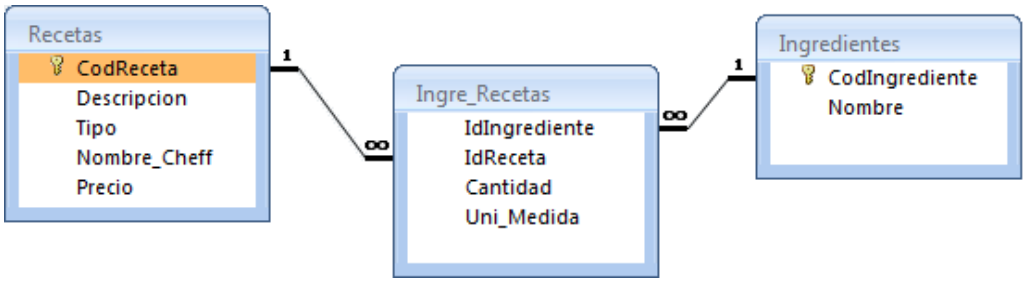
Seminario de Lenguajes - Opción Delphi. - 19/06/2013

1) Se dispone de información de recetas de cocina organizada en tres tablas de la siguiente forma:



El detalle de los campos es el siguiente

RECETAS

Nombre del campo	Tipo de datos
CodReceta	Autonumérico
Descripcion	Texto
Tipo	Texto
Nombre_Cheff	Texto
Precio	Número

INGRE_RECETAS

Nombre del campo	Tipo de datos
IdIngrediente	Número
IdReceta	Número
Cantidad	Número
Uni_Medida	Texto

INGREDIENTES

Nombre del campo	Tipo de datos
CodIngrediente	Autonumérico
Nombre	Texto

Indique la manera de obtener

a) La descripción, tipo y nombre del cheff de todas las recetas ordenadas por el nombre del cheff en forma descendente y por tipo en forma ascendente. Muestre el resultado en una grilla.

Rta:

ADOConnection: Para acceder a la base de datos debe utilizarse un componente ADOConnection. En él se indica cuál es la base de datos con la que nos queremos conectar (entre otras cosas). Como no utilizamos login seteamos la propiedad LoginPromt en FALSE. Finalmente, para establecer la conexión seteamos la propiedad connected en TRUE. En este punto, ya estamos conectados con la base.

ADOQUERY

Para hacer una consulta y obtener un *dataset* como resultado, utilizamos un componente ADOQuery cuya propiedad *name* tendrá el valor ADOQUERY1. La consulta se escribe en la propiedad SQL (es un TSTring) de la siguiente forma:

```
//Propiedad SQL del componente ADOQuery1:
select R.descripcion, R.tipo, R.nombre_cheff
from Recetas R
order by R.nombre_cheff desc, R.tipo
```

DBGrid: Es la component donde se visualizará el conjunto de registros obtenidos como resultado de la consulta. Se relaciona con un dataset a través de un componente DataSource. Es decir que el componente **DataSource** es el nexo entre los datos que se desean mostrar (el dataset resultado de la consulta) y la

componente que los va a visualizar (el DBGrid). Las propiedades que establecen estas relaciones pueden hacerse desde el inspector de objetos o desde el código de la siguiente forma:

```
DataSource1.dataSet := ADOQuery1;
DBGrid1.dataSource := DataSource1;
```

Luego, el Onclick del botón será el siguiente

```
//OnClick del botón:
procedure TForm1.Button1Click(Sender: TObject);
begin
   // cierra y abre la consulta para que se actualice la información
   // recuerde que la consulta es como una "foto" obtenida al momento
   // de abrirla. De esta forma nos aseguramos de mostrar el resultado
   // actualizado
   ADOQuery1.close;
   ADOQuery1.open;
   // Las siguientes dos instrucciones pueden hacerse a través del
   // Inspector de Objetos
   DataSource1.dataSet := ADOQuery1;
   DBGrid1.dataSource := DataSource1;
end;
```

