SEMINARIO DE LENGUAJES – OPCION DELPHI – MODULO II
Ejercicios de Repaso

1. Cuales de las siguientes clases de componentes pueden ser pegadas en un Módulo de Datos:

   A- TSaveDialog
   B- TPanel
   C- TADOQuery
   D- TLabel
   E- TImageList
   F- TADOConnection

   Rta: Para poder ser pegado en un módulo de datos, las componentes no deben ser controles, es decir no deben tener parte visual. Por lo tanto, pueden pegarse las opciones A, C, E y F

2. Dado el código de la derecha, responda las siguientes preguntas:

   a) ¿Existe algún error detectable en tiempo de compilación? Si es así explique cómo subsanarlo.

   b) Asumiendo que se ha arreglado cualquier error de compilación que pudiese existir. ¿Hay alguna otra corrección que debería realizarse para que el programa funcione correctamente en tiempo de ejecución?

```
procedure TForm6.Button1Click(Sender: TObject);
var i:integer; vector:array of string;
begin
 setlength(vector,ControlCount);
 for i := 1 to ControlCount do
  if Controls[i] is TButton
   then vector[i]:=Controls[i].Caption
   else vector[i]:='No es botón';
end;
```

Rta:

a) El error que se detecta en tiempo de compilación es el de intentar acceder a la propiedad Caption del componente Controls[i] sin convertirlo previamente al tipo de componente apropiado, en este caso un botón. Para resolver este problema se debe escribir (Controls[i] as TButton).Caption.

b) Tanto los controles como los elementos del vector se acceden con índices numéricos comenzando en 0, no en 1. Debería cambiarse el for haciendo que i tome valores entre 0 y ControlCount-1.

El código correcto sería:

```
procedure TForm6.Button1Click(Sender: TObject);
var i:integer; vector:array of string;
begin
  setlength(vector,ControlCount);
  for i := 0 to ControlCount-1 do
    if Controls[i] is TButton
      then vector[i] := (Controls[i] as TButton).Caption
      else vector[i]:='No es botón';

 end;
```

3. Se dispone de información de películas de un video club organizada de la siguiente forma:

Categorías

Cod_Categoria	Número
Descripción	Texto

Ejemplos de categorías son: Drama, Suspenso, Comedia, etc.

Películas

Cod_Pelicula	Número
Cod_Categoria	Número
Título	Texto
Actor_Principal	Texto
Director	Texto
Fecha_estreno	Fecha

Indique la forma de obtener los títulos de las categorías a las que pertenecen las películas estrenadas en un año dado. El año se ingresa mediante un componente **`TEdit`** y la información solicitada se recupera al hacer click sobre un botón. Verifique que no aparezcan categorías repetidas. Visualice la información en un **`ListBox`**.

Rta: Consulta cargada en la propiedad SQL de un component ADOQUERY asociado a la BBDD

```
select  distinct descripcion
from Peliculas P , Categorias C
where (P.cod_categoria = C.cod_categoria) and
      (fecha_estreno >= :fecha1) and
      (fecha_estreno <= :Fecha2)
```

(OnClick del botón)

```
procedure TForm1.Button1Click(Sender: TObject);
var FInicio, FFin : String;
begin
    FInicio := '01/01/'+edit1.text;
    FFin := '31/12/'+edit1.text;

    ADOQuery1.Close;
    ADOQuery1.Parameters.ParamByName('Fecha1').Value :=
                                                StrToDate(FInicio);
    ADOQuery1.Parameters.ParamByName('Fecha2').Value :=
                                                StrToDate(FFin);
    ADOQuery1.Open;

    // Cargando los nombres de las categorias en el ListBox
    ListBox1.Clear;
    while not ADOQuery1.eof do
    begin
      ListBox1.items.Add(ADOQuery1.FieldByName('descripcion').AsString);
      ADOQuery1.Next;
    end;

end;
```

4. ¿Qué es un campo calculado? Indique la manera de agregar un campo calculado a la tabla CATEGORIAS del ejercicio anterior a fin de conocer la cantidad de películas que hay de cada categoría. Realice lo solicitado
    a) SIN utilizar consultas
    b) Utilizando consultas

Rta: La definición de Campo Calculado y la manera de definir un nuevo campo está indicado en las transparencias vistas en clase.

