FEUP Universidade do Porto
Faculdade de Engenharia

Industrial Informatics

[Informática Industrial]

2022/23 edition

Lazarus DB – basic read access

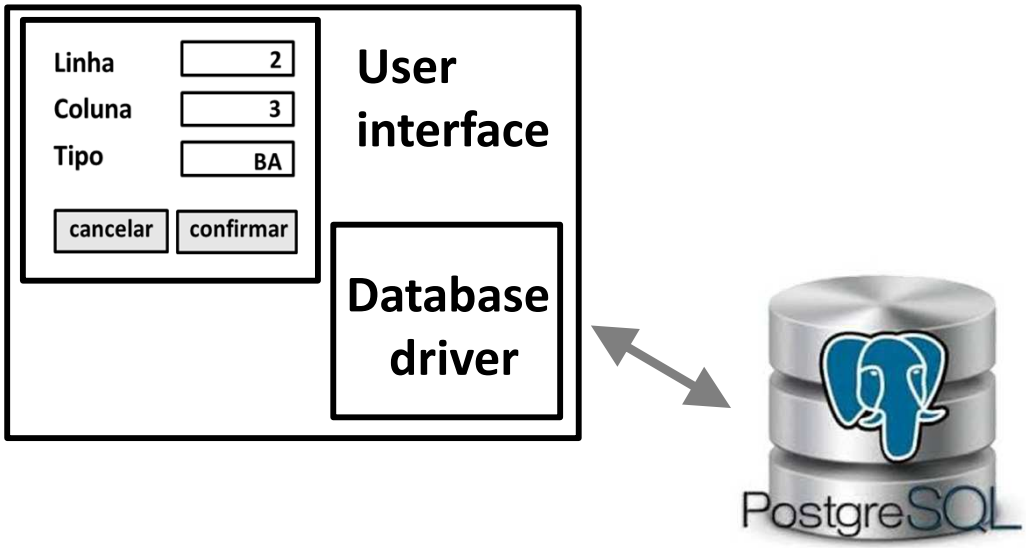
V1.2

José Faria, Andry Pinto

Contents

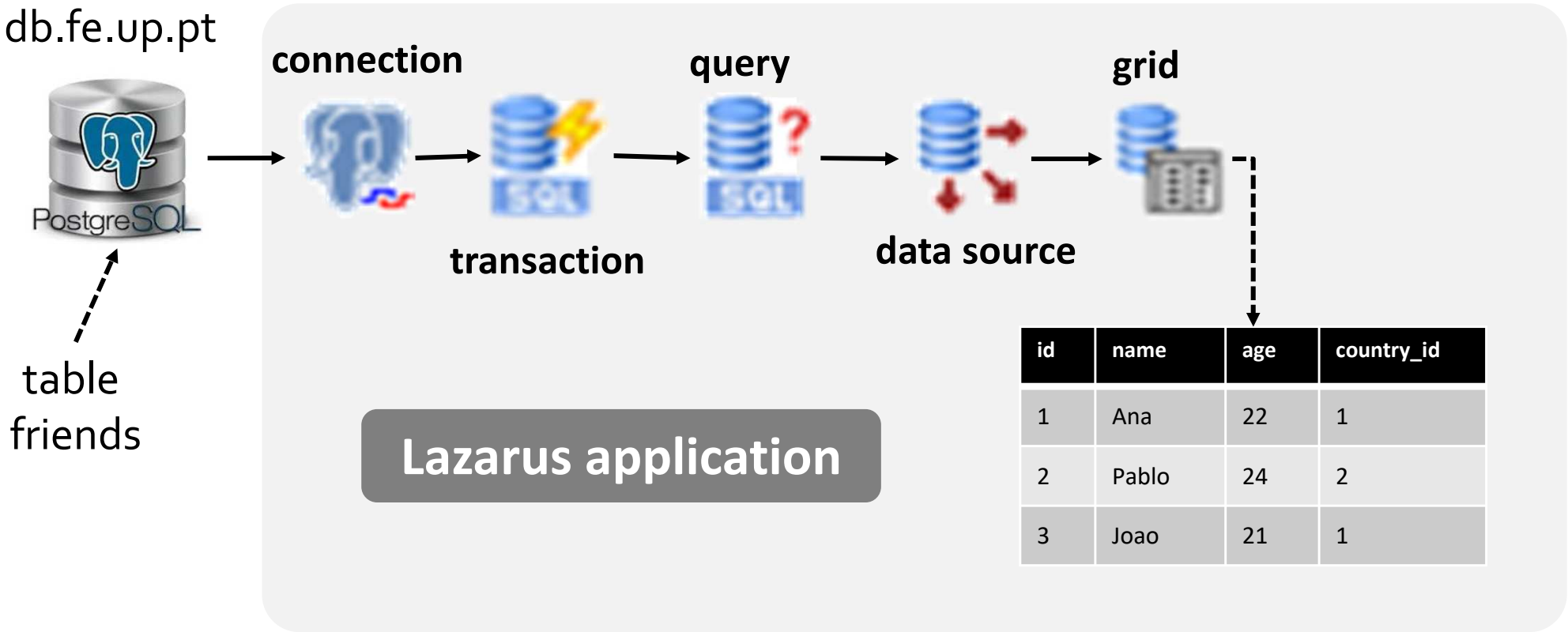
1. Introduction

- To **interact with a PostgreSQL** database server from a Lazarus application,



we **need the 5 components** described in the following slide …

Component	Icon	Menu	Description
TPQConnection		SQLdb	Establishes the connection between the Lazarus application and the DB server
TSQLTransaction		SQLdb	Every query should be executed within a transaction
TSQLQuery		SQLdb	Executes the SQL query and stores the result returned by the DB server
TDataSource		Data Access	Mediates the communication between the data source and the data control in the user interface
TDBGrid		Data Controls	Displays the data in the user interface



let’s do it right now !

2. Basic data access and simple exercises

Intro

- The 1st application we are going to develop implements a **simple access and display** of the records in table ***testinfi.friends***.
- We'll start by a direct display in Lazarus' user interface of the records in the DB.
- Then we'll add some simple functionalities so that you start **becoming familiar and at ease with Lazarus DB** components.

