

Lazarus и базы данных

Этот материал об использовании встраиваемой БД SQLite в проектах Lazarus. Статья на английском, но будущих магистров это не должно пугать)))

Using a SQLite database in a Lazarus program

Version 1

Contents

Introduction

A Lazarus program can modify, interrogate and view the contents of an existing SQLite database.

This tutorial is a step-by-step guide. However, it is worth looking at the layout of the [Form](#) and the [source code](#) now. They are at the end of this document.

I am working with Lazarus 0.9.28.2 beta in MS Windows.

You will connect using components from the SQLdb component palette. This comes with Lazarus. This palette has a specific SQLite3 component which we will use.

Preparations

Create a folder called SQLiteConnect. To save a Lazarus application into the folder

- Click File/New/Application and the familiar Form1 should appear
- Click File/Save All and navigate to your 'SQLiteConnect' folder
- Click Save (for the unit) and Save (for the project).
- Click the green run icon and check that a blank form appears
- Close the blank form.

Lazarus is ready.

You will need SQLite3 to create the database. See Appendix A.

You need to create an SQLite database beforehand. My database is called A.db. See Appendix B.

A.db must be in the SQLiteConnect folder with your Lazarus project. Copy and paste A.db, if necessary

You need the file SQLite3.dll. It is an easy download. See Appendix C.

Everything is now in place.

Summary

You need

- A Lazarus project in your project folder called *SQLiteConnect*
- A SQLite database called *A.db* - Student(*StudentID*, First, Second) is in *SQLiteConnect*
- The file SQLite3.dll is in *SQLiteConnect*

Putting the Lazarus components on the form

In Lazarus, you need to find the SQLdb component palette at the end of the row of tabs holding components. On the right of SQLdb component palette you will find a component called TSQLite3Connection.

- Click on it.
- Click on Form1 and drag open a window
- Release and the component will appear. TSQLite3Connection is not a visible component. It does not appear on the Form when the program runs. We will change its properties dynamically as the program runs.

The TSQLTransaction component is at the other end of the SQLdb component palette.

Drop a TSQLTransaction component underneath the TSQLite3Connectioncomponent on the form using the same method as before. This component passes Database transactions to the Connection component.

The TSQLQuery component is in the SQLdb component palette near the TSQLTransaction component.

Drop a TSQLQuery underneath the TSQLTransaction component. The TSQLQuery component stores the results from any query that you have run on the database. It also passes any changes to its contents to the TSQLite3Connectioncomponent.

The TDataSource component is in the Data Access component palette.

Drop a TDataSource component on the form underneath the TSQLQuery1 component. This takes data from the TSQLQuery component and feeds it to the database-aware components that you will shortly put on your form.

The TDBGrid component is in the DB controls component palette.

Drop a DBGrid component onto your form under the TDataSource component. This is a visible component so you must drag it to a reasonable size on your form. It is a database-aware component and it displays whatever it is fed from the datasource.

The TButton component is in the Standard component palette.

- Drop a button onto your form at the top and resize it.
- Click on the Button once.
- Check that the Object inspector shows the button and change the 'caption' property from 'Button 1' to 'Connect'
- The button should now have the word 'Connect' on it.
- Click the green run button. A form with just your button should appear.
- Close the form.

Summary

Components on the form

- SQLite3Connection
- SQLTransaction1
- SQLQuery1
- Datasource1
- DBGrid1

- Button1 : caption = 'Connect'

Connecting

You now set up the connection between the program and the database. When you click the button 'Connect' then a connection to a.db, (the SQLite database) is made and the data from your database will appear in the DBgrid.