Los eventos OnCalcFields de cada inciso se describen a continuación.

```
//Sin utilizar consultas
procedure TForm1.ADOTableCategoriasCalcFields(DataSet: TDataSet);
var cod, cant : Integer;
begin
  cod := ADOTableCategorias.fieldbyname('cod_Categoria').AsInteger;

  cant := 0;
  ADOTablePeliculas.open;
  ADOTablePeliculas.First;
  while not ADOTablePeliculas.Eof do
  begin
  if ADOTablePeliculas.fieldbyname('cod_categoria').AsInteger=cod then
            cant := cant + 1;
      ADOTablePeliculas.Next;
  end;
  ADOTableCategorias.FieldByName('Cant_Peliculas').AsInteger := cant;
end;
```

```
Sentencia select del componente ADOQuery2

select count(*) as cantidad
from peliculas
where cod_categoria = :cod

//Usando consultas
procedure TForm1.ADOTableCategoriasCalcFields(DataSet: TDataSet);
var cod, cant : Integer;
begin
  cod := ADOTableCategorias.fieldbyname('cod_categoria').AsInteger;
  ADOQuery2.close;
  ADOQuery2.parameters.parambyname('cod').value := cod;
  ADOQuery2.Open;
  cant := ADOQuery2.FieldByName('cantidad').AsInteger;
  ADOTableCategorias.FieldByName('Cant_Peliculas').AsInteger := cant;
end;
```

5. Indique las similitudes y diferencias entre los componentes **DBComboBox** y **DBLookupComboBox**. ¿Cuál resultaría más adecuado para ingresar el valor del campo **Cod_Categoría** de la tabla PELICULAS definida en el ejercicio 7? Justifique.

   Rta: Los componentes DBComboBox y DBLookupComboBox fueron definidos y ejemplificados en clase (ver transparencias). En este caso particular, para ingresar el valor del campo Cod_categoría, debe utilizarse un **DBLookupComboBox** ya que tiene la capacidad de seleccionar un registro de un dataset y copiar el valor de un campo indicado (propiedad KeyField) de dicho dataset (propiedad ListSource) en otra tabla (propiedades datasource y datafield). La lista que se visualiza en el componente DBLookupComboBox (propiedad listfield) se encuentra directamente relacionada con el contenido de un dataset . Además, la información visualizada en el momento de la selección no necesariamente tiene que coincidir con lo que se va a almacenar. En este caso, podría seleccionarse una categoría por su nombre y almacenar el código correspondiente. En cambio, el componente DBCombobox permite seleccionar un string de una lista o ingresar uno nuevo (propiedad text del componente DBComboBox). En este caso el campo Cod_categoría es numérico, lo que implica una operación adicional para convertir el texto seleccionado o ingresado en un número. Además, la lista de códigos disponibles debería cargarse expresamente en el componente DBComboBox.

6. Codifique el procedimiento Clasificar cuyo encabezado se muestra a continuación:

   procedure Clasificar(Texto: TMemo; listaEnteros, listaNoEnteros: TListBox);

   Este procedimiento debe agregar a la lista listaEnteros todas las líneas de Texto que representen números enteros y a la lista listaNoEnteros todas las restantes. Además debe agregar como última línea de la lista listaEnteros la suma de todos los números que fueron agregados.

