Lazarus 与 SQLite3 开发入门

——开源 RAD 工具 Lazarus 初探

写在前面的话

我不是学计算机专业的，但是一直在努力自学计算机技术。学习开发的入门工具是 Delphi，后来又接触了 C++ Builder（控件还是 Delphi 那些），学习时间比较长的是 Java，.NET 也有所接触。2006 年 9 月真正进入 IT 公司后，主要做的是基于 Java 的开发。闲时愿意玩儿 Linux，总愿意尝试在不同的操作系统下都有相应版本的软件，以在 Linux 和 Windows 下用同样的软件，做同样的事情为乐。这不，为了在 Windows 下和 Linux 下都能用同样的 RAD 开发工具，重温一下开发入门的感觉，又找上了 Qt，Lazarus。现在让我们来看看 Lazarus。

Lazarus 是基于 Free Pascal 的 Object Pascal 语言集成开发环境(IDE)。Lazarus 与 Delphi 高度兼容，并被视作后者的开源替代品（本段文字来自维基百科，如果想知道更多请自行学习）。

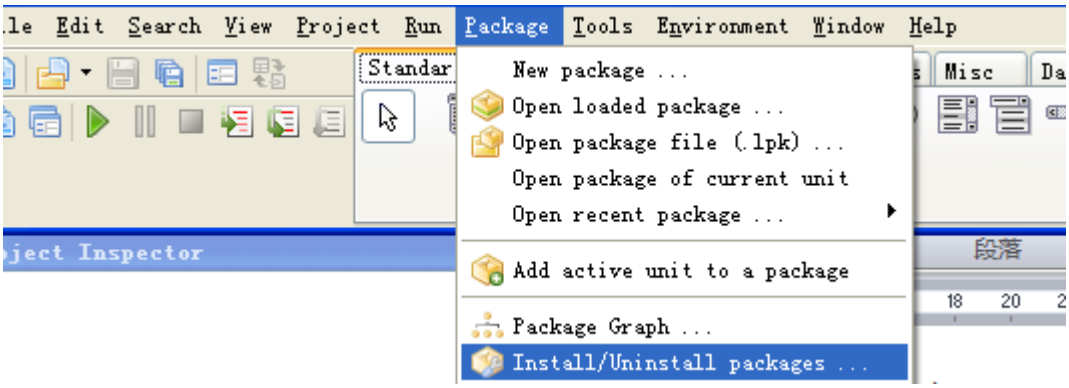
SQLite3 就更不用多说了。本文是基于已有的入门教程进行的整理，但并不是简单的汉化。对于一些必备的知识也做了一些讲解，并增加了每一步的详细的截图，力求做到讲解清楚。有不正之处请大家指正。

学习前的准备工作

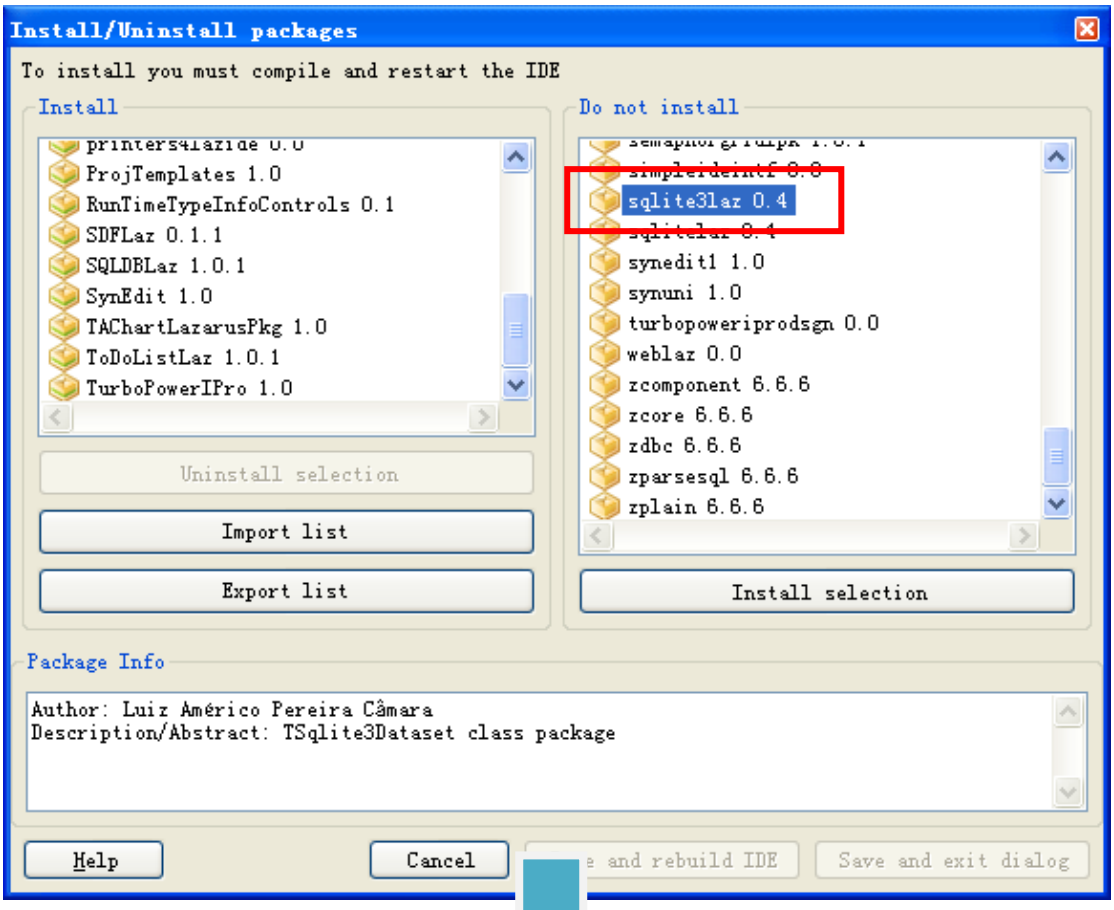
首先、要获取 Lazarus 的安装包。本例是在 Windows 下进行的（版本是 0.9.30），但是你完全可以在 Linux 下进行同样的工作。

