Manual de Referencia de Manejo de Office en VB6

Carlos Reyes
ereyes@inf.utfsm.cl

22 de Agosto

Índice

1. Introducción.

Este es un documento que intenta recopilar múltiples ideas de la programación con Visual Basic 6 y sus aplicaciones con Office a través de la herramienta ADO (ActiveX Data Object). Los códigos que se presentan como ejemplos fueron creados especialmente para ser introducidos en este documento, algunas ideas se tomaron de la realización de una aplicación que manejaba Excel y Access, y que fue desarrollada por mí y otras de la constante búsqueda en Internet de soluciones para finalizar esta aplicación.

1.1. ¿Cuál es la idea de este documento?

La idea de crear este documento es facilitar el manejo de las distintas aplicaciones que estemos desarrollando. Supongo que nadie entregará un informe en block de notas, ¿¿o no?? .

Bueno, tal como se especifico anteriormente, nos enfocaremos en interactuar con Word, Excel y Access, y no nos detendremos en explicar como se utiliza Visual Basic, pues daremos por conocidas muchas cosas que son básicas, tal como los bucles, declaración de variables globales, etc..

1.2. He leído el documento y creo que estas mal, o tengo un mejor ejemplo.

Si alguien quisiera hacer un aporte para que otros programadores no se quiebren la cabeza tal como lo hicimos nosotros, sería excelente, ese es otro punto importante.

Todos los aportes, comentarios, y otros, serán bienvenidos y agregados con una referencia de quien lo envía y su mail.

2. Excel.

Antes de empezar debemos agregar la referencia que vamos a ocupar.

```
Proyecto > Referencias > Microsoft Excel 10.0 Object Library
```

Consideraremos que tenemos un archivo existente llamado `test1.xls` para todos los efectos de llenado.

2.1. Declaración de Variables.

```
Dim objExcel As Object
Dim objLibro As Object
```

`ObjExcel` será el objeto que utilizaremos para referirnos a la aplicación Excel como tal, y `ObjLibro` nos indicara que estamos trabajando directamente sobre el archivo `test1.xls` en cuestión.

2.2. Ejemplo Demostrativo.

```
Dim objExcel As Object
Dim objLibro As Object
Dim ruta As String

On Error Resume Next
Set objExcel = GetObject(, "Excel.Application")     'Seteamos los objetos
If Err.Number = 429 Then
  Err.Clear
  Set objExcel = CreateObject("Excel.Application")
End If

ruta = App.Path  & "\test1.xls"     'Ruta de un archivo de Excel existente

If Len(Dir(ruta)) > 0 Then     'Verificamos la existencia del archivo
  Set objLibro = objExcel.Workbooks.Open(ruta)     'Abrimos un libro existente
  'Escribimos
  objLibro.Worksheets(1).Range(''A1'').Value = ''Texto1''
  objLibro.Worksheets(1).Range(''A2'').Value = ''Texto de Prueba''
  objLibro.Worksheets(1).Range(''A3'').Value = ''Texto 3''
  objExcel.Visible = True     'Mostramos los cambios en pantalla
Else

    MsgBox "El archivo no existe"
  End If

Set objLibro = Nothing     'Reseteamos las variables
Set objExcel = Nothing
```

2.3. Del ejemplo anterior sacamos las siguientes ideas.

Escribir: `objLibro.Worksheets(Hoja).Range(''Celda'').Value = ''Texto''`

Hoja: Corresponde a el numero de la hoja en la cual deseamos escribir.

Celda: Es la posición especifica en que escribiremos.

Note que la celda debe ir entre comillas, pero solamente cuando es fija.

2.4. Ejemplos.

2.4.1. Ejemplo N° 1.

```
Dim casilla As String
Dim i as Integer
i = 2
casilla = "D" \ & i

objLibro.Worksheets(1).Range(''A1'').Value = ''Texto1''
objLibro.Worksheets(1).Range(casilla).Value = ''Texto2''
```

Estas son dos formas de llenar una Hoja, lógicamente la primera es más útil para llenar datos específicos, que debieran estar en una casilla determinada, y la segunda se utiliza para rellenar con datos que no se saben cuando van a terminar.

Ej: Extracción de datos de un archivo y traspasarlos a casillas verticales a través de un Bucle (while, for, etc)

Leer: Se lee de la misma forma en que se escribe pero esta vez el valor se coloca en un texto, label o donde sea que lo necesiten.