Rta:

```
Procedure Clasificar(Texto: TMemo; listaEnteros, listaNoEnteros:
TListBox);
Var
   suma, numero, i:integer;
   linea:String;
Begin
   suma := 0;
   listaEnteros.items.clear;
   listaNoEnteros.items.clear;
   for i:= 0 to Texto.lines.count - 1 do begin
        linea := Texto.lines.strings[i];
        try
             numero := StrToInt(linea);
             suma := suma + numero;
             listaEnteros.items.add(linea);
        except
             on EConvertError do begin
                  listaNoEnteros.items.add(linea);
             end;
        end;
   end;
   linea := IntToStr(suma);
   listaEnteros.items.add(linea);
End;
```

7. Enunciar tres propiedades y tres métodos del componente **TADOTable**

Rta:

Propiedades: Connection, Tablename, Active, IndexFieldNames, Filtered, Filter, …

Métodos: Open, Close, Next, First, FieldByName, Append, Edit, Delete, Cancel, Post, …

8. Se dispone de información acerca de los sueldos depositados a los trabajadores de una empresa en tablas con la siguiente estructura:

Sueldo		
Codigo_Empleado	Número	Código del empleado
Monto	Real	Monto del Sueldo
Fecha_Vencimiento	Fecha	Fecha en que debe depositarse el sueldo
Fecha_Deposito	Fecha	Fecha en que se realizó el depósito

Empleado	
Codigo	Número
CUIL	Texto
Fecha_Nacimiento	Fecha
Apellido_Nombre	Texto
Fecha de Ingreso	Fecha
Nacionalidad	Texto
Cuenta_Sueldo	Texto

   Se requieren las siguientes estadísticas:

   a) Listado de cuentas con depósitos realizados fuera de fecha. Visualice el resultado en un ListBox.

b) Nombre, apellido y fecha de ingreso del empleado más antiguo. Utilice ShowMessage para visualizarlo.

c) Cantidad de Empleados extranjeros (que no sean Argentinos) que ingresaron a la empresa luego del 01/01/1990. Muestre el resultado en un Edit.

d) Sueldo promedio por nacionalidad. Visualice el resultado en un Memo.

Rta:

--a) Listado de cuentas con depósitos realizados fuera de fecha. Visualice el resultado en un ListBox.

Propiedad SQL del componente ADOQuery3:

```
select E.cuenta_sueldo
from Empleado E, Sueldo S
where (E.codigo = S.codigo_empleado)
        and (S.fecha_deposito > S.fecha_vencimiento)

procedure TForm1.Button1Click(Sender:TObject);
var
        cuenta_sueldo: String;
begin
        ADOQuery3.close;
        ADOQuery3.open;
        listbox1.items.clear;
        while not ADOQuery3.EOF do begin
                cuenta_sueldo := ADOQuery3.fieldByName('cuenta_sueldo').asString;
                listbox1.items.add(cuenta_sueldo);
                ADOQuery3.next;
        end;
end;
```

--b) Nombre, apellido y fecha de ingreso del empleado más antiguo. Utilice ShowMessage para visualizarlo.

```
//Para obtener el empleado más antiguo se puede ordenar la tabla Empleado por fecha de ingreso
//en orden ascendente, de forma que el empleado con la fecha más antigua quede en el primer lugar
//en el resultado de la consulta.
```

Propiedad SQL del componente ADOQuery4:

```
select apellido_nombre, fecha_ingreso
from Empleado
order by fecha_ingreso

procedure TForm1.Button2Click(Sender:TObject);
var
        apellido_nombre: String;
        fecha_ingreso: TDateTime;
begin
        ADOQuery4.close;
```

```
        ADOQuery4.open;
        apellido_nombre := ADOQuery3.fieldByName('apellido_nombre').asString;
        fecha_ingreso := ADOQuery3.fieldByName('apellido_nombre').asTDateTime;
        showMessage('El empleado más antiguo es ' + apellido_nombre + ', e ingresó en la fecha ' +
DateToStr(fecha_ingreso));
end;
```

--c) Cantidad de Empleados extranjeros (que no sean Argentinos) que ingresaron a la empresa luego del 01/01/1990. Muestre el resultado en un Edit.

