Базы данных

Лабораторная работа №4

«**Музыкальный магазин**»

Задача. В среде разработки Lazarus создать интерфейс для работы с базой данных типа SQLite с условным названием «Музыкальный магазин», состоящей из следующих взаимосвязанных таблиц:

Таблица **Artists:**

Назначение поля	Название поля	Тип	Размер
Идентификатор	ID	INTEGER	
Имя исполнителя	Name	VARCHAR	80

Таблица **Albums:**

Назначение поля	Название поля	Тип	Размер
Идентификатор	ID	INTEGER	
Название альбома	Name	VARCHAR	100
Указатель на исп-ля	Artist	INTEGER	

Таблица **Genres:**

Назначение поля	Название поля	Тип	Размер
Идентификатор	ID	INTEGER	
Название жанра	Name	VARCHAR	80

Таблица **Songs:**

Назначение поля	Название поля	Тип	Размер
Идентификатор	ID	INTEGER	
Номер трека в альбоме	Track	INTEGER	
Название трека	Name	VARCHAR	160
Указатель на альбом	Album	INTEGER	
Жанр (указатель)	Genre	INTEGER	

В программе предусмотреть поиск по следующим параметрам (раздельно):

1) по **имени исполнителя** с выводом списка названий его альбомов;

2) поиск треков **по названию жанра и по исполнителю**;

3) поиск по **фрагменту названия трека** с выводом названия жанра, альбома, трека и исполнителя.

Выполнение проекта

Для начала поработаем с главной формой. Поместим сюда главное меню со следующей структурой:

База данных:
- Создание БД
- Жанры

Поиск:
- Исполнитель
- Жанр и исполнитель
- Название трека

Пунктам меню дадим осмысленные названия, например:

MenuDB:
- MenuDBCreate
- MenuGenres

MenuSearch:
- MenuSearchArtist
- MenuSearchGenres
- MenuSearchTrack

На форме настройки оставим исходные, изменим только

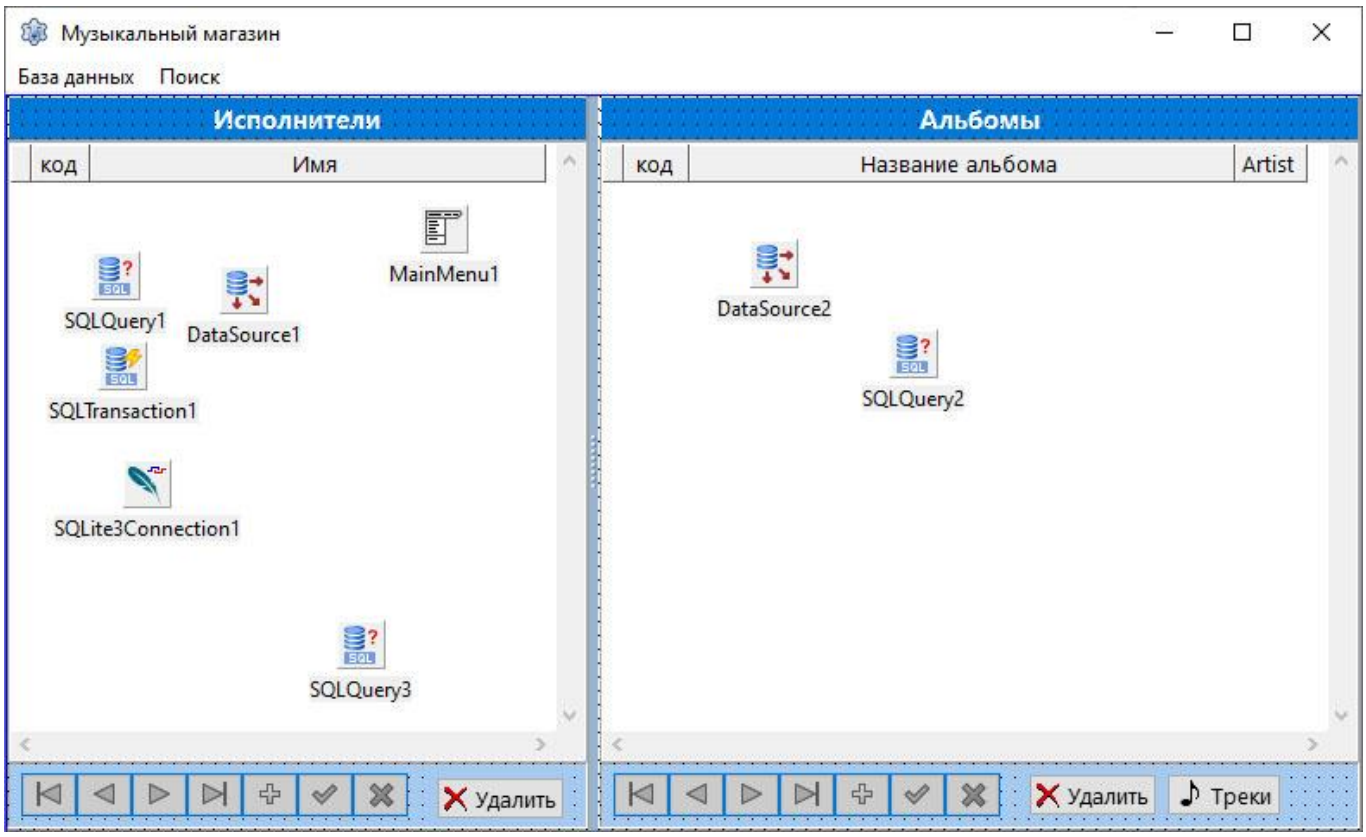
`Position = poScreenCenter`.