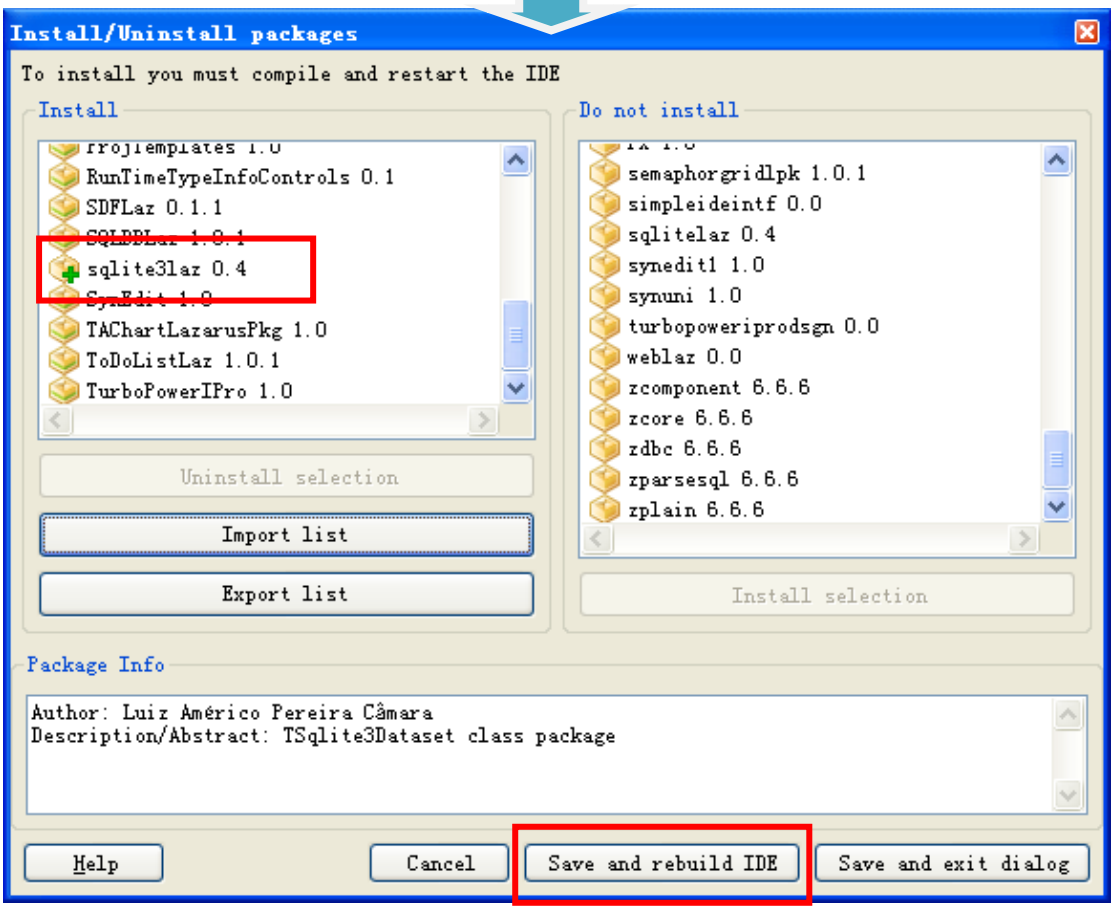
第二、要有 SQLite3 的运行支持库，这个就不详细说明了。

第三、在 Lazarus 下安装支持 SQLite3 的组件。在菜单 Package 下有 Install/Uninstall packages ...一项，如图：

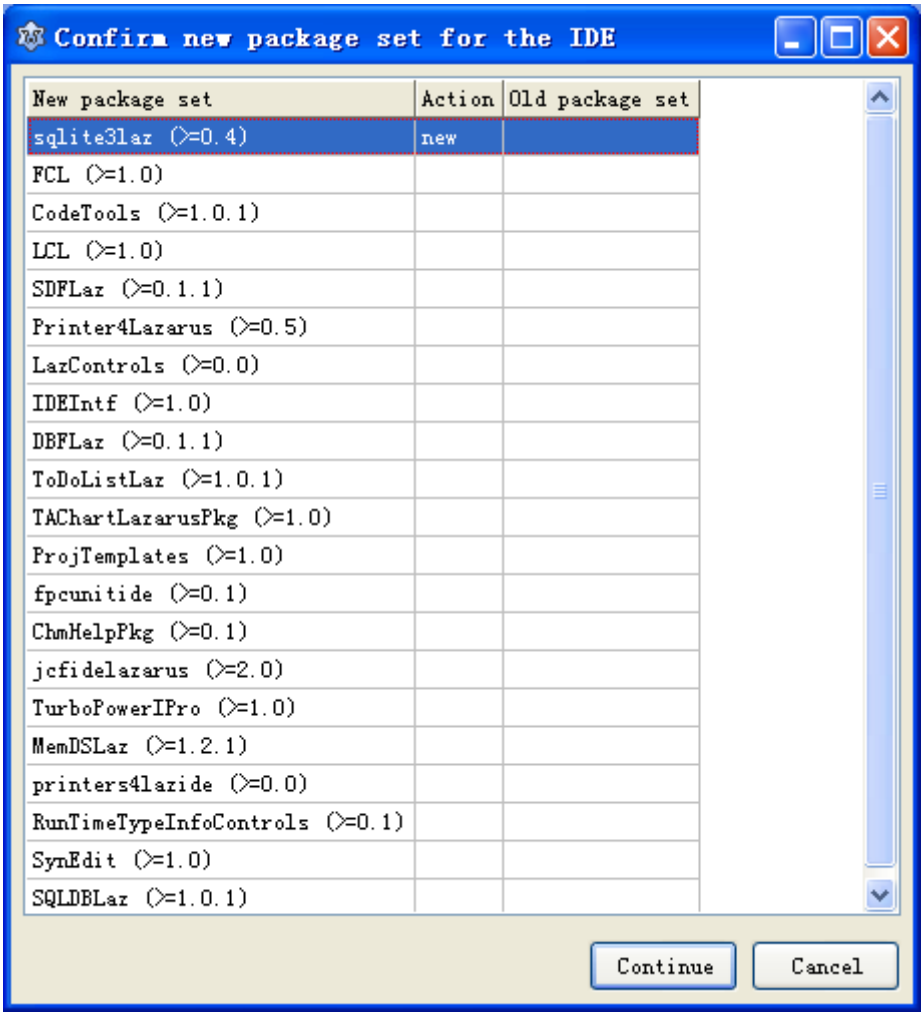


点击后，会出现对话框，在右边的(Do not install)列表里找到 sqlite3laz 0.4，点击按钮 Install selection，将其加入到左边的(Install)列表中，如下图：

