En el Ejemplo 5 se muestra CondiciónFiltro en el formulario de NombreCampo [NOT] BETWEEN Inicio_Rango AND Fin_Rango

Este ejemplo comprueba si los valores del campo están dentro de un intervalo de valores especificado.

```
customer.postalcode BETWEEN 90000 AND 99999
```

Ejemplo 6

En el Ejemplo 6 se muestra CondiciónFiltro en el formulario de [NOT] EXISTS (Subconsulta)

Este ejemplo comprueba si al menos una línea cumple los criterios de la subconsulta. Cuando la condición de filtro incluye EXISTS, la condición de filtro se evalúa como verdadera (.T.) a no ser que la subconsulta sea un conjunto vacío.

```
EXISTS ;
(SELECT * FROM orders WHERE customer.postalcode =                orders.postalcode)
```

Ejemplo 7

En el Ejemplo 7 se muestra CondiciónFiltro en el formulario de NombreCampo [NOT] IN Conjunto_Valor

Cuando una condición de filtro incluye IN, el campo debe contener uno de los valores antes de que el registro se incluya en los resultados de la consulta.

```
customer.postalcode NOT IN ("98052","98072","98034")
```

Ejemplo 8

En el Ejemplo 8 se muestra CondiciónFiltro en el formulario de NombreCampo [NOT] IN ( Subconsulta)

Aquí, el campo debe contener uno de los valores devueltos por la subconsulta antes de que su registro se incluya en los resultados de la consulta.

```
customer.cust_id IN ;
(SELECT orders.cust_id FROM orders WHERE orders.city="Seattle")
```

Ejemplo 9

En el Ejemplo 9 se muestra CondiciónFiltro en el formulario de NombreCampo [NOT] LIKE cExpresión

```
customer.country NOT LIKE "Reino Unido"
```

Esta condición de filtro busca cada uno de los campos que coinciden con cExpresión.

Puede utilizar el signo de porcentaje (%) y subrayado ( _ ) como parte de cExpresión. El signo de porcentaje representa a cualquier secuencia de caracteres desconocidos en la cadena. El subrayado representa un solo carácter desconocido en la cadena.

GROUP BY ColumnaGrupo [, ColumnaGrupo ...] Agrupa las filas de la consulta basándose en los valores de una o más columnas. ColumnaGrupo puede ser el nombre de un campo normal de una tabla, o un campo que incluya una función de campo SQL, o una expresión numérica indicando la posición de la columna en la tabla resultado (la columna más a la izquierda tiene el número 1).

HAVING CondiciónFiltro Especifica una condición de filtro que los grupos deben satisfacer para quedar incluidos en el resultado de la consulta. HAVING debe utilizarse con GROUP BY. Puede incluir tantas condiciones de filtro como se deseen, conectadas con el operador AND u OR. También puede utilizar NOT para invertir el valor de una expresión lógica.

CondiciónFiltro no puede contener una subconsulta.

Una cláusula HAVING sin una cláusula GROUP BY actúa como una cláusula WHERE. Puede utilizar alias locales y funciones de campo en la cláusula HAVING. Utilice una cláusula WHERE para acelerar el rendimiento si su cláusula HAVING no contiene funciones de campo. No olvide que la cláusula HAVING debería de aparecer antes de una cláusula INTO porque, de lo contrario, se producirá un error de sintaxis.

[UNION [ALL] ComandoSELECT] Combina el resultado final de una SELECT con el resultado final de otra SELECT. De forma predeterminada, UNION comprueba el resultado combinado y elimina las filas duplicadas. Puede utilizar paréntesis para combinar múltiples cláusulas UNION.

Utilice la palabra clave opcional ALL para impedir que UNION elimine filas duplicadas de los resultados combinados.

Las cláusulas UNION siguen las reglas siguientes:

· No puede utilizar UNION para combinar subconsultas.
· La salida de ambos SELECT debe tener el mismo número de columnas.
· Cada columna de los resultados de la consulta de un SELECT debe tener el mismo tipo de dato y anchura que su columna correspondiente en el otro SELECT.
· Únicamente el SELECT final puede tener una cláusula ORDER BY, que debe referirse a las columnas de salida por su número. Si se incluye otra cláusula ORDER BY, afectará al resultado completo.

También puede usar la cláusula UNION para simular una combinación externa.

Cuando combina dos tablas en una consulta, solamente se incluyen en la salida los registros que tengan valores coincidentes en los campos de combinación. Si un registro de la tabla primaria no tiene un registro correspondiente en la tabla secundaria, el registro de la tabla primaria no se incluye en la salida. Una combinación externa le permite incluir todos los registros de la tabla primaria en la salida, junto con los registros coincidentes de la tabla secundaria. Para crear una combinación externa en Visual FoxPro, necesita utilizar un comando SELECT anidado, como en el siguiente ejemplo:

```
SELECT customer.company, orders.order_id, orders.emp_id ;
        FROM customer, orders ;
        WHERE customer.cust_id = orders.cust_id ;
UNION ;
        SELECT customer.company, " ", " " ;
        FROM customer ;
        WHERE customer.cust_id NOT IN ;
        (SELECT orders.cust_id FROM orders)
```

Nota Asegúrese de incluir el espacio que aparece justo delante de cada punto y coma. De lo contrario, recibirá un error.