Можно также настроить параметры

`Constraints`: `MaxHeight, MaxWidth, MinHeight, MinWidth`.

Особенно два последних – поставить пределы, чтобы при уменьшении формы невозможно было спрятать важные элементы на форме.

Настройки формы лучше делать до того, как она вся будет закрыта другими элементами, так как потом может быть сложно в Инспекторе Объектов сделать её активной для работы с методами и свойствами формы. Поэтому желательно заранее включить события формы `OnCreate` и `OnShow` – они нам понадобятся.



Две панели, разделённые сплиттером.

Для работы с базой данных нам понадобятся следующие компоненты: с вкладки SQLdb берём: SQLite3Connection1, SQLTransaction1, три SQLQuery, а с вкладки Data Access – два DataSource. С вкладки Data Controls берём два DBGrid и два DBNavigator.

Настроить их в Инспекторе Объектов надо следующим образом:

```
SQLTransaction1.DataBase = SQLite3Connection1
SQLite3Connection1.DatabaseName = <Путь к базе данных>
SQLQuery1.Database = SQLite3Connection1
SQLQuery2.Database = SQLite3Connection1
SQLQuery3.Database = SQLite3Connection1
```

`DataSource1.DataSet = SQLQuery1` (аналогично для DataSource2)

`DBGrid1.DataSource = DataSource1` (аналогично для DBGrid2)

`DBNavigator1.DataSource = DataSource1`

(аналогично для DBNavigator2)

Чтобы набор данных мог редактироваться, для SQLQuery следует указать в `Options` значения `[sqoKeepOpenOnCommit,sqoAutoApplyUpdates, sqoAutoCommit]`, это лучше сделать в Инспекторе Объектов.

Также удобнее сделать в Инспекторе Объектов следующие настройки:

`SQLQuery1.SQL = SELECT * FROM Artists`

`SQLQuery2.DataSource = DataSource1` (это вместо `MasterSource`)

SQLQuery2.SQL = SELECT * FROM Albums WHERE Artist = :ID

в SQLQuery2 в Params добавить параметр ID и указать у него

DataType = ftInteger

В начале работы программы определим путь к базе данных в событии Form1.OnCreate:

```
var
  Form1: TForm1;
  BaseName: string;

implementation

{$R *.lfm}

uses Unit_Genres, Unit_Songs, SearchGenre, SearchTrack;

var SearchName: string;

procedure TForm1.FormCreate(Sender: TObject);
begin
  BaseName:= ExtractFilePath(Application.ExeName) + 'mus_shop.sqlite';
end;
```

BaseName здесь глобальная переменная. Такой вариант удобен тем, что программу можно помещать в любой каталог и не надо в тексте программы делать исправления в его названии. Хотя при проектировании удобно указать этот путь в SQLite3Connection1 в свойстве DatabaseName.

Пока строку с **uses** закомментируйте, так как модули будут созданы позднее.

Чтобы идентификаторы ID в таблицах рассчитывались автоматически, объявляем их главными ключами (Primary Key) со свойством AutoIncrement.