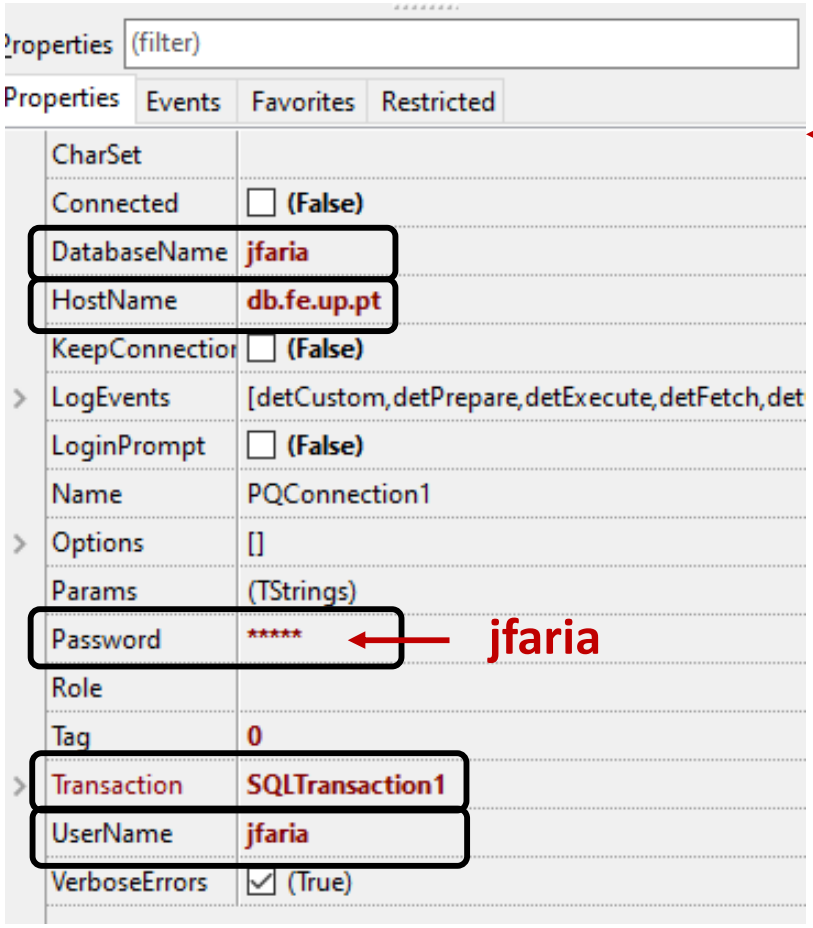
1. Start a new Lazarus application

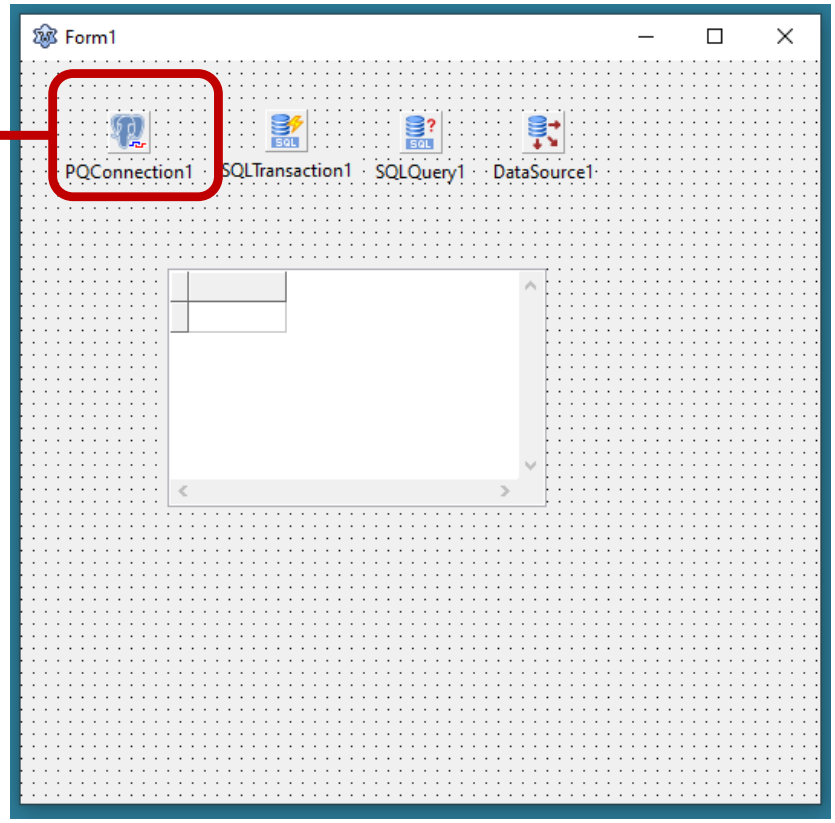
2. Create these 5 objects and

Component	Find it in menu	Icon
TPQConnection	SQLdb	
TSQLTransaction	SQLdb	
TSQLQuery	SQLdb	
TDataSource	Data Access	
TDBGrid	Data Controls	

3. Configure their properties as shown in the following slides …

PQConnection

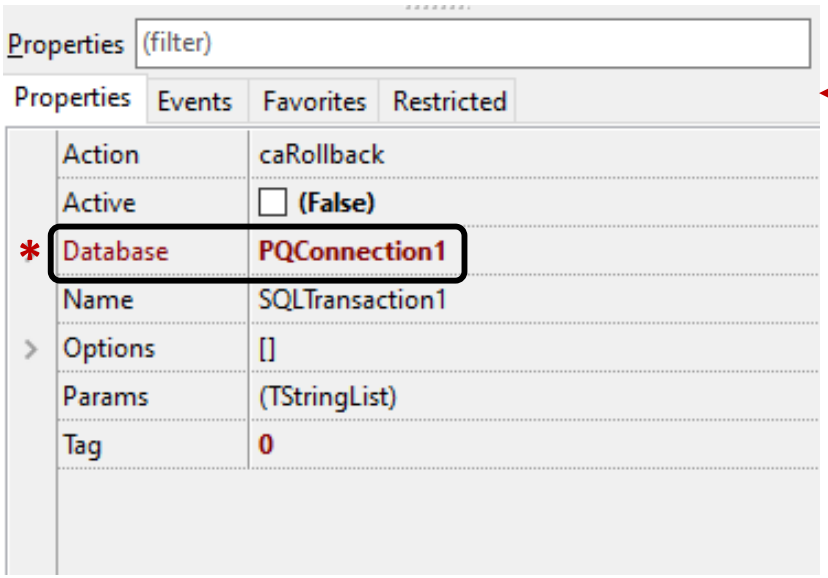


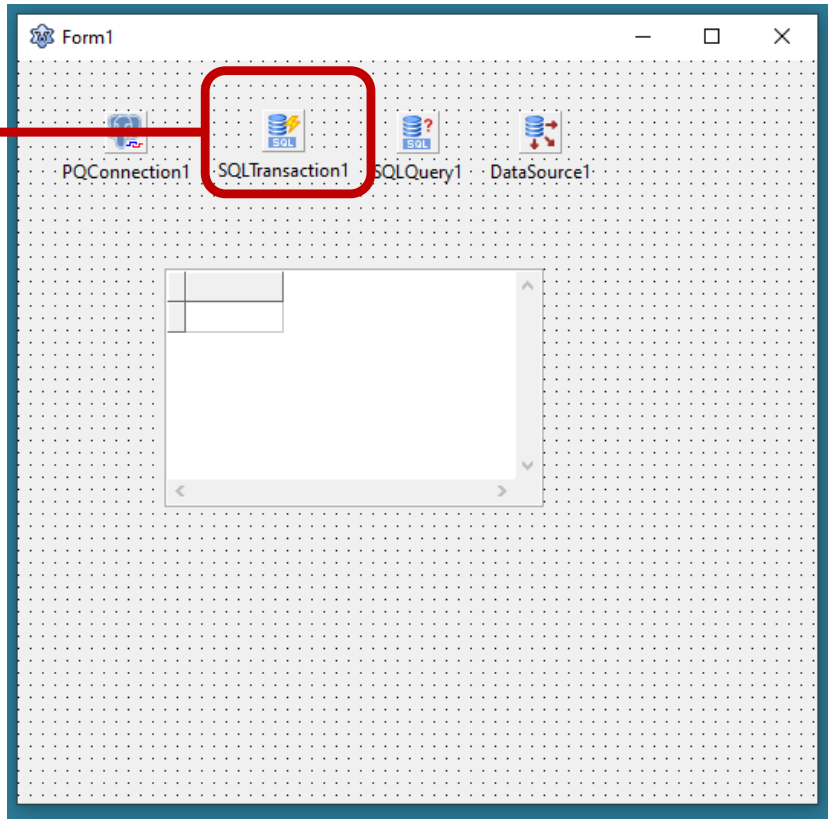


HINT

- You are setting a connection to a database called jfaria and held in server db.fe.up.pt
- The credentials of the owner of jfaria database are: username = jfaria; pass = jfaria