La sección del comando situada antes de la cláusula UNION selecciona los registros de ambas tablas que contienen valores coincidentes. Las empresas cliente que no tengan facturas asociadas no se incluyen. La sección del comando situada tras la cláusula UNION selecciona los registros de la tabla customer que no tienen registros coincidentes en la tabla orders.

En lo que respecta a la segunda sección del comando, observe lo siguiente:

· La instrucción SELECT incluida entre paréntesis se procesa en primer lugar. Esta instrucción da como resultado una selección de todos los números de clientes de la tabla orders.
· La cláusula WHERE busca todos los números de cliente de la tabla customer que no están en la tabla orders. Puesto que la primera sección del comando proporcionó todas las empresas que tenían un número de cliente en la tabla orders, todas las empresas de la tabla customer están incluidas en los resultados de la consulta.

· Puesto que las estructuras de las tablas incluidas en UNION deben ser idénticas, hay dos marcadores de posición en la segunda instrucción SELECT para representar orders.order_id y orders.emp_id de la primera instrucción SELECT.

Nota Los marcadores de posición deben ser del mismo tipo que los campos que representan. Si el campo es de tipo Date, el marcador de posición deberá ser { / / }. Si el campo es de tipo Character, el marcador de posición deberá ser la cadena vacía ("").

ORDER BY Elemento_Orden Ordena el resultado de la consulta basándose en los datos de una o varias columnas. Cada Elemento_Orden debe corresponder a una columna del resultado de la consulta, y puede ser uno de los siguientes:

· Un campo de una tabla FROM que también es un elemento de selección en la cláusula principal SELECT (no en una subconsulta).
· Una expresión numérica que indica la ubicación de la columna en la tabla resultante. (La columna de la izquierda es la número 1.)

ASC Especifica un orden ascendente para los resultados de la consulta, de acuerdo con el elemento o los elementos de orden, y es el valor predeterminado para ORDER BY.

DESC Especifica un orden descendente para los resultados de la consulta.

Los resultados de la consulta aparecerán desordenados si no especifica un orden con ORDER BY.

Comentarios

SELECT es un comando SQL que está incorporado en Visual FoxPro como cualquier otro comando de Visual FoxPro. Cuando utiliza SELECT para componer una consulta, Visual FoxPro interpreta la consulta y recupera los datos especificados de las tablas. Puede crear una consulta SELECT:

· En la ventana Comandos
· En un programa Visual FoxPro (como cualquier otro comando de Visual FoxPro)
· El Diseñador de consultas

Cuando emite SET TALK ON y ejecuta SELECT, Visual FoxPro muestra la duración de la consulta y el número de registros del resultado. _TALLY contiene el número de registros del resultado de la consulta.

SELECT no respeta la condición de filtro actual especificada con SET FILTER.

Una subconsulta, a la que se hace referencia en los argumentos siguientes, es un comando SELECT dentro de otro SELECT y debe incluirse entre paréntesis. Puede tener múltiples subconsultas al mismo nivel (no anidadas) en la cláusula WHERE (consulte esta sección de los argumentos). Las subconsultas pueden contener múltiples condiciones de combinación.

Cuando se obtiene el resultado de una consulta, las columnas se denominarán según las siguientes reglas:

· Si un elemento seleccionado es un campo con un nombre único, el nombre de la columna de resultado es el nombre del campo.
· Si hay más de un elemento seleccionado con el mismo nombre, se añadirán un signo de subrayado y una letra al nombre de la columna. Por ejemplo, si una tabla llamada Cliente tiene un campo llamado CALLE, y una tabla llamada Empleados también tiene un campo llamado CALLE, las columnas de resultado se llamarán Extensión_A y Extensión_B (CALLE _A y CALLE _B). En el caso de un elemento seleccionado con un nombre de 10 caracteres, se truncará el nombre para añadir el símbolo de subrayado y la letra. Por ejemplo, DEPARTMENT se convertiría en DEPARTME_A.

· Si un elemento seleccionado es una expresión, su columna de resultado se llamará EXP_A. Cualquier otra expresión recibirá el nombre de EXP_B, EXP_C, y así sucesivamente.
· Si un elemento seleccionado contiene una función de campo como, por ejemplo, COUNT( ), la columna de resultado se llamará CNT_A. Si otro elemento seleccionado contiene SUM( ), su columna de resultado se llamará SUM_B.