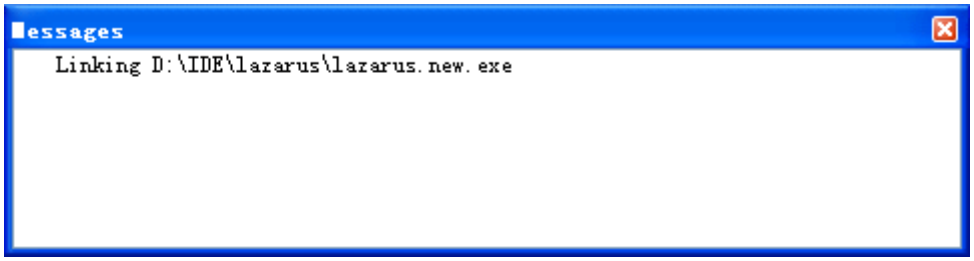

Install/Uninstall packages
To install you must compile and restart the IDE
Install
ProjTemplates 1.0
RunTimeTypeInfoControls 0.1
SDFLaz 0.1.1
SQLDBLaz 1.0.1
SynEdit 1.0
TAChartLazarusPkg 1.0
ToDoListLaz 1.0.1
TurboPowerIPro 1.0
Uninstall selection
Import list
Export list
Do not install
sqlite3laz 0.4
synedit1 1.0
synuni 1.0
turbopoweriprodsgn 0.0
weblaz 0.0
zcomponent 6.6.6
zcore 6.6.6
zdbc 6.6.6
zparsesql 6.6.6
zplain 6.6.6
Install selection
Package Info
Author: Luiz Américo Pereira Câmara
Description/Abstract: TSqlite3Dataset class package
Help
Cancel
Save and exit dialog


Install/Uninstall packages
To install you must compile and restart the IDE
Install
RunTimeTypeInfoControls 0.1
SDFLaz 0.1.1
sqlite3laz 0.4
TAChartLazarusPkg 1.0
ToDoListLaz 1.0.1
TurboPowerIPro 1.0
Uninstall selection
Import list
Export list
Do not install
semaphorgridlpk 1.0.1
simpleideintf 0.0
sqlitelaz 0.4
synedit1 1.0
synuni 1.0
turbopoweriprodsgn 0.0
weblaz 0.0
zcomponent 6.6.6
zcore 6.6.6
zdbc 6.6.6
zparsesql 6.6.6
zplain 6.6.6
Install selection
Package Info
Author: Luiz Américo Pereira Câmara
Description/Abstract: TSqlite3Dataset class package
Help
Cancel
Save and rebuild IDE
Save and exit dialog

然后点击按钮(Save and rebuild IDE)，会出现下面的对话框，继续(Continue)即可。

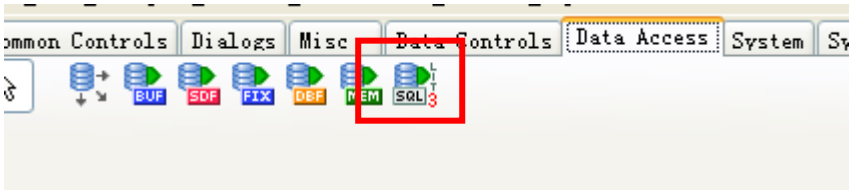


这时 Lazarus 开始重新编译自身，时间不会太长……



编译完成后，Lazarus 会自动重启，这时查看 Data Access 标签下的组件会

发现多了 TSQLite3Dataset。



好了，现在我们可以开始工作了！Let’s go!