Here is how:

Double-click the button. The source editor appears. In the Button1click procedure add the following code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

SQLite3Connection.DatabaseName:='A.db';
SQLite3Connection.Transaction:=SQLite3Transaction1;

SQLite3Transaction1.Database:=SQLite3Connection;

SQLiteQuery1.Database:=SQLite3Connection;
SQLiteQuery1.Transaction:=SQLite3Transaction1;
SQLiteQuery1.SQL.Text:='SELECT * FROM Student';

Datasource1.dataset:=SQLiteQuery1;
DBGrid1.DataSource:=Datasource1;
end;
```

This code

- Tells the TSQLite3Connection the name of the database and the transaction
- Tells the TSQLite3Transaction the name of the TSQLite3Connection
- Tells the TSQLiteQuery the name of the TSQLite3Connection and the TSQLite3Transaction.
- Tells the datasource the name of TSQLiteQuery
- Tells the DBGrid the name of its datasource

You can see that it prepares a route from the database to the DBGrid. It also prepares (but does not execute) a query to get all the data from the database using
SELECT * FROM Student

- Run the program.
- Click the button

There should be no errors. You should see the same as before.

If the program doesn't run or there is an error when you click the button then you must stop and check everything. It is pointless to proceed since this problem is going to block you every time.

Add a label to Form1. The TLabel component is in the 'Standard' component palette. This label will change to show you are connected when you connect.

To do this, add the 5 lines that are highlighted just before the end;

```
Datasource1.dataset:=SQLQuery1;  
DBGrid1.DataSource:=DataSource1;
```

```
SQLite3Connection.Open;  
If SQLite3Connection.Connected then  
begin  
Label1.caption:='connected -great';  
end;  
end;
```

- Rerun the program
- Click the button
- “Connected to SQLite database –great” should appear in the label.

If the label doesn’t change then you must stop and check everything. It is pointless to proceed since there is no connection.

Next add the one line of code immediately after the code you just wrote and just before the end of the procedure:

SQLQuery1.open;

This single line executes the query that was prepared earlier. The table of results is stored in SQLQuery1 ‘Select * from Student’ is passed to the TSQLite3Connection and then to the database and the result is passed back through the chain to the TSQLQuery, the datasource and into the DBGrid.

- Run the program
- It should compile without errors
- Click the button

The label should say 'Connected to SQLite database - great' The DBGrid should fill up with the data from your database.

If it does then that is great. You can go on. If it doesn’t then you must stop and check everything. It is pointless to proceed since there is no connection.

Summary

Components

- Label1

Procedures

- Button1.Click

- Initialise components
- Connect
- Open Query

SELECTing data

You can interrogate the data in the database using SELECT.

Add a new button (Button2) and in the Object Inspector change the button's caption to 'SELECT'. Also add an editbox (Edit2) onto the form so that the user can enter their SELECT command. It needs to be fairly wide.

Add the procedure below

```
procedure TForm1.Button2Click(Sender: TObject);
begin
SQLQuery1.Close;
SQLQuery1.SQL.text:=edit1.text;
SQLQuery1.Open;
end;
```

Closing SQLQuery1 clears the contents in preparation for a new query. Then the new query is loaded. Then the new query is executed and the table of results made available in SQLQuery1.

- Run the program.
- Click the Connect button
- Check that the data appears in the DBGrid
- Enter SELECT First FROM Student;
- Click the Select button

The list of first names only should appear in the DBGrid.

Summary

Components

- Button2 : caption = 'SELECT'
- Editbox1

Procedures

- Button2.Click

- Deactivates the SQLQuery1
- Copies the query from the edit box into SQLQuery
- Executes the Query
- Reactivates SQLQuery

INSERTing data directly

There are two ways of inserting data into the database. Both work and both are good. The first method uses a transaction

Add a new button (Button3) and in the Object Inspector change its caption to 'Insert'. Also add an editbox (Edit2) so that the user can enter their INSERT command. It needs to be fairly wide.

Double-click on the button and add this source code:

```
procedure TForm1.Button3Click(Sender: TObject);  
begin  
SQLTransaction1.commit;  
SQLTransaction1.StartTransaction;  
SQLite3Connection.ExecuteDirect(edit2.text);  
SQLTransaction1.commit;  
end;
```

The first line completes any transactions that may be outstanding from procedures that we have yet to write. The next three lines simply perform the transaction. This procedure changes the database directly according to the INSERT command in the edit box.

So if you write