2.4.2. Ejemplo N° 2.

```
Texto1.Text = objLibro.Worksheets(1).Range(''A1'').Value
```

Guardar: Si es necesario guardar los cambios en nuestro trabajo debemos agregar el siguiente comando:

```
objLibro.Save
```

Cerrar: Para finalizar una operación en nuestra hoja de trabajo cerramos el libro con el siguiente comando.

```
objLibro.Close
```

2.4.3. Ejemplo N° 3.

```
Dim objExcel As Object
Dim objLibro As Object
Dim ruta As String

On Error Resume Next
Set objExcel = GetObject(, "Excel.Application")     'Seteamos los objetos
If Err.Number = 429 Then
  Err.Clear
  Set objExcel = CreateObject("Excel.Application")
End If

ruta = App.Path \ & " test1.xls"     'Ruta de un archivo de Excel existente
Set objLibro = objExcel.Workbooks.Open(ruta)     'Abrimos un libro existente
  'Escribimos
objLibro.Worksheets(1).Range(''A1'').Value = ''Texto de Prueba''
  'Leemos
Text1.Text = objLibro.Worksheets(1).Range("A1").Value
  'Escribimos lo leido}
objLibro.Worksheets(1).Range(''A2'').Value = Text1.Text

objLibro.Save     'Guardamos el libro
objLibro.Close     'Cerramos el libro

objExcel.Quit     'Eliminamos variables
Set objLibro = Nothing     'Reseteamos las variables
Set objExcel = Nothing
```

La idea de trabajar en Excel es crear documentos tales como Facturas, Boletas, Listados de Repuestos, etc. más que aquellos que son tan solo para rellenar con datos y hacer cálculos. Es por esto que es recomendable tener la Hoja preparada una vez que vayamos a escribir.

Es por esto que no seguiremos con el tema de Excel (aunque habrán unos ejemplos más completos) pues la escritura es lo único necesario para estos requerimientos. También es útil tener los cálculos hechos en la Hoja de Excel, en caso que se deban realizar cambios. De esta forma se podrán realizar manualmente y no tener que volver a la aplicación.

2.4.4. Ejemplo Nº 4.

Para este ejemplo debe existir un "archivo1.txt" que debe tener la siguiente estructura:

dato1, dato2, dato3

después de dato3 el cursor debe quedar en la línea inferior

```
Dim objExcel As Object
Dim objLibro As Object
Dim i As Integer
Dim archivo, var1, var2, var3 As String
Dim ruta, casilla1, casilla2, casilla3 As String

On Error Resume Next
Set objExcel = GetObject(, "Excel.Application")
If Err.Number = 429 Then
  Err.Clear
  Set objExcel = CreateObject("Excel.Application")
End If

ruta = App.Path & "\test1.xls"

Set objLibro = objExcel.Workbooks.Open(ruta)

archivo = App.Path & "\archivo1.txt"
Open archivo For Input As #1

i = 0
While Not EOF(1)
  Input #1, var1, var2, var3

  i = i + 1
  casilla1 = "A" & i
  casilla2 = "B" & i
  casilla3 = "C" & i

  objLibro.Worksheets(1).Range(casilla1).Value = var1
  objLibro.Worksheets(1).Range(casilla2).Value = var2
  objLibro.Worksheets(1).Range(casilla3).Value = var3

Wend
Close #1

objLibro.Worksheets(1).Range("E1").Value = Date
objExcel.Visible = True
Set objLibro = Nothing
Set objExcel = Nothing
```

3. Word

Antes de empezar debemos agregar la referencia que vamos a ocupar.

```
Proyecto > Referencias > Microsoft Word 10.0 Object}
```

Consideraremos que tenemos un archivo existente llamado test1.doc para todos los efectos de escritura.

3.1. Definición de variables.

```
Dim ObjWord As Object
Dim DocdeWord As Object
```

De la misma forma que seteamos los objetos para Excel lo haremos para Word. ObjWord será la variable que utilizaremos para referirnos a la aplicación Word, mientras que DocdeWord nos dirá que estamos trabajando sobre el documento en si y nos permitirá acceder a todas la funciones básicas de Word, como negritas, crear tablas, cambiar el tipo de letra y su color, etc.