Сначала надо создать базу данных и таблицы. Сделаем это в обработчике пункта меню «Создание БД»:

```
procedure TForm1.MenuDBCreateClick(Sender: TObject);
begin
  SQLite3Connection1.Connected:= False;
  SQLTransaction1.Active:= False;
  if FileExists(BaseName) then begin
    DeleteFile(BaseName);
    ShowMessage('Файл удалён: ' + BaseName);
  end;
  SQLite3Connection1.CreateDB;
  ShowMessage('База данных создана');
  SQLQuery3.Close;
  SQLite3Connection1.Connected:= True;
  SQLTransaction1.Active:= True;
  with SQLQuery3 do begin
    SQL.Clear;
    SQL.Add('CREATE TABLE Artists');
    SQL.Add(' (ID INTEGER PRIMARY KEY AUTOINCREMENT, Name VARCHAR(80))');
    ExecSQL;
    SQL.Clear;
    SQL.Add('CREATE TABLE Albums');
    SQL.Add(' (ID INTEGER PRIMARY KEY AUTOINCREMENT, Name VARCHAR(80),');
    SQL.Add('  Artist INT NOT NULL REFERENCES Artists)');
    ExecSQL;
    SQL.Clear;
    SQL.Add('CREATE TABLE Genres');
    SQL.Add(' (ID INTEGER PRIMARY KEY AUTOINCREMENT, Name VARCHAR(80))');
    ExecSQL;
    SQL.Clear;
    SQL.Add('CREATE TABLE Songs (ID INTEGER PRIMARY KEY AUTOINCREMENT, ');
    SQL.Add('  Track INT, Name VARCHAR(160), ');
    SQL.Add('  Album INT NOT NULL REFERENCES Albums, ');
    SQL.Add('  Genre INT NOT NULL REFERENCES Genres)');
    ExecSQL;
  end;
  SQLTransaction1.Commit;
  FormShow(nil);
  ShowMessage('Таблицы созданы');
end;
```

После этого можно проверить результат в SQLite Studio.

Сделаем так, что при запуске программы в событии `Form1.OnShow` загружаются списки исполнителей с их альбомами:

```
procedure TForm1.FormShow(Sender: TObject);
begin
  SQLite3Connection1.DatabaseName:= BaseName;
  if not FileExists(BaseName) then exit;
  SQLite3Connection1.Connected:= True;
  SQLTransaction1.Active:= True;
  SQLQuery1.Open;
  SQLQuery2.Open;
end;
```

Чтобы настроить столбцы в DBGrid, надо временно включить `SQLite3Connection1` и `SQLQuery`:

`Connected = true` и `Active = true` соответственно.

При этом проверьте в SQLQuery запрос в свойстве SQL.

Столбцы с кодами (ID) и ссылками защитим от редактирования (`ReadOnly=True`), так как значение ID будет определяться автоматически (`AutoIncrement`), а ссылки в записях будут подсчитываться программно:

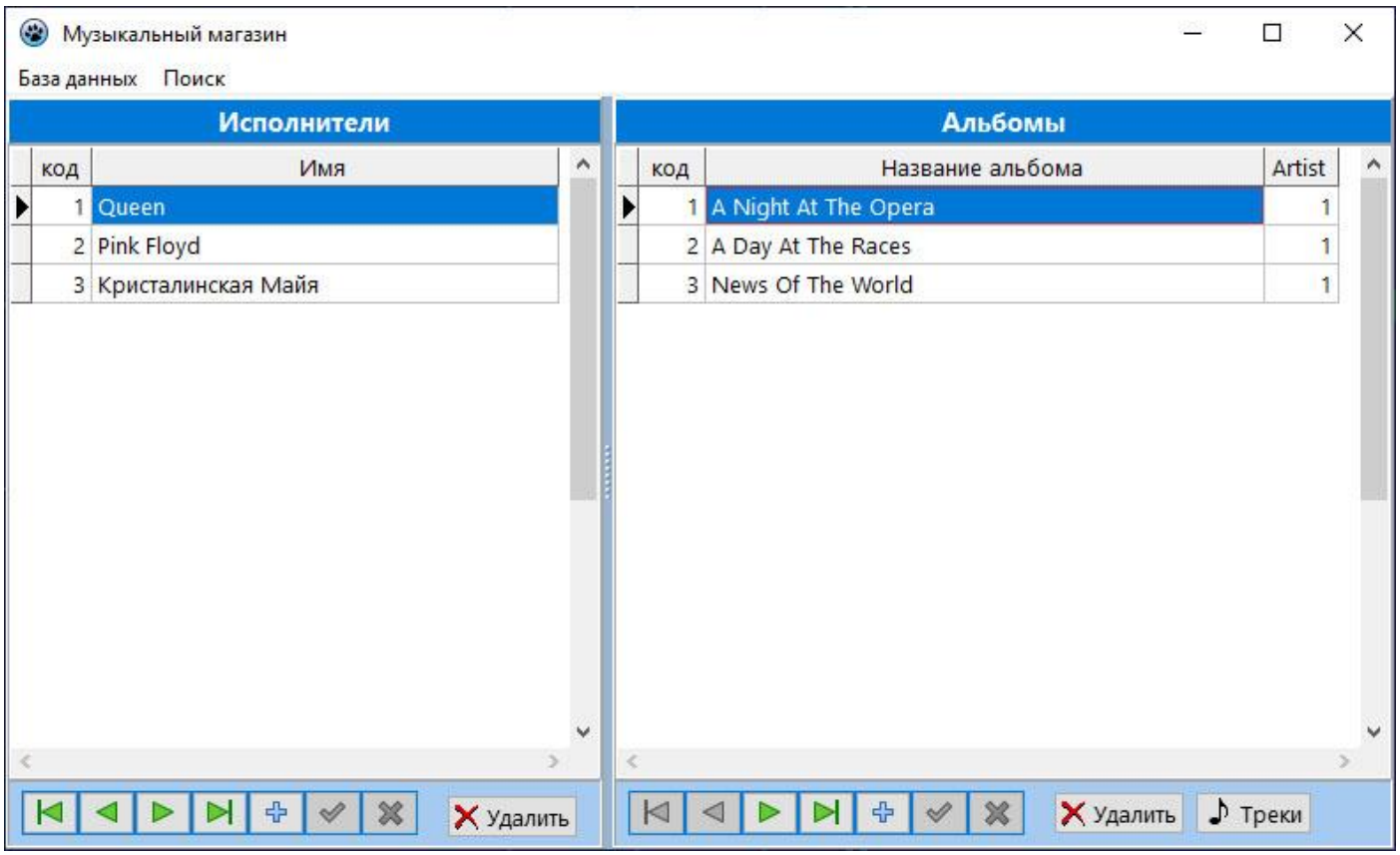
```
procedure TForm1.SQLQuery2BeforePost(DataSet: TDataSet);
begin
  if DataSet.FieldByName('Artist').IsNull then begin
     DataSet.FieldByName('Artist').Value:= SQLQuery1.Fields[0].Value;
  end;
end;
```