Propiedad SQL del componente ADOQuery5:

```
select count(*) as cantidad
from Empleado
where (nacionalidad <> 'Argentino') and (fecha_ingreso > #01/01/1990#)

procedure TForm1.Button3Click(Sender:TObject);
var
        cantidad:integer;
begin
        ADOQuery5.close;
        ADOQuery5.open;
        cantidad := ADOQuery5.fieldByName('cantidad').asInteger;
        edit1.text := IntToStr(cantidad);
end;
```

--d) Sueldo promedio por nacionalidad. Visualice el resultado en un Memo.

Propiedad SQL del componente ADOQuery6:

```
select E.nacionalidad, avg(S.monto) as sueldo_promedio
from Empleado E, Sueldo S
where E.codigo = S.codigo_empleado
group by E.nacionalidad

procedure TForm1.Button4Click(Sender:TObject);
var
        nacionalidad, linea:String;
        sueldo_promedio:real;
begin
        ADOQuery6.close;
        ADOQuery6.open;
        memo1.lines.clear;
        while not ADOQuery6.EOF do begin
                nacionalidad := ADOQuery6.fieldByName('nacionalidad').asString;
                sueldo_promedio := ADOQuery6.fieldByName('sueldo_promedio').asFloat;
```

```
            linea := nacionalidad + ' ' + FloatToStr(sueldo_promedio);
            memo1.lines.add(linea);
            ADOQuery6.next;
        end;
end;
```

9. Indicar Verdadero o Falso:
   a) Las componentes ListBox, ComboBox y ADOQuery tienen una propiedad TStrings.
   b) La función LOW aplicada a un arreglo dinámico siempre devuelve cero.
   c) Los tipos AnsiString y String son iguales.
   d) Los tipos AnsiString y String utilizan una representación basada en un contador de referencia.
   e) Si X e Y son dos variables de tipo String[255], la asignación X := Y podría llegar a producir un incremento de la memoria heap.
   f) La función SetLength se aplica tanto a strings como a arreglos dinámicos.
   g) El componente DBListBox puede utilizarse para visualizar una lista con el contenido de un campo de una tabla.
   h) Realizar un INNER JOIN entre las tablas EMPLEADO y SUELDO del ejercicio anterior, dará como resultado un conjunto de registros de mayor tamaño que el que contiene la tabla de EMPLEADO por sí sola.
   i) Si el valor indicado en la propiedad HelpContext de una componente no se encuentra definido dentro del archivo de ayuda, la instrucción **Application.HelpCommand(HELP_CONTEXT,10)**, levantará una excepción.

Rta:

a) Verdadero. Esta propiedad se llama Items en el ListBox y en el ComboBox, y SQL en el ADOQuery.

b) Verdadero. Esto es porque los arreglos dinámicos siempre tienen índice numérico y este siempre empieza en cero y Low retorna el índice de la primera posición de un arreglo.

c) Falso en Delphi 2010. En Delphi 2010 el tipo String es un alias de UnicodeString. Si bien tanto AnsiString como UnicodeString son dinámicos y utilizan contador de referencias, UnicodeString utiliza 2 bytes por carácter, mientras que AnsiString utiliza 1 byte por carácter.

d) Verdadero en Delphi 2010. Ver respuesta c)

e) Falso. Al declararlas como String[255] X e Y son variables de tipo ShortString, el cual es estático y por lo tanto no utiliza memoria heap.

f) Verdadero.

g) Falso. DBListBox permite escribir en un campo de una tabla un valor elegido de una lista predefinida. Para visualizar el contenido de un campo de una tabla se necesita utilizar un DBLookupListBox.

h) Falso, porque no se cumple siempre. Sólo es verdadero en el caso de que la tabla Sueldo tenga más registros que la tabla Empleado, lo que significaría que hay empleados que tienen más de un depósito a su nombre.

i) Falso. La instrucción indicada abrirá el archivo de ayuda en la página que tenga el código 10, sin importar el valor de la propiedad HelpContext de ningún componente. Además, HelpCommand no lanza una excepción cuando el número de una página de ayuda no está definido en el archivo de ayuda. El valor de la propiedad HelpContext de una componente sólo tendrá importancia cuando el usuario presione la tecla F1 y el componente en cuestión tenga el foco en ese momento. En ese caso se abrirá el archivo de ayuda en la página que tenga el código especificado por la propiedad HelpContext del componente.