教程之一　一个(非常)基础的地址簿

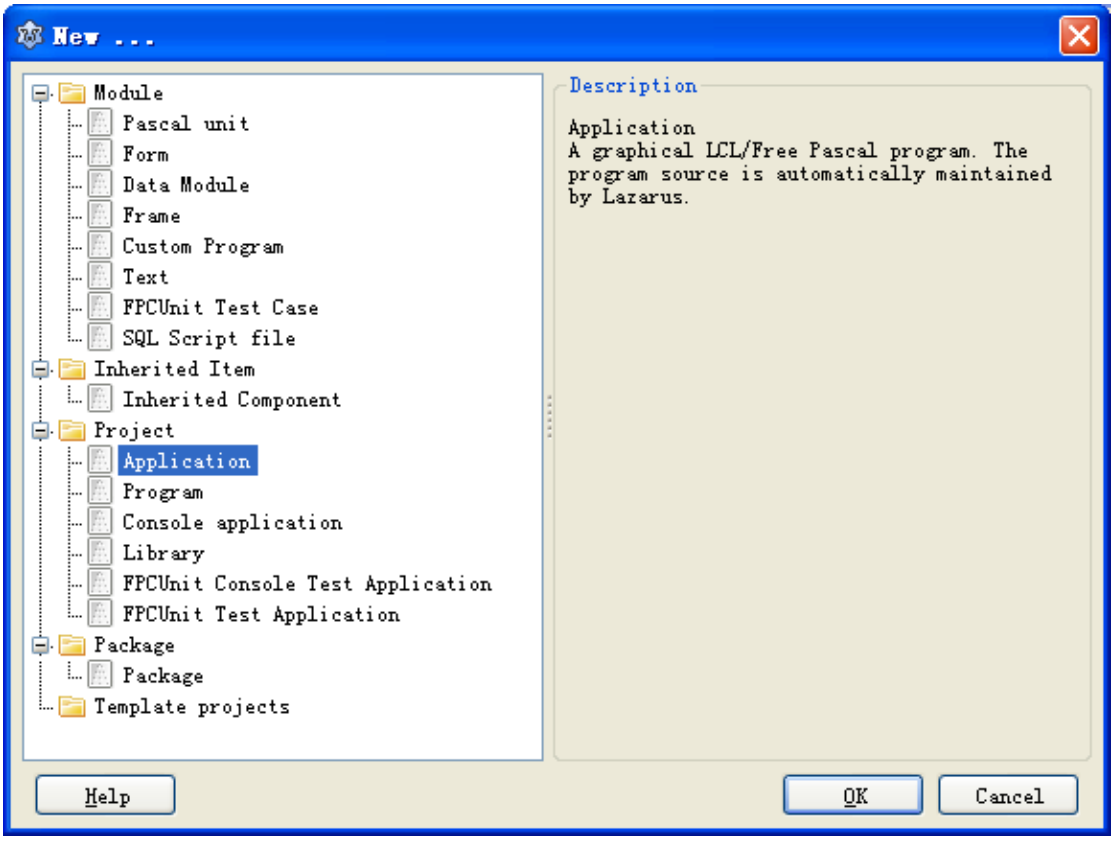
准备工作

创建一个空的应用程序，放置下列组件，并按照指定设置属性。

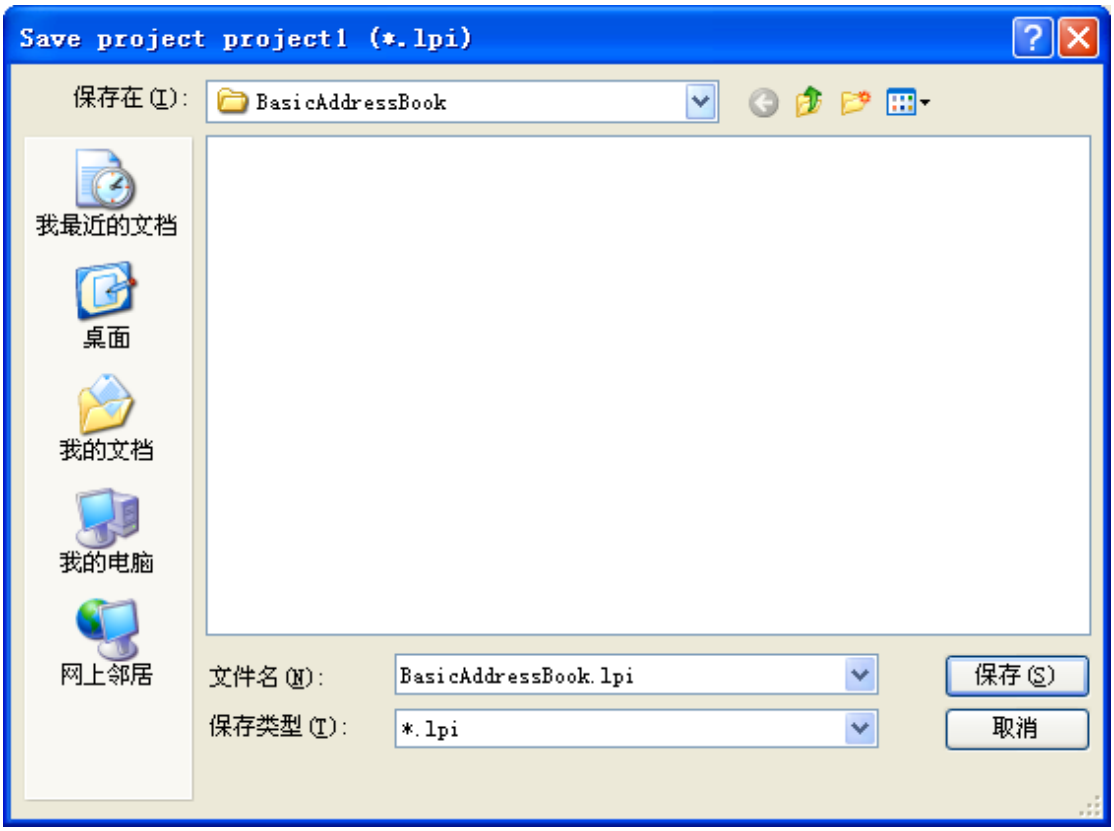
组件	属性	设定值
TSqlite3Dataset	Name	ContactsDataset
	TableName	Contacts
TDataSource	Name	ContactsDataSource
	Dataset	ContactsDataset
TDBGrid	Name	ContactsGrid
	DataSource	ContactsDataSource
	Columns	Name
	•FieldName	Email
TButton	Name	AddButton
	Caption	添加
TButton	Name	DeleteButton
	Caption	删除
TButton	Name	SaveButton
	Caption	保存