3.2. Ejemplo Demostrativo:

```
Dim ObjWord As Object
Dim DocdeWord As Object
Dim ruta As String

On Error Resume Next
Set ObjWord = GetObject( , "Word.Application")
  If Err.number = 429 Then
    Err.Clear
    Set ObjWord = CreateObject("Word.Application")
  End If

ruta = App.Path \ "\test1.doc"          'Ruta de un archivo de Word existente
Set DocDeWord = ObjWord.Documents.Open(ruta)  'Abrimos el documento existente

ObjWord.Visible = True                  'Mostramos en pantalla
ObjWord.Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter     'centramos

ObjWord.Selection.Font.Size = 20        'seteamos el tamaño de letra
ObjWord.Selection.TypeText "Titulo"     'Escribimos
ObjWord.Selection.TypeParagraph         'Salto de línea
ObjWord.Selection.TypeParagraph         'Salto de línea
ObjWord.Selection.Font.Size = 12        'seteamos el tamaño de letra
ObjWord.Selection.ParagraphFormat.Alignment = wdAlignParagraphJustify   'Justificar

ObjWord.Selection.TypeText "Aqui ponemos el texto"     'Escribimos
```

```
ObjWord = Nothing                          'Reseteamos las variables
DocdeWord = Nothing
```

3.3. Ejemplo.

En este ejemplo realizaremos la misma idea del ejemplo demostrativo, pero agregaremos un `With` con el objeto `ObjWord`, con el fin de no tener que repetir numerosas veces lo mismo y tendremos un control sobre la existencia del archivo.

```
Dim ObjWord As Object
Dim DocdeWord As Object
Dim ruta As String

On Error Resume Next
Set ObjWord = GetObject( , "Word.Application")
  If Err.number = 429 Then
    Err.Clear
    Set ObjWord = CreateObject("Word.Application")
  End If

ruta = App.Path & "\test1.doc"     'Ruta de un archivo de Word existente

If Len(Dir(ruta)) > 0 Then     'Verificamos la existencia del archivo
  Set DocDeWord = ObjWord.Documents.Open(ruta)       'Abrimos el documento existente

ObjWord.Visible = True

With ObjWord.Selection
 .ParagraphFormat.Alignment = wdAlignParagraphCenter       'centrar
 .Font.Size = 20                                           'seteamos el tamaño de letra
 .TypeText "Titulo"                                        'Escribimos
 .TypeParagraph                                            'Salto de línea
 .TypeParagraph                                            'Salto de línea
 .Font.Size = 12                                           'seteamos el tamaño de letra
 .ParagraphFormat.Alignment = wdAlignParagraphJustify      'Justificar
 .TypeText "Aquí ponemos el texto"                         'Escribimos
End With

Else
  MsgBox "El archivo no existe"
 End If

ObjWord = Nothing                                  'Reseteamos las variables
DocdedWord = Nothing
```

3.4. Opciones.

He aquí un pequeño listado de opciones, las cuales lograran satisfacer nuestras necesidades básicas, de no ser así, en el "Examinador de Objetos" de VB6 seleccionar solo la referencia de Word. Ahí se encontraran todos los elementos que provee VB para estos efectos. Estas opciones están ligadas a un `With` para simplicidad de la escritura y no tener que volver a repetir el objeto.

Cambiar el Font	`.Font.name = ''Verdana''`
Tamaño del Font	`.Font.Size = 20`
Color del Font	`.Font.color = wdColorBlack`
Negritas	`.Font.Bold = True 'False para desactivar`
Subrayado	`.Font.Underline = True ' False para desactivar`
Cursiva	`.Font.Italic = True 'False para desactivar`
Escribir	`.TypeText ''Texto a escribir''`
Salto de Línea	`.TypeParagraph`
Justificar	`.ParagraphFormat.Alignment = wdAlignParagraphJustify`
Centrar	`.ParagraphFormat.Alignment = wdAlignParagraphCenter`
Izquierda	`.ParagraphFormat.Alignment = wdAlignParagraphLeft`
Derecha	`.ParagraphFormat.Alignment = wdAlignParagraphRight`

3.5. Manejo de Tablas.

A continuación seguimos listando opciones, pero ahora será especificas para el manejo de tablas. El numero que llevan las tablas es un índice, que generalmente es 1, de hecho cuando se genera una nueva tabla su índice es uno y se elimina de la tabla anterior, eso significa que creamos otra tabla no podremos ingresar datos a una tabla anterior.