```
INSERT INTO Student VALUES (23, 'Fred','Jones');
```

then this will enter Fred Jones as student 23 into the database.

- Run the program
- Click the connect button
- Add the above In the editbox
- Click the Insert button.
- Enter in the select edit box - SELECT * FROM Student;
- Click the select button
- the new entry in the DBGrid.
- Close the application.

Summary

Components

- Button3 : caption='INSERT'
- Editbox2

Procedures

- Button3.Click

- Inserts data directly

INSERTing data with TSQLQuery

This is the second way. This method uses SQLQuery and is very similar to SELECT

Double-click on the button3 and add this source code:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
//Not now needed - we will use SQLQuery1 instead //SQLTransaction1.commit;
//SQLTransaction1.StartTransaction;
//SQLite3Connection.ExecuteDirect(edit2.text);
//SQLTransaction1.commit;
SQLQuery1.Close;
SQLQuery1.SQL.text:=edit1.text;
SQLQuery1.ExecSQL;
end;
```

Using // comments out a line in Lazarus. The technique above is a neat way of making code invisible to the compiler but retaining it for reference or later use.

This is very similar to the SELECT code. The rule is -

- if you are using SELECT or expect data to be returned to SQLQuery then use **SQLQuery1.open** to execute the query.
- if you are using INSERT/DELETE/CREATE and expect NO data to be returned to SQLQuery then use **SQLQuery1.ExecSQL** to execute the query.

Now if you write

```
INSERT INTO Student VALUES (24, 'Stefani','Germanotta');
```

then this will enter 'Stefani Germanotta' as student 24 in the database.

- Run the program
- Click the connect button
- Add the above in the INSERT editbox
- Click the Insert button.
- Enter in the select edit box - SELECT * FROM Student;
- Click the select button
- Check the new entry is in the DBGrid.
- Close the application.

Summary

Components

- Button3 : caption='INSERT'
- Editbox2

Procedures

- Button3.Click

- Insert data with SQLQuery

Viewing the database in a memo box

You can see the whole database in a memobox. The memobox can't be edited and it doesn't change with each new SELECT because it is not a database-aware component. It's still nice to see the database contents initially (and it's easy!).

Add a new button (Button4) and in the Object Inspector change its caption to 'Show Whole Database'. Add a memo box from the Standard component palette.

```
procedure TForm1.Button4Click(Sender: TObject);  
begin  
SQLQuery1.Close  
SQLQuery1.SQL.Text:='Select * FROM Student';  
SQLQuery1.Open;  
while not SQLQuery1.Eof do  
begin  
memo1.lines.add(  
' ID: ' + SQLQuery1.FieldName('StudentID').AsString +  
' First name: ' + SQLQuery1.FieldName('First').AsString +  
' Second name: ' + SQLQuery1.FieldName('Second').AsString) ;  
SQLQuery1.Next;  
end;  
end;
```

The first 3 lines execute the query

```
SELECT * FROM Student;
```

so the whole table is copied to SQLQuery1 automatically. It then appears in the DBGrid because DBGrid is database-aware. This is just as before.

The while block analyses the database table row-by-row. For each row it dumps a string into the memo box. The string is created by concatenating 4 smaller strings, two of which are simple labels and two of which are the entries in that row of the database.

- Run the program
- Click Connect
- Click Show Whole Database

The database should appear in the DBGrid and the memo box.

Summary

Components

- Button4 caption='View the Whole Database'
- Memobox1

Procedures

- Button4.Click

- Runs SQLQuery1
- Cycles through the database table in SQLQuery1
- Creates a formatted string of the contents of each row
- Puts the string into the memo box.

DELETE data

The method for deleting data is easy. Remember that the code for INSERT isn't just for INSERT but for any SQL command that does not return a result from the database. DELETE is just such a command. Therefore the code for DELETE is identical to the code for INSERT. Therefore you can enter a DELETE command in the INSERT edit box and it will work. Remember to then run 'SELECT * FROM Student' in order to see the effect.

The SQL command

```
DELETE FROM Student WHERE StudentID=23
```

will delete student 23 Fred Jones from the database

To test this out write

- Run the program
- Click the connect button
- Add the above in the Insert editbox
- Click the Insert button.
- Enter in the select edit box - SELECT * FROM Student
- Click the select button
- Check the entry has gone from the DBGrid.
- Close the Form.

