Universidad Adventista de Centro América

MANUAL
DE
EVENTOS Y PROPIEDADES

de VISUAL BASIC

por CESAR MIRANDA

Escuela de Ingeniería de Sistemas Computacionales

corregido y aumentado por

Lic. Jair del Valle

UNADECA, Alajuela Costa Rica
25 de Octubre de 2000

Tabla de Contenido

Los Eventos en Visual Basic

Un evento es una circunstancia ante la cual un programa debe responder realizando alguna instrucción, por ejemplo, en el momento en que un usuario da click sobre un objeto de la pantalla. Esta capacidad le da al programa una serie de ventajas, como la que puede ser el mostrarse de una manera más amigable y entendible para los usuarios o tener una estructura más <<compacta>>.

Con anterioridad, y en lenguajes de programación no visuales como Pascal y C++, las respuestas a los eventos debían de ser planificadas completamente por el programador, ya que estos lenguajes no proporcionaban objetos capaces de controlar o interactuar con los dispositivos de hardware u otros programas. Con la llegada de los lenguajes visuales como Delphi y Visual Basic se tiene la posibilidad de no preocuparse por otorgar al programa de órdenes de código que le permitan conocer cuándo se producen los eventos, ya que se proporcionan objetos que tienen la capacidad de responder por sí mismos y de una manera independiente.

Entre los eventos más importantes están los movimientos del mouse, el 'click' derecho e izquierdo, la opresión de teclas, el cerrar o abrir una ventana, etc... , de los cuales algunos son producidos porque el usuario necesita que el programa realice cierta función y manipula los dispositivos de entrada y otros son completamente producidos por las instrucciones internas del programa, como el cargar una ventana u ocultar algún otro objeto.

El propósito de la presente obra es dar una explicación de los eventos más importantes que se encuentran en Visual Basic y cómo pueden ser utilizados para hacer programas más amigables y funcionales.

El evento <CLICK>

Este evento se origina, obviamente, al dar click sobre algún objeto de la ventana o de la ventana misma. Algunos de los objetos que responden a este evento son los COMMAND, LABEL, TEXT, FORM, IMAGE,COMBO, LIST , etc... los cuales ejecutan un procedimiento que tiene más o menos el siguiente formato:

```
Private Sub  <nombre del objeto> _Click()

End Sub
```

Como puede verse, se trata de un procedimiento privado del objeto en cuestión y el cual se puede programar simplemente dando click en el objeto. Cada vez que se da click sobre uno de estos objetos se manda a llamar este procedimiento y así se puede ejecutar todas las veces que sea necesario sin necesidad de tener grandes cantidades de código disperso a lo largo del programa.

Ejemplo de código que muestra una imagen distinta al dar click.

```
Private Sub  image1 _Click()

    I=I+1           'I es una variable global de tipo integer
    If  I>2 then
       I=0
    End if
    Select case I
        0:          image1.picture="C:\my documents\New York.bmp"
        1:          image1.picture="C:\my documents\San Francisco.bmp"
        2:          image1.picture="C:\my documents\Chicago.bmp"

    End select

End Sub
```

El evento <LOAD>

Atendiendo a su significado en inglés, el evento load se produce al 'cargar' o mostrar algunos objetos como las formas. Este es un ejemplo de evento que se produce sin que haya intervención del usuario sino que es puramente producido por las órdenes internas del programa.

Este evento es útil sobre todo para inicializar variables globales u objetos que deben de funcionar de una manera determinada desde el momento en que empiezan a recibir órdenes. El formato de este evento es el siguiente:

```
Private Sub < nombre del objeto >_load()

End Sub
```

El evento load de una forma se ejecuta una sóla vez al cargarse esta y no se vuelve a realizar a menos que se vuelva a dar una instrucción para cargarla de nuevo.

Ejemplo de código que muestra el nombre y la fecha al iniciarse una ventana

```
Private Sub form_load()
    N="César"
    A="Miranda"
    Text1.text= "Nombre: " + N
    Test2.text= "Apellido: " +  A
    Text3.text= date
End Sub
```

El evento <CHANGE>

Este evento lo poseen objetos sobre los cuales el usuario puede introducir información a travez del teclado, por ejemplo, el TEXT y el COMBO. Se trata aquí de una manera de responder ante lo que está tratando de introducir el usuario. El formato de este evento es:

Private Sub < *nombre del objeto* >_change()

End Sub

Es importante notar que el objeto no esperará una segunda orden para ejecutar su código: este se realizará ante el más mínimo cambio y sólo cuando termine realizará la siguiente instrucción.

Ejemplo de código que despliega una ventana nueva en el instante que se lee la palabra 'Salir' en un text

```
Private Sub text1_change()
    If text1.text = "Salir" then
       Msgbox  "Encontraste la clave"
       VentanaNueva.show                    'Ventananueva es el nombre de una forma distinta
    End if
End Sub
```

El evento <GOTFOCUS>

En términos estrictamente computacionales, se define Focus como el indicador del objeto que está en dispocisión de responder a un evento. El focus permite al programa definir qué objeto responderá ante un evento en un momento determinado, o de otra manera todos los demás responderían al mismo tiempo.

Este vento GOTFOCUS se produce al dar click sobre un objeto, por lo que siempre cuando se da el evento click también se dará el evento gotfocus, sin embargo este último también se produce al oprimir la tecla TAB. El formato de este procedimiento es:

Private Sub < *nombre del objeto* >_gotfocus()

End Sub

Se puede usar esta propiedad para preparar al objeto en cuestión a responder de buena manera a los requirimientos que se le harán a continuación, por ejemplo, deshabilitarse en el momento en que se trata de introducir información en un text.

Ejemplo de código que deshabilita un TEXT a ciertas horas del día usando el evento gotfocus

```
Private Sub text1_gotfocus()
   A = hour (time)
   If ( a = 17) or ( a= 11 ) then
      Text1.enabled = false
   End if
End Sub
```

El evento <LOSTFOCUS>

Este evento se produce cuando un objeto tiene el focus y lo pierde porque se ha dado click en otro objeto o porque se ha presionado la tecla TAB. Si se ha dado click en otro objeto y por esta razón el objeto pierde el focus, el procedimiento lostfocus se ejecutará antes que el evento CLICK del otro objeto. El formato para este procedimiento es:

Private Sub < *nombre del objeto* >_lostfocus()

End Sub

Objetos que hacen uso de este evento son el TEXT, el COMMAND,el COMBO, el LIST, etc..., sin embargo el IMAGE y el LABEL no lo poseen.

Ejemplo de código para desactivar un TEXT cuando lo último en escribirse fue "GOOD BYE".

```
Private Sub text1_lostfocus()

   If text1.text = "GOOD BYE" then
      Text1.enabled = false
      Command1.enabled = false
   Else
      Msgbox "Puedes seguir introduciendo información"
   End if

End Sub
```

El evento <DBLCLICK>

Este evento, como es de esperarse, se produce al dar doble click en un objeto que tiene esta propiedad. Siguiendo los patrones generalmente aceptados, el click se usa para ejecutar botones, abrir menúes, etc..., y el doble click para ejecutar programas, abrir archivos o cargar ventanas. Sin embargo, Visual Basic da la posibilidad utilizar este evento para cualquier aplicación , incluso aquellas que por lo general 'son propias del evento click'. No obstante, en haras de desarrollar programas amigables y lo más entendibles posible es conveniente seguir estos parámetros normalmente aceptados.

El formato de este evento es el siguiente:

Private Sub < *nombre del objeto* >_dblclick()

End Sub

Este evento es una buena opción para sobrecargar objetos, es decir, la posibilidad de realizar distintas tareas dependiendo de si se ha dado un click o un doble click. Por ejemplo, sería apropiado en una aplicación de manipulación de archivos tener un botón que en el evento click despliegue información de un archivo y al darse doble click en él abra el archivo.

Ejemplo de código que permite elegir entre dos imágenes al darse doble click

```
Private Sub image1_dblclick()

   If primera then
   Image1.picture = "C:\my documents\una foto.bmp"
   Primera = false
   Else
      Image1.picture = "C:\my documents\my photos\otra foto.bmp"
      Primera = true
   End if

End Sub
```

El evento <KEYPRESS>

Siendo el teclado uno de los dispositivos de entrada más importantes por medio del cual el usuario dará instrucciones al programa, es de vital importancia que este responda de una manera eficaz a las pulsaciones que se producen en este dispositivo. Es importante diferenciar las teclas de control de las demás, ya que teclas como enter y tab no introducen caracteres al sistema sino que actúan como verdaderos comandos que se mandan a ejecutar directamente del teclado.

Cuando se habla del evento KEYPRESS es muy importante tener en cuenta que se produce en el objeto que tiene el focus. A diferencia del click, que le otorga el focus al objeto que está apuntando, el oprimir una tecla de caracter no le otorga el focus a un objeto en particular. El formato del evento keypress es el siguiente:

Private Sub <*nombre del objeto*>_KeyPress(KeyAscii As Integer)

End Sub

Como puede verse, este procedimiento difiere de muchos otros un tanto más simple ya que devuelve un parámetro que es de suma importancia, ya que contiene el número ASCII para la tecla que se ha oprimido. Este parámetro es de tipo integer y por lo tanto puede ser usado para hacer operaciones matemáticas o servir de variable de control para una estructura FOR.

Ejemplo de código que al oprimirse una tecla despliega todas las letras anteriores a ella en el código Ascii

```
Private Sub Form_KeyPress(KeyAscii As Integer)

    List1.clear
    List1.additem ( "Lista de letras anteriores a esa" )
    For I = 0 to KeyAscii
       List1.additem ( chr ( keyascii ) )
    Next

End Sub
```

El evento < MOUSEMOVE >

El evento mousemove se origina cuando el apuntador del mouse cambia en la pantalla. Este evento es muy utilizado en aplicaciones de dibujo debido a sus características, que permiten saber con precisión las coordenadas en que se produce el movimiento del mouse.

La sintaxis de este evento es como sigue:

Private Sub <nombre del objeto>_MouseMove(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)

End sub

Este procedimiento regresa cuatro parámetros, los cuales son button,shift,x e y.

Button: Es un entero que representa el estado de los botones del mouse y equivale a un

- 0 si ninguno está oprimido
- 1 click izquierdo
- 2 click derecho
- 3 click derecho e izquierdo
- 4 botón central
- 5 click izquierdo más botón central
- 6 click derecho más botón central
- 7 click izquierdo más click derecho más botón central

Shift: Es un entero que representa el estado de las teclas control,shift y alt, siendo estos sus valores:

- 0 los tres están sin oprimir

1 shift está oprimido

2 control

3 control+shif

4 alt

5 alt+shift

6 alt+control

7 alt+control+shift

Y: Un número que regresa la coordenada en y de la pantalla en la que se encuentra el apuntador.

X: Un número que regresa la coordenada en x de la pantalla en la que se encuentra el apuntador.

Ejemplo de código que permite dibujar una línea que representa el movimiento del mouse

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  ForeColor = RGB(X, Y, 0)
  PSet (X, Y)
End Sub
```

El evento < KEYDOWN >

Al igual que con el evento KeyPress, el KeyDown se origina al oprimir una tecla, pero en el momento justo en que se encuentra oprimida. La sintaxis para este procedimiento es:

```
Private Sub < nombre del objeto >_keydown(KeyCode As Integer, Shift As Integer )
End Sub
```

Como puede verse, este evento añade el parámetro Shift, cuyos valores representan lo siguiente:

0 los tres están sin oprimir

1 shift está oprimido

2 control

3 control+shif

4 alt

5 alt+shift

6 alt+control

7 alt+control+shift

Ejemplo de código que permite dibujar una línea al oprimir cualquier tecla usando Keydown, y alternando colores con las teclas alt, control y shift

```
Private Sub text1_keydown(KeyCode As Integer, Shift As Integer )

  Select case shift
     Case  1:        ForeColor = RGB(100,0, 0)
     Case  1:        ForeColor = RGB( 0,100,0)
     Case  1:        ForeColor = RGB( 0,0,100 )
  End select
  Dim x as integer
  For x=0 to 100
  PSet ( X , 7 )
next

End Sub
```

El evento < KEYUP >

Al igual que con el evento KeyPress, el KeyUp se origina al oprimir una tecla, pero en el momento justo en que se suelta. La sintaxis para este procedimiento es:

Private Sub < *nombre del objeto* >_keyup(KeyCode As Integer, Shift As Integer)

End Sub

Como puede verse, este evento añade el parámetro Shift, cuyos valores representan lo siguiente:

0 los tres están sin oprimir

1 shift está oprimido

2 control

3 control+shif

4 alt

5 alt+shift

6 alt+control

7 alt+control+shift

Ejemplo de código que permite cambiar el color de un text mientras se mantenga cualquier tecla oprimida usando Keydown y Keyup

```
Private Sub text1_keydown(KeyCode As Integer, Shift As Integer )
    If keycode > 100 then
     Text1.forecolor=vbred
    Else
     Text1.forecolor=vbblue
    End if
End Sub

Private Sub text1_keyup(KeyCode As Integer, Shift As Integer )
    Text1.forecolor=vbblack
End Sub
```

El evento <MOUSEUP>

El evento mouseup se origina cuando el apuntador del mouse es movido hacia arriba por el usuario. El evento mouseup tiene una sintaxis muy similar al mouse move y se diferencia de este únicamente en que se activa sólo si el movimiento del mouse se da hacia arriba.

La sintaxis de este evento es como sigue:

Private Sub <nombre del objeto>_Mouseup(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)

End sub

Este procedimiento regresa cuatro parámetros, los cuales son button,shift,x e y.

Button: Es un entero que representa el estado de los botones del mouse y equivale a un

9 si ninguno está oprimido

10 click izquierdo

11 click derecho

12 click derecho e izquierdo

13 botón central

14 click izquierdo más botón central

15 click derecho más botón central

16 click izquierdo más click derecho más botón central

Shift: Es un entero que representa el estado de las teclas control,shift y alt, siendo estos sus valores:

8	los tres están sin oprimir
9	shift está oprimido
10	control
11	control+shif
12	alt
13	alt+shift
14	alt+control
15	alt+control+shift

Y: Un número que regresa la coordenada en y de la pantalla en la que se encuentra el apuntador.

X: Un número que regresa la coordenada en x de la pantalla en la que se encuentra el apuntador.

Ejemplo de código que permite dirigir al usuario hacia un punto específico de la pantalla usando mouse up y mouse down

```
Private Sub Form_Mouseup   (Button As Integer, Shift As Integer, X As Single, Y As Single)
   If  lapocision < y  then
      Label1.forecolor = vbblue
      Label1.caption = "Enfriándote"
  End if
End Sub