Crear Tabla	`.Tables.Add .Range, n°columnas, n°filas`
Agregar Fila	`.Tables(1).Rows.Add`
Bordes	`.Tables(1).Cells.Borders = True`
Escribir en la	`.Tables(1).Cell(n°columna, n°fila).Select`
Tabla	`.TypeText ''Marca''`
Salir de la tabla	`.MoveDown`
AutoAjuste	`.Tables(1).AllowAutoFit = True`
Alineación	`.Tables(1).Rows.Alignment=wdAlignRowCenter`
Selección Columna	`.Tables(1).Columns(n°columna).Select`

Consejo A medida que se escribe y se va generando un documento cada vez más complejo, es bueno ejecutar el código que estamos creando, eso nos mostrará en pantalla lo que llevamos realizado hasta el momento. Una vez visto la página de Word, avanzamos a una nueva página en blanco y ejecutamos nuevamente el código para hacer lo mismo, pero ahora nos cambiamos rápidamente a la ventana de Word, para ver como es que va escribiendo todo lo realizado. Esto es muy útil para generar una idea final de lo que se debe hacer para obtener el resultado esperado, especialmente cuando se esta trabajando con las tablas. (Se recomienda lo mismo si se trabaja con Excel)

3.6. Ejemplos.

3.6.1. Ejemplo Nº 1:

Este ejemplo es sumamente simple, es por esto que se eliminaron los comentarios. Lo que hace es cambiar el color de letra y aplicarle sucesivamente Cursiva, Subrayado y Negrita. (Para conocer los colores que pueden adquirir las letras, ir a "Examinador de objetos", elegir la referencia de Word y buscar por "wdColor")

```
Dim DocdeWord As Object
Dim ObjWord As Object
Dim ruta As String

On Error Resume Next
Set ObjWord = GetObject(, "Word.Application")
If Err.Number = 429 Then
  Err.Clear
  Set ObjWord = CreateObject("Word.Application")
End If

ruta = App.Path & "\test1.doc"
Set DocdeWord = ObjWord.Documents.Open(ruta)

With ObjWord.Selection
   .Visible = True
   .ParagraphFormat.Alignment = wdAlignParagraphCenter
   .Font.Size = 20
   .Font.Color = wdColorBlack
   .Font.Bold = True
   .TypeText "Titulo"
   .Font.Bold = False
   .TypeParagraph
   .Font.Size = 12
   .TypeParagraph
   .ParagraphFormat.Alignment = wdAlignParagraphJustify
   .Font.Color = wdColorRed
   .TypeText "Este texto es de color Rojo"
   .TypeParagraph
   .Font.Color = wdColorBlue
   .Font.Italic = True
   .TypeText "Este texto tiene Cursivas y de color Azul"
   .TypeParagraph
   .Font.Color = wdColorViolet
   .Font.Underline = True
   .TypeText "Este texto tiene Cursivas, Subrayado y es de color Violeta"
   .TypeParagraph
   .Font.Bold = True
   .TypeText "Este texto tiene Cursivas, Subrayado, Negritas y es de color Violeta"
End With
```