Поскольку альбомы вводятся только при сохранённой записи исполнителя, то надо бы в момент перехода с левого списка на правую панель автоматически сохранять запись исполнителя. Это сделаем в событии `DBGrid.OnExit` (аналогично и для DBGrid2):

```
procedure TForm1.DBGrid1Exit(Sender: TObject);
begin
  if (SQLQuery1.State in [dsEdit, dsInsert]) and SQLQuery1.Modified then
     SQLQuery1.Post;
  { понятие Modified в Lazarus отличается от того же в Delphi -
    там оно True может быть только в режиме редактирования,
    а в Lazarus остаётся True и после Post - со смыслом, что запись
    вообще-то была изменена. Нас интересует тот момент, когда запись
    изменена, но не сохранена  }
end;
```

```
procedure TForm1.DBGrid2Exit(Sender: TObject);
begin
  if (SQLQuery2.State in [dsEdit, dsInsert]) and SQLQuery2.Modified then
     SQLQuery2.Post;
end;
```

В результате можно получить следующую картину:



Для удаления записей в таблице исполнителей задействуем отдельную кнопку, так как требуется проверка на наличие ссылок из таблицы альбомов на эту запись:

```
procedure TForm1.BtnDelArtistClick(Sender: TObject);
begin
  if SQLQuery2.RecordCount > 0 then
     Application.MessageBox('Удалять запись нельзя - есть ссылки!',
         PChar(Caption), MB_ICONSTOP)    // MB_xxx  in LCLType
   else
     if Application.MessageBox('Удалить запись?', PChar(Caption),
       MB_ICONQUESTION + MB_YESNO) = mrYES then SQLQuery1.Delete
end;
```

Аналогично и для удаления альбома:

```
procedure TForm1.BtnDelAlbumClick(Sender: TObject);
var
  k : integer;
begin
  if SQLQuery2.RecordCount = 0 then
    exit; // если таблица ещё пустая
  SQLQuery3.Close;
  SQLQuery3.SQL.Clear;
  SQLQuery3.SQL.Add('SELECT COUNT(*) FROM Songs');
  SQLQuery3.SQL.Add('WHERE Album = ' + SQLQuery2.FieldByName('ID').AsString);
  SQLQuery3.Open;
  k:= SQLQuery3.Fields[0].AsInteger;
  SQLQuery3.Close;
  if k > 0 then
     Application.MessageBox('Удалять запись нельзя - есть ссылки!',
         PChar(Caption), MB_ICONSTOP)
   else
     if Application.MessageBox('Удалить запись?', PChar(Caption),
       MB_ICONQUESTION + MB_YESNO) = mrYES then SQLQuery2.Delete
end;
```

Вызов списка треков текущего диска:

```
procedure TForm1.BtnTracksClick(Sender: TObject);
begin
  FrmSongs.ShowModal
end;
```

Этот обработчик можно создать заранее, но пока модуля с формой FrmSongs нет, содержимое процедуры надо закомментировать.