Private Sub Form_Mousedown   (Button As Integer, Shift As Integer, X As Single, Y As Single)
   If  lapocision > y  then
      Label1.forecolor = vbred
      Label1.caption = "Calentándote"
  End if
End Sub
```

El evento <MOUSEDOWN>

El evento mousedown se origina cuando el apuntador del mouse es movido hacia abajo por el usuario. Al igual que con el mouseup y el mousemove, este evento permite conocer la pocisión del mouse en la pantalla así como el estado de las teclas ALT , SHIFT y CONTROL.

La sintaxis de este evento es como sigue:

Private Sub <nombre del objeto>_Mousedown(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)

End sub

Este procedimiento regresa cuatro parámetros, los cuales son button,shift,x e y.

Button: Es un entero que representa el estado de los botones del mouse y equivale a un

- 17 si ninguno está oprimido
- 18 click izquierdo
- 19 click derecho
- 20 click derecho e izquierdo
- 21 botón central
- 22 click izquierdo más botón central
- 23 click derecho más botón central
- 24 click izquierdo más click derecho más botón central

Shift: Es un entero que representa el estado de las teclas control,shift y alt, siendo estos sus valores:

- 16 los tres están sin oprimir
- 17 shift está oprimido
- 18 control
- 19 control+shif
- 20 alt
- 21 alt+shift
- 22 alt+control
- 23 alt+control+shift

Y: Un número que regresa la coordenada en y de la pantalla en la que se encuentra el apuntador.

X: Un número que regresa la coordenada en x de la pantalla en la que se encuentra el apuntador.

Ejemplo de código que permite colocar un label en la posición del apuntador usando mouseup y mousedown

```
Private Sub Form_Mousedown  (Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label1.top = y
End Sub

Private Sub Form_Mouseup  (Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label1.top = y
End Sub
```

Propiedades de Objetos en Visual Basic

Cada objeto en Visual Basic tiene un conjunto de propiedades que le son inherentes y que difieren en mayor o menor grado con las de otros objetos. Estas propiedades comprenden su altura, anchura, color, etc... y tienen unos valores predeterminados que varían de objeto a objeto. Esto, sin embargo, no significa que estas propiedades son constantes sino que más bien el hecho de que se encuentren agrupadas dentro de cada objeto es para que sean más fáciles de manipular y acceder.

En el momento en que se inserta un objeto en una forma de Visual Basic se establecen unos valores predeterminados para cada una de sus propiedades. Aunque algunas de ellas no se pueden cambiar, la mayoría pueden ser predefinidas por el programador, esto por medio de la tabla de propiedades del objeto que puede ser desplegada dando clic derecho sobre el objeto en cuestión. Además de esta manera, se pueden dar nuevos valores a las propiedades de cada objeto directamente desde el código de fuente, siendo esta la manera que más resultados provee ya que permite variar dichas propiedades de acuerdo a los eventos que se desarrollen durante la ejecución del programa.

Con anterioridad, y en estilos de programación no orientados a objetos, era realmente difícil tener cierto grado de flexibilidad en el programa, ya que no se podía utilizar una misma porción de código para tareas similares y el cuerpo del programa se convertía en un verdadero 'mar de instrucciones'. El hecho de que lenguajes como Visual Basic posean ahora un estilo de programación orientado a objetos facilita en general las tareas de programación y disminuye en no poco el tamaño del código fuente.

A continuación se dará una breve explicación de algunas de las propiedades más comunes en Visual Basic y ejemplos de cómo pueden ser utilizadas para realizar ciertas tareas de una manera más fácil y abreviada.

Propiedad < ENABLED >

Imagine que estuviera desarrollando una ventana con un botón para eliminar registros de una base de datos, pero existe la posibilidad que en un momento determinado durante la ejecución del programa se diera la posibilidad de borrar un registro que no debe ser eliminado bajo ninguna circunstancia. En esta situación el programador debe de pensar en algún método para desactivar esa posibilidad en el momento en que se visualice el registro que no debe ser borrado. Esto se puede lograr fácilmente usando la propiedad ENABLED del botón y con una sola línea de código como la siguiente:

COMMAND1.ENABLED = FALSE

Como era de esperarse, desactivar ese botón sin disponer de esta propiedad hubiera llevado tal vez una gran cantidad de trabajo y una cantidad bastante apreciable de líneas de código. La propiedad ENABLED facilita este trabajo y está presente en todas las formas y controles de Visual Basic. La sintaxis para asignar un valor a esta propiedad es:

*Objeto.*Enabled = *true* o *false*

Como puede verse, el valor que se le asigna a esta propiedad es de tipo boolean y puede hacerse en cualquier lugar del código del programa. La aplicación más común para esta propiedad es la de desactivar objetos dependiendo de los eventos que se desarrollen durante la ejecución del programa, esto con el objetivo de anticiparse a posibles errores.

Ejemplo de código que habilita un botón cuando se ha introducido la palabra clave "VER"

```
Private Sub Command1_Click()

  If text1.Text = "VER" Then
    Command1.Enabled = True
  End If

End Sub
```

La propiedad < VISIBLE >

Al igual que la propiedad Enabled, la propiedad VISIBLE tiene un valor de tipo boolean que representa, como es de esperar, uno de dos estados: visible cuando es TRUE e invisible cuando es FALSE. Se puede saber si un objeto es visible con la siguiete comparación:

```
If objeto.visible then

   Msgbox " Puedes ver este objeto? "

End if
```

La propiedad visible puede asignarse en cualquier lugar del programa con una sentencia como esta:

*Objeto.*visible = false

Esta propiedad es muy útil cuando se quiere ocultar información durante cierto tiempo o dependiendo de los eventos que se produzcan durante la ejecución del programa. Es muy importante tener en cuenta que si

se ha declarado un objeto con su propiedad VISIBLE como false, no se podrá visualizar ese objeto nunca más a menos que en otro lugar se de una sentencia para hacer esa propiedad verdadera de nuevo.

Ejemplo de código que muestra una serie de imágenes distinta según la pocisióndel apuntador

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    If x > 100 then
       If  y > 50 then
            For I =  0 to 5
               Image1( I ). Visible = True
               Image2( I ). Visible = False
            Next
       End  if
    Else
        If  y < 50 then
            For I =  0 to 5
               Image1( I ). Visible = False
               Image2( I ). Visible = True
            Next
       End  if
    End  if


End Sub
```

La propiedades < CAPTION y TEXT >

Estas dos propiedades tienen en común que representan una serie de caracteres visualizados por el objeto en cuestión, por lo que puede verse que se trata de un tipo de dato STRING. La propiedad CAPTION pertenece a objetos como LABEL, FORM, FRAME, COMMAND, etc.., pero el objeto TEXT tiene la propiedad TEXT que no obstante desempeña la misma función.

Se puede asignar a estas propiedades valores de tipo string desde la ventana de propiedades del objeto o directamente desde la fuente de código. Esto se logra con una sentencia como la siguiente:

```
Label1.Caption = "Aquí hay un mensage"
Text1.text = Alfa   ( donde alfa es una variable de tipo string )
```

Esta propiedad es útil, por ejemplo, cuando se usan objetos multifuncionales, ya que un botón podría ser usado tanto para borrar como para escribir según sea la ocación, pero en un momento su Caption debe ser "BORRAR" y en el otro "ESCRIBIR", según sea la función que está realizando.

Ejemplo de código que despliega en un Label la función que se está realizando

```
Private Sub Command1_Click()

    Label1.caption = "Copiando los primeros 5 elementos de la  lista"
    For I = 0 to 4
       List1.additem ( list2.list ( i ) )
```

```
    Next
    Label1.caption = "Clasificando la información"

    For I = 0 to 4
       If (list1.list (i)<> "Allan") or (list1.list(i)<>"Ernesto") then
        List3.additem ( list1.list ( i ) )
       End if
    Next
    Label1.caption = "Operación realizada con éxito"

End Sub
```

La propiedad <LISTINDEX>

Las listas y los combobox son objetos de suma utilidad cuando se trata de conjuntos de información que está relacionada entre sí, por ejmplo, los nombres de los empleados de una empresa. Estos objetos funcionan de una manera muy similar a los arreglos, es decir, que se puede accesar cualquier elemento de la lista empleando un subíndice que le es distintivo. Sin embargo, estos objetos incorporan la propiedad LISTINDEX la cual permite al usuario seleccionar cualquier elemento de la lista usando el mouse. Al dar click en un elemento determinado, la propiedad LISTINDEX adquiere un valor de tipo integer que representa el subíndice del elemento sobre el que se dió click.

Se puede obtener el valor de la propiedad LISTINDEX de un objeto por medio de una línea de código así:

```
I = List1.listindex          ( donde I debe ser una variable de tipo integer )
```

Además de el LIST y el COMBOBOX, otros objetos como el DRIVE, DIR y FILE, que actúan como verdaderas listas de archivos o directorios, tienen la propiedad Listindex, lo que les proporciona una gran flexibilidad también.

Es muy importante tener en cuenta que un elemento de la lista debe ser seleccionado para que listindex obtenga el valor de su subíndice, o de otra manera LISTINDEX tendrá un valor de –1.

Ejemplo de código que cambia el color de un label de acuerdo al elemento de una lista que está seleccionado

```
Private Sub List1_Click()

    Num=list1.listindex
    If list1.listindex = -1 then
       Msgbox " Seleccione un elemento... "
    Else
          Select case Num
             Case 0:     label1.forecolor = vbblack
             Case 1:     label1.forecolor = vbwhite
             Case 2:     label1.forecolor = vbgreen
             Case 3:     label1.forecolor = vbblue
             Case 4:     label1.forecolor = vbgray
             Case 5:     label1.forecolor = vbred
             Case 6:     label1.forecolor = vbyellow
             Case 7:     label1.forecolor = vbwhite
```

```
        End select
    End if

End Sub
```

La propiedad < STRETCH >

La propiedad STRETCH , que en inglés significa 'estirar ' , permite a los objetos IMAGE y PICTUREBOX adaptarse al tamaño de la imágen que está desplegando. El valor predeterminado para esta propiedad es FALSE, y puede ser cambiado en cualquier momento de la ejecusión del programa de la siguiente manera:

```
Image1.strecth = True
```

Si la propiedad STRETCH es verdadera, la imágen ocupará el tamaño total del control de imágen sin importar las proporciones que deba adquirir para ello. Esto ovbiamente puede traer problemas de estética si el tamaño del cuadro no corresponde bien con el de la imágen. Por otro lado, este problema no se presentará si la propiedad STRETCH es puesta en falso, ya que en este caso la imágen aparecerá en su tamaño y proporción original. Sin embargo, cuando se requiere llenar un cuadro determinado con una imágen más pequeña o sobre todo si es más grande, la propiedad STRETCH debe ser puesta en true.

Ejemplo de código que encoge la imágen si es más grande que el cuadro en donde se verá.

```
Private Sub Command1_Click()
    Image1.picture = "C:\una foto.bmp"
    If (image1.heigth > image1.picture.heigth) or (image1.width > image1.picture.width) then
        Image1.strecth = True
    End if
End Sub
```

La propiedad < HEIGHT >

Atendiendo a su significado en el idioma inglés, la propiedad HEIGHT regresa un entero que representa la altura de un objeto medida en pixeles. Esta es una propiedad que la tienen la mayoría de los objetos, desde las formas hasta las imágenes. Se puede obtener la altura de un objeto en un momento determinado con una sentencia de código así:

```
Altura = Image1.height
```

donde altura es una variable de tipo integer que almacenará la cantidad de pixeles que hay desde el inicio del objeto hasta su extremo derecho.

Debido a las necesidades que se presentan en el manejo de las imágenes, la propiedad HEIGHT se aplica muy bien a la hora de tener que ajustar las dimensiones de un control de imágen de acuerdo al tamaño del archivo que cargará. Si una imágen es más grande que el tamaño del cuadro donde se despliega, se puede decrementar la propiedad HEIGHT del control hasta que se ajuste al tamaño real de la imágen.

Ejemplo de código que agranda el tamaño de una imágen al dar click en un botón

```
Private Sub Command1_Click()
```

```
    Dim N as integer
    N = Text1.text      ' Aquí el usuario introduce el número de pixeles que quiere aumentar

    Image1.picture.height = Image1.picture.height + N

    Label1.caption = Image1.picture.height      ' Depliega el nuevo tamaño

End Sub
```

La propiedad < WIDTH >

Al diferencia de la propiedad Height, WIDTH regresa el número de pixeles que hay desde el extremo ixquierdo hasta el extremo derecho de un objeto. Este valor es de tipo integer y por lo tanto puede ser manipulado en operaciones matemáticas que sirven para calcular el ancho y el alto ideal de una imágen en particular.

Las siguientes sentencias usando la propiedad Width de un Command son válidas:

```
If Command1.width = 45 then
   Command1.height = 100
End if

Int = Command1.width + 100
```

Siempre que se asignen nuevos valores a WIDTH es necesario cuidar no se pierda el sentido de proporción en el objeto, por lo que la manipulación de WIDTH está muy ligada a HEIGHT y viceversa.

Ejemplo de código que mantiene las proporciones de un label al cambiar su ancho.

```
Private Sub Command1_Click()

    Proporcion = 0.4
    For I = 0 to 100
        Label1.height = I
        Label1.width = label1.height * proporcion
    Next

End Sub
```

La propiedad < BACKCOLOR >

Una de las características más importantes de todo objeto visual es su color de fondo, el cual puede ser definido por medio de la propiedad BACKCOLOR. Al asignar un color en especial a la propiedad backcolor de un objeto, este cambiará inmediatamente el color de su fondo al nuevo que se le está asignando. Lo anterior se puede conseguir por medio de la siguiente sentencia:

```
Label1.backcolor = vbred
```

En este ejemplo, el valor 'vbred' representa el rojo y está siempre antecedido de las siglas de Visual Basic. Así, para mencionar al color azul, se debe escribir vbblue.

Ejemplo de código para que el usuario elija un color para la ventana actual

```
Private Sub Form_Load()

    List1.additem ( " 0 : Negro " )
    List1.additem ( " 1 : Azul " )
    List1.additem ( " 2 : Rojo " )
    List1.additem ( " 3 : Verde " )
    List1.additem ( " 4 : Blanco " )

End Sub

Private Sub List1_Click()

    If  list1.listindex <> -1 then
    I = list1.listindex

        Select case I
            0 :  form1.backcolor = vbblack
            1 :  form1.backcolor = vbblue
            2 :  form1.backcolor = vbred
            3 :  form1.backcolor = vbgreen
            4 :  form1.backcolor = vbwhite

        End select

    End if
End Sub
```

La propiedad < FORECOLOR >

Una de las características más importantes de todo objeto visual es el color de su fuente, el cual puede ser definido por medio de la propiedad FORECOLOR. Al asignar un color en especial a la propiedad forecolor de un objeto, este cambiará inmediatamente el color de su fuente al nuevo que se le está asignando. Lo anterior se puede conseguir por medio de la siguiente sentencia:

```
Label1.forecolor = vbred
```

En este ejemplo, el valor 'vbred' representa el rojo y está siempre antecedido de las siglas de Visual Basic. Así, para mencionar al color azul, se debe escribir vbblue.