3.6.2. Ejemplo Nº 2:

```
Dim DocDeWord As Object
Dim ObjWord As Object
Dim ruta As String

On Error Resume Next
Set ObjWord = GetObject(, "Word.Application")
If Err.Number = 429 Then
  Err.Clear
  Set ObjWord = CreateObject("Word.Application")
End If

ruta = App.Path & "\test1.doc"

Set DocDeWord = ObjWord.Documents.Open(ruta)

With ObjWord.Selection
   .Tables.Add .Range, 1, 6            'Creación una tabla de 1x6
   .Tables(1).Rows.Add                 'Adición de una columna

   .Tables(1).Rows(1).Select           'Selección de  la 1ª fila
   .Font.Bold = True                   'y Negritas

   .Tables(1).Cell(1, 1).Select        'Selección celda
   .TypeText "Marca"                   'Escritura
   .Tables(1).Cell(1, 2).Select
   .TypeText "Modelo"
   .Tables(1).Cell(1, 3).Select
   .TypeText "Color"
   .Tables(1).Cell(1, 4).Select
   .TypeText "Año"
   .Tables(1).Cell(1, 5).Select
   .TypeText "Puertas"
   .Tables(1).Cell(1, 6).Select
   .TypeText "Patente"
   .Tables(1).Cell(2, 1).Select
   .TypeText "Chrevrolet"
   .Tables(1).Cell(2, 2).Select
   .TypeText "Cavalier"
   .Tables(1).Cell(2, 3).Select
   .TypeText "Rojo"
   .Tables(1).Cell(2, 4).Select
   .TypeText "2000"
   .Tables(1).Cell(2, 5).Select
   .TypeText "4"
   .Tables(1).Cell(2, 6).Select
   .TypeText "DFGT56"

   .MoveDown                                    'salimos de la tabla
```

```
   .TypeParagraph
   .TypeParagraph

   .Tables.Add .Range, 4, 4                          'Creación de tabla 4x4
   .Tables(1).AllowAutoFit = True
   .Tables(1).AllowPageBreaks = False
   .Tables(1).Rows.Alignment = wdAlignRowCenter
   .Tables(1).Columns(1).Select                      'Selección de Columna
   .Font.Bold = True                                 'Negritas
   .Tables(1).Columns(3).Select                      'Selección de Columna
   .Font.Bold = True                                 'Negritas

   .Tables(1).Cell(1, 1).Select
   .TypeText "Cia. Seguros"
   .Tables(1).Cell(2, 1).Select
   .TypeText "Clientes"
   .Tables(1).Cell(3, 1).Select
   .TypeText "Tercero"
   .Tables(1).Cell(4, 1).Select
   .TypeText "Siniestro N°"
   .Tables(1).Cell(1, 3).Select
   .TypeText "Liquidador"
   .Tables(1).Cell(2, 3).Select
   .TypeText "Póliza"
   .Tables(1).Cell(3, 3).Select
   .TypeText "Telefono1"
   .Tables(1).Cell(4, 3).Select
   .TypeText "Telefono2"

   .Tables(1).Cell(1, 2).Select
   .TypeText "CorpV"
   .Tables(1).Cell(2, 2).Select
   .TypeText "Oscar G."
   .Tables(1).Cell(3, 2).Select
   .TypeText "-----"
   .Tables(1).Cell(4, 2).Select
   .TypeText "4563"
   .Tables(1).Cell(1, 4).Select
   .TypeText "-----"
   .Tables(1).Cell(2, 4).Select
   .TypeText "34545"
   .Tables(1).Cell(3, 4).Select
   .TypeText "4586625"
   .Tables(1).Cell(4, 4).Select
   .TypeText "09574815"

   .MoveDown                              'salir de la tabla
   .TypeParagraph

End With
```

3.7. Explicación ejemplo anterior.

En el ejemplo anterior se crean dos tablas. La primera de ellas de 1x6, y luego se le agrega una segunda fila, finalmente la tabla queda de 2x6. Esta es una forma de crear tablas que no cuentan con un numero fijo de de filas (o columnas). Después se debe seleccionar la casilla para poder escribir en ella, también se puede seleccionar filas o columnas y darles características especiales, tales como Negritas. ¿Como salir de la tabla?, de la misma forma que se hace regularmente se sitúa el cursor en la ultima casilla y se baja. La segunda tabla ya estaba establecida de antemano y su tamaño es de 4x4, lo demás es de la misma forma que en la tabla anterior. Cabe notar que nos referimos cada vez y a cada tabla con el número 1 de la siguiente forma:

```
.Tables(1).Cell(x, x).Select
```

esto para indicar que se trata de la tabla en la cual estamos situados en este momento. A la segunda tabla se le agregan una serie de características que no explicare ya que tan solo de modo de ejemplo, y los encontrarán en el "Examinador de Objetos"

Me cuesta imaginarme como es que se escriben las tablas ¿Qué puedo hacer? Tal como se mencionó anteriormente la mejor manera de saber si el resultado es lo que realmente se espera es ejecutar el código, y abrir rápidamente el documento en que se escribirá, de esta manera se genera una idea global de lo que estamos haciendo y como Visual lo va creando.