10. Codifique un procedimiento que elimine todas las líneas de un control Memo que contengan la palabra tipeada por el usuario en un TEdit llamado EditPalabra.
    Ayuda: Recuerde que la función Pos(substr, str) devuelve la posición de substr dentro de str y puede utilizarse para saber si un String contiene a otro.

Solución 1:

```
procedure ExcluirLineas(memo:TMemo; editPalabra:TEdit);
var
   linea:String;
   i:integer;
begin
   i := 0;
   while (i <  memo.lines.count) do begin
         linea := memo.lines.strings[i];
         if pos(editPalabra.text, linea) > 0 then
               memo.lines.delete(i)
         else
               i := i + 1;
   end;
end;
```

Notar que se usó un While y no un For porque al borrar una línea se modifica la cantidad de líneas del TStrings y además la línea i+1 pasa a ser la nueva línea i. Por ello sólo se incrementa i cuando no se encuentra el String buscado.

Solución 2:

```
procedure ExcluirLineas(memo:TMemo; editPalabra:TEdit);
var
  linea:String;
  i:integer;
begin
  for I := memo.lines.count - 1 downto 0 do begin
    linea := memo.Lines.Strings[i];
    if pos(editPalabra.Text, linea) > 0 then
      memo.Lines.Delete(i);
  end;
end;
```

Note que para borrar las líneas utilizando FOR en lugar de WHILE debe hacerlo comenzando por la última línea. De esta forma, el número de las líneas que restan analizar no se modifica cada vez que se borra una línea del memo.

11. La Facultad de Informática lleva un registro de los finales rendidos por cada alumno organizado de la siguiente manera:

ALUMNOS

Campo	Tipo del Campo
Nro_Alum	Numero
Ap_y_nom	Texto
Nacionalidad	Texto

MATERIAS

Campo	Tipo del Campo
Cod_mat	Numero
Nombre	Texto
Anio	Numero

NOTAS

Campo	Tipo del Campo
Nro_alum	Numero
Cod_mat	Numero
Nota_Final	Numero
Fecha_Examen	Fecha

Indique la manera de obtener:

a) Las calificaciones que obtuvieron los alumnos que rindieron cierta materia durante el mes de julio de 2012. El nombre de la materia se indica como parámetro y se ingresa a través de un Edit. Para cada alumno se debe visualizar nombre y apellido y calificación obtenida. Note que la tabla calificaciones tiene registrado el día de cada examen como día/mes/año y NO únicamente el mes. La información debe aparecer ordenada por nota en forma decreciente y los alumnos que hayan obtenido una misma calificación deberán aparecer en orden alfabético. Visualice la información en un Memo

b) Nombre y apellido y cantidad de finales aprobados (aprobado = nota≥4) de aquellos alumnos que hayan aprobado más de 6 materias.

c) Para cada materia, el nombre de la materia, calificación mínima, calificación máxima y calificación promedio ordenado en forma creciente por nota promedio.

Rta:

-- a) Las calificaciones que obtuvieron los alumnos que rindieron cierta materia durante el mes de julio de 2012.