Ejemplo de código para cambiar el color de la fuente de la ventana de acuerdo al movimiento del mouse

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Form1.forecolor = rgb ( x,y,x )

End Sub
```

Conclusión

La programación orientada a objetos es una de las mayores ventajas que ofrecen los lenguajes de cuarta generación, ya que entre otras cosas permiten controlar de una manera eficaz y sencilla los eventos que se producen en el programa y porque hace de la programación una tarea ordenada y más entendible.

Para que un programa sea lo más amigable posible , es necesario que sepa responder bien a los eventos a los que está sujeto o en los que se espera que realice una tarea determinada. Entre más eventos sepa manejar un lenguaje de programación, más eficaz será y más fácil de programar. Visual Basic incorpora gran cantidad de eventos para cada uno de los controles de que dispone, haciendo de la manipulación de estos una tarea relativamente rápida y eficaz.

El hecho de que los objetos tengan reunidos entre sí sus propiedades y la forma en que interactúan ante los eventos, es una medida de mucho orden que permite entre otras cosas un ahorro inestimable de líneas de código durante la implementación del programa, permitiendo en gran medida la creación de proyectos de programación de un tamaño cada vez mayor, sin que ello signifique la creación de interminables conjuntos de instrucciones que terminan siendo inmanejables para cualquier programador.

Es de esperar que ha medida pase el tiempo vendrán nuevas formas de manipular los eventos y propiedades de una manera eficaz lo que se traducirá en lenguajes de programción que permitan hacer programas cada vez máz poderosos y de una manera muy ordenada.

Crystal Report

Herramienta que permite el diseño de reportes de manera estandarizada. Con esta herramienta se pueden lograr reportes de alta calidad que incluyen gráficos, formato condicional de datos, reportes sumarios que permiten el análisis a detalle, reportes de referencia cruzada, mapas, etc. Con **Seagate Crystal Reports 7.0** se adquiere una enorme flexibilidad y dinamismo en el manejo de su información.

Seagate Crystal Reports 7.0 permite rápida y fácil creación de reportes. Control completo en el diseño de reportes. Una forma más sencilla de análisis de información. Generación de reportes de diversos tipos de datos Fácil y rápida distribución de reportes dentro de aplicaciones. Servidor de reportes Web. Y muchas ventajas más para desarrolladores de aplicaciones.

Algunas aplicaciones usando Crystal Reports

Crystal Reports es el generador de Reportes de Visual Basic y con el diseñaremos los reportes de nuestras aplicaciones. Ahora posee un objeto llamado Printer para imprimir datos, su utilización además de compleja es trabajosa pues todo debe ser codificado. Al contrario, Crystal Reports utiliza una interface gráfica a partir de donde podemos construir cualquier reporte que necesitemos.

Podemos iniciar Crystal Reports a través de la opción Report Designer.. del menú Add-Ins o por el icono correspondiente en la barra de programas de Visual Basic en Windows.

Creando un Nuevo Reporte

Vamos a generar un reporte basado en una tabla de nombre Agenda que se encuentra en una Base de Datos de nombre Controle. Nuestro reporte deberá obedecer a los siguientes parámetros:

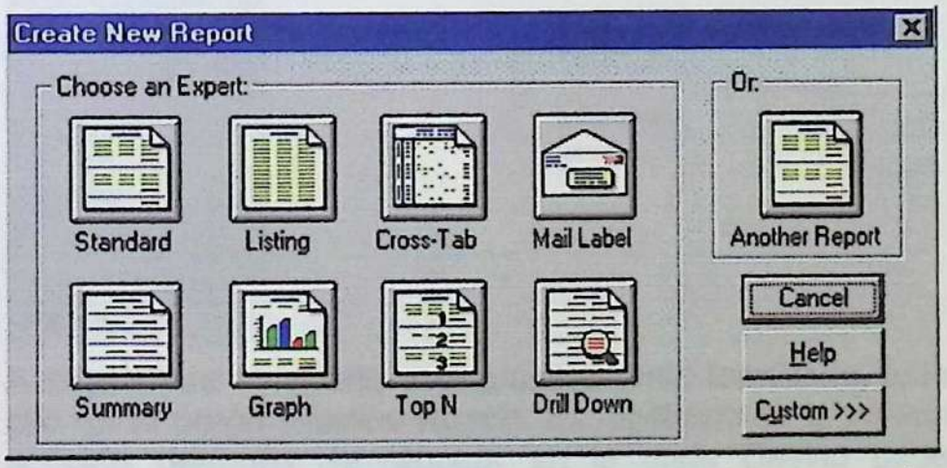
1 Campos a ser impresos: Sobrenombre, Dirección y Fecha de Nacimiento

2 El reporte deberá ser ordenado por campo Sobrenombre.

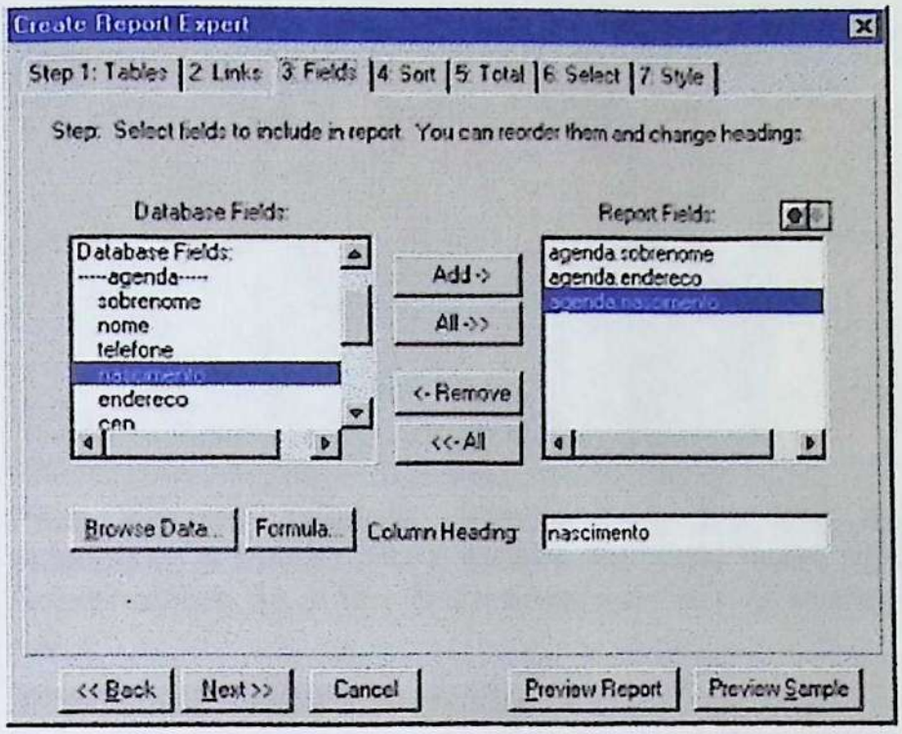
3 Debemos permitir inicialmente la visualización del reporte antes de imprimirlo.

4 El nombre del reporte será Agenda.rpt

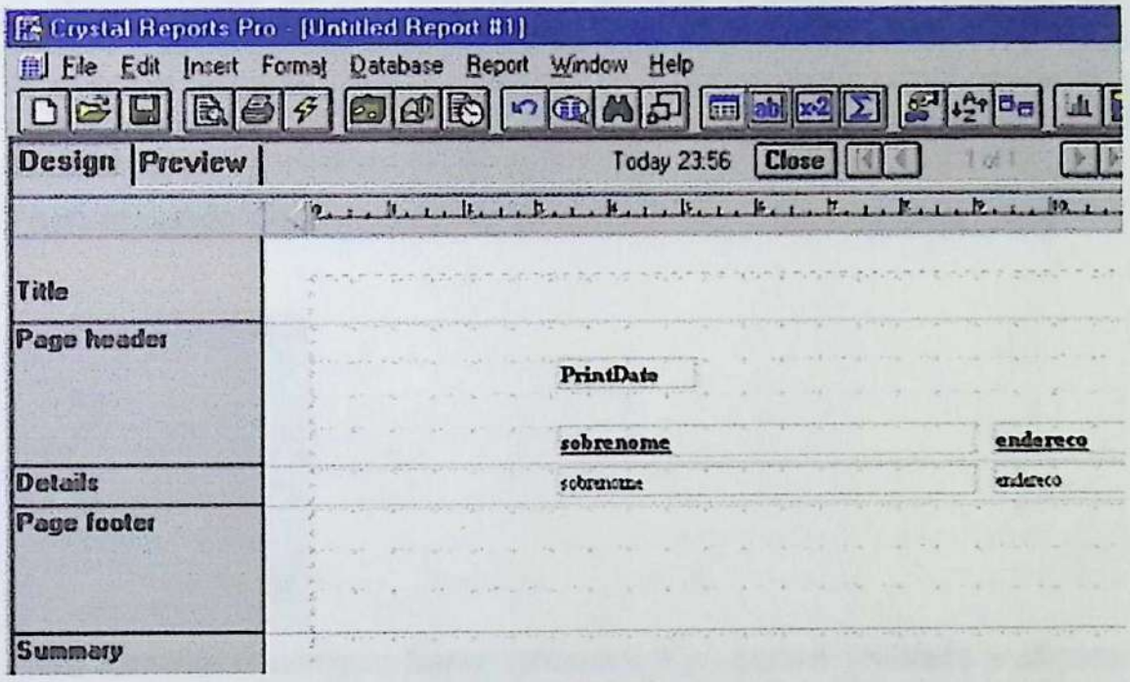
Seleccionando la opción New del menú File tendremos el cuadro de la Figura 1.0 de abajo.



Después de seleccionar el botón Standard, debemos seleccionar la Base de Datos en la opción Data File, para nuestro caso Controle.mdb. Al continuar tenemos una lista de todas las tablas y consultas grabadas en la Base de Datos. Excluya todos los elementos de la lista, excepto la tabla Agenda y haga clic en el botón Next para proseguir. Como nuestro reporte está basado solamente en la tabla Agenda, el próximo paso Links puede ser pasado por alto, por lo tanto haga clic nuevamente en el botón Next.

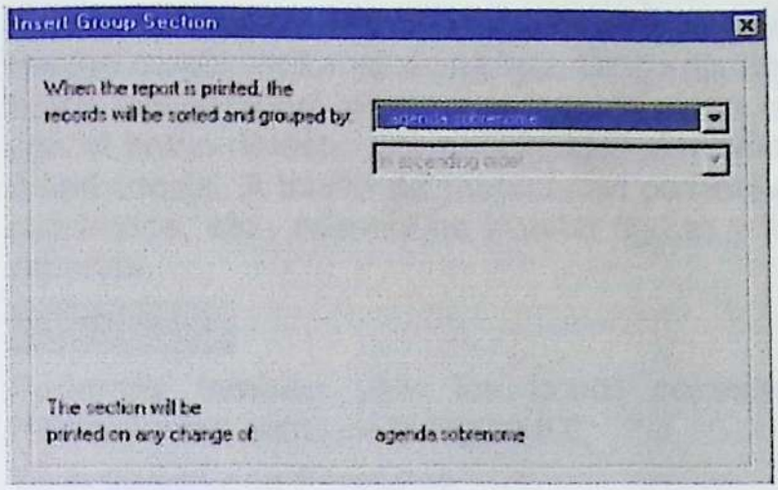


Vamos a seleccionar los campos de la tabla que serán impresos en el reporte. Seleccione cada campo y haga clic en el botón Add. Al final debemos tener algo parecido a la figura 2.0 de abajo:



A esta altura el reporte está prácticamente terminado, para visualizarlo haga clic en el botón Preview Report. Es mostrado en la ventana de la figura 3.0 adonde después de cliquear en la oreja Desing podemos notar cinco secciones: 1 Title: para el título de la aplicación 2 Page Header: contiene los elementos de encabezado de página. 3 Details: contiene los campos a ser impresos 4 Page Footer: se refiere a pie de página 5 Summary: impresión de resúmenes

Agrupando y Ordenando Resumenes

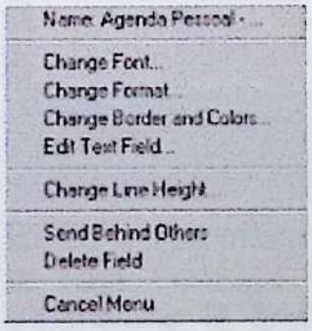


Para agrupar registros, ordenándolos por una determinada columna seleccione la opción Group Section del menú Insert. Agrupando por el campo Sobrenombre en orden ascendente veremos un cuadro igual al de la Figura 4.0

Insertando Títulos y Leyendas

Vamos a insertar un título para nuestro reporte y una leyenda para el campo Sobrenombre, para eso seleccione la opción Text Field... del menú Insert. En la caja de diálogo Enter Text escriba el título: Agenda Personal y haga clic en el botón Accept. Al lado del puntero del mouse hay un rectángulo que usted deberá posicionar en el lugar deseado, o sea, en la sección Title. Para crear la leyenda Nombre para el campo Sobrenombre como encabezado de grupo, seleccione Text Field... nuevamente escriba Nombre, cliqueando en Accept y posicionando la leyenda en el mismo lugar de la leyenda Sobrenombre.

Formateando Campos, Campos Especiales y Diseño de Lineas



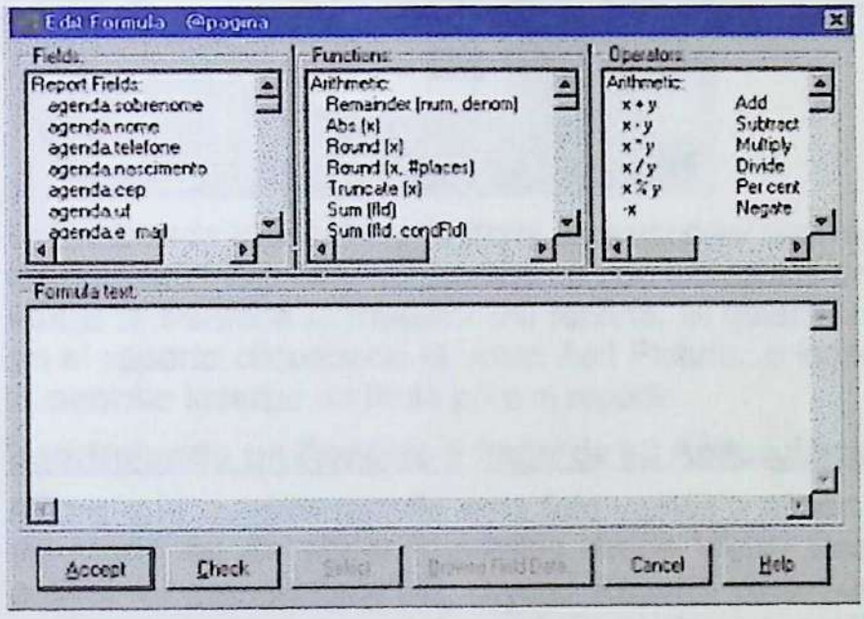
Para formatear campos basta seleccionar el campo deseado y cliquear en la opción Format o haciendo clic en el botón derecho del mouse sobre el campo aparecerá un menú pop-up como la figura 5.0 de abajo: Por este menú podemos acceder a las opciones pertinentes de un determinado campo del reporte. Para nuestro caso seleccionamos el título Agenda Personal, y vamos a modificar la fuente (Change Font...) para un tamaño 14 y estilo negrita. Para modificar más de un campo los seleccionamos manteniendo apretada la tecla Shift. Aprovechando vamos a insertar un campo referente a la Fecha en la esquina superior izquierda. Seleccione la opción Special Field... del menú Insert y escoja la opción Print Date y posiciónelo en el lugar indicado. Cliquee con el botón derecho del mouse sobre el campo Fecha de nacimiento y seleccione la opción Change Format... escogiendo el formato DMY (día-mes-año) y haga clic en OK. Finalmente vamos diseñar un rectángulo alrededor

del título. Seleccione la opción Box del menú Insert, notará que le puntero del mouse cambia la forma a un Lápiz. Dibuje alrededor del título manteniendo el botón izquierdo del mouse apretado. Si quiere colorear el rectángulo cliquee con el botón derecho del mouse sobre el mismo y rellénelo con el color que usted escoja. A través del menú Insert podemos diseñar líneas, rectángulos, cuadrados, etc., además de insertar figuras y también gráficos en nuestros reportes.

Podemos también usar los iconos correspondientes en la Barra de Herramientas como en la Figura 6.0

Trabajando con Fórmulas

Vamos a poner una fórmula para imprimir el Nº de página en el pie de página del reporte. Para eso usamos el editor de fórmulas de Crystal Reports que puede ser abierto a través del icono (x2) o de la opción Formula Field... del menú Insert. Después de eso usted debe escribir el nombre de la fórmula en el campo Formula Field para nuestro caso escriba "página" y haga clic en el botón OK.



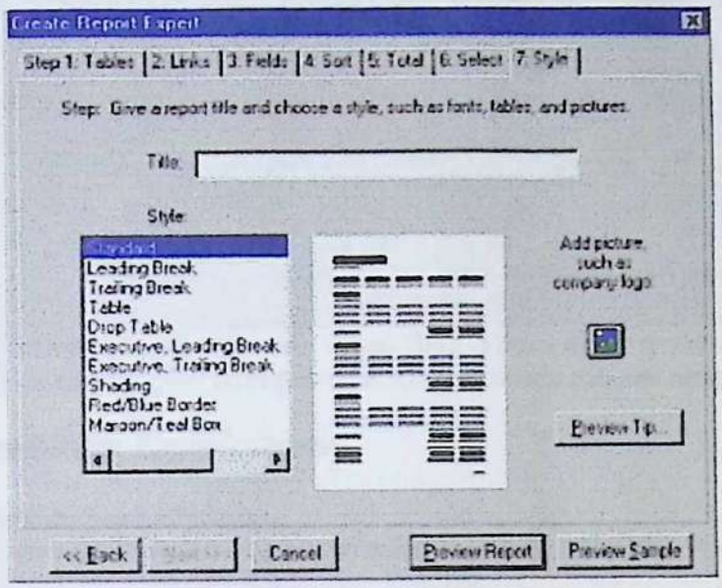
El Editor es mostrado en la figura 7.0 Ahora basta con escribir la fórmula en la caja Formula Text, vamos ya, escriba: "Página: " + y a continuación seleccione la función TrimLeft en la lista de Funciones, en la misma lista seleccione la función ToText y finalmente seleccione el ítem Page Number en el final de la lista Functions. Al final debemos tener lo siguiente en la caja Formula Text: "Página: " + TrimLeft (ToText(PageNumber,0)) La función PageNumber retorna un valor numérico del número de página por eso usamos la función ToText para convertirla en una String, y a continuación usamos la función TrimLeft para remover los espacios en blanco a la derecha. Vamos a verificar la fórmula haciendo clic en el botón Check, si todo estuviera correcto Crystal Reports informará con el mensaje No errors found indicando que la sintaxis está correcta. Ahora basta hacer clic en el botón Accept y posicionar la fórmula en la izquierda de la sección Page Footer. Terminado el reporte basta guardarlo a través de la opción Save del menú File y escribir el nombre para el reporte (nuestro caso escriba Agenda). Debemos resaltar que el lenguaje de fórmulas de Crystal Reports es diferente

al de Visual Basic, así por ejemplo, si usamos la propiedad Selection Formula de Crystal Reports que permite definir las condiciones para la impresión en nuestro reporte de forma que se impriman solamente los nombres que empiecen con la letra "J" tendríamos algo como:

Crystal Report1: SelectionFormula= "{Agenda.Nombre}>=" & """ & "J" & """

Nótese que la referencia a los campos de la tabla está hecha entre llaves ({}).

Determinando el Estilo e Insertando una Figura en su Reporte



Usted puede utilizar la guía Style para escoger una forma de presentación de su reporte. Para esto seleccione uno de los estilos de la caja de lista Style y vea a la derecha la muestra del reporte. Si quiere puede insertar una figura en el reporte cliqueando el botón Add Picture...o icono. La caja de texto Title le permite insertar un título para el reporte.

Imprimiendo un Reporte a Partir de su Aplicación en Visual Basic

Ahora que nuestro reporte esta listo vamos a asociarlo a nuestra aplicación de modo tal de poder imprimirlo desde Visual Basic. Para eso debemos activar el componente de Crystal Reports para nuestra aplicación con la opción Components... del menú Proyects y a continuación seleccionar el control Crystal Reports y copiarlo para nuestro formulario. A continuación vamos a definir algunas propiedades para el control Crystal Report1 CopiesToPrinter: Determina el número de copias del reporte. Escriba uno (1) Destination: Direcciona la impresión: En el cuadro 1- Para impresora 2-A un archivo. Escriba cero (0) ReportFileName: Indica la localización del reporte (archivo.RPT) a ser impreso. WindowTitle: Título de la ventana Preview, escriba Agenda. SortFields: Configura el orden de ordenación (ver abajo). Finalmente cree un botón de comando en el formulario que irá a disparar la impresión del reporte con la leyenda de Imprime y después asocie el siguiente código al botón Imprime:

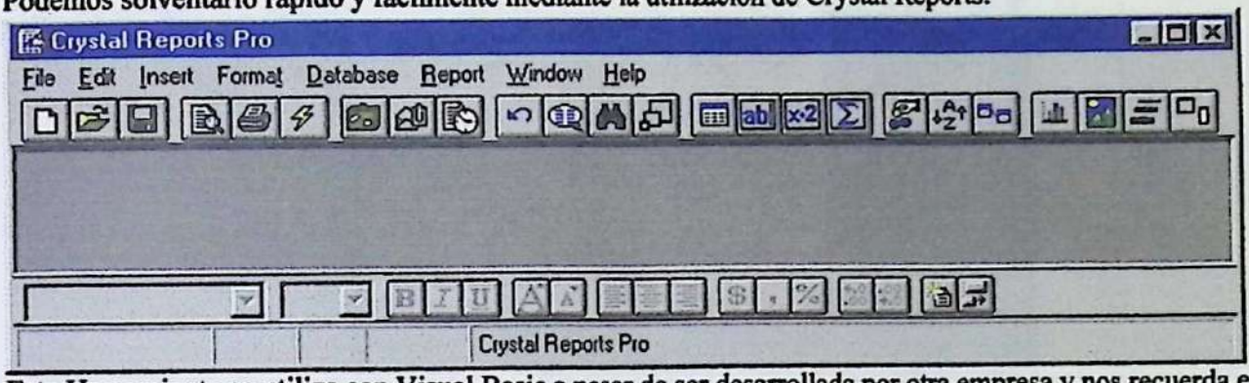
```
Private Sub Imprime_Click()