To get a 'good' DELETE button you would first have the code for INSERT and then have the code for opening the query 'SELECT * FROM Student' after it. Then the result will then appear instantaneous.

Appendix A

Getting SQLite3.exe

The program can be obtained from [the SQLite website](#) . It can be installed and run on a memory stick (flash drive).

Appendix B

Creating a simple SQLite database called A.db

You need to create an SQLite database. My database is called A.db, and it has one table called Student, with three columns - StudentID(number), First(text) and Second(text). There are three rows in the table. To produce the database

- Open a text editor like 'Notepad'
- Save a text file with filename A.txt
- Copy and paste the SQL below into it.

-- SQL for the Student table

```
CREATE TABLE Student(  
StudentID INTEGER PRIMARY KEY NOT NULL,  
First VARCHAR(20),  
Second VARHAR(20));
```

-- Some Student Values

```
INSERT INTO Student VALUES (1, 'David', 'Beckham');  
INSERT INTO Student VALUES (2, 'William', 'Shakespeare');  
INSERT INTO Student VALUES (3, 'Reginald', 'Dwight');
```

-- End of SQL

Then -

- Save the file into the same folder as SQLite3.exe
- Double-click SQLite3.exe
- Type .read A.txt (just copy and paste this text)
- Type SELECT * FROM Student (at the SQLite prompt)

You should see the three records.

- Type .backup A.db

The Student table should have been imported along with the three records. Check they are there by browsing them.