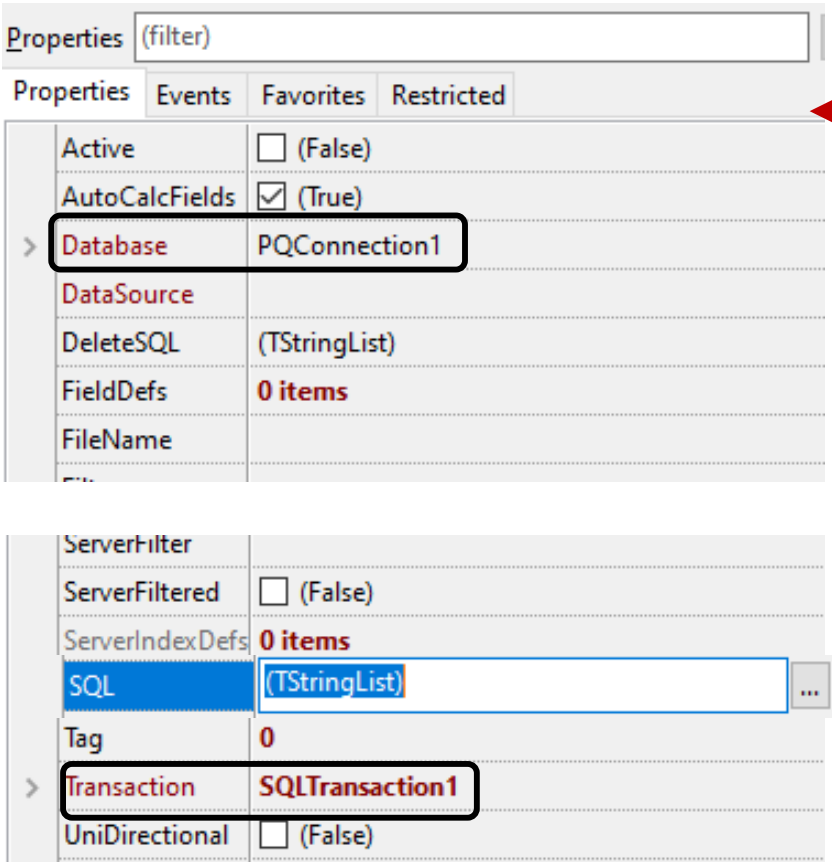
SQLTransaction

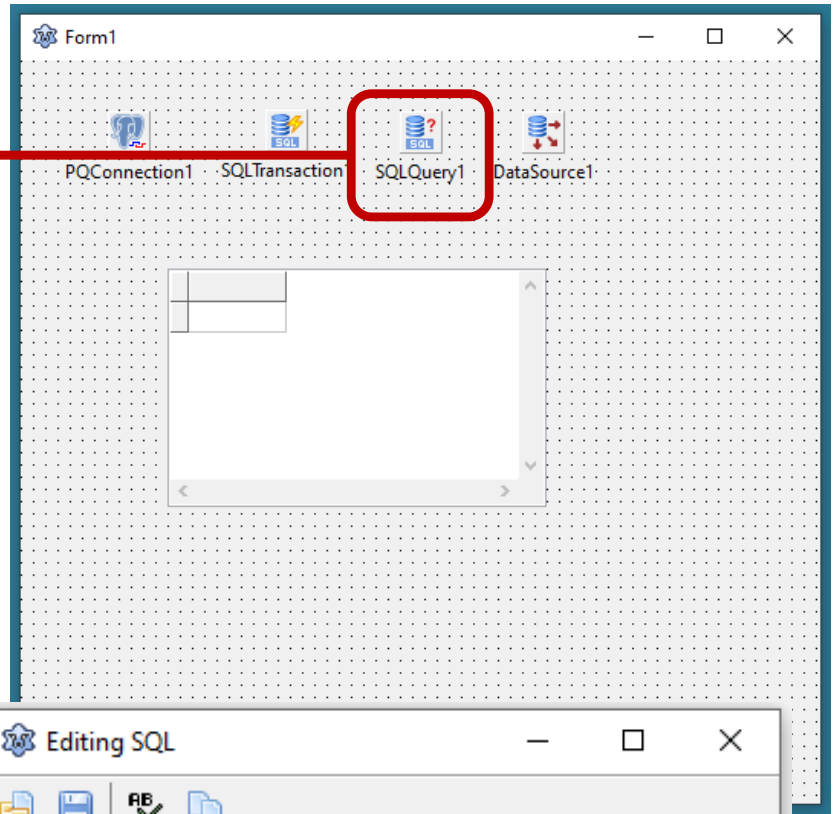


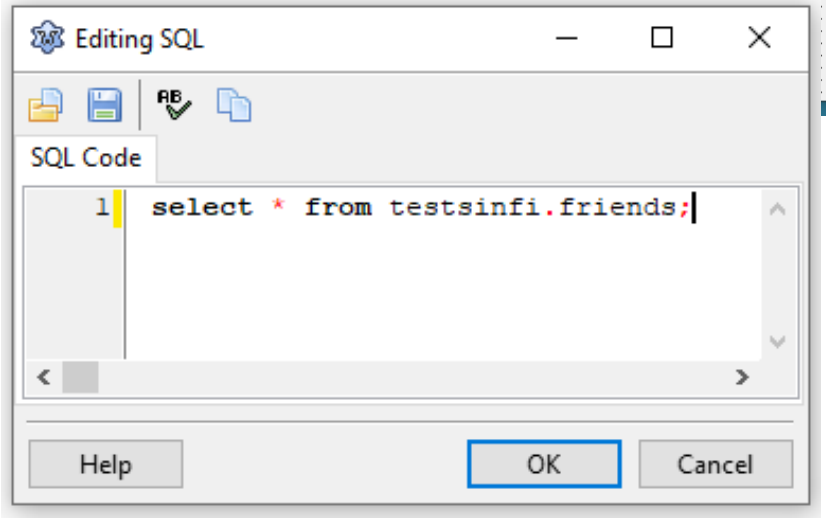


* Just confirm that the Database property is set to SQLTransaction1

SQLQuery



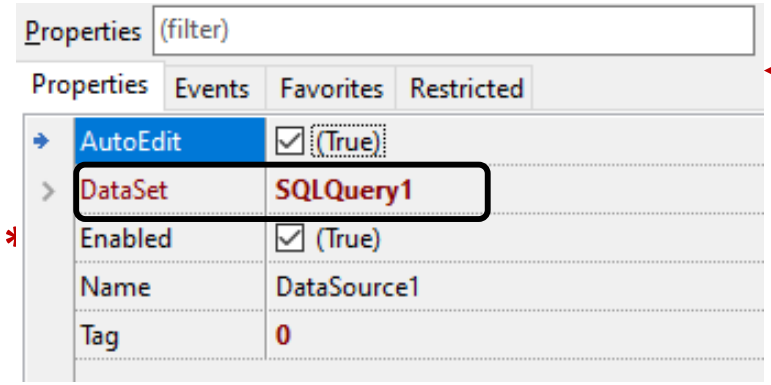


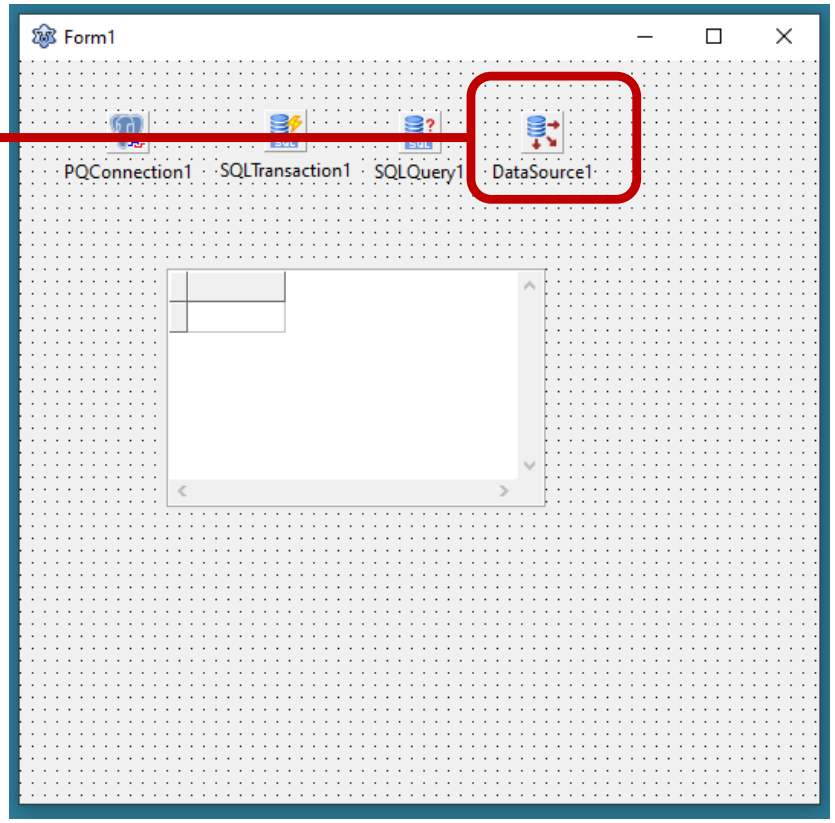