        CrystalReport1.Destination = 0

        CrystalReport1.ReportFileName = "C:\Controle\Agenda.rpt"
```

Creación de Informes por Impresora.

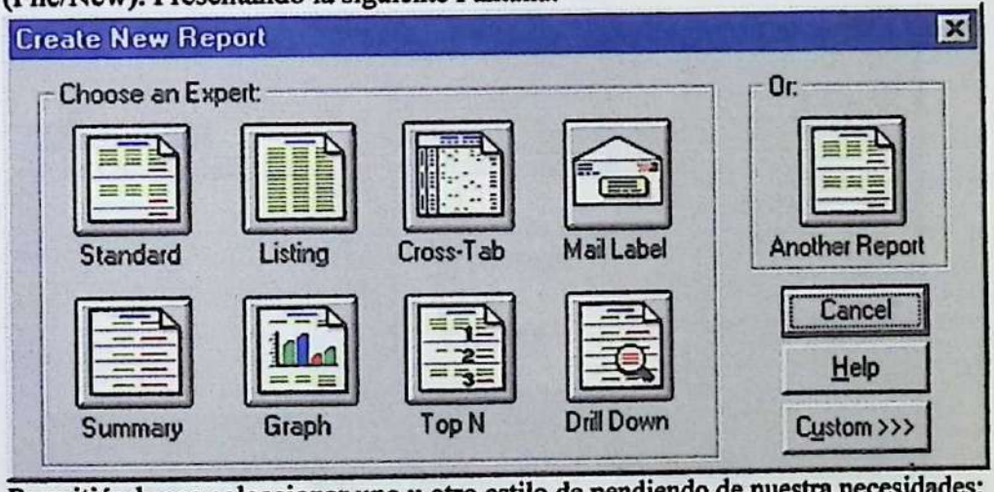
A pesar de que podamos realizar tareas de impresión directamente desde VB. Imprimiendo incluso sobre el puerto PRN o LPT1 como si fuera un fichero de texto. Esto conlleva un espectacular incremento sobre las tareas a realizar dentro de la programación y todas sus rutinas.
Esto se puede complicar aun más cuando tratamos BB.DD. formadas por múltiples tablas y se requiere un conocimiento amplio del manejo de SQL.
Podemos solventarlo rápido y fácilmente mediante la utilización de Crystal Reports.



Esta Herramienta se utiliza con Visual Basic a pesar de ser desarrollada por otra empresa y nos recuerda en algunas ocasiones al diseño de informes de Access mezclado con una antiguo dBase.

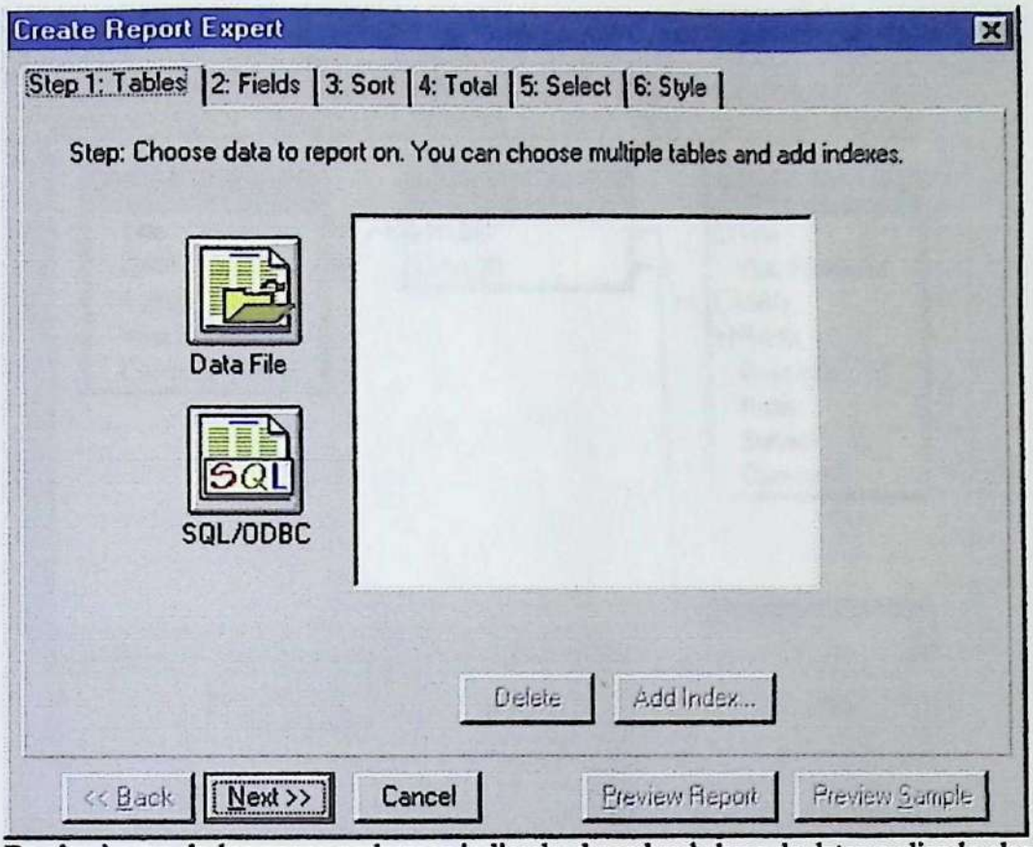
Utilización de Crystal Report

Creando un Nuevo Informe
La primera labor es crear un nuevo informe para lo cual podemos utilizar el Bóton del folio para realizar esta tarea o (File/New). Presentando la siguiente Pantalla:

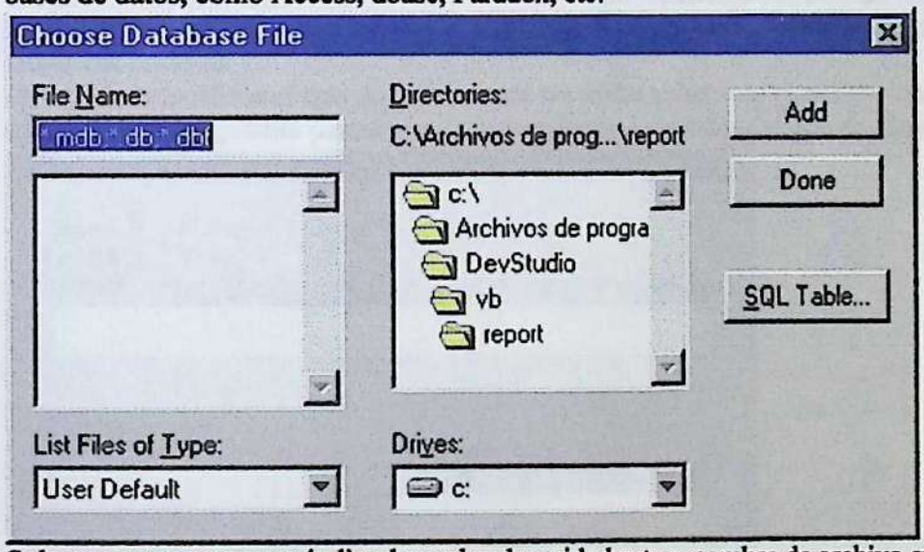


Permitiéndonos seleccionar uno u otro estilo de pendiendo de nuestra necesidades:

Standard
No guiará a través de un asistente el cual nos permitirá de una forma cómoda poder realizar las labores más o menos tediosas del informe:

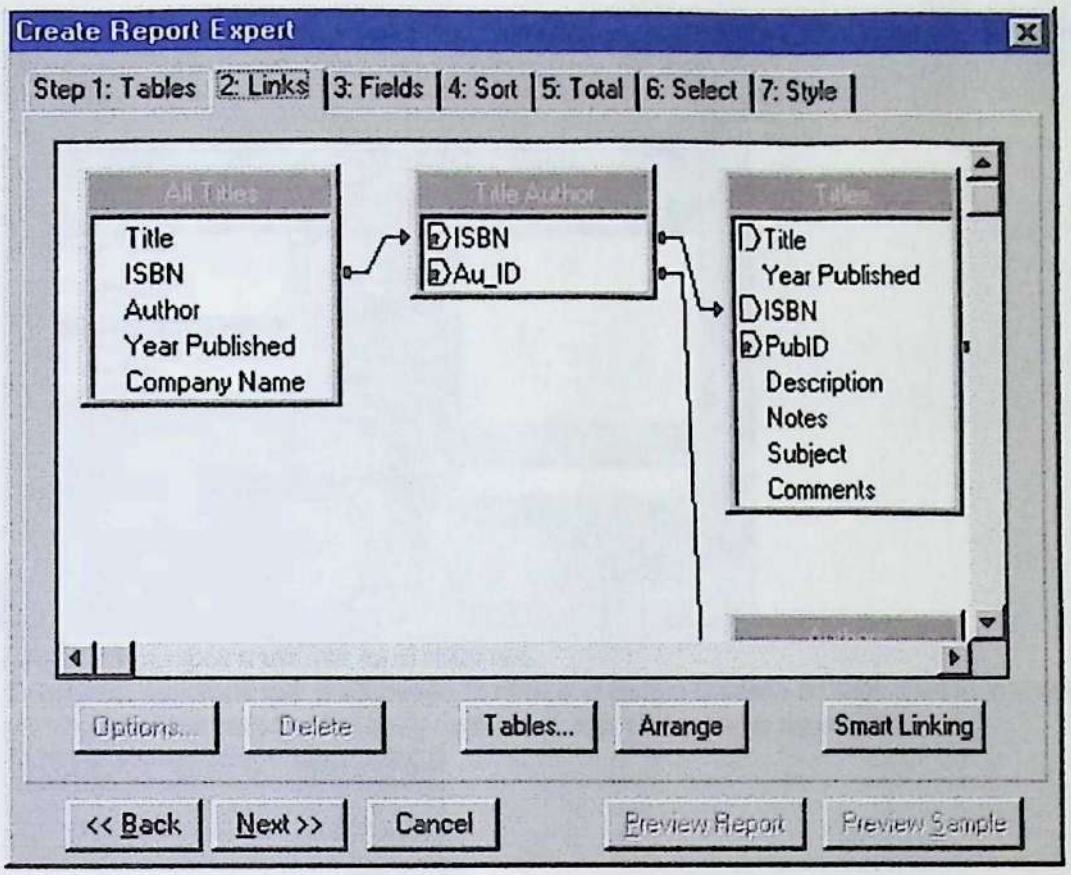


En el primero de los pasos podremos indicarle el nombre la base de datos pudiendo elegir entre múltiples tipos de bases de datos, como Access, dbase, Paradox, etc.



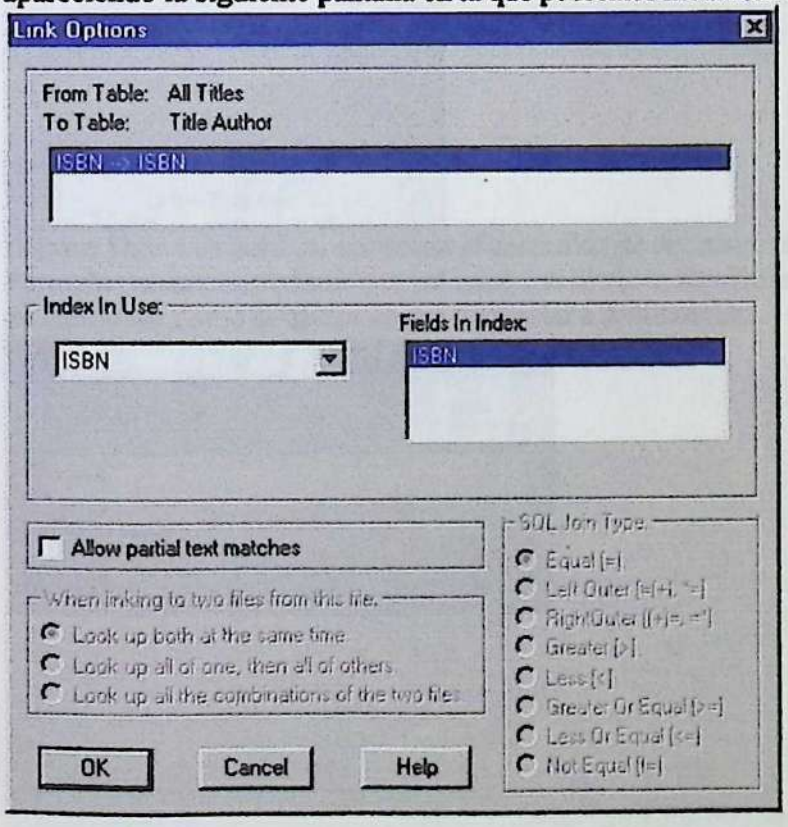
Solamente tenemos que indicarle cual es la unidad ruta y nombre de archivo en de la cual deseamos realizar el informe.

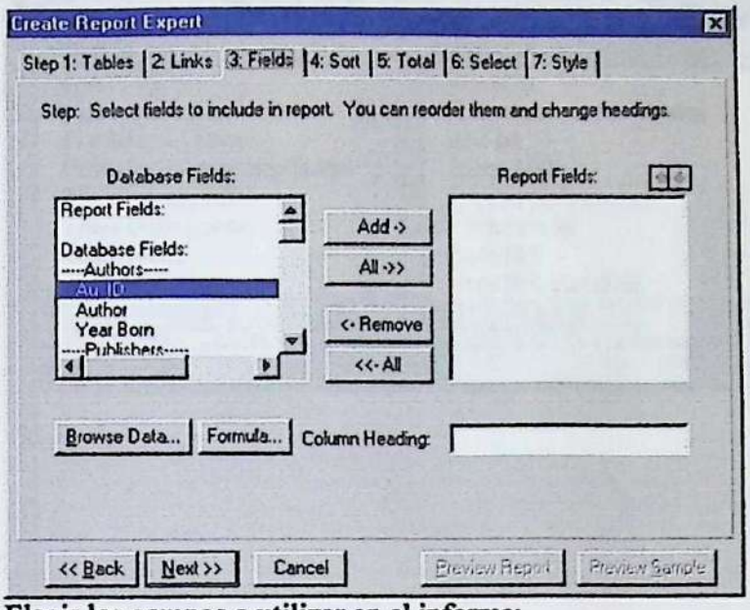
Una vez seleccionada la Base de Datos o la consulta de SQL o EL ODBC creado para el acceso a la base de datos por ejemplo SQL Server u Oracle. Pasaremos a la definición de los Link (Relaciones entre tablas)



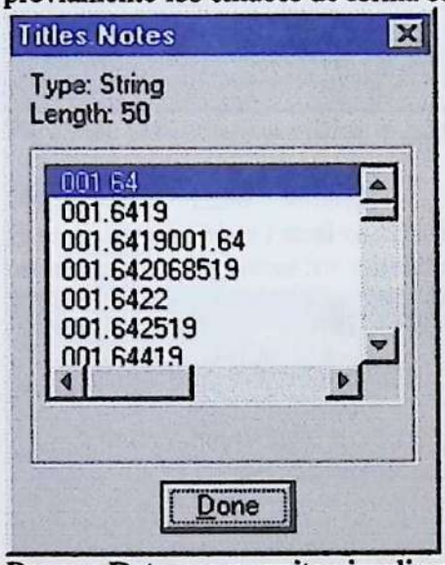