При закрытии формы:

```
procedure TForm1.FormClose(Sender: TObject; var CloseAction: TCloseAction);
begin
  Panel5.SetFocus;
  SQLQuery2.Close;
  SQLQuery1.Close;
  SQLite3Connection1.Connected:= false;
  CloseAction:= caFree; // на главной форме только так или не указывать
end;
```

Дело в том, что если мы закрываем форму обычным способом (через так называемую иконку-кнопку SystemMenu), то можем не сохранить последние изменения в таблице, даже если выходим при этом из DBGrid и у нас есть обработчик `DBGrid.OnExit`. Для этого фокус курсора должен покинуть DBGrid, а иконка SystemMenu не имеет чувствительности курсора, поэтому его искусственно в этот момент направляем на Panel, которая имеет такое свойство, и тогда событие `DBGrid.OnExit` сработает. Конечно, если пользователь перед закрытием не забыл нажать на кнопку сохранения записи на Навигаторе, то всё сохранится и без этого. Обычно записи автоматически

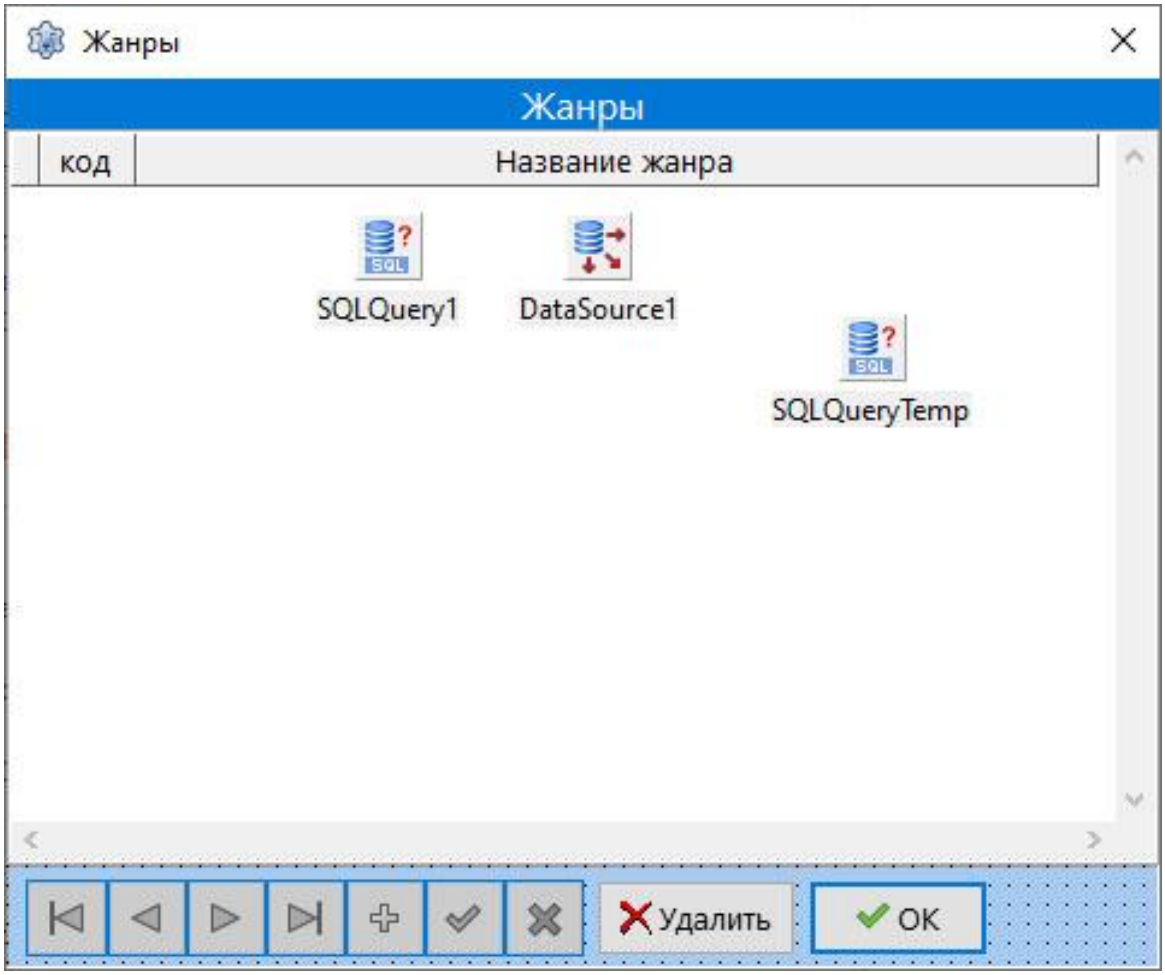
сохраняются при переходе от одной записи к другой, и при вводе новой записи прежние изменения сохраняются. Поэтому пользователи привыкают, что можно обойтись и без кнопки сохранения. Так что эти дополнительные подстраховки в профессиональной разработке не лишние.