Funciones definidas por el usuario con SELECT Aunque la utilización de funciones definidas por el usuario en la cláusula SELECT ofrece unas ventajas evidentes, también debería tener en cuenta las siguientes limitaciones:

· Es posible que la velocidad de realización de las operaciones con SELECT se vea limitada por la velocidad a la que se ejecutan las funciones definidas por el usuario. Las manipulaciones de un gran volumen que impliquen funciones definidas por el usuario se pueden realizar mejor utilizando funciones API y funciones definidas por el usuario escritas en C o en lenguaje ensamblador.
· No se puede prever nada acerca de la entrada/salida de Visual FoxPro (E/S) ni del entorno de la tabla en funciones definidas por el usuario invocadas a partir de SELECT. Generalmente, no se puede saber qué área de trabajo se ha seleccionado, ni el nombre de la tabla actual, ni los nombres de los campos que se están procesando. El valor de dichas variables depende del lugar específico, dentro del proceso de optimización, en el que se invoque la función definida por el usuario.

· En funciones definidas por el usuario invocadas desde SELECT, no es seguro cambiar la E/S de Visual FoxPro ni el entorno de la tabla. Por norma general, los resultados son impredecibles.
· La única forma segura de pasar valores a funciones definidas por el usuario invocadas desde SELECT es mediante la lista de argumentos pasada a la función cuando es invocada.
· Si prueba y descubre una manipulación teóricamente prohibida que funciona correctamente en una versión determinada de FoxPro, eso no significa que también funcione en versiones posteriores.

Salvando dichas limitaciones, las funciones definidas por el usuario son aceptables en la cláusula SELECT. Sin embargo, recuerde que la utilización de SELECT puede ralentizar el rendimiento.

Las siguientes funciones de campo están disponibles para ser utilizadas con un elemento seleccionado que sea un campo o una expresión que implique a un campo:

· AVG(Elemento_Selección), que realiza una media de una columna de datos numéricos.
· COUNT(Elemento_Selección), que cuenta el número de elementos seleccionados en una columna. COUNT(*) cuenta el número de filas en el resultado de la consulta.
· MIN(Elemento_Selección) determina el menor valor de Elemento_Selección en una columna.
· MAX(Elemento_Selección) determina el mayor valor de Elemento_Selección en una columna.

· SUM(Elemento_Selección) que proporciona el total de la suma de una columna de datos numéricos.

No se pueden probar las funciones de campo.

Combinaciones Visual FoxPro acepta sintaxis de combinación de 1992 SQL ANSI, lo que le permite crear consultas que vinculen las filas en dos o más tablas mediante la comparación de los valores de campos especificados. Por ejemplo, una combinación interna selecciona filas procedentes de dos tablas sólo cuando los valores de los campos combinados son iguales. Visual FoxPro admite combinaciones anidadas.

Dado que SQL se basa en la teoría de conjuntos matemática, se puede representar a cada tabla con un círculo. La cláusula ON que especifica las condiciones de la combinación determina el punto de intersección, el cual representa el conjunto de filas que coinciden. En el caso de una combinación interna, la intersección tendrá lugar en el interior o en una parte “interna” de los dos círculos. Una combinación externa incluye tanto las filas coincidentes que se han encontrado en la sección de intersección interna de las tablas, como las filas de la parte externa del círculo a la izquierda, o a la derecha, de la intersección.

Importante Tenga presente la siguiente información a la hora de crear condiciones de combinación:

· Si incluye dos tablas en una consulta y no especifica una condición de combinación, cada registro de la primera tabla se combinará con cada registro de la segunda tabla hasta que surtan efecto las condiciones del filtro. Una consulta tal puede producir unos resultados interminables.

· Sea prudente al utilizar, en condiciones de combinación, funciones tales como DELETED( ), EOF( ), FOUND( ), RECCOUNT( ), y RECNO( ), que aceptan un área de trabajo o un alias opcional. La inclusión de un alias o de un área de trabajo en dichas funciones puede producir resultados inesperados. SELECT no utiliza sus áreas de trabajo; realiza lo equivalente a USE ... AGAIN. Las consultas de una única tabla que utilizan estas funciones sin un área de trabajo o un alias opcional, tendrán resultados correctos. De todas formas, las consultas de varias tablas que utilicen dichas funciones (incluso sin un área de trabajo o un alias opcional) pueden tener resultados inesperados.

· Sea prudente al combinar tablas que contengan campos vacíos porque Visual FoxPro concuerda campos vacíos. Por ejemplo, si combina CUSTOMER.ZIP e INVOICE.ZIP, y CUSTOMER contiene 100 códigos postales vacíos e INVOICE contiene 400 códigos postales vacíos, el resultado de la consulta contendrá 40.000 registros más, como resultado de los campos vacíos. Use la función EMPTY( ) para eliminar los registros vacíos del resultado de la consulta.

Para obtener más información sobre combinaciones, consulte “Definición y modificación de condiciones de combinación” en el capítulo 8, “Creación de vistas”, en el Manual del programador.