Pudiendo eliminar , añadir o modificar el sistema de relaciones entre tablas estas serán vitales para el buen funcionamiento posterior de los informes ya que en caso contrario podemos encontrarnos verdaderos desastres en la salida del informe.

Para poder modificar el tipo de enlace entre las tablas solamente es necesario realizar un doble clic sobre el enlace apareciendo la siguiente pantalla en la que podremos modificar el tipo de acceso realizado.

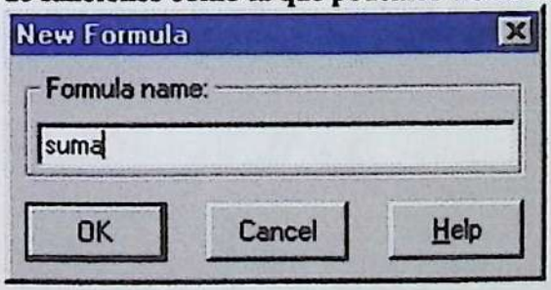


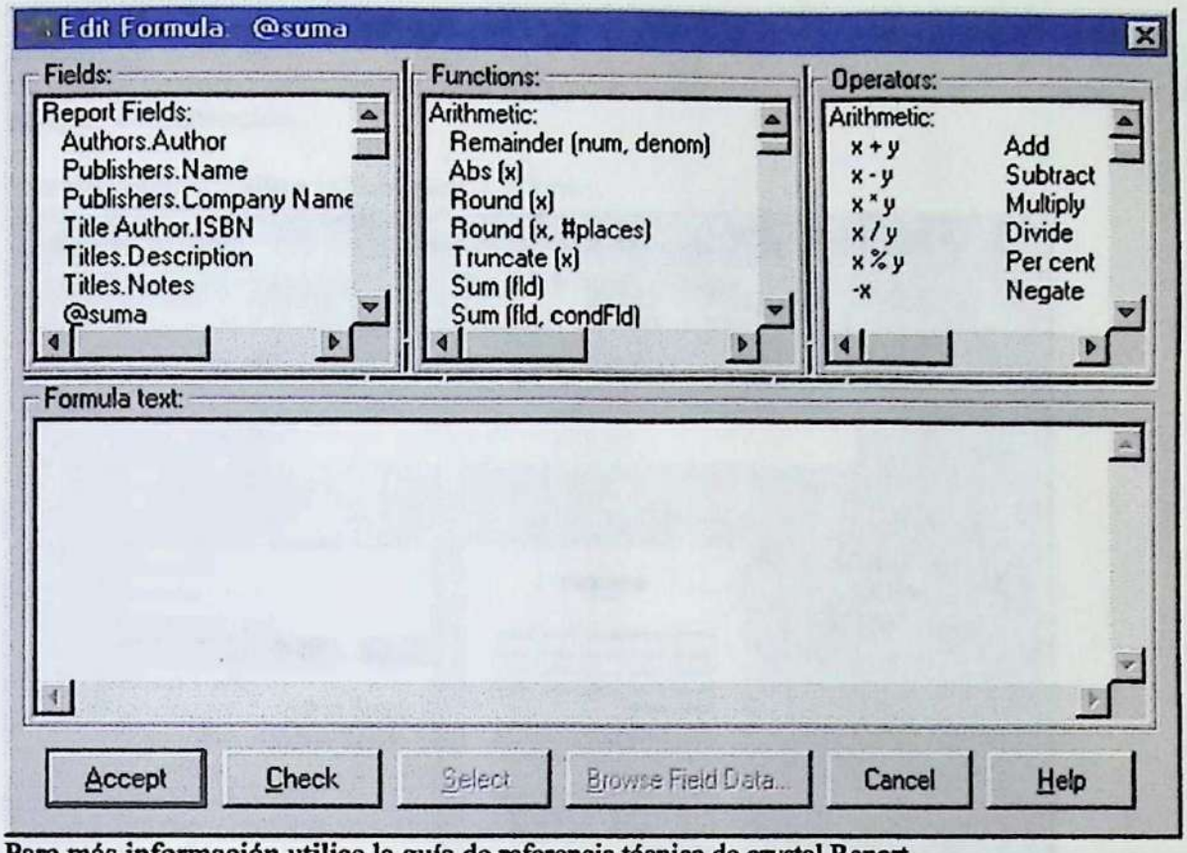


Elegir los campos a utilizar en el informe:
Solamente tenemos que ir eligiendo la tabla y el campo deseado de cada tabla en cuestión, si hemos realizado previamente los enlaces de forma correcta no tenemos por que temer.



Browse Data nos permite visualizar el contenido de un campo antes de poder utilizarlo como seleccionado
Formula permite introducir una columna calculada en función de una formula en la cual podemos utilizar una lista de funciones como la que podemos visualizar a continuación

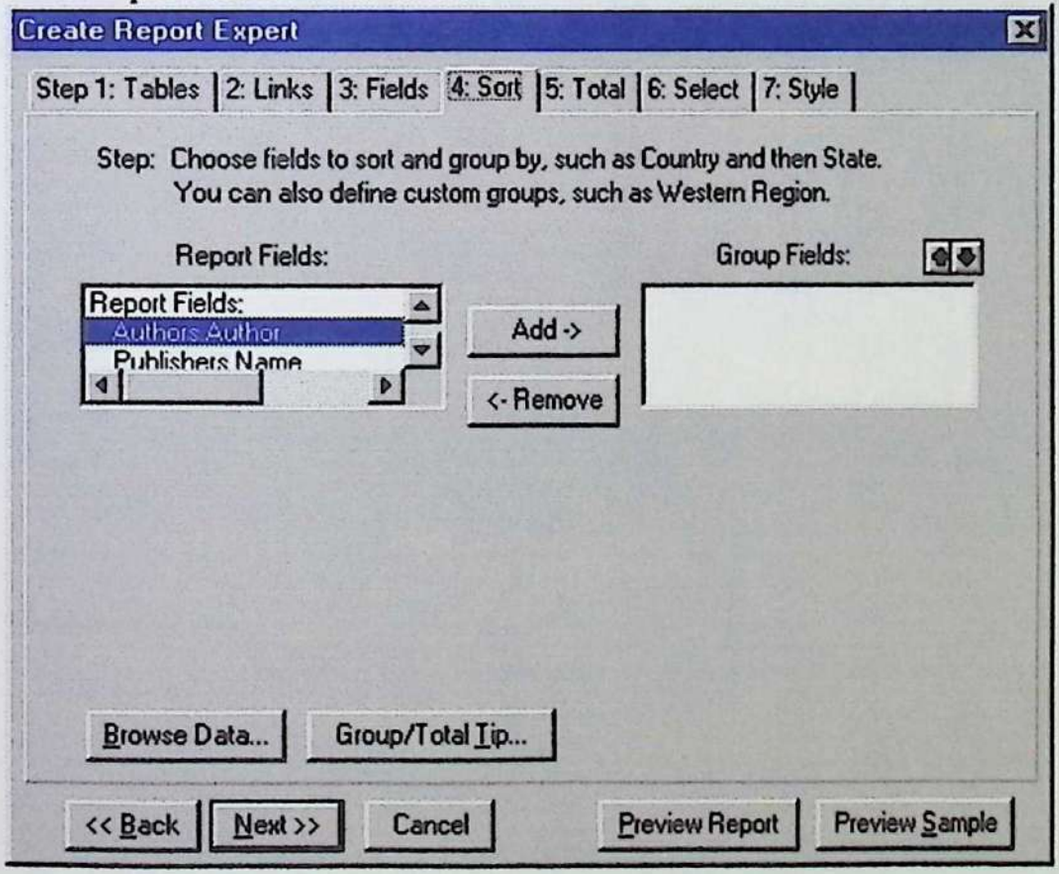




Para más información utilice la guía de referencia técnica de crystal Report.

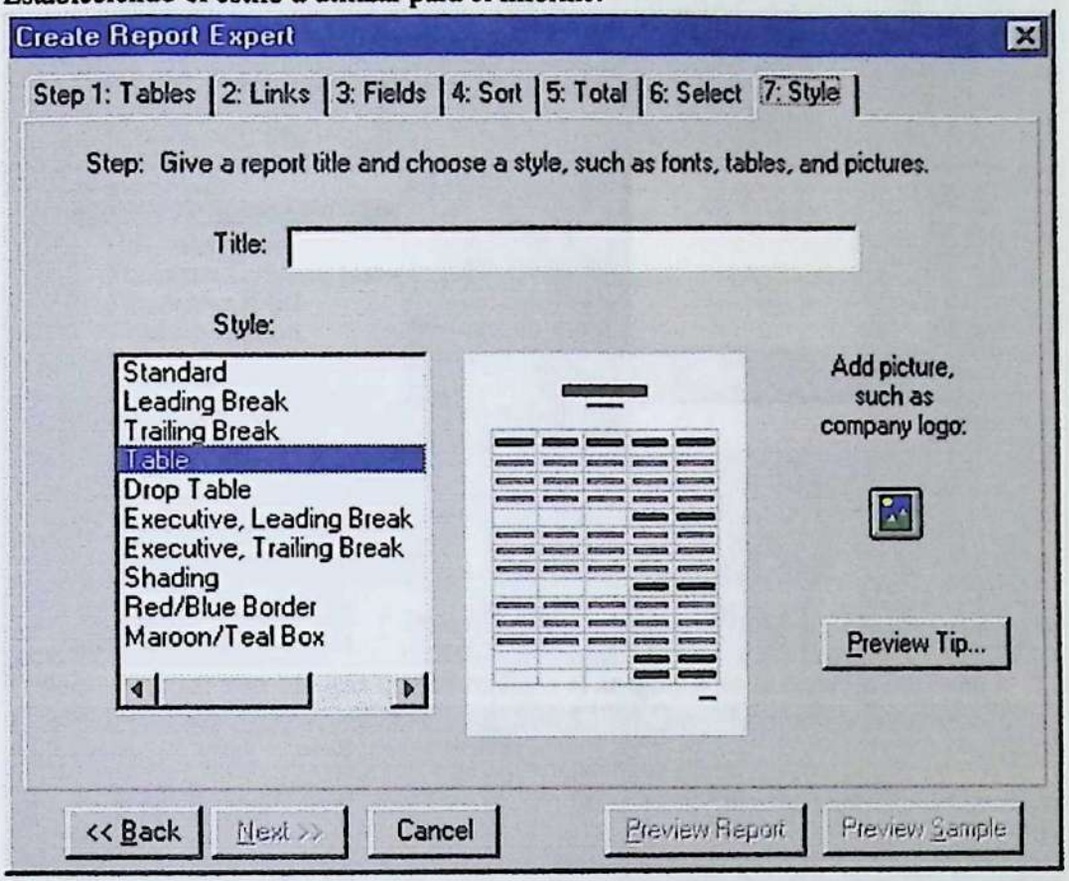
Indicando el Orden de Salida:

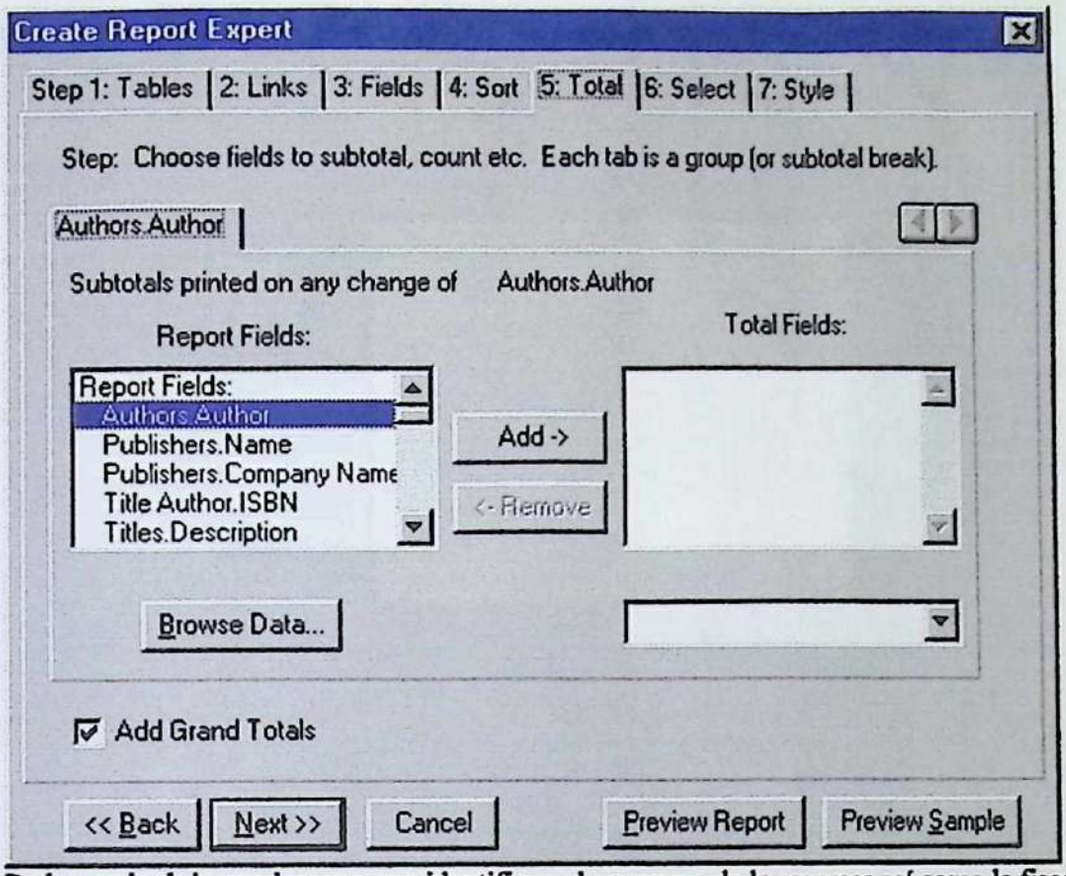
Para poder establecer cual es el orden de salida que deseamos realizar basta con establecer los campos deseados y el orden en que deseamos visualizarlos:



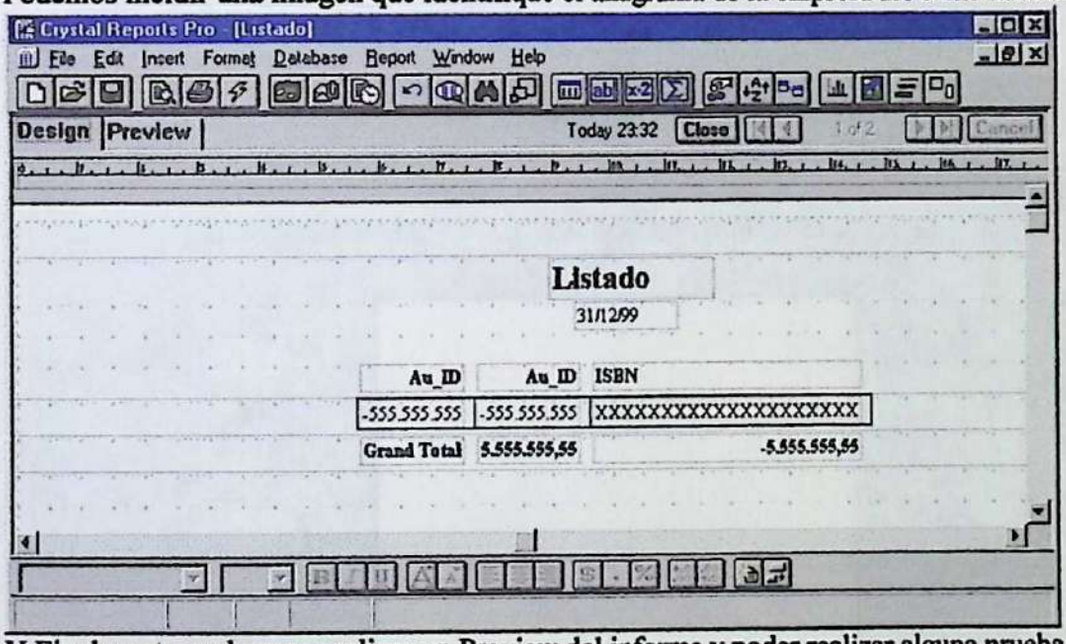
Totales:
En el caso de querer establecer un subtotal o grupo de subtotales podemos indicarle el campo por el cual se desea agrupar la información.

Estableciendo el estilo a utilizar para el Informe:



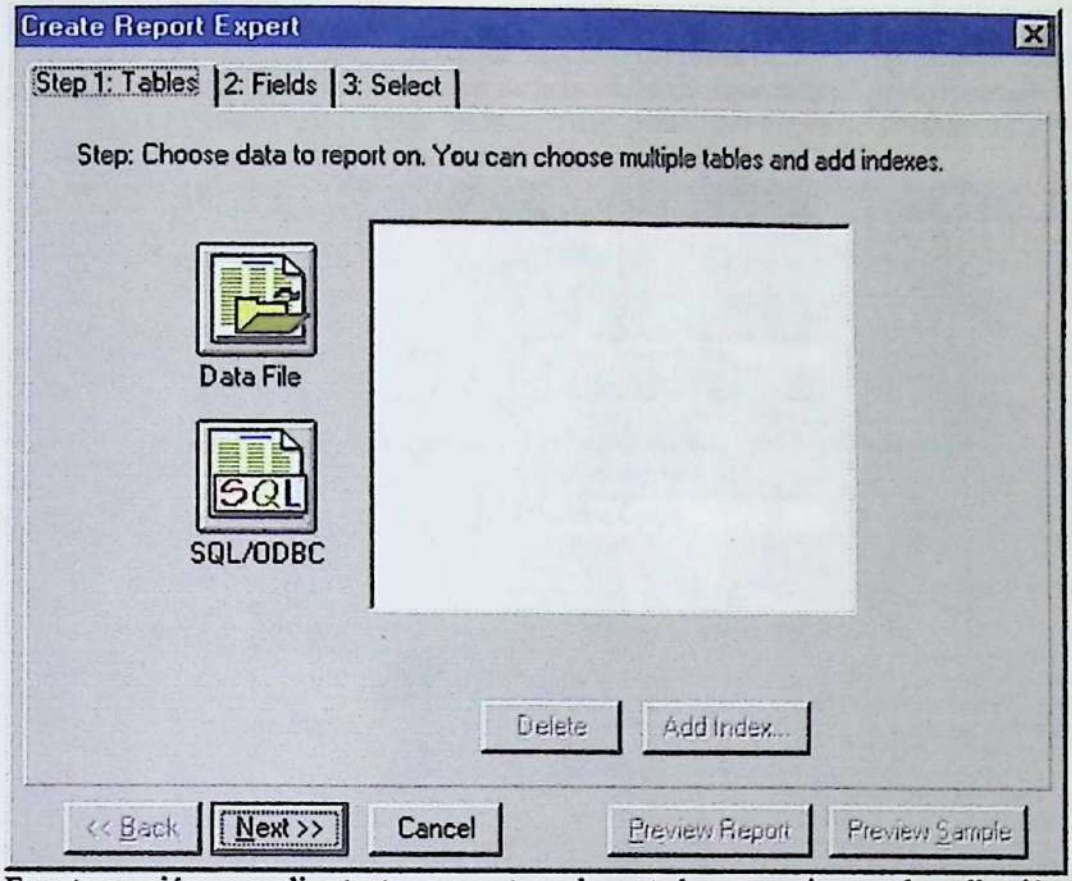


Podemos incluir una imagen que identifique el anagrama de la empresa así como la fisonomía general de informe.



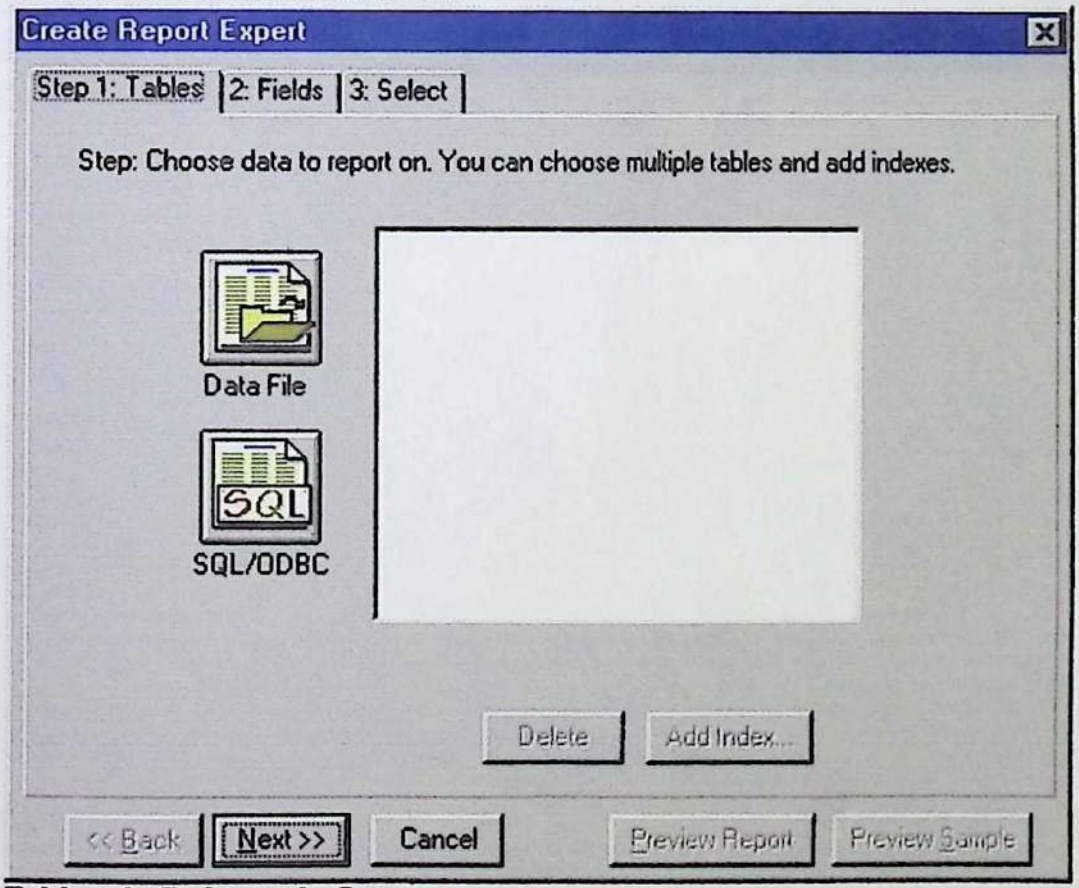
Y Finalmente podremos realizar un Preview del informe y poder realizar alguna prueba en pantalla o modificar algunos elementos del listado.

Listado Simple