Propiedad SQL del componente ADOQuery7:

```
select A.ap_y_nom, N.nota_final
from Alumnos A, Notas N, Materias M
where (A.nro_alum = N.nro_alum) and (N.cod_mat = M.cod_mat)
    and (M.nombre = :nombreMateria) and (N.fecha_examen >= #7/1/2012#)
    and (N.fecha_examen <= #7/31/2012#)
order by N.nota_final desc, A.ap_y_nom
```

```
procedure TForm1.Button5Click(Sender:TObject);
var
   nombreMateria, ap_y_nom, linea:String;
   nota_final: integer;
begin
   ADOQuery7.close;
   nombreMateria := edit2.text;
   ADOQuery7.parameters.paramByName('nombreMateria').value := nombreMateria;
   ADOQuery7.open;
   memo2.lines.clear;
   while not ADOQuery7.EOF do begin
      ap_y_nom := ADOQuery7.fieldByName('ap_y_nom').asString;
      nota_final := ADOQuery7.fieldByName('nota_final').asInteger;
      linea := IntToStr(nota_final) + ' ' + ap_y_nom;
      memo2.lines.add(linea);
      ADOQuery7.next;
   end;
```

```
end;
```

```
-- b) Nombre y apellido y cantidad de finales aprobados (aprobado = nota>=4) de aquellos alumnos que hayan aprobado más de 6 materias.

select A.ap_y_nom, count(*) as cantidad_finales
from Alumnos A, Notas N
where (A.nro_alum = N.nro_alum) and (N.nota_final >= 4)
group by A.ap_y_nom
having (count(*) > 6)

-- c) Para cada materia, el nombre de la materia, calificación mínima, calificación máxima y calificación promedio ordenado en forma creciente por nota promedio.
select M.nombre, min(N.nota_final) as calificacion_minima, max(N.nota_final) as calificacion_maxima, avg(N.nota_final) as calificacion_promedio
from Materias M, Notas N
where (M.cod_mat = N.cod_mat)
group by M.nombre
order by 4
```

12. La tabla ALUMNOS presenta algunos problemas en el campo NACIONALIDAD. Los valores correctos para este campo son: NATIVO, NATURALIZADO o EXTRANJERO. Por error, en donde debería decir NATIVO aparece la palabra ARGENTINO. Escriba el segmento de código que modifique el campo NACIONALIDAD de la tabla ALUMNOS reemplazando el término ARGENTINO por NATIVO. Por simplicidad, la palabra ARGENTINO siempre aparece en mayúsculas.

Rta:

```
procedure TForm1.Button6Click(Sender: TObject);
var
        nacionalidad: String;
begin
        //TablaAlumnos es el componente TADOTable que referencia a la tabla Alumnos de la BBDD.
        //Es necesario usar Open y First porque no sabemos si la tabla está abierta, y en caso de que
        //esté abierta no sabemos a qué registro está apuntando el puntero de la tabla.
        TablaAlumnos.open;
        TablaAlumnos.first;
        while not TablaAlumnos.EOF do begin
                nacionalidad := TablaAlumnos.fieldByName('nacionalidad').asString;
                if nacionalidad = 'ARGENTINO' then begin
                        TablaAlumnos.edit;
                        TablaAlumnos.fieldByName('nacionalidad').value := 'NATIVO';
                        TablaAlumnos.post;
                end;
                TablaAlumnos.next;
        end;
end;
```

<u>IMPORTANTE</u>: Notar que para modificar la tabla es necesario ponerla en modo de edición mediante el método Edit y para guardar los cambios se debe usar el método Post. El método Post <u>NO</u> mueve el puntero de la tabla, por lo que es necesario usar el método Next para avanzar al siguiente registro.

13. Se desea generar una impresión del contenido de la tabla MATERIAS agrupando las asignaturas por ANIO y dentro del mismo año ordenadas alfabéticamente. Cuáles de las siguientes afirmaciones son verdaderas:

    A- Es necesario utilizar una componente TADOQuery para generar el reporte.

    B- Los datos del reporte se obtienen directamente de la tabla.

    C- Para hacer el reporte es necesario utilizar una componente TADOQuery que en su propiedad SQL agrupe los datos por el campo ANIO.

    D- Todas las bandas que intervienen en el reporte son TQRBand.

    E- Los registros deben estar ordenados por año.

    F- El componente TQuickRep toma la información a través de su propiedad DataSet.

Rta:

A – Falso. Todos los datos involucrados están en la tabla Materias, alcanza con ordenar la tabla por año y luego por nombre.

B – Verdadero. Ver respuesta A.

C – Falso. Si se agrupara por ANIO en una consulta no podrían obtenerse los nombres de las materias.

D – Falso. Se requiere además el uso de un componente TQRGroup para agrupar en el reporte los registros de la tabla mediante la propiedad Expression.