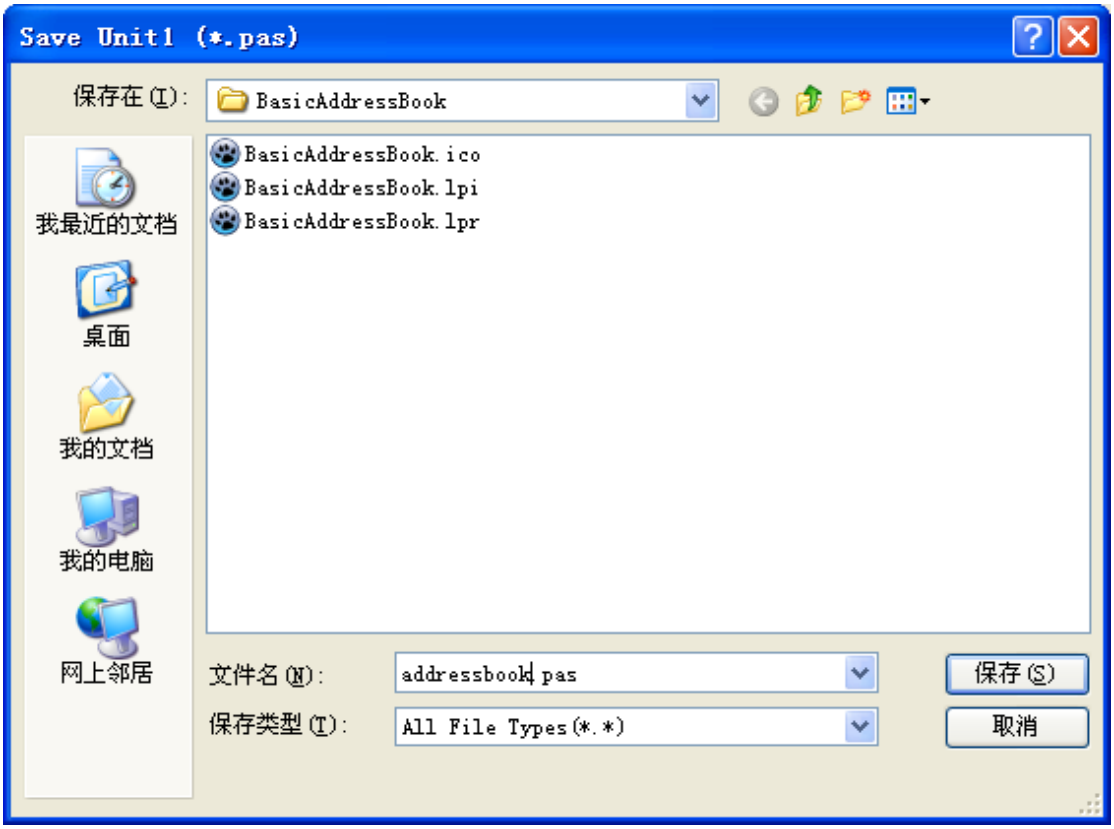
具体可以参考下面：

点击菜单(File)→(New)会出现向导对话框：

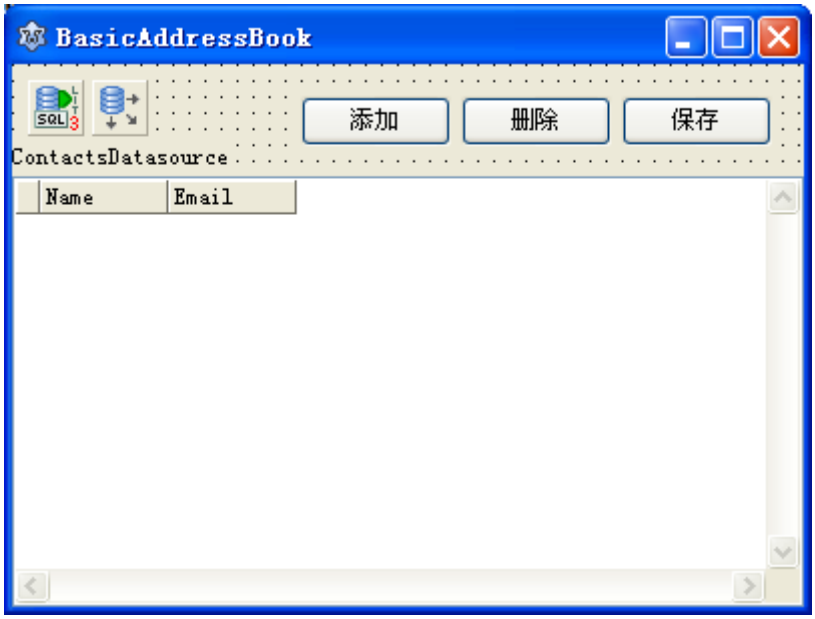


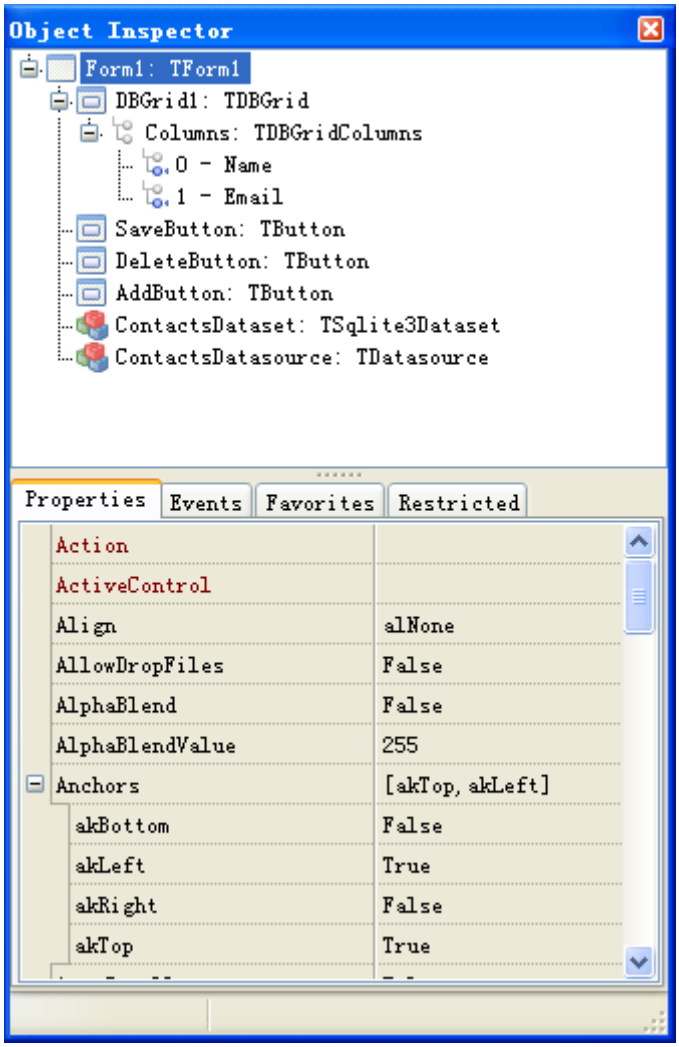
点击 OK，会创建新的应用程序，点击保存，在自己指定的目录下保存工程文件。





布置好的窗体以及组件的属性请参考：





随教程会提供工程文件的，所以更细节的部分请参照工程文件，这里就不再详细抓图了。

创建和载入数据库

在程序中创建数据库文件有三种方法：

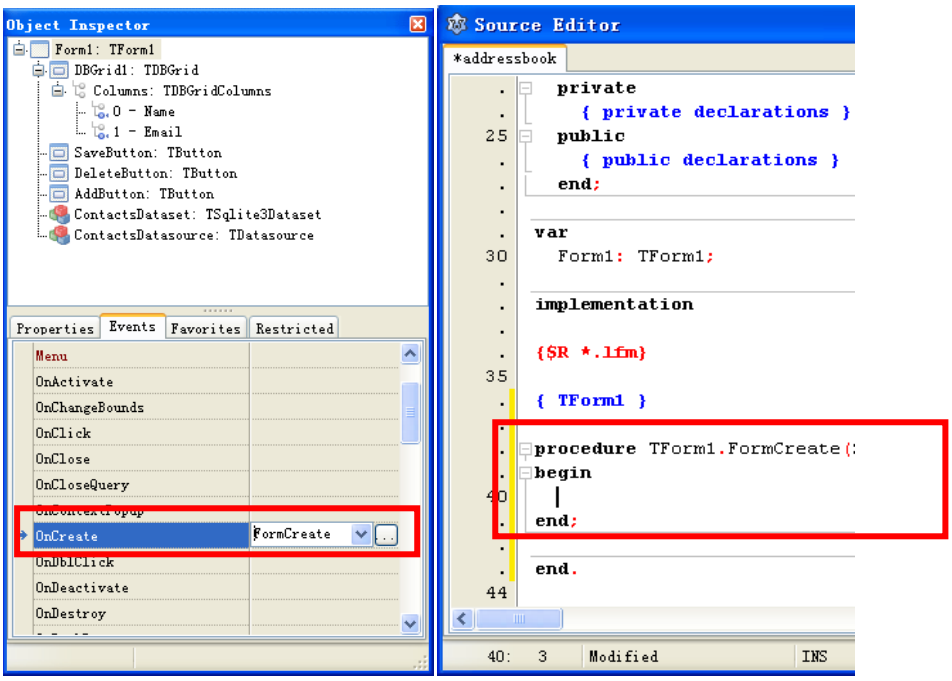
① 使用外部的数据库管理器。不利之处是必须和程序一同发布数据文件，并且在解析数据类型的时候，管理器与 TSqlite3Dataset 之间可能会出现兼容性的问题。

② 右键单击 TSqlite3Dataset 组件，使用内建的表管理器。不利之处是必须和程序一同发布数据文件。

③ 在运行时创建数据文件。这是推荐的方法，因为没有上述的不利之处，本教程将使用这种方法。

为窗体创建在 OnCreate 事件中调用的方法（在窗体属性的 OnCreate 事件

上双击左键)：



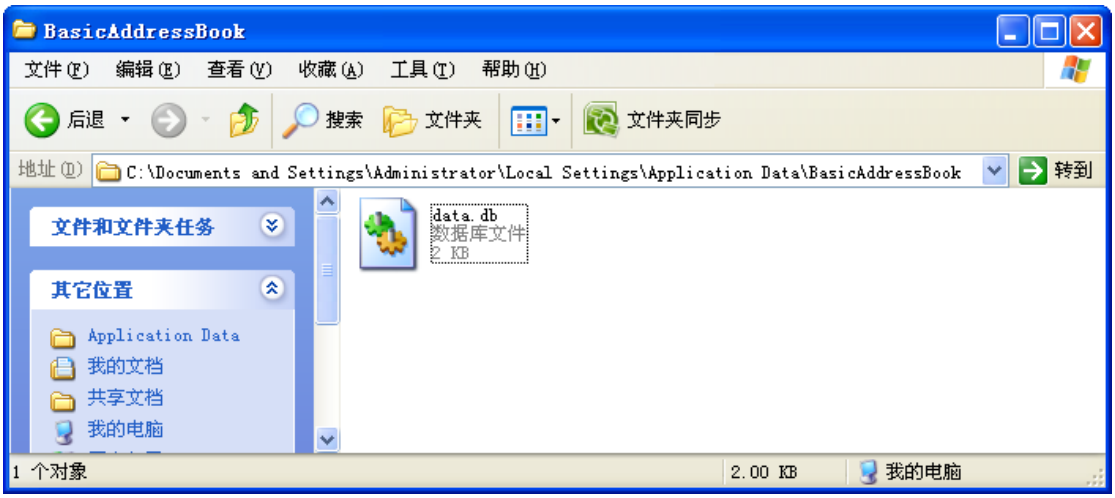
输入下面的代码：

```
1.  procedure TForm1.FormCreate(Sender: TObject);
2.  begin
3.    ForceDirectoriesUTF8(GetAppConfigDirUTF8(False));
4.    with ContactsDataset do
5.    begin
6.      FileName := IncludeTrailingPathDelimiter(GetAppConfigDirUTF8(False
    )) + 'data.db';
7.      if not TableExists then
8.      begin
9.        FieldDefs.Clear;
10.       FieldDefs.Add('Id', ftAutoInc);
11.       FieldDefs.Add('Name', ftString);
12.       FieldDefs.Add('Email', ftString);
13.       CreateTable;
14.     end;
15.     Open;
16.   end;
17. end;
```

下面一行一行地说明代码：

```