En esta ocasión no realiza tantas preguntas solamente las necesarias para la realización de un listado simple (Ver la opción standard para comprobar el significadode las fichas)

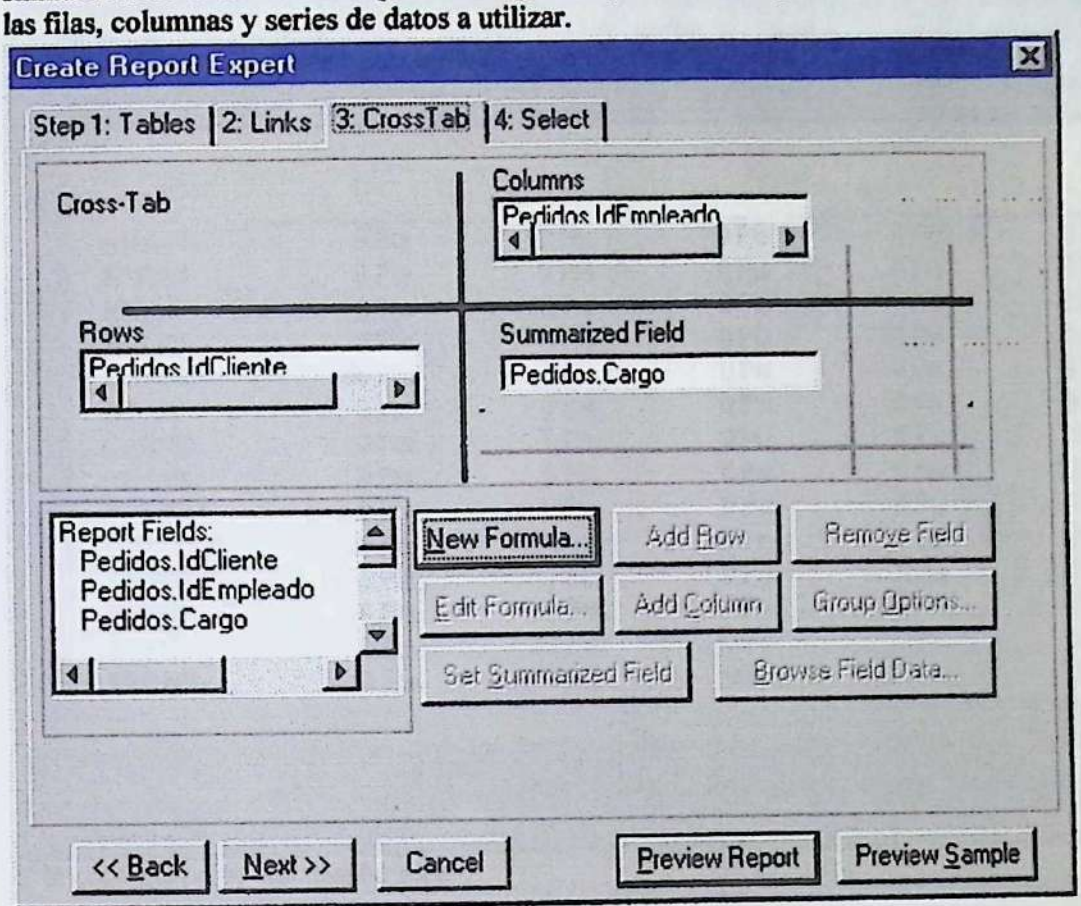
El Resultado es un informe Tabular



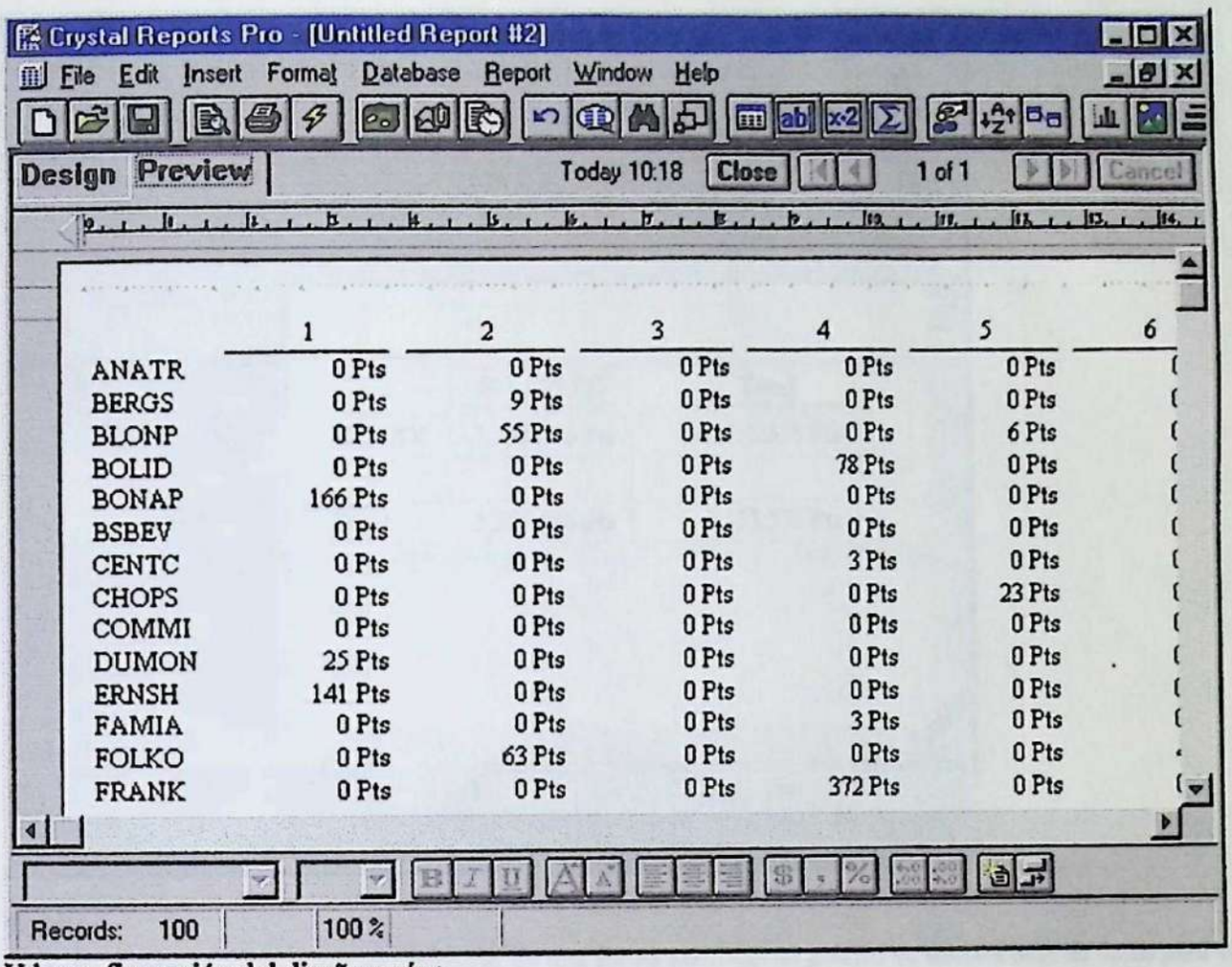
Tablas de Referencia Cru

zada

Permite realizar informes agrupados en filas y en columnas de una forma cómoda además las columnas pueden dilatarse automáticamente en función de la cantidad de datos diferetes que tengamos. Los pasos a realizar son similares a los anteriores excepto en la siguiente pantalla en la que debemos de definir cual es la configuración de las filas, columnas y series de datos a utilizar.

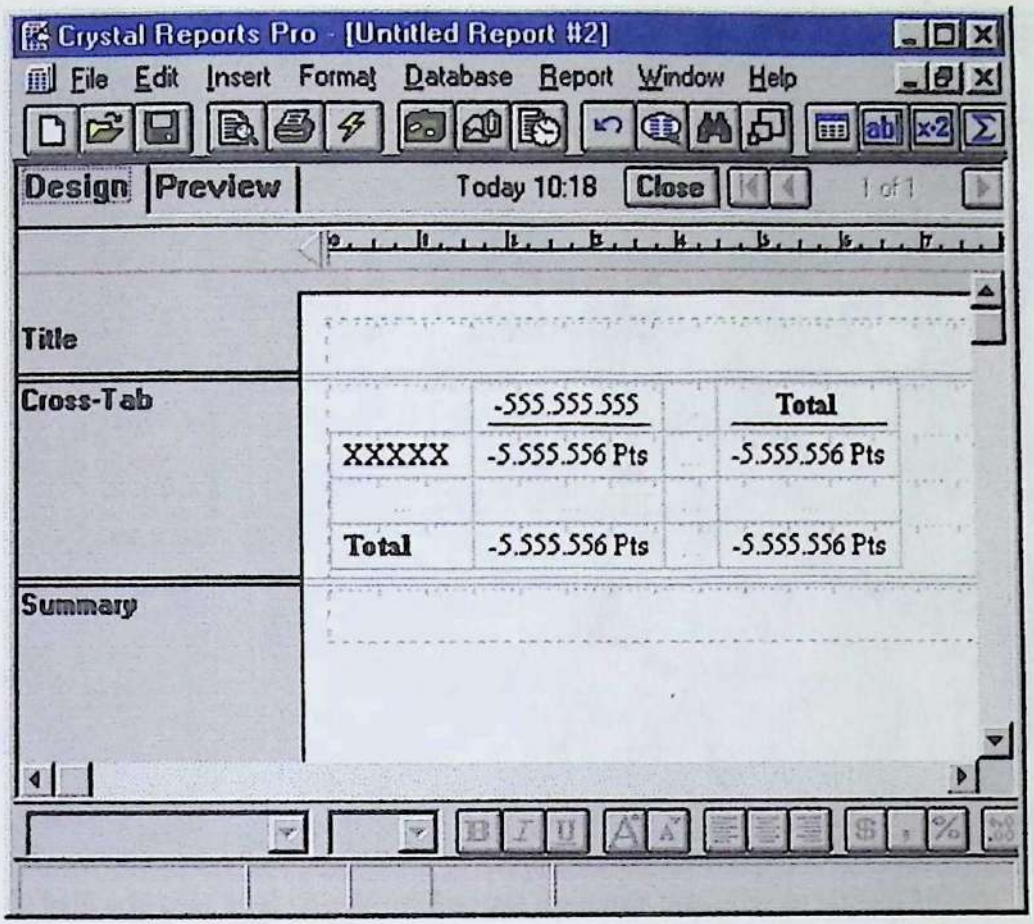


La salida sería similar a :



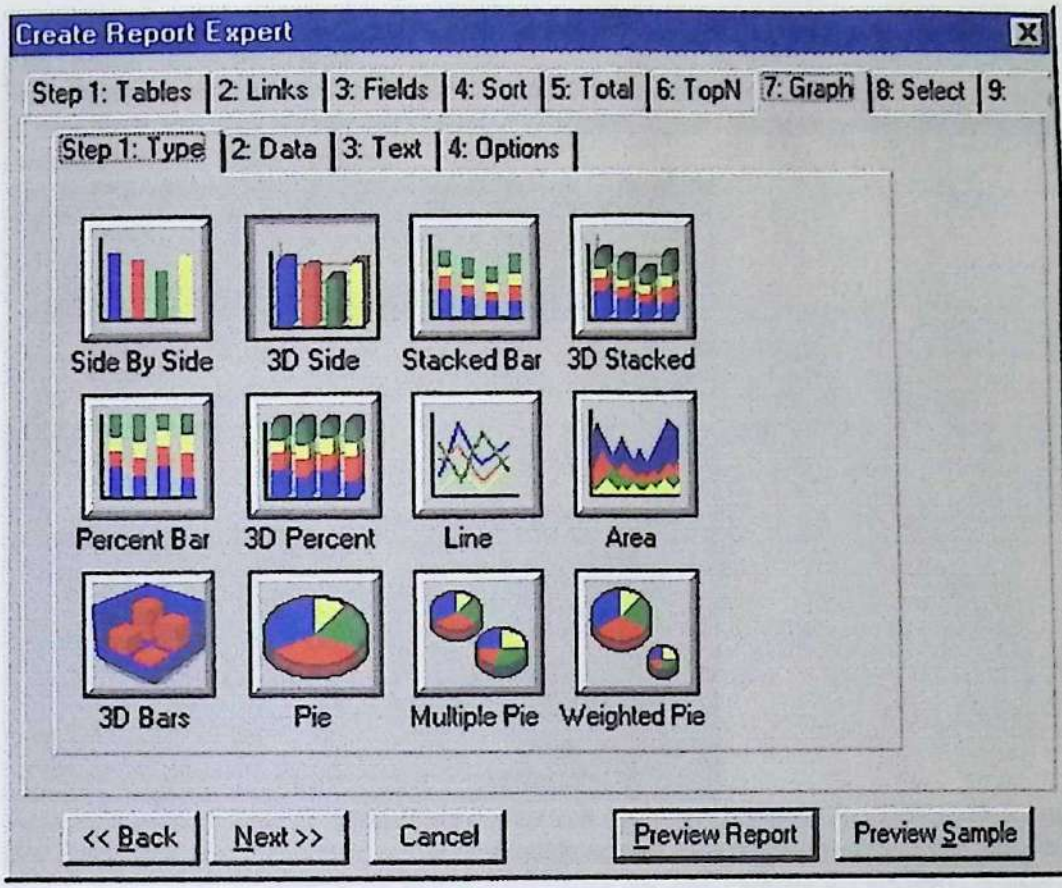
	1	2	3	4	5	6
ANATR	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
BERGS	0 Pts	9 Pts	0 Pts	0 Pts	0 Pts	(
BLONP	0 Pts	55 Pts	0 Pts	0 Pts	6 Pts	(
BOLID	0 Pts	0 Pts	0 Pts	78 Pts	0 Pts	(
BONAP	166 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
BSBEV	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
CENTC	0 Pts	0 Pts	0 Pts	3 Pts	0 Pts	(
CHOPS	0 Pts	0 Pts	0 Pts	0 Pts	23 Pts	(
COMMI	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
DUMON	25 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
ERNSH	141 Pts	0 Pts	0 Pts	0 Pts	0 Pts	(
FAMIA	0 Pts	0 Pts	0 Pts	3 Pts	0 Pts	(
FOLKO	0 Pts	63 Pts	0 Pts	0 Pts	0 Pts	
FRANK	0 Pts	0 Pts	0 Pts	372 Pts	0 Pts	(

Y la configuración del diseño sería :



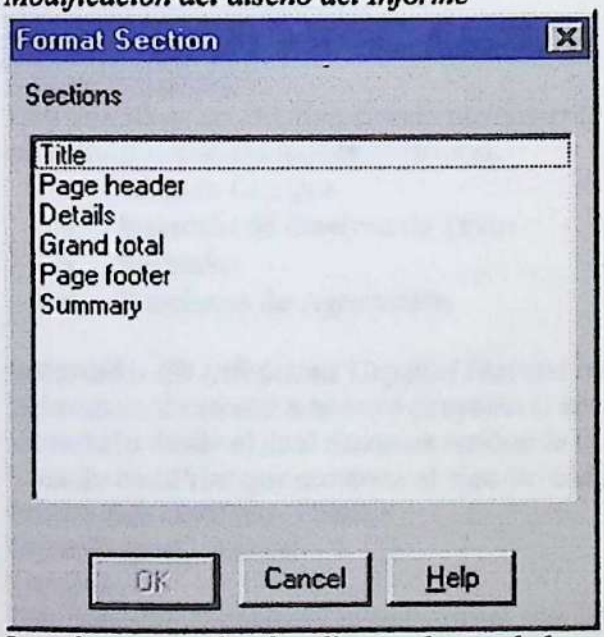
Gráfico

Este tipo de informe nos permite Mostrar de una forma sencillas un gráfico en nuestra base de datos para ello deberemos de seguir en gran parte los mismos pasos anteriores solamente difiren en algunas pantallas.



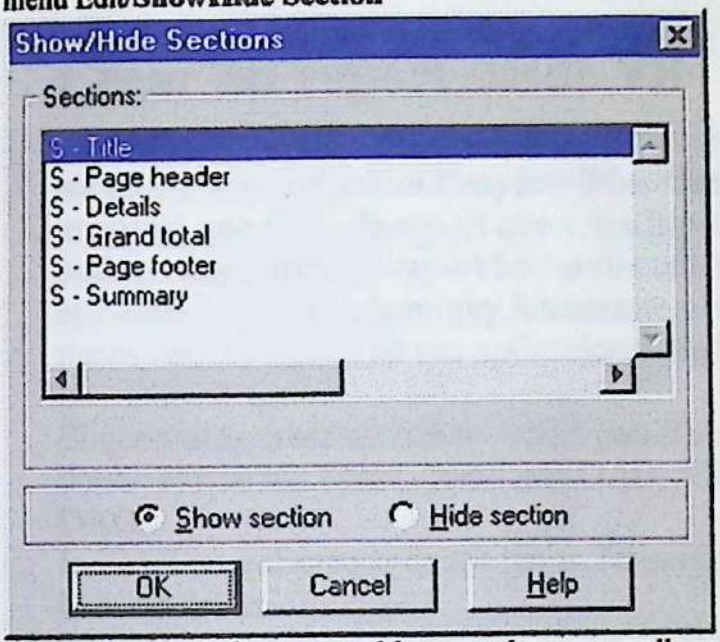
En esta pantalla deberemos de indicarle cual es el tipo de gráfico que deseamos visualizar en nuestro informe
El resto de los tipos solamente difieren en pequeños detalles respecto a los anteriores

Modificación del diseño del Informe