b) El código y la descripción de la receta que tiene el mayor número de ingredientes distintos. Muestre el resultado en dos componentes Edits.

```
Rta:
//Propiedad SQL del componente ADOQuery2:
select R.codReceta, R.descripcion, count(*) as cantidad_ingredientes
from Recetas R, Ingre_Recetas IR
where R.codReceta = IR.idReceta
group by R.codReceta, R.descripcion
order by 3 desc

//OnClick del botón:
procedure TForm1.Button2Click(Sender: TObject);
var
   codReceta:integer;
   descripcion:String;
begin
   ADOQuery2.close;
   ADOQuery2.open;
   codReceta := ADOQuery2.fieldByName('codReceta').asInteger;
   descripcion := ADOQuery2.fieldByName('descripcion').asString;
   editCodReceta.text := IntToStr(codReceta);
   editDescripcion.text := descripcion;
end;
```

c) Los nombres de los ingredientes de una receta cuya descripción se indica como parámetro a través de un edit. Muestre el resultado en un ListBox.

```
Rta:
//Propiedad SQL del componente ADOQuery3:
select I.nombre
from Recetas R, Ingre_Recetas IR, Ingredientes I
where (R.codReceta = IR.idReceta) and (I.codIngrediente = IR.idIngrediente)
      and (R.descripcion = :descripcion_receta)

//OnClick del botón:
procedure TForm1.Button3Click(Sender: TObject);
var
   nombre_ingrediente:String;
begin
   ADOQuery3.close;
   ADOQuery3.parameters.paramByName('descripcion_receta').value := edit1.text;
   ADOQuery3.open;
   listbox1.items.clear;
   while not ADOQuery3.EOF do begin
         nombre_ingrediente := ADOQuery3.fieldByName('nombre').asString;
         listbox1.items.add(nombre_ingrediente);
         ADOQuery3.next;
   end;
end;
```

2) Debido a un incremento de precios generalizado deben incrementarse los valores del campo precio de la tabla Recetas en un 10%.

```
Rta:
procedure TForm1.Button4Click(Sender: TObject);
var
   precio:real;
begin
   //TablaRecetas es el componente TADOTable que referencia a la tabla Recetas.
   TablaRecetas.open;
   TablaRecetas.first;
   while not TablaRecetas.EOF do begin
         precio := TablaRecetas.fieldByName('precio').asFloat;
         TablaRecetas.edit;
         TablaRecetas.fieldByName('precio').value := precio * 1.1;
         TablaRecetas.post;
         TablaRecetas.next;
   end;
end;
```

3) Se desea obtener el reporte que aparece a la derecha

   a) Indique el valor de la propiedad SQL de la componente ADOQuery que se asociará al reporte a través de la propiedad DataSet.

   b) Indique cuantas bandas se necesitan para hacer el reporte y si son del mismo tipo. Explique.

   c) ¿Aparecen en el reporte las recetas que no tienen ingredientes cargados?

¿Qué necesito para hacer … ?

Arroz con Leche
1 litro Leche entera
200 grs. Arroz común
200 grs. Azúcar
50 grs. Cáscara de limón
1 cdita. Esencia de Vainilla

Bizcochuelo clásico de Vainilla
250 grs. Azúcar
300 grs. Harina
8 Huevos
1 cdita. Esencia de Vainilla

Budín de Pan
... (lista de ingredientes)

Flan Casero
...(lista de ingredientes)

Rta:

a) Indique el valor de la propiedad SQL de la componente ADOQuery que se asociará al reporte a través de la propiedad DataSet.

```
select R.descripcion, IR.cantidad, IR.uni_medida, I.nombre
from Recetas R, Ingre_recetas IR, Ingredientes I
where (R.codReceta = IR.idReceta) and (I.codIngrediente = IR.idIngrediente)
order by R.descripcion
```

b) Indique cuantas bandas se necesitan para hacer el reporte y si son del mismo tipo. Explique.

Se necesitan 3 bandas, y no son todas del mismo tipo.
Se necesita una banda para mostrar el título del reporte, otra para agrupar los ingredientes de cada receta y escribir el nombre de la receta, y una tercera para mostrar la cantidad, la unidad de medida y el nombre de cada ingrediente.

Por lo tanto, se necesitan 2 TQRBand y un TQRGroup.

c) ¿Aparecen en el reporte las recetas que no tienen ingredientes cargados?

No, las recetas que no tienen ingredientes cargados no aparecen en el reporte porque no están presentes en el resultado de la consulta que une las 3 tablas, ya que no hay ninguna fila de la tabla Ingre_receta que les corresponda.