4. Access.

Antes de empezar debemos agregar la referencia que vamos a ocupar.

```
Proyecto > Referencias > Microsoft DAO 3.51 Object Library}
```

4.1. Declaración de Variables.

```
Public Base As DAO.Database
Public Tabla As DAO.TableDef
Public campo As DAO.Field
Public db As DAO.Recordset
```

La definición de las variables se realizan dentro de un modulo o de forma Explicita (`Option Explicit`) al principio de un formulario, para que puedan alcanzar el ámbito global de la aplicación que se desea desarrollar, de esta forma se podrán llamar de cualquier lugar sin tener que definir nuevamente todas las variables cada vez que se deba acceder a la base de datos.

A continuación se presenta una definición más detallada para cada una de las variables que se podrán utilizar en el transcurso de una aplicación:

`Public Base As DAO.Database`**:** Base será la variable que se referirá a la base de datos como archivo (extensión .mdb), para crear, abrir o cerrar una determinada base.

`Public Tabla As DAO.TableDef` **y** `Public campo As DAO.Field`**:** Tabla se refiere a una tabla específica dentro de la base, esto solo es útil para la creación de la base de datos, proceso en el cual debemos establecer los atributos de la tabla, es decir, el nombre de esta. La variable campo realiza funciones semejantes pero a nivel de datos. Esta variable establecerá el nombre y el tipo de los datos en una tabla especifica durante la creación de una base de datos.

`Public db As DAO.Recordset`**:** Por ultimo, la variable bd será utilizada para efectuar todos los movimientos dentro de la base de datos, sean estos consultas, agregar, eliminar o actualizar datos dentro de esta.

Importante: En caso de que la base de datos sea creada directamente a través de código en Visual Basic 6 esta se encontrará en el formato dado para OFFICE 97, por lo tanto en caso de querer visualizar la base de datos creada en otra versión de OFFICE, pedirá "Convertir la base de datos", lo cual impedirá un manejo posterior de esta. Lo que se debe hacer es "Abrir la base de datos" y por ningún motivo convertirla.

4.2. Creación de la base de Datos. Forma Genérica.

```
Set Base = CreateDatabase("Nombre_base_de_datos", dbLangGeneral)
Set Tabla = Base.CreateTableDef("Nombre_tabla")
Set campo = Tabla.CreateField("Nombre_campo", tipo)
Tabla.Fields.Append campo    'Se agrega el campo a la tabla
Base.TableDefs.Append Tabla      'Se agrega la tabla a la base de datos
Base.Close
```

4.3. Ejemplo Demostrativo:

```
Set Base = CreateDatabase("test1", dbLangGeneral)
Set Tabla = Base.CreateTableDef("Ingreso")
Set campo = Tabla.CreateField("Clave", dbText, 6)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Login", dbText, 15)
Tabla.Fields.Append campo
Base.TableDefs.Append Tabla
Set Tabla = Base.CreateTableDef("Cliente")
Set campo = Tabla.CreateField("Login", dbText, 15)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Saldo", dbInteger)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Direccion", dbText, 30)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Ciudad", dbText, 30)
Tabla.Fields.Append campo
Base.TableDefs.Append Tabla
Base.Close
```

En este ejemplo se ha creado una base de datos llamada "`test1.mdb`", que contiene dos tablas: "Ingreso" y "Cliente". Los datos contenidos en estas tablas son en su mayoría String a los cuales se les debe agregar la longitud de estos. El campo saldo corresponde a un entero.

4.4. Manejo de la base de Datos. Consultas.

```
Set db = Base.OpenRecordset _
(‘‘Select * from Nombre_Tabla’’, dbOpenDynaset, dbOptimistic)
```

4.4.1. Ejemplo Demostrativo.

```
Dim ruta As String
ruta = App.Path & "\test1.mdb"
Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
(‘‘Select * from Cliente’’, dbOpenDynaset, dbOptimistic)

bd.Close
Base.Close
```

En el ejemplo demostrativo se realiza una consulta en la tabla cliente y selecciona todos los datos que se encuentran en esta tabla. En caso que se desee seleccionar un grupo específico de datos se realiza como una consulta SQL común y corriente. Suponga que debe recuperar los datos de un cliente, cuyo login es “Policarpo”:

4.5. Ejemplos.

4.5.1. Ejemplo N^{o} 1.

```
Dim ruta As String

ruta = App.Path & "\test1.mdb"
Set Base = OpenDatabase(ruta)

Set db = Base.OpenRecordset _
("Select * from Cliente Where Login = 'Policarpo'", dbOpenDynaset, dbOptimistic)

Text1.Text = db!Nombre
Text2.Text = db!Ciudad
Text3.Text = db!Saldo

bd.Close
Base.Close
```

4.5.2. Ejemplo N° 2.

El siguiente ejemplo se utiliza para realizar consultas con una variable, es decir, haremos lo mismo que en el ejemplo anterior pero ahora se ingresara el login en `Text1.text`.

```
Dim ruta As String

ruta = App.Path & "\test1.mdb"
Set Base = OpenDatabase(ruta)

Set db = Base.OpenRecordset _
"Select * from Cliente Where Login = '" & Text1.text & "'", dbOpenDynaset, dbOptimistic)

Text2.Text = db!Nombre
Text3.Text = db!Ciudad
Text4.Text = db!Saldo

bd.Close
Base.Close
```

Note que lo que se ha realizado no es más que una concatenación de la variable `Text1.text` con el string de consulta, y en caso de que no se entienda bien la sentencia se explica a continuación con la siguiente ecuación:

```
simple + dobles + & + variable + & + dobles + simple + dobles
```

Lógicamente simple y dobles se refieren a las comillas, esto se hizo para facilitar la escritura, de esta forma se visualizar mejor el orden de los signos.

En este ejemplo también se realizo una lectura de la base de datos, a través de la siguiente línea de código

```
Text2.Text = db!Nombre
```

Se entiende que esta es la forma de rescatar datos de la base, utilizando la variable de consulta, en este caso `db`. La estructura es sumamente fácil, y corresponde a la variable de consulta con el campo deseado unidos por un signo de exclamación.

```
db!campo
```

Nota: Recuerde que `db` es la variable de consulta utilizada para este caso y no corresponde a un nombre genérico.

4.6. Agregar, Eliminar y Modificar datos.

4.6.1. Ejemplo

Para el siguiente ejemplo se recreará el caso de una automotriz para lo cual se creará una nueva base de datos con los siguientes atributos:

OT = orden de trabajo

Creación de la base de datos.

```
Set Base = CreateDatabase("Automotriz", dbLangGeneral)
Set Tabla = Base.CreateTableDef("Orden")
Set campo = Tabla.CreateField("OT", dbText, 7)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Patente", dbText, 6)
Tabla.Fields.Append campo
Base.TableDefs.Append Tabla
Set Tabla = Base.CreateTableDef("Cliente")
Set campo = Tabla.CreateField("Patente", dbText, 6)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Nombre", dbText, 50)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Direccion", dbText, 50)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Ciudad", dbText, 20)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Telefono", dbText, 9)
Tabla.Fields.Append campo
Base.TableDefs.Append Tabla
Set Tabla = Base.CreateTableDef("Valores")
Set campo = Tabla.CreateField("OT", dbText, 7)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Desabolladura", dbText, 10)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Pintura", dbText, 10)
Tabla.Fields.Append campo
Set campo = Tabla.CreateField("Reparacion", dbText, 10)
Tabla.Fields.Append campo
Base.TableDefs.Append Tabla
Base.Close
```

Agregar datos (AddNew)

Una vez creada la base de datos, debemos proceder a llenarla.

```
ruta = App.Path & "\Automotriz.mdb"
Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
("Select * from Cliente", dbOpenDynaset, dbOptimistic)
db.AddNew
db!Patente = patente.Text
db!Nombre = nombre.Text
db!Direccion = direccion.Text
db!Ciudad = ciudad.Text
db!Telefono = telefono.Text
db.Update
db.Close
Base.Close
```

Modificar datos (Edit)

En caso que se deban actualizar datos se debe hacer lo siguiente. Se selecciona solamente aquellos datos que se desean cambiar, en este caso se supone que una patente es única para cada cliente, por lo tanto un cliente se puede reconocer por su patente.

```
ruta = App.Path & "\Automotriz.mdb"
Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
("Select * from Cliente Where  = '" & patente.Text & "'", dbOpenDynaset, dbOptimistic)
db.Edit
db!Patente = patente.Text
db!Nombre = nombre.Text
db!Direccion = direccion.Text
db!Ciudad = ciudad.Text
db!Telefono = telefono.Text
db.Update
db.Close
Base.Close
```

La línea de comando `db.Update` lo que hace es actualizar los datos por eso se realiza para agregar y modificar datos, pero no para eliminarlos como se vera a continuación, porque en caso de realizar una consulta a datos eliminados simplemente no se encontrara nada, pero se podría llegar a error en caso de consultar a una base datos que no ha sido actualizada.