Appendix B

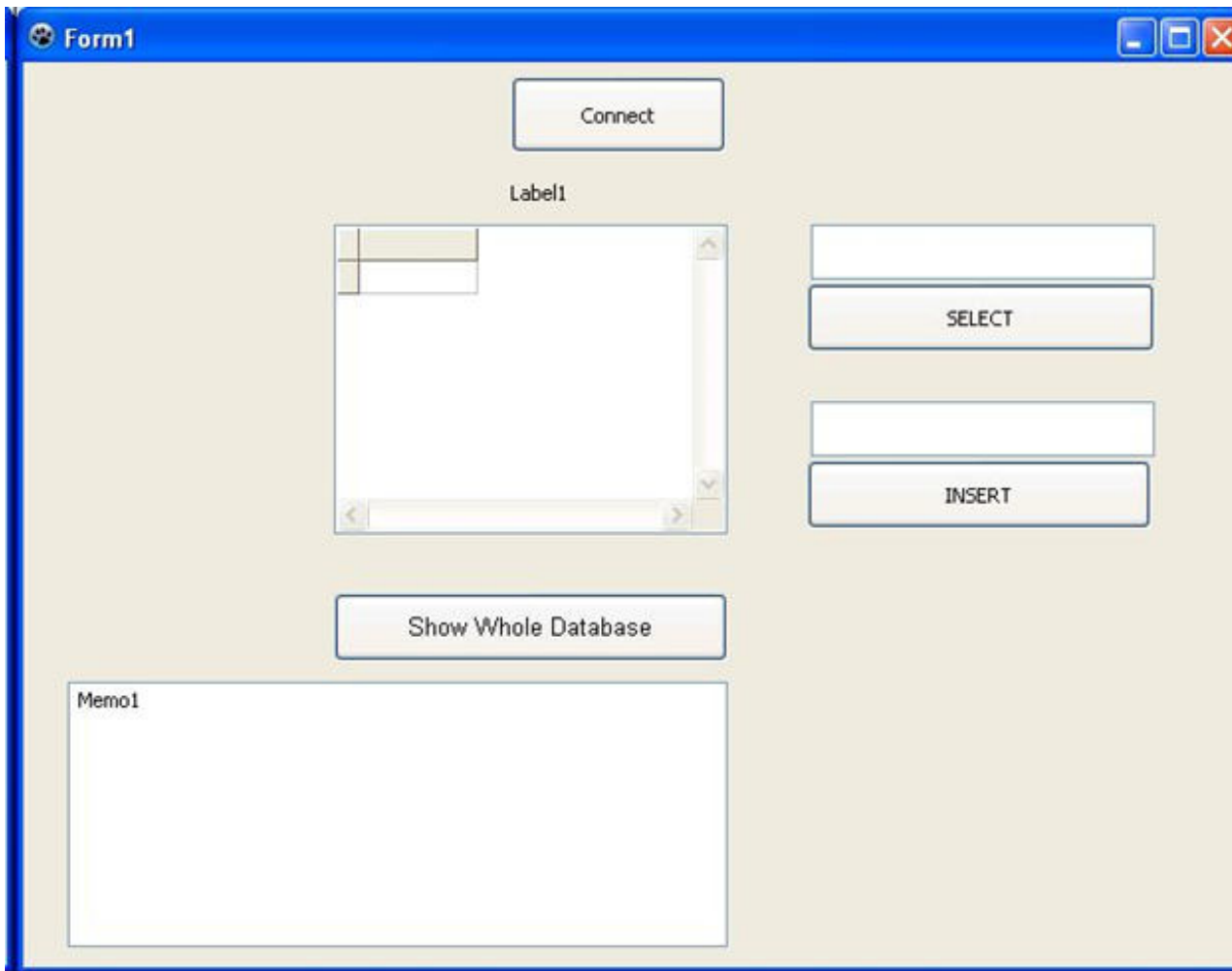
Getting SQLite3.dll

This file can be obtained from [the SQLite website](#).

You download an installer for the file. The installer should be executed. It will create a folder called SQLite3.dll and this contains the SQLite3.dll file. The file needs to go into your SQLite3Connect folder with your project

The link between Lazarus and A.db is made.

Screenshot of the form



Source Code

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, SQLite3Conn, sqldb, db, FileUtil, LResources, Forms, Controls,
  Graphics, Dialogs, DBGrids, StdCtrls;
type
  { TForm1 }
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
```

```

Button3: TButton;
Button4: TButton;
Datasource1: TDataSource;
DBGrid1: TDBGrid;
Edit1: TEdit;
Edit2: TEdit;
Label1: TLabel;
Memo1: TMemo;
SQLite3Connection: TSQLite3Connection;
SQLQuery1: TSQLQuery;
SQLTransaction1: TSQLTransaction;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
private
{ private declarations }
public
{ public declarations }
end;
var
Form1: TForm1;

implementation

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin

SQLite3Connection.DatabaseName:='A.db';
SQLite3Connection.Transaction:=SQLTransaction1;

SQLTransaction1.Database:=SQLite3Connection;

SQLQuery1.Database:=SQLite3Connection;
SQLQuery1.Transaction:=SQLTransaction1;
SQLQuery1.SQL.Text:='SELECT * FROM Student';

Datasource1.dataset:=SQLQuery1;
DBGrid1.DataSource:=DataSource1;

SQLite3Connection.Open;
If SQLite3Connection.Connected then

```

```

begin
Label1.caption:='connected -great';
end;
SQLQuery1.open;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
SQLQuery1.Close;
SQLQuery1.SQL.text:=edit1.text;
SQLQuery1.Open;

end;

procedure TForm1.Button3Click(Sender: TObject);
begin
//Not now needed - we will use SQLQuery1 instead //SQLTransaction1.commit;
//SQLTransaction1.StartTransaction;
//SQLite3Connection.ExecuteDirect(edit2.text);
//SQLTransaction1.commit;
SQLQuery1.Close;
SQLQuery1.SQL.text:=edit1.text;
SQLQuery1.ExecSQL;

end;

procedure TForm1.Button4Click(Sender: TObject);
begin
SQLQuery1.Close
SQLQuery1.SQL.Text:='Select * FROM Student';
SQLQuery1.Open;
while not SQLQuery1.Eof do
begin
memo1.lines.add(
' ID: ' + SQLQuery1.FieldName('StudentID').AsString +
' First name: ' + SQLQuery1.FieldName('First').AsString +
' Second name: ' + SQLQuery1.FieldName('Second').AsString) ;
SQLQuery1.Next;
end;

end;

```

```
initialization
{$! unit1.lrs}

end.
```