* Enter the SQL query

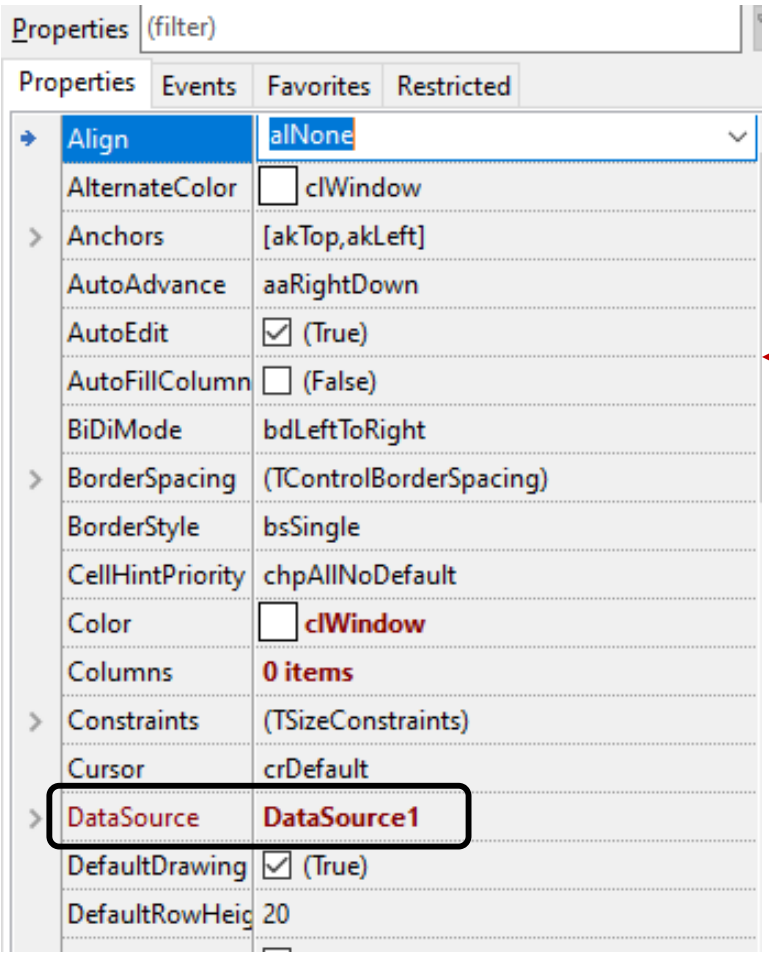
Select * from testsinfi.friends

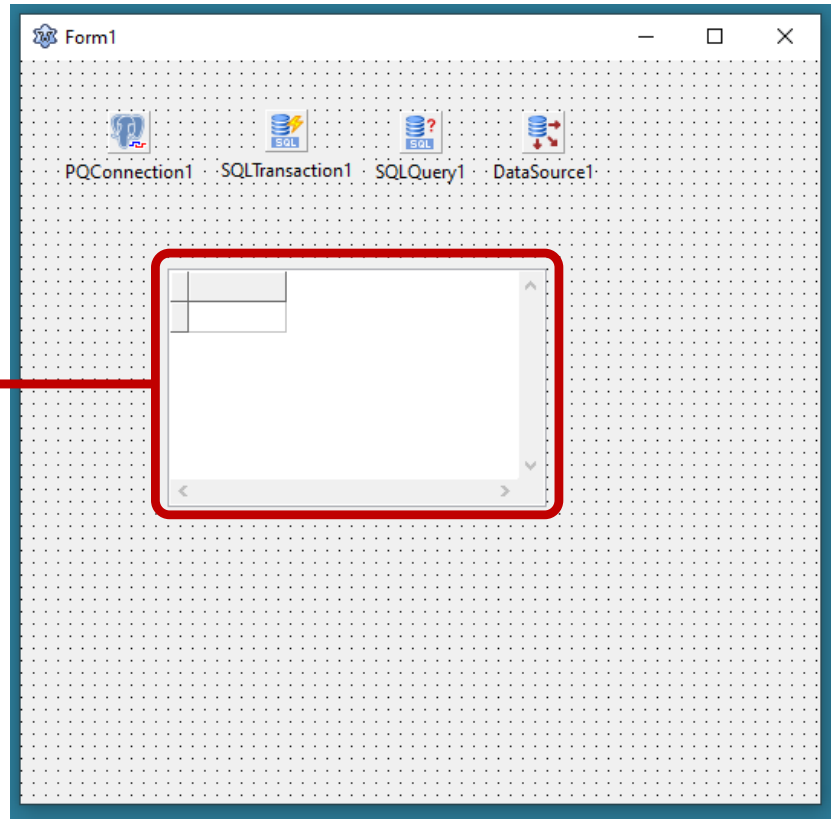
DataSource


Properties
(filter)
Properties
Events
Favorites
Restricted
AutoEdit
(True)
DataSet
SQLQuery1
Enabled
(True)
Name
DataSource1
Tag
0