E – Verdadero. Ordenar por año permite agrupar en el reporte las materias del mismo año, para aplicar luego el segundo criterio de ordenación sobre las mismas. Si la tabla está desordenada el reporte no se visualizará correctamente.

F – Verdadero. En este caso la propiedad DataSet del TQuickRep debería apuntar al componente TADOTable que referencia a la tabla Materias.

14. En base a la información registrada en la tabla **PELICULAS** se ha generado el reporte que se muestra en el margen derecho.
    La tabla contiene la siguiente información

Peliculas	
IdPelicula	Longint(+) Pk
Nombre	Text(50)
Comentario	Memo
Imagen	Ole
Actores	Text(255)
Genero	Text(25)

01/07/2007

Listado de Películas

Nombre

A
ALERTA SOLAR
ARTHUR Y LOS MINIMOYS

C
COLORIN COLORADO,

E
EL AMOR Y LA CIUDAD

L
LA FAMILIA DEL FUTURO
LA VIDA DE LOS OTROS
LAS TORTUGAS NINJA

N
NUMERO 23

P
PARANOIA
PAREJAS
PRUEBA DE FE

    a. Analice cuidadosamente el reporte e identifique la mayor cantidad de componentes que le sea posible.
    b. Especifique las propiedades mínimas que deben configurarse en cada banda para obtener el reporte de la Figura 2.
    c. Indique la forma de visualizar las letras iniciales A, C, E, etc.
    d. Indique las componentes necesarias para obtener la información de la tabla y accederla desde el reporte. Detalle, para cada una, las propiedades indispensables para realizar esta tarea.
    e. ¿La tabla debe estar ordenada? En caso afirmativo indique cómo se especifica el índice a utilizar.
    f. ¿Dónde se ingresa el título de reporte?

Rta:

a – El reporte (componente TQuickRep) contiene 3 TQRBands, un TQRGroup, 2 TQRSysData, 2 TQRLabel y un TQRDBText.

b - Para visualizar el reporte se necesita:

- un TQRBand para mostrar el título del reporte (propiedad BandType = rbTitle), que contenga un componente TQRSysData para obtener la fecha (propiedad Data = qrsDate) y otro TQRSysData para obtener el título del reporte (propiedad Data = qrsTitle).

- un TQRBand de fondo gris que sirva como encabezado de columna (propiedad BandType = rbColumnHeader), y que contenga un TQRLabel con el texto 'Nombre'.

- un TQRGroup para agrupar en el reporte las películas que empiezan con la misma letra, que contenga un componente TQRLabel.

- un TQRBand que sirva como detalle (propiedad BandType = rbDetail), y que contenga un TQRDBText para mostrar el nombre de la película.

c - Para visualizar las iniciales se debe configurar el TQRGroup para especificar que se agruparán en el reporte aquellas películas que empiecen con la misma letra. Para ello se debe colocar la expresión copy(nombre, 1, 1) en la propiedad Expression del TQRGroup. El TQRGroup se imprime en el reporte cada vez que la expresión contenida en su propiedad Expression cambia su valor al ser evaluada sobre el registro actual de la tabla. Por ello, en el evento OnBeforePrint el cual se ejecuta antes que de que el TQRGroup se imprima, se debe escribir el código que actualice el contenido del TQRLabel que está dentro del TQRGroup colocando la letra inicial.

d - El TQuickRep debe referenciar a la tabla Peliculas en su propiedad DataSet.

El TQRDBText debe referenciar a la tabla Peliculas en su propiedad DataSet, y al campo 'nombre' en su propiedad DataField.

e – Es necesario que la tabla esté ordenada para que el agrupamiento por letra inicial funcione correctamente. Esto se logra indicando en la propiedad IndexFieldNames del componente TADOTable cuáles con los campos por los que se quiere ordenar la tabla. En este caso podemos colocar en dicha propiedad el valor 'nombre'.

f – El título debe estar escrito en la propiedad ReportTitle del componente TQuickRep.