1.      FileName := IncludeTrailingPathDelimiter(GetAppConfigDirUTF8(False
    )) + 'data.db';
```

这是数据文件将被保存的路径。Windows 下是(当前用户 Administrator) :



那么 ,为什么使用 GetAppConfigDirUTF8 而不是放在与程序文件相同的目录或者子目录下呢？如果是在 Windows(Vista 之前)下也可以 ,所有的用户都被允许在程序文件夹下写文件，但是在 Linux 和 Vista 下就无效了。有关 GetAppConfigDir*的更多信息请查阅 fpc 站点的文档。不论以何种方式，这取决于程序员选择在哪里存储数据文件。

GetAppConfigDirUTF8 与 GetAppConfigDir 具有相同的行为，除了返回 UTF8 格式编码的字符串。Sqlite 要求 UTF8 格式编码的字符串，因此建议使用 FileUtil 中的*UTF8 函数。

```
1.      if not TableExists then
```

检查表 Contacts(属性 TableName 的值)是否已经存在于数据文件中。

```
1.  FieldDefs.Clear;
2.  FieldDefs.Add('Id', ftAutoInc);
3.  FieldDefs.Add('Name', ftString);
4.  FieldDefs.Add('Email', ftString);
5.  CreateTable;
```

如果表不存在，将会创建包含字段 Id，Name 和 Email 的表。

那么为什么创建自增的字段 Id 呢？

要使 TSqlite3Dataset 的保存函数(ApplyUpdates)正常工作，必须要有一个主键字段，也就是说，一个非空的唯一值的字段。使用 TSqlite3Dataset 时，设定一个自增字段是获取主键的最简单的方法。

```
1.      Open;
```

这会使数据集进入活动状态，所有保存在数据文件中的记录将被载入并且显示在网格组件中。

添加和删除记录、保存所做的改变

在 AddButton 的 OnClick 事件中输入：

```
procedure TForm1.AddButtonClick(Sender: TObject);
begin
  ContactsDataset.Append;
  ContactsDataset.FieldByName('Name').AsString := 'New Contact';
  ContactsDataset.FieldByName('Email').AsString := 'xxx@yyy.com';
  ContactsDataset.Post;
end;
```