Form1
PQConnection1
SQLTransaction1
SQLQuery1
DataSource1

DBGrid

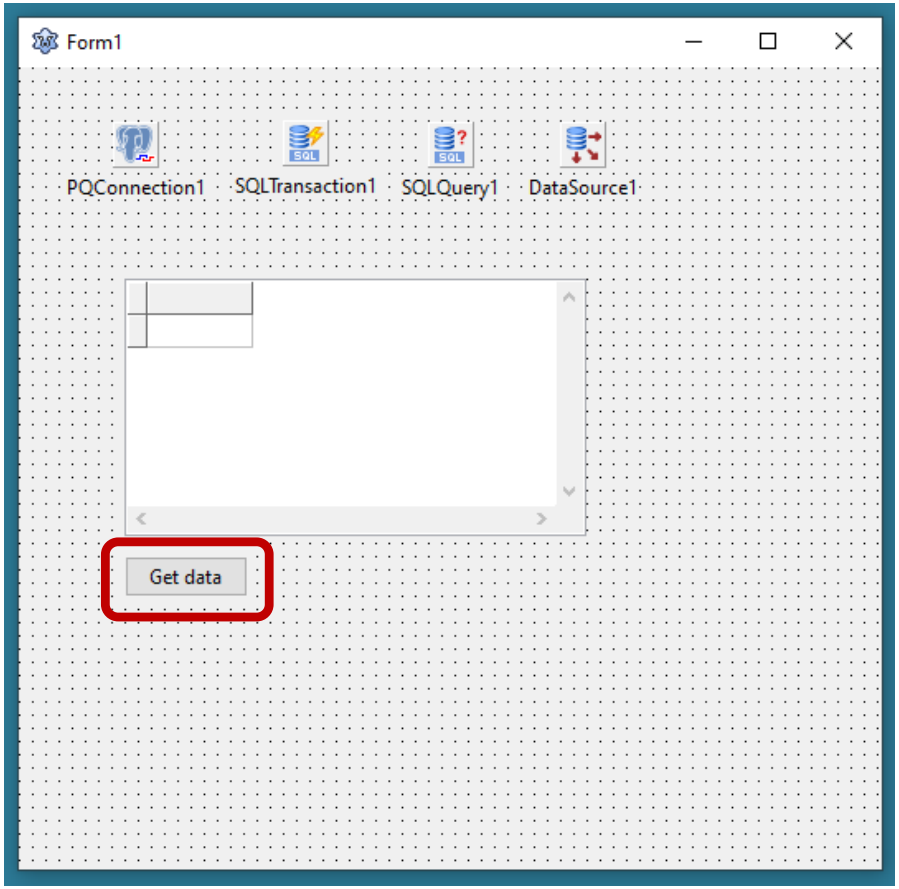

Properties
(filter)
Properties
Events
Favorites
Restricted
Align
alNone
AlternateColor
clWindow
Anchors
[akTop,akLeft]
AutoAdvance
aaRightDown
AutoEdit
(True)
AutoFillColumn
(False)
BiDiMode
bdLeftToRight
BorderSpacing
(TControlBorderSpacing)
BorderStyle
bsSingle
CellHintPriority
chpAllNoDefault
Color
clWindow
Columns
0 items
Constraints
(TSizeConstraints)
Cursor
crDefault
DataSource
DataSource1
DefaultDrawing
(True)
DefaultRowHeig
20


Form1
PQConnection1
SQLTransaction1
SQLQuery1
DataSource1

- Now **add a button** to the form:

  Name: btGetData

  Caption: Get data



- Add the following code to the Click event:

```
procedure TForm1.btGetDataClick(Sender: TObject);
begin
    SQLQuery1.Active := true;
end;
```

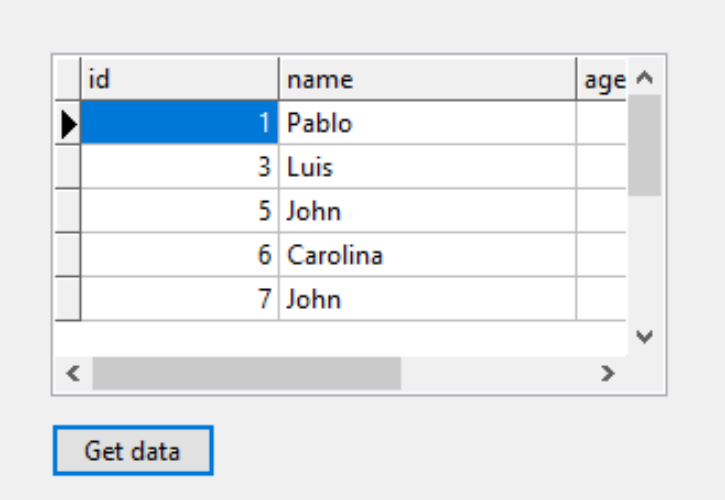
Now, you are **ready to run the application** and display your friends in the Lazarus form:

1. Check the VPN connection if you are outside FEUP

2. Then click run

for

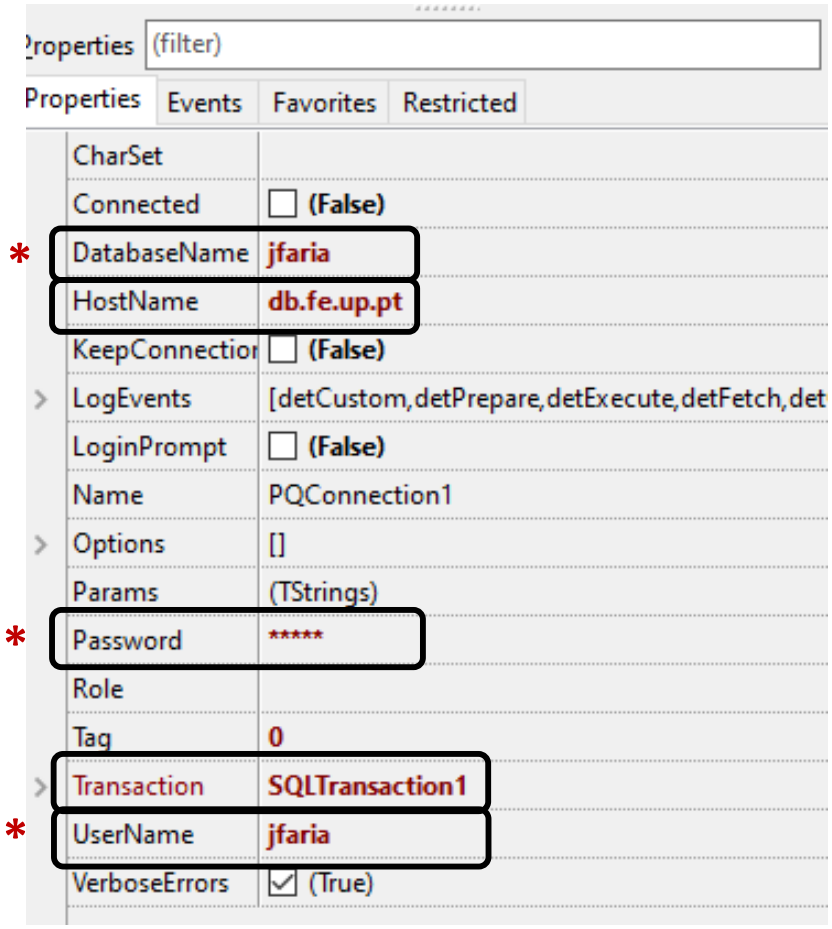
to happen ☺ !



Access your database

- Change as appropriate the properties of the connection object, so that the queries are sent to your database, not mine (jfaria),
- and run the application.

PQConnection



2.1. a *tricky* stuff

a *tricky* stuff

- If you update the data in the DB server using phpPgAdim, the data being displayed in the grid won't change.

- This is because the query in **SQLQuery1 is executed** only in the **activation of the object**: *SQLQuery1.Active := true;*

- To **execute a new select query**, you should add to *Get data click* event:

```
SQLQuery1.Active := false;
SQLQuery1.Active := true;
```

- After this change in the code of the click event, you'll see that:
  - in the first click, the query executes ok, but **generates an error** in the second click ☹ !

- This is due to a **configuration of DB server** db.fe.up.pt as, by default:
  - it **closes the connections that remain inactive** for several milliseconds.