Lo primero es saber localizar cada una de las secciones del Informe para ello podemos utilizar **Format/Section**
De esta manera seremos capaces de reconocer donde estamos insertando la información y que resultados obtendremos posteriormente.:
Titulo: El título del Informe solo aparederá al principio del mismo.
Encabezados de Página (Page Header) utilizado para insertar toda aquella infromación que queremos que aparezca una sola vez y al principio de la página, por ejemplo el nombre de la tabla o el número de la Página o la fecha.
Detalles (Details) En esta zona se incorpora todos aquellos campos que serán posteriormente la información que veamos en la salida impresa como líneas de Registros.

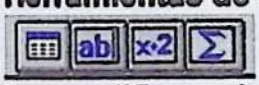
Grand Total: Utilizado para inserción de subtotales.
Page footer : Pies de Página
Resumen General (Summary)utilizado para sumas generales o totales.
Si deseamos mostrar alguna otra sección o ocultar alguna existente solamente deberemos de indicarlo mediante el menú Edit/ShowHide Section



Aquí es donde podremos establecer cuales son aquellas secciones que realmente desamos visualizar y cuales no. Por ejemplo que deseamos ocultar la línea de totales para el grupo bastará con seleccionar Grand total y posterioremente pulsa en Hide sectión Ello hará que desaparezca del informe tal zona.

También podemos utilizar la opción Edit Delete Section para eliminar algunas de las secciones previamente ocultadas.

Herramientas de Diseño



Para modificar un informe creado previamente podremos utilizar diferentes herramientas entre ellas nos encontramos con los iconos anteriores.

- Lista de Campos
- Inserción de Cuadros de Texto
- Formulas
- Funciones de Agrupación

Inserción de Infromes Crystal Report desde V.Basic 5.0.

Deberemos de añadir a nuestro proyecto el control Crystal Report e insertar posteriormente dicho control sobre el formulario desde el cual desemos realizar la operación de Impresión por jemeplo insertamos un botón de comando llamado cmdPrint que contiene el siguiete código:

```
Private Sub cmdPrint_Click()
CrystalReport1.Action = 1
End Sub
```

Esto hará que el control CrystalReport1 que antes hemos insertado en el formulario y que ya contiene la información sobre el informe a realizar muestre dicho informe **ReportFileName**

.

Aplicando Condiciones en el Informe

```
CrystalReport1.SelectionFormula = "{pedidos.idempleado}=1"
CrystalReport1.Action = 1
```

Uso del entorno de Crystal Reports 6.0 con bases de datos DAO.

Crear un report desde VB

Una vez instalado Crystal Reports la aplicación funciona como un diseñador ActiveX, accesible desde el menú **Proyecto/Mas diseñadores** de Visual Basic. Pulsando sobre el nos pregunta que clase de report queremos hacer. Una característica muy interesante es que podemos importar un report hecho con una versión anterior de Crystal Reports, (la 4.6 por ej.). Esta es una función muy interesante y muy utilizada por todos aquellos que tienen la necesidad de readaptar sus aplicaciones realizadas con versiones anteriores de Visual Basic.

Si queremos crear un nuevo report para nuestra aplicación; Crystal Reports nos preguntará de donde extrae los datos, (mediante DAO, ADO, RDO o Jet). Para este ejemplo elegiremos DAO...
Indicaremos el origen de los datos, (el camino y nombre de la base de datos), y el tipo de base de datos.
Si el sistema encuentra la base de datos solicitada nos preguntará de que tebla de las halladas en la base de datos deseamos generar el informe; Opcionalemnte, (o casisiempre), podemos introducir una instrucción SQL que nos aglutine datos de varias tablas y/o nos filtre datos de las mismas.
Con esto concluimos la conexión con la base de datos.

Seguidamente el sistema nos preguntará por :

- Los campos que deseamos incluir en el informe.
- Por que campos, (y en que orden), deseamos ordenar el informe.
- Que campos deseamos totalizar.
- Sobre que campos deseamos realizar un gráfico.
- Que estilo de informe deseamos.

Por último, Crystal Reports pregunta si deseas crear un formulario "tipo" para tus informes. Es saludable responder que si con el primer informe y no para el resto, y utilizar el formulario creado para mostrar todos los informes. También podemos crear nuestro propio formulario de informes puesto que Crystal Reports nos provee de un control accesible desde la ventana de Tools. Una vez hecho esto Crystal Reports creará un informe automáticamente a partir de la información otorgada. Este informe podrá ser manipulado gracias a un editor muy paedico al de versiones anteriores, (aunque en mi modesta opinión me parece menos "agil").

Una vez obtenido el informe deseado lo salvamos con formato RPT dirigiendonos al menú flotante, (boton derecho del ratón), y a **Reports/Save to Crystal Reports File.** El diseño se salvará con extensión DSR junto a nuestra aplicación.

NOTA :

- La versión 6.0 de Crystal Reports no funciona correctamente en entornos SDI de Visual Basic. Debereis, (si quereis trabajar comodamente), cambiar el entorno de Visual Basic a MDI, (menú **Herramientas/Opciones** y la pestaña **Avanzado**).
- No existe en esta versión, (6.0), ninguna función de presentación preliminar. Por lo que para ver el resultado de la ejecución del informe, habra que ejecutarlo junto al programa que haga uso de el.

Lanzar el report desde nuestra aplicación.

Para lanzar un listado hecho con Crystal Reports podemos utilizar una función parecida a la que sigue :
En un módulo BAS ...

```
' Abre un listado de Crystal Reports.
Sub AbrirListado(Listado As String, OrigenDatos As Recordset, Titulo As
String)

    Dim t As Integer ' Contador.
    Dim MostrarListadoPos As Integer ' Posición de la matriz en la que se
    mostrará la ventana de listado.


    ' Comprueba que el listado no esté presente.
    For t = 1 To EstadoListados(0).Estado

        If EstadoListados(t).Nombre = Listado Then

       VentanasListados(t).ZOrder
       Exit Sub

    End If

Next t

' En caso contrario busca un hueco.
MostrarListadoPos = 0
For t = 1 To EstadoListados(0).Estado

   If EstadoListados(t).Estado = 0 Then

        MostrarListadoPos = t
        EstadoListados(t).Estado = 1
        EstadoListados(t).Nombre = Listado
        Exit For

    End If

Next t
```

```
' Si no hay hueco abre uno.
If MostrarListadoPos = 0 Then

    EstadoListados(0).Estado = EstadoListados(0).Estado + 1
    ReDim Preserve EstadoListados(EstadoListados(0).Estado)
    EstadoListados(EstadoListados(0).Estado).Estado = 1
    EstadoListados(EstadoListados(0).Estado).Nombre = Listado
    ReDim Preserve VentanasListados(EstadoListados(0).Estado)
    MostrarListadoPos = EstadoListados(0).Estado

End If

' Abre el listado.
VentanasListados(MostrarListadoPos).Tag = MostrarListadoPos
VentanasListados(MostrarListadoPos).Caption = Titulo
VentanasListados(MostrarListadoPos).Show

Dim app As New CRAXDRT.Application
Dim rep As Report

On Error GoTo ErrorListado
Set app = New CRAXDRT.Application
Set rep = app.OpenReport(EstadoListados(MostrarListadoPos).Nombre)

rep.Database.SetDataSource OrigenDatos

VentanasListados(MostrarListadoPos).CRViewer1.ReportSource = rep
VentanasListados(MostrarListadoPos).CRViewer1.ViewReport

' Salida.
SalirListado:
Exit Sub

' Error al intentar abrir el listado.
ErrorListado:
Resume SalirListado

End Sub
```

Por este sistema un listado con el mismo título sólo puede ser abierto una vez. Esta es una buena política porque si se abre varias veces un mismo listado... ¿Cual es el que muestra la información correcta?.
Este código ha sido optimizado para funcionar en un entorno MDI, por lo que el formulario de informes deberá tener a **True** la propiedad **MdiChild**.
Para funcionar correctamente en un entorno MDI, en el mismo módulo BAS, en el apartado de declaraciones, hay que añadir el siguiente código :

```
' Ventanas de listados.
Type TipoEstadoListados

    Nombre As String
    Estado As Integer

End Type
Public EstadoListados() As TipoEstadoListados ' Estado de las ventanas de listados.
Public VentanasListados() As New Listados ' Matriz de ventanas de listados.
```

Y por último; en el **Form_Load** del formulario principal, (aquel que abre la aplicación), incluir el siguiente código :

```
' Control de ventanas de listados MDI.
ReDim VentanasListados(0)
ReDim EstadoListados(0)
```

Para realizar la llamada a esta función, primero se han de filtrar los datos que se desean listar; para ello hay que abrir la base de datos y seleccionar en un **RecordSet** que información deseamos se muestre en nuestro informe.
Un ejemplo de esto seria :

```
Dim Base As Database ' Base de datos.
Dim Empresas As Recordset ' Empresas.

' Conectar con la base de datos.
Set Base = OpenDatabase("MiBase.Mdb")

' Abrir Recordset.
Set Empresas = Base.OpenRecordset("SELECT * FROM Empresas ORDER BY
CodEmpresa", dbOpenDynaset, dbConsistent, dbPessimistic)
AbrirListado "MiInforme.rpt", Empresas, "Listado de empresas"
```

Si no se desea un entorno MDI baste reseñar que el código que abre nuestro informe es el siguiente :

```
Dim app As New CRAXDRT.Application
Dim rep As Report

Set app = New CRAXDRT.Application
Set rep = app.OpenReport("MiInforme.rpt")

rep.Database.SetDataSource OrigenDatos

MiFormInforme.CRViewer1.ReportSource = rep
MiFormInforme.CRViewer1.ViewReport
```