Append 将会在数据集的最后添加一条记录。调用 FieldByName 为新记录添加缺省值。Post 通知数据集编辑完成。警告：Post 并不能保存对数据文件的改变。

在 DeleteButton 的 OnClick 事件中输入：

```
procedure TForm1.DeleteButtonClick(Sender: TObject);
begin
  if ContactsDataset.RecordCount > 0 then
    ContactsDataset.Delete;
end;
```

Delete 函数从数据集中移除当前的记录。RecordCount 检查是为了避免空数据及的异常。

在 SaveButton 的 OnClick 事件中输入：

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
  ContactsDataset.ApplyUpdates;
end;
```

这将把所有对数据集的改变(添加、删除以及值改变)保存到数据文件中。下次程序打开的时候，新的值会显示。如果 ApplyUpdates 没被调用，本次会话所做的改变将会丢失。

那么 SQL 到底在哪儿？

我们注意到程序员一条 SQL 命令都没用到。TSqlite3Dataset 具有不使用 SQL 命令操作数据集的能力，同时还提供了访问 SQL/Sqlite 所有特性的方法。在后面我们会接触更多。

那么在内部，TSqlite3Dataset 做了些什么？

①在 Sql 属性为空的情况下（如本教程）调用 Open，它使用形如’ Select * from [TableName]’ 的 Sql 命令来载入数据。有必要在此提醒，如果 Sql 属性为空的话，TableName 属性就不能为空。

②ApplyUpdates 通过 TDataset 调用（Append、Delete、Edit）修改记录来创建 SQL 命令。想要正常工作，主键（本教程中的 Id 字段）和 TableName 属性的有效值是必须的。

工程文件可以到这里下载。

教程之二　主表/从表

在本教程中，我们将为教程一中的程序增加为每个联系人存储电话号码的功能。这将通过设定一对多的主表/从表关系来实现，例如，每个联系人可以拥有一个或多个电话号码。讲解关系数据库的理论不在本教程的范围，有任何问题可以请教你的谷歌老师。

让我们从在教程一完成的工作开始。打开程序的一个副本并加入下面的组件：

组件	属性	设定值
TSqlite3Dataset	Name	PhonesDataset
	TableName	Phones
	MasterDatasource	ContactsDatasource
	MasterFields	Id
	IndexFieldNames	ParentId
	SaveOnClose	True
	SaveOnRefetch	True
TDataSource	Name	PhonesDatasource
	Dataset	PhonesDataset
TDBGrid	Name	PhonesGrid
	DataSource	PhonesDatasource
	Columns •FieldName	Phone

TButton	Name	AddPhoneButton
	Caption	+
	Name	DeletePhoneButton
	Caption	-

设定 ContactsDataset 的 SaveOnClose 属性为 True。

另外，删除 SaveButton 以及它的 OnClick 事件对应的方法（稍后详述）。

本教程涉及的新内容是建立主表/从表关系所需的属性。

- MasterSource：连接至主表数据集(ContactsDataset)的 TDatasource。
- MasterFields：主表数据集中可以唯一确定一条记录的字段，通常为主键。
- IndexFieldNames：从表数据集关联至主表数据集中一条记录的字段，又名外键。

一些注意事项：

①在 MasterKeys 和 IndexFieldNames 中可以设定多个字段名（用“;”分隔），这两个属性中的字段数量必须一致。

②所有与主表/从表相关的属性都是在从表的数据集中设定的，不要改动主表数据集的设定。

创建新表

我们来修改 OnCreate 事件对应的方法以创建 Phones 表：

【接前次教程……】

```
1.  with PhonesDataset do
2.  begin
3.    FileName := IncludeTrailingPathDelimiter(GetAppConfigDirUTF8(False))
     + 'data.db';
4.    if not TableExists then
5.    begin
6.      FieldDefs.Clear;
7.      FieldDefs.Add('Id', ftAutoInc);
8.      FieldDefs.Add('Phone', ftString);
9.      FieldDefs.Add('ParentId', ftInteger);
10.     CreateTable;
11.   end;
12.   Open;
13. end;
```

可以看到，这与创建 Contacts 表特别类似。要注意到文件名和 ContactsDataset 是一样的，我们增加了额外的字段（外键）来对应主表/从表关系。

为添加/删除电话号码增加代码

在 AddPhoneButton 的 OnClick 事件中加入：

```
1. procedure TForm1.AddPhoneButtonClick(Sender: TObject);
2. begin
3.   PhonesDataset.Append;
4.   PhonesDataset.Post;
5. end;
```

在 DeletePhoneButton 的 OnClick 事件中加入：

```
1. procedure TForm1.DeleteButtonClick(Sender: TObject);
2. begin
3.   if ContactsDataset.RecordCount > 0 then
4.     ContactsDataset.Delete;
5. end;
```

这里就不多说了，我们用 IsEmpty 属性来取代检查 RecordCount。

当删除一个联系人的时候要一起删除相关联的所有电话号码，要实现这个功能，就要在 ContactsDataset 的 BeforeDelete 事件中加入代码：