Don't be dismayed!

to **get the solution**, just look at the **following slide ☺ !**

- To *tell* the DB server that it should keep the connection open even it doesn't have activity, add the following code to the ***FormCreate* event**:

```
begin

   PQConnection1.Connected := True;
   PQConnection1.ExecuteDirect('Begin Work;');
   PQConnection1.ExecuteDirect('set idle_in_transaction_session_timeout = 0');
   PQConnection1.ExecuteDirect('Commit Work;');

end;
```

0 ≡ ∞

HINT

You'll understand better these instruction in the next class.

By now, just copy/paste them !

Close de connection when no longer needed

- Any **serious application** must make sure that all open database **connections are properly closed** when not needed anymore.
- In this case, we should close the connection when the form is closed.
- **Add** the following code to the ***FormClose* event**:

```
begin
   PQConnection1.Connected:= False;
end;
```

Automatic grid filling

- Finally, if you want the data grid to be filled automatically on application startup, just **add** the following instruction to the ***FormCreate* event**:

```
SQLQuer1.Active := True;
```

.

- Having arrived here, if **your application works fine** and updates the DBGrid every time you click *Get Data* button, then:

and you are **ready to *meet* the following slide !**

2.2. Simple exercises

Two simple exercises

1. Making use of the property ***Columns[nr].Width*** of the DBGrid's:
   - **hide column *id*** in the grid being displayed
   - **adjust the width of the other columns** to the length of the data they display

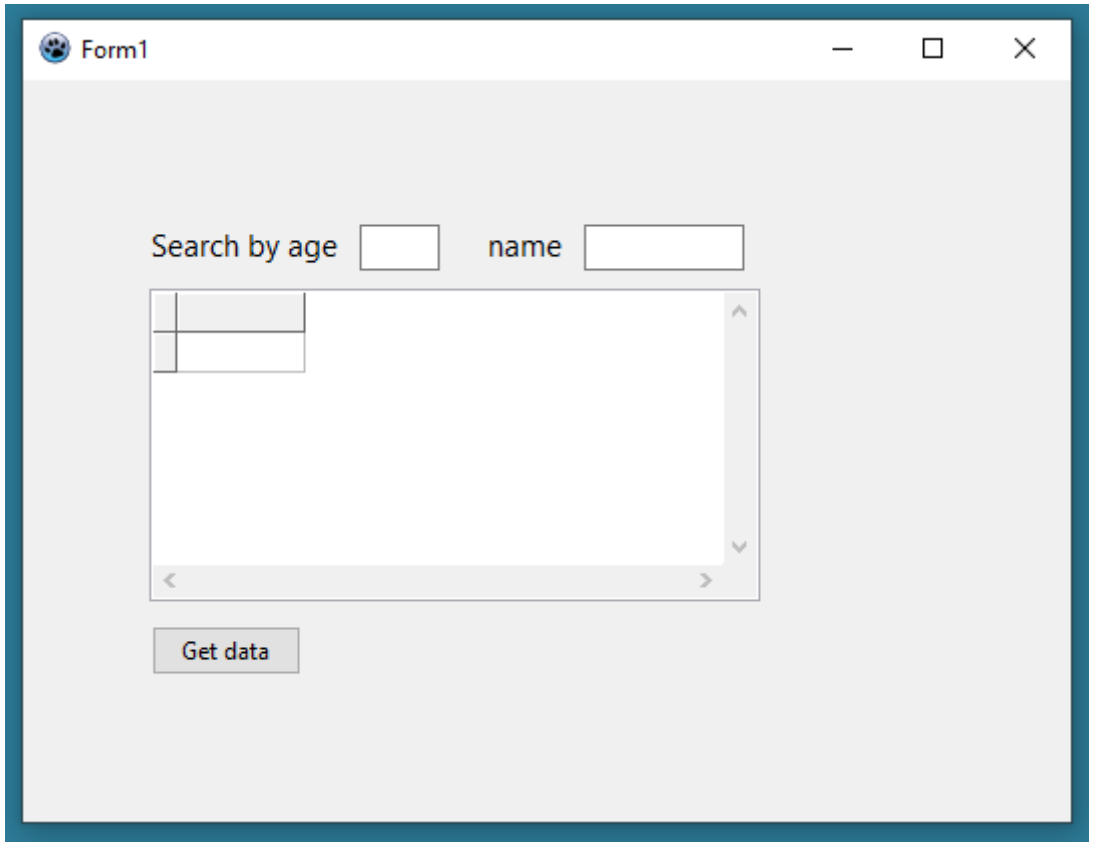
2. Modify the SQL query so that the grid **displays**:
   - your **friend's name and age** and
   - the **name of his/her country**.

3. Dynamic queries and final exercise

Dynamic queries

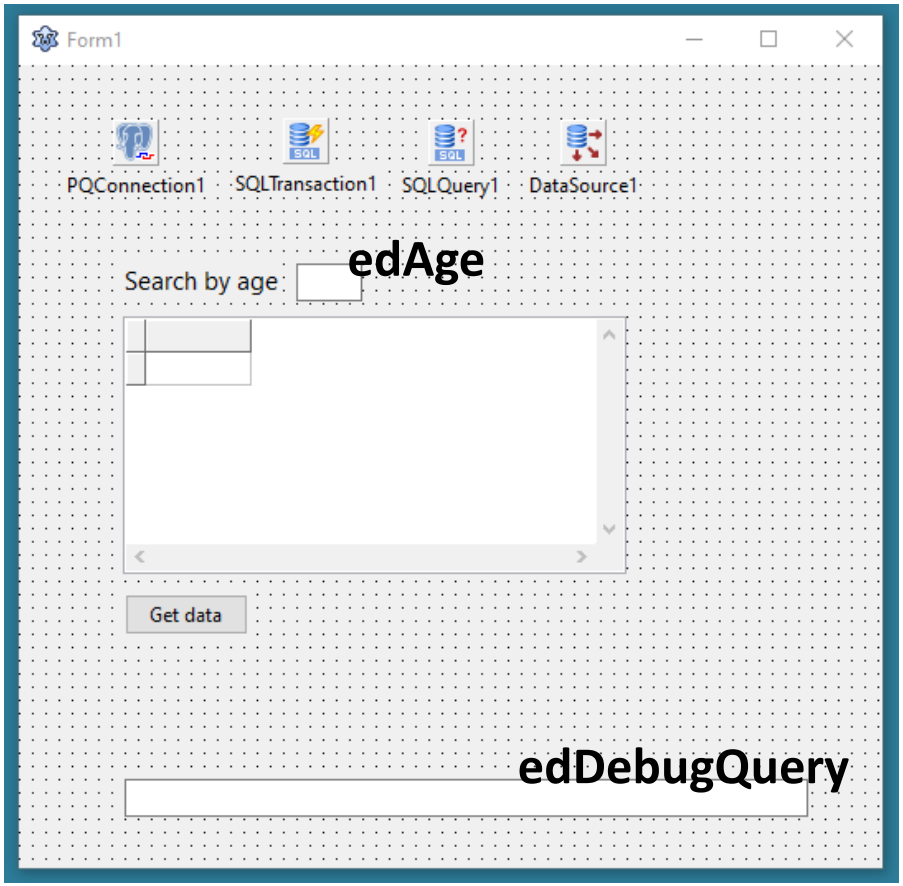
- Now, we are going to address another **very important topic**:

prepare dynamic queries, i.e., queries that **depend on data the user introduces** in the application interface



1. Numerical field

1. Add 3 new controls to the form:
   - A label with caption *Search by age*
   - A textbox named *edAge*
   - A textbox named *edDebugQuery*
2. Interpret and add the following code to *btGetData.Click*:



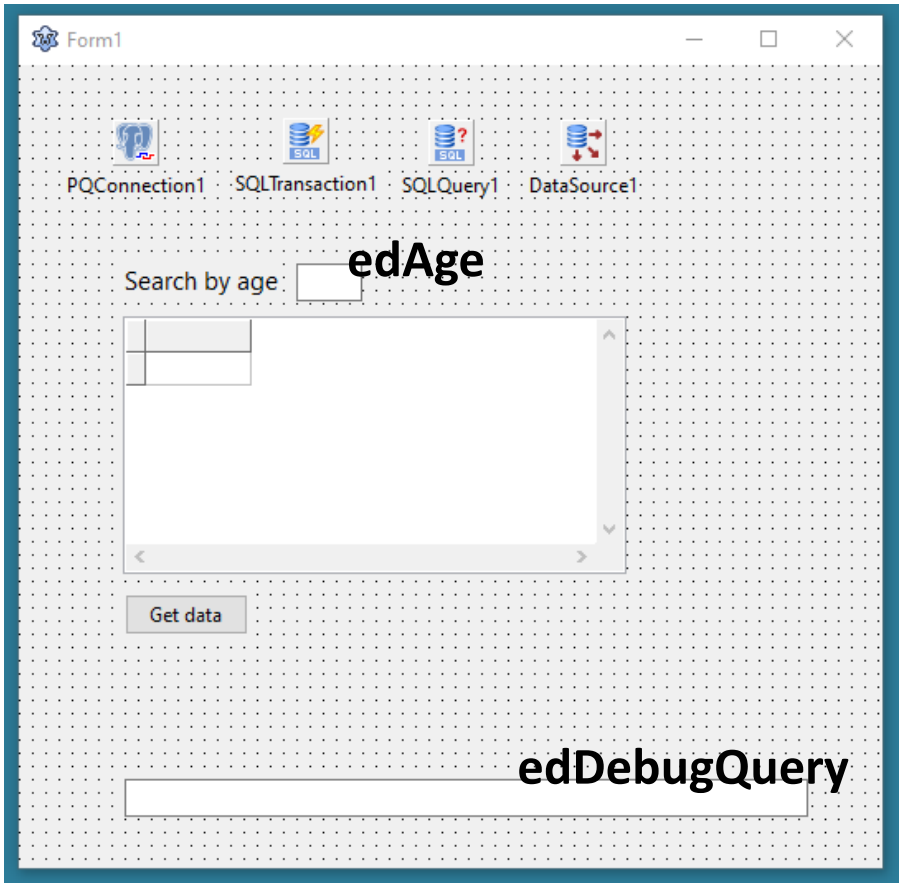

```
procedure TForm1.btGetDataClick(Sender: TObject);
var query : string;

begin

      query := 'select * from testsinfi.friends where age = ' + edAge.Text;
      edDebugQuery.Text   := query;
      SQLQuery1.SQL.Text  := query;
      SQLQuery1.Active    := true;

end;
```

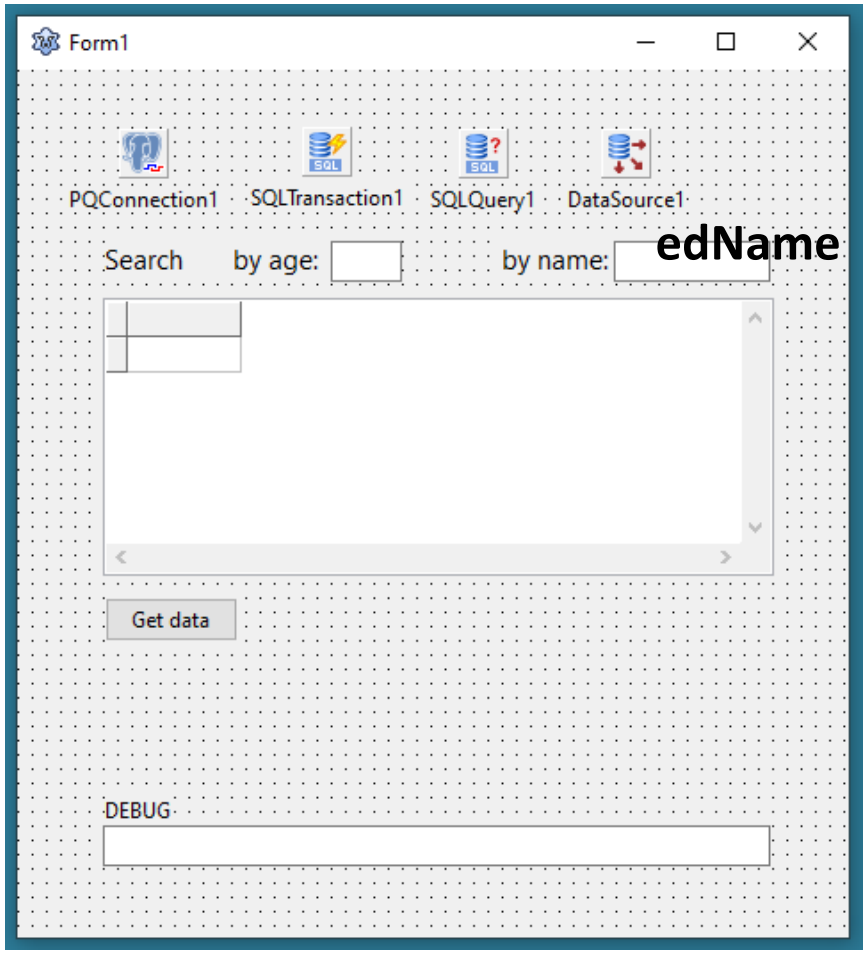
1. Numerical field



3. Run the application
4. Enter an age in *edAge*
5. Click button *Get data*
6. See the dynamic query sent to the DB server in *edDebugQuery*

5. See the result of the query in the DBGrid

2. Text field (string)



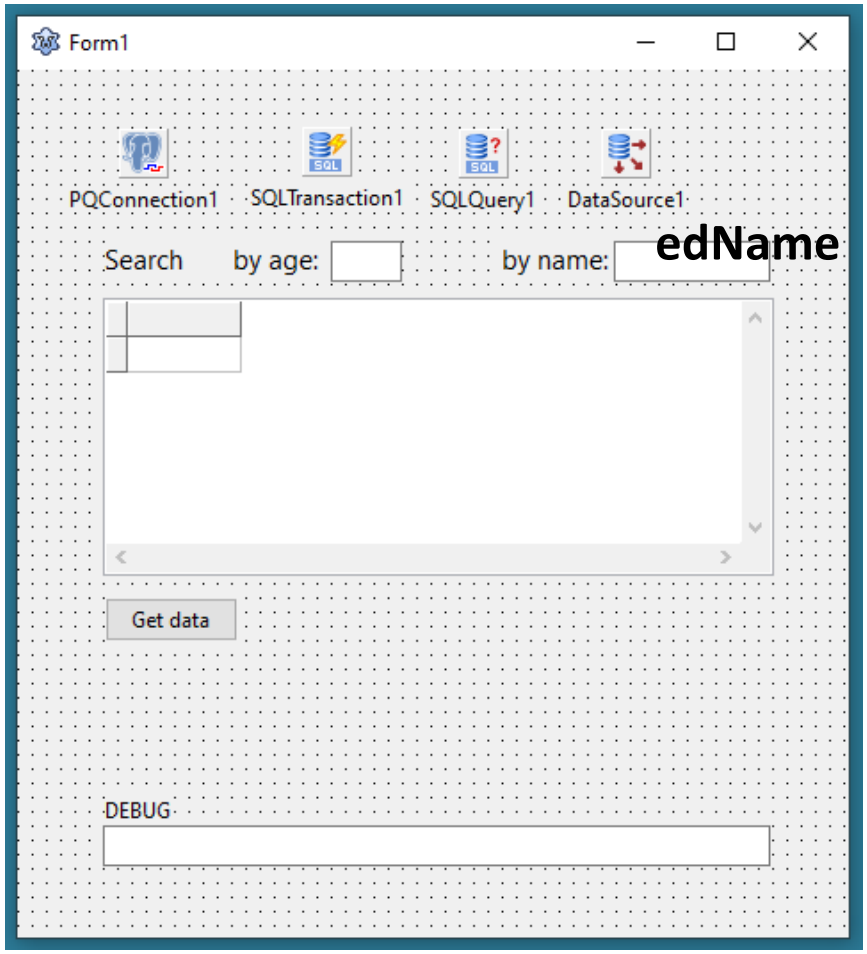
1. Add 2 new controls to the form:
   - A label with caption *by name*
   - A textbox named *edName*