Eliminar datos (`Delete`)

Cuando se desea eliminar se debe ser muy cuidadoso en seleccionar tan solos los datos no deseados, pues un error puede llevar a perder toda la información de la base de datos. Además el programador deberá tomar en cuenta las referencias lógicas de la base de datos, dado que Access no lo hace y esto podría causar problemas con la coherencia de los datos

```
ruta = App.Path & "\Automotriz.mdb"
Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
("Select * from Cliente Where  = '" & patente.Text & "'", dbOpenDynaset, dbOptimistic)
If Not db.EOF Then
  db.Delete
End If
db.Close
Base.Close
```

4.7. Ejemplos.

4.7.1. Ejemplo Nº 1.

En caso de que al realizar una consulta se encuentre más de una solución se debe recorrer las soluciones posibles hasta encontrar la deseada simplemente mostrarlas todas. En este ejemplo se supone que existe una patente con más de una OT y todas estas OT se agregarán a una lista.

```
Dim patente, ruta, n_ot As String
patente = patente.Text
List1.Clear
ruta = App.Path & "\Automotriz.mdb"
Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
("Select OT from Orden Where Patente = '" & patente & "'", dbOpenDynaset, dbOptimistic)
If Not db.EOF Then
  While Not db.EOF
    n_ot = db!OT
    List1.AddItem n_ot
    db.MoveNext
 Wend
End If
db.Close
Base.Close
```

El comando `db.MoveNext` es el que realiza el movimiento entre las distintas columnas de datos ubicadas en la variable de consulta.

Movimiento dinámico: Tomando en cuenta el ejemplo anterior, con la diferencia que la variable de consulta y la base de datos quedan abiertas para una próxima interacción:

```
'Ir al Inicio
If db.RecordCount <> 0 Then
  db.MoveFirst
End If


'Ir al Final
If db.RecordCount <> 0 Then
  db.MoveLast
End If


'Ir al Proximo
'Si se encuentra al Final vuelve al Inicio
If db.RecordCount <> 0 Then
  db.MoveNext
  If db.EOF Then
    db.MoveFirst
  End If
End If


'Ir al Anterior
'Si se encuentra al Final vuelve al Inicio
If db.RecordCount <> 0 Then
  db.MovePrevious
  If db.BOF Then
    db.MoveLast
  End If
End If
```

La línea de comando db.RecordCount es un contador que se mueve como un puntero sobre la columna de datos de la variable de consulta, si es cero significa que la variable de consulta esta vacía. Si se realiza un movimiento en una variable vacía se provocaría un error que posiblemente haría caer nuestra aplicación.

4.7.2. Ejemplo Nº 2.

```
Dim ruta, pat, n_ot As String

n_ot = n_ot.Text
ruta = App.Path & "\Automotriz.mdb"

Set Base = OpenDatabase(ruta)
Set db = Base.OpenRecordset _
("Select Patente from Cliente where OT = '" & n_ot & "'", dbOpenDynaset, dbOptimistic)

pat = db!patente

Set db = Base.OpenRecordset _
("Select * from Cliente where patente = '" & pat & "'", dbOpenDynaset, dbOptimistic)

Nombre.Text = db!Nombre
Direccion.Text = db!Direccion
Ciudad.Text = db!Ciudad
telefono.Text = db!telefono

db.Close

Set db = Base.OpenRecordset _
("Select * from Valores where OT = '" & n_ot & "'", dbOpenDynaset, dbOptimistic)

desabolladura.Text = db!desabolladura
pintura.Text = db!pintura
Reparacion.Text = db!Reparacion

db.Close
Base.Close
```

Referencias

Visual Basic
www.lawebdelprogramador.com
SQL
www.sqlzoo.net

En estas direcciones encontrarán gran cantidad de información sobre estos lenguajes.

Agradecimientos

Agradecimientos a:
Luis Arévalo
Roberto Bonvallet

Dos grandes amigos, gracias por su apoyo y por su ayuda a traspasar este documenton a latex.