4) Dado el siguiente código Delphi

```
var a,b : array of integer;                          (1)
begin
     writeln(sizeof(b));                             (2)
     setLength(a,2);
     b := a;
     a[low(a)] := 24;
     b[low(b)] := 30;
     b[high(b)] := 12;
     writeln('a[0]=', a[0], '  b[0] =', b[0]);      (3)
     if (a=b) then writeln('Son Iguales');           (4)
     setLength( b, 2 );
     writeln('a[1]=', a[1], '  b[1] =', b[1]);      (5)
end;
```

Indique el valor de verdad de las siguientes afirmaciones:

a) La declaración (1) está incompleta porque no puede determinarse si es un vector o una matriz.
b) La instrucción (2) imprime el valor 0.
c) La instrucción de (3) muestra los valores 24 y 30.
d) El mensaje de (4) NO aparece en pantalla.
e) La instrucción de (5) da error porque a[1]no tiene valor asignado.
f) No puede decirse a priori lo que imprime la instrucción de (5) porque a[1]no tiene valor asignado.

Rta:

a) Falso, la declaración es correcta. Para declarar un arreglo dinámico no se necesita especificar los índices, y la declaración se interpreta como un arreglo de una dimensión. Si se quiere declarar una matriz dinámica de enteros se debe escribir *array of array of integer*.

b) Falso. La función *SizeOf* retorna el tamaño en bytes de la variable, no la cantidad de elementos que contiene. Como un arreglo dinámico se implementa internamente con un puntero, *SizeOf(b)* da el valor 4, que es la cantidad de bytes que ocupa un puntero.

c) Falso. La instrucción *b := a* hace que *b* y *a* apunten a la misma área de memoria, con lo que la instrucción *b[low(b)] := 30* hace que tanto *a[0]* como *b[0]* tengan el valor 30.

d) Falso. El mensaje sí aparece porque *a* y *b* apuntan a la misma área de memoria, es decir, que su valor (son punteros) es el mismo y por lo tanto el valor de verdad de *a=b* es TRUE.

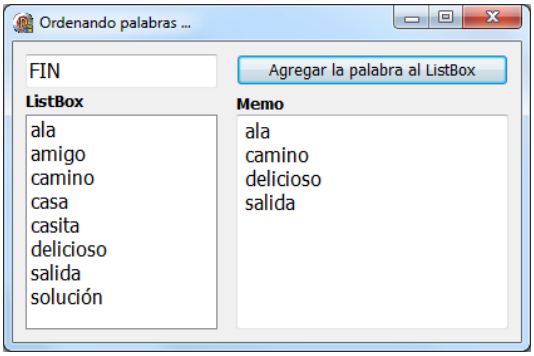
e) Falso. Como *a* y *b* apuntaban a la misma área de memoria, *a[1]* tiene el valor de *b[1]* que fue asignado previamente. Luego del *SetLength*, *b* apunta a una nueva área de memoria (con una copia de los elementos originales) pero *a* sigue apuntando al área original.

f) Falso. Ver respuesta e).

Ejercicio Adicional

Sobre el formulario principal se encuentran pegados: 1 Edit, 1 Botón, 1 ListBox y 1 Memo.

Programe el OnClick del botón para que agregue en forma ordenada en el ListBox, cada palabra que se ingrese en el Edit. Cuando se ingrese la palabra FIN deberá visualizar en el Memo la primera palabra de cada letra.



Rta:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i, lon: Integer;
  inicial : char;
  palabra : String;
begin
    if (edit1.Text<>'FIN') then begin
           lon := ListBox1.Items.Count;
           i:=0;
           while (i<lon) and (edit1.Text>ListBox1.Items.Strings[i])  do
                i:= i+ 1;
           ListBox1.Items.Insert(i, edit1.Text);
           edit1.Text := '';
       end
  else begin
         Memo1.Clear;
         inicial := char(0);
         for I := 0 to ListBox1.items.Count - 1 do
         begin
             palabra := ListBox1.Items.Strings[i];
             if inicial<>palabra[1] then begin
               Memo1.Lines.Add(palabra);
               inicial := palabra[1];
             end;
         end;
      end;
end;
```