2. In *btGetData.Click*, replace the instruction that creates the dynamic query by:

```
query := 'select * from testsinfi.friends where name = ''' + edName.Text + '''';
```

3. Run the application and see the dynamic query in *edDebugQuery*.

2. Text field (string)



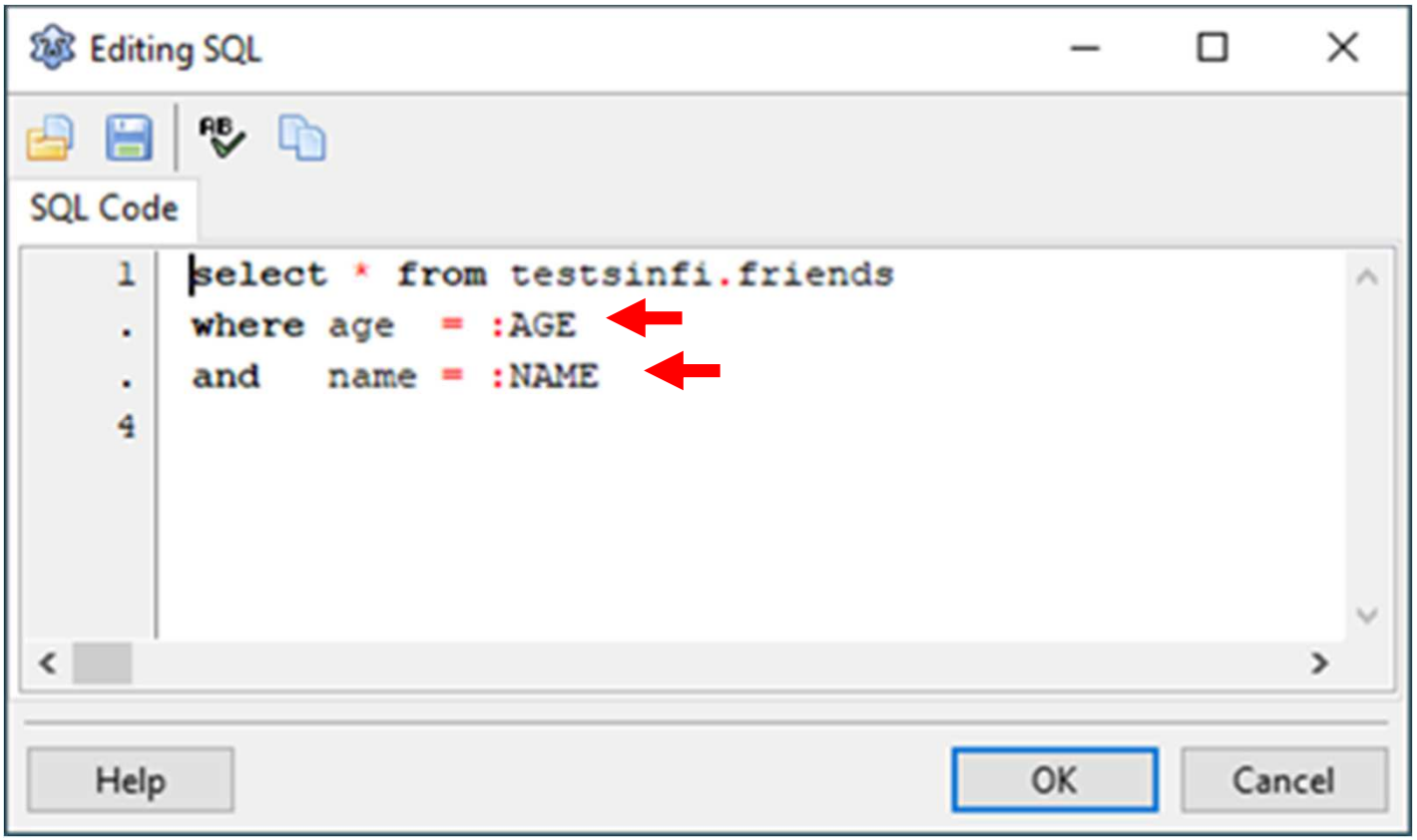
3. Run the application
4. Enter a name in *edName*
5. Click button *Get data*
6. As before, see the dynamic query sent to the DB server in *edDebugQuery and* the result of the query in the DBGrid.

3.1. Queries parameters

Intro

- When the queries contain several dynamic items, the **use of the string concatenation operator** + may turn the queries a **bit confusing** and **prone to errors**.
- In these cases, the use of **query parameters** may be a **more efficient** approach.
- In order to create a dynamic query containing 2 parameters, with the age and name entered by the user, proceed as described in the following slide …

1. Edit the SQL property in SQLQuery1 with two parameters: AGE and NAME



2. Assign to the parameters the values entered in the text boxes *edAge* and *edName*:

```
SQLQuery1.Params.ParamByName('AGE').AsInteger    := StrToInt(edAge.Text);

SQLQuery1.Params.ParamByName('NAME').AsString  := edName.Text;
```

So that the code of *btGetDataClick* becomes:

```
procedure TForm1.btGetDataClick(Sender: TObject);
var query : string;

begin

     SQLQuery1.Params.ParamByName('AGE').AsInteger := StrToInt(edAge.Text);
     SQLQuery1.Params.ParamByName('NAME').AsString := edName.Text;

     SQLQuery1.Active    := false;
     SQLQuery1.Active    := true;

end;
```

- When executing the query (*SQLQuery.Active := true*), Lazarus replaces the parameters by the values entered in the text boxes *edAge* and *edName*.

- As when setting the value of a parameter we also specify its type, Lazarus will know if it should enclose the value within ' (if a string parameter), or not (if a numerical parameter):

```
SQLQuery1.Params.ParamByName('AGE').AsInteger   := StrToInt(edAge.Text);
SQLQuery1.Params.ParamByName('NAME').AsString  := edName.Text;
```

- This way, you don't have to worry no more about doubling '' ☺ !

3.2. Final exercise

Start by taking a breath ☺ !

Then, edit the code of the event *btGetData.Click* so that it:

a) detects if the user has filled *edAge*, *edName* or both text boxes

b) creates the appropriate dynamic query

c) if the user let the two text boxes empty, the DBGrid lists all the records in the DB table *friends*

d) you may create the dynamic queries with string concatenation or parameters

one last thing: **I wish you good luck ☺ !**

thank you

and see you in two weeks!