```
1. procedure TForm1.ContactsDatasetBeforeDelete(DataSet: TDataSet);
2. begin
3.   while not PhonesDataset.IsEmpty do
4.     PhonesDataset.Delete;
5. end;
```

完成

很好，但是之前说过，需要调用 ApplyUpdates 来保存改变，可是在这里又要求移除保存的代码。出什么问题了吗？

这个问题很好。如果必要的时候，TSqlite3Dataset 有自动调用 ApplyUpdates 的能力，基本上，当数据缓冲被释放的时候，也就是说在关闭(关闭、销毁)和取回数据(重取数据、内部的主/从机制)。为了达到这个目的，提供

了两个属性：SaveOnClose 和 SaveOnRefetch。

因此，在 ContactsDataset 和 PhonesDataset 中设定 SaveOnClose 属性为 True，并且将 PhonesDataset(从表数据集)的 SaveOnRefetch 属性设定为 True。

一些提示：

①在不久的将来，Sqlite 的所有布尔型的属性，比如 SaveOn*将在选项属性中分成一组(一组元素)，行为保持不变。

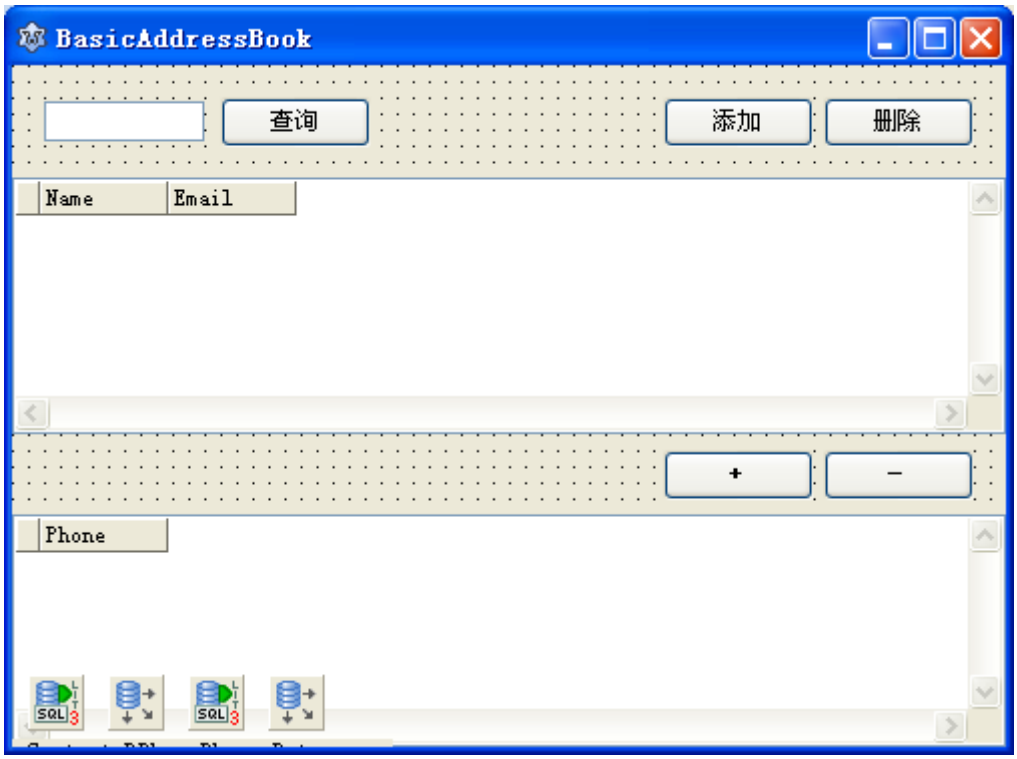
再有，SQL 知识/用法不是必须的。SQL 的逻辑工作已经由 TSqlite3Dataset 完成了，不需要程序员的干预。

工程文件可以在这里下载。

教程之三　定位记录

给我们的小程序加上查找的功能。备份教程二的工程后，加入下面的组件：

组件	属性	设定值
TEdit	Name	SearchEdit
	Text	
TButton	Name	SearchButton
	Dataset	查询



设计概述

设计由一个允许用户输入要查找的文字（联系人姓名的部分或全部）的编辑组件以及一个关联的按钮构成。当第一次按下按钮的时候，从第一条记录开始查找。再次按下按钮的时候，从当前的记录开始继续向后查找。改变查找文字会重置查找，再次查找会再从第一条记录开始。如果查找不到会将编辑组件的颜色变为红色。

还有其它更有效地方法实现这个特性，比如活动搜索，但是为了保持代码的简单而选择了当前的设计。

实现

在窗体的实现中加入一个私有变量 FSearching：

```
1.  unit addressbook;
2.
3.  [...]
4.  type
5.
6.    { TForm1 }
7.    [...]
8.    TForm1 = class(TForm)
9.      AddPhoneButton: TButton;
10.   private
11.     { private declarations }
12.     FSearching: boolean;
13.   public
14.     { public declarations }
15.   end;
```

在 SearchButton 的 OnClick 事件中加入：

```
1.  procedure TForm1.NameButtonClick(Sender: TObject);
2.  var
3.    SearchText: string;
4.    Options: TLocateOptions;
5.    Ok: boolean;
6.  begin
7.    SearchText := Trim(SearchEdit.Text);
```

```
8.    if SearchText = '' then
9.      Exit;
10.
11.   Options := [loCaseInsensitive, loPartialKey];
12.
13.   if FSearching then
14.     Ok := ContactsDataset.LocateNext('Name', SearchText, Options)
15.   else
16.     Ok := ContactsDataset.Locate('Name', SearchText, Options);
17.   FSearching := True;
18.   if Ok then
19.     SearchEdit.Color := clWindow
20.   else
21.     SearchEdit.Color := clRed;
22. end;
```

解说：

```
1.  SearchText := Trim(SearchEdit.Text);
2.  if SearchText = '' then
3.    Exit;
```

以上代码用来从编辑组件取得查找文字，并检查是否为空。

```
1.  Options := [loCaseInsensitive, loPartialKey];
```

对 Options 的设置将传递给 Locate*函数。loCaseInsensitive 意味着搜索时忽略大小写。loPartialKey 意味着匹配以查找文字开头的记录。

也可以在查找关键字中用通配符，例如，查找’*Paul*’将会匹配任何包含’Paul‘的值，忽略位置。要实现这个功能，在 ContactsDataset.options 中设定 soWildCardKey。

```
1.  if FSearching then
2.    Ok := ContactsDataset.LocateNext('Name', SearchText, Options)
3.  else
4.    Ok := ContactsDataset.Locate('Name', SearchText, Options);
5.    FSearching := True;
```

FSearching 是个全局属性，用来标记查找是否开始。

Locate 和 LocateNext 以要查找的字段名为第一个参数 ,值必须和第二个参数匹配。Option 传给第三个参数。查找到匹配的记录都会返回 True，不同之处在于 Locate 从第一条记录开始查找而 LocateNext 从当前活动的记录开始查找。

在 SearchEdit 的 OnChange 事件中加入：

```
1.  procedure TForm1.SearchEditChange(Sender: TObject);
2.  begin
3.    FSearching := False;
4.    SearchEdit.Color := clWindow;
5.  end;
```

重置查找标记以及编辑框的颜色。

完成

可以利用 TSqlite3Dataset 以多种方式查找/过滤记录。我们以后会看到这些。

工程文件可以从这里下载

我无法抗拒地制作了活动搜索的版本 ;-)

（声明：文中的链接均来自 **http://sqlite4fpc.yolasite.com** ）

（内容均来自 http://sqlite4fpc.yolasite.com 以及自己的探索）