Чтобы в навигаторе работали подсказки, в Инспекторе объектов укажем:

```
DBNavigator.ShowHint = True
```

Жанры

Для работы с этой таблицей создадим новую форму.



Сохраним её под именем FrmGenres, а модуль – Unit_Genres.

Теперь надо сделать связь между двумя нашими модулями – через **uses**. Тогда появится возможность настроить SQLQuery1, указав ему в Инспекторе Объектов:

```
SQLQuery1.Database = Form1.SQLite3Connection1
SQLQueryTemp.Database = Form1.SQLite3Connection1
DataSource1.Dataset = SQLQuery1
DBGrid1.DataSource = DataSource1
```

В SQLQuery1 запрос в свойстве SQL:

```
SELECT * FROM Genres
```

У формы для жанров желательно сделать следующие настройки:

```
Position = poMainFormCenter
BordersIcon = [biSystemMenu],
```

т.е. оставить только `biSystemMenu = true`

Вызываться эта форма будет через соответствующее меню на главной форме:

```
procedure TForm1.MenuGenresClick(Sender: TObject);
begin
  FrmGenres.Tag:= 0;
  FrmGenres.ShowModal;
end;
```

Здесь для формы используем свободное свойство Tag. Через него будем в модуль Unit_Genres передавать режим работы формы списка жанров: при `Tag=0` вызывается список для редактирования жанров, а при `Tag=1` – список для выбора жанров в таблице треков (Songs). При `Tag=2` – список для выбора исполнителя при поиске в варианте «Жанр и исполнитель». Таблицы по структуре одинаковые, поэтому одну форму с DB-элементами можно применить для двух таблиц – при загрузке меняется только название таблицы.

Вся работа формы организуется в обработчике события `OnShow`:

```
implementation

{$R *.lfm}

uses Unit_Main, SearchGenre;

procedure TFrmGenres.FormShow(Sender: TObject);
begin
  SQLQuery1.Close;
  SQLQuery1.SQL.Clear;
  if Tag < 2 then begin
    Caption:= ' Жанры';
    SQLQuery1.SQL.Add('SELECT * FROM Genres');
  end
  else begin
    Caption:= ' Исполнители';
    SQLQuery1.SQL.Add('SELECT * FROM Artists');
  end;
  if Tag = 0 then begin  // работаем со списком жанров
    SQLQuery1.Options:= [sqoKeepOpenOnCommit, sqoAutoCommit,
      sqoAutoApplyUpdates]; // чтобы можно было редактировать в DBGrid
    DBGrid1.Options:= DBGrid1.Options - [dgRowSelect] + [dgEditing];
    DBNavigator1.VisibleButtons:=
       [nbFirst, nbPrior, nbNext, nbLast, nbInsert, nbPost, nbCancel];
    BtnDelGenre.Visible:= True;
    BtnOK.Visible:= False;
  end
  else begin  { вызов списка для выбора жанра (из Unit_Track)
                или исполнителя (из SearchGenre) }
    DBGrid1.Options:= DBGrid1.Options + [dgRowSelect];
    DBNavigator1.VisibleButtons:= [nbFirst, nbPrior, nbNext, nbLast];
    BtnDelGenre.Visible:= False;
    BtnOK.Visible:= True;
  end;
  SQLQuery1.Open;
  if (Tag = 1) and (Genre > 0) then
    SQLQuery1.Locate('ID', Genre, []);
  if (Tag = 2) and (Artist > 0) then
    SQLQuery1.Locate('ID', Artist, []);
end;
```

Добавим сюда такой же обработчик события `DBGrid1.OnExit`, что и на главной форме:

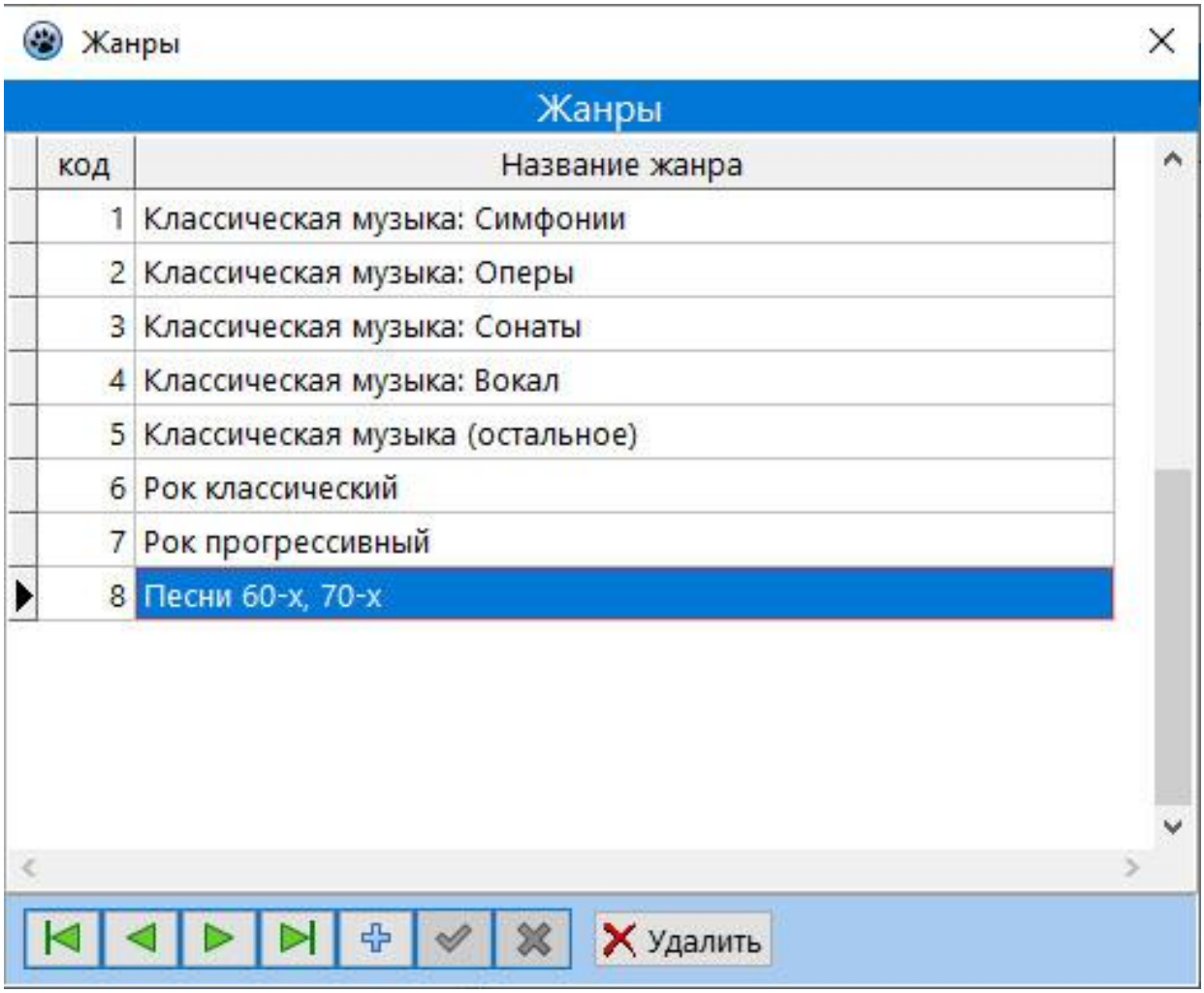
```
procedure TFrmGenres.DBGrid1Exit(Sender: TObject);
begin
  if (SQLQuery1.State in [dsEdit, dsInsert]) and SQLQuery1.Modified then
     SQLQuery1.Post;
end;
```

Удаление жанра – с учётом наличия ссылок в треках:

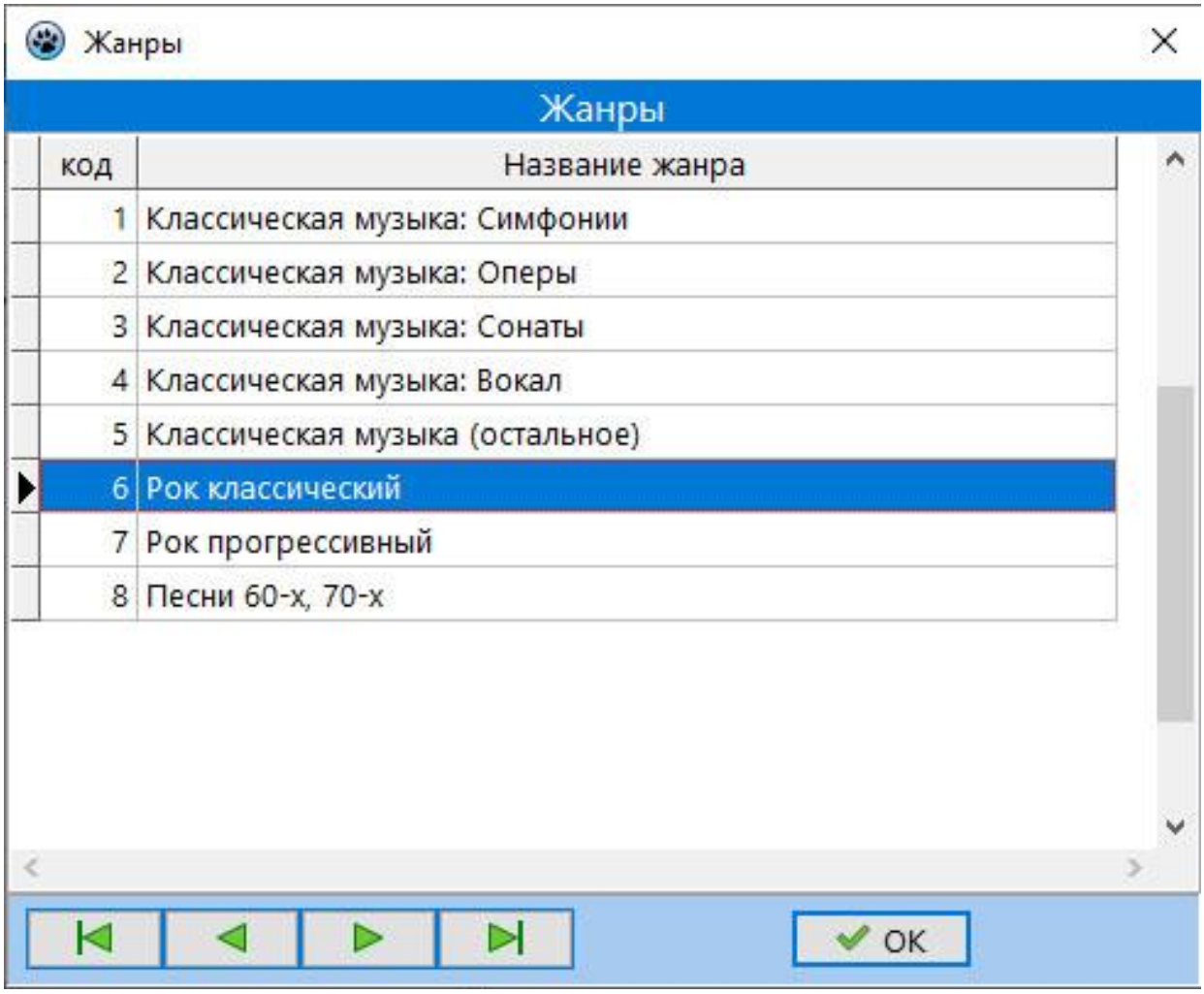
```
procedure TFrmGenres.BtnDelGenreClick(Sender: TObject);
var
  k : integer;
begin
  if SQLQuery1.RecordCount = 0 then
    exit; // если таблица ещё пустая
  SQLQueryTemp.Close;
  SQLQueryTemp.SQL.Clear;
  SQLQueryTemp.SQL.Add('SELECT COUNT(*) FROM Songs');
  SQLQueryTemp.SQL.Add('WHERE Genre = ' + SQLQuery1.FieldByName('ID').AsString);
  SQLQueryTemp.Open;
  k:= SQLQueryTemp.Fields[0].AsInteger;
  SQLQueryTemp.Close;
  if k > 0 then
     Application.MessageBox('Удалять запись нельзя - есть ссылки!',
         PChar(Caption), MB_ICONSTOP)
   else
     if Application.MessageBox('Удалить запись?', PChar(Caption),
       MB_ICONQUESTION + MB_YESNO) = mrYES then SQLQuery1.Delete
end;
```

Результат работы с таблицей жанров.

1) В режиме корректировки списка жанров:



2) В режиме вызова списка для выбора жанра:



Для кнопки OK обработчик следующий:

```
procedure TFrmGenres.BtnOKClick(Sender: TObject);
begin
  if Tag < 2 then begin
    Genre:= SQLQuery1.Fields[0].AsInteger;
    GName:= SQLQuery1.Fields[1].AsString;
  end
  else begin
    Artist:= SQLQuery1.Fields[0].AsInteger;
    ArtistName:= SQLQuery1.Fields[1].AsString;
  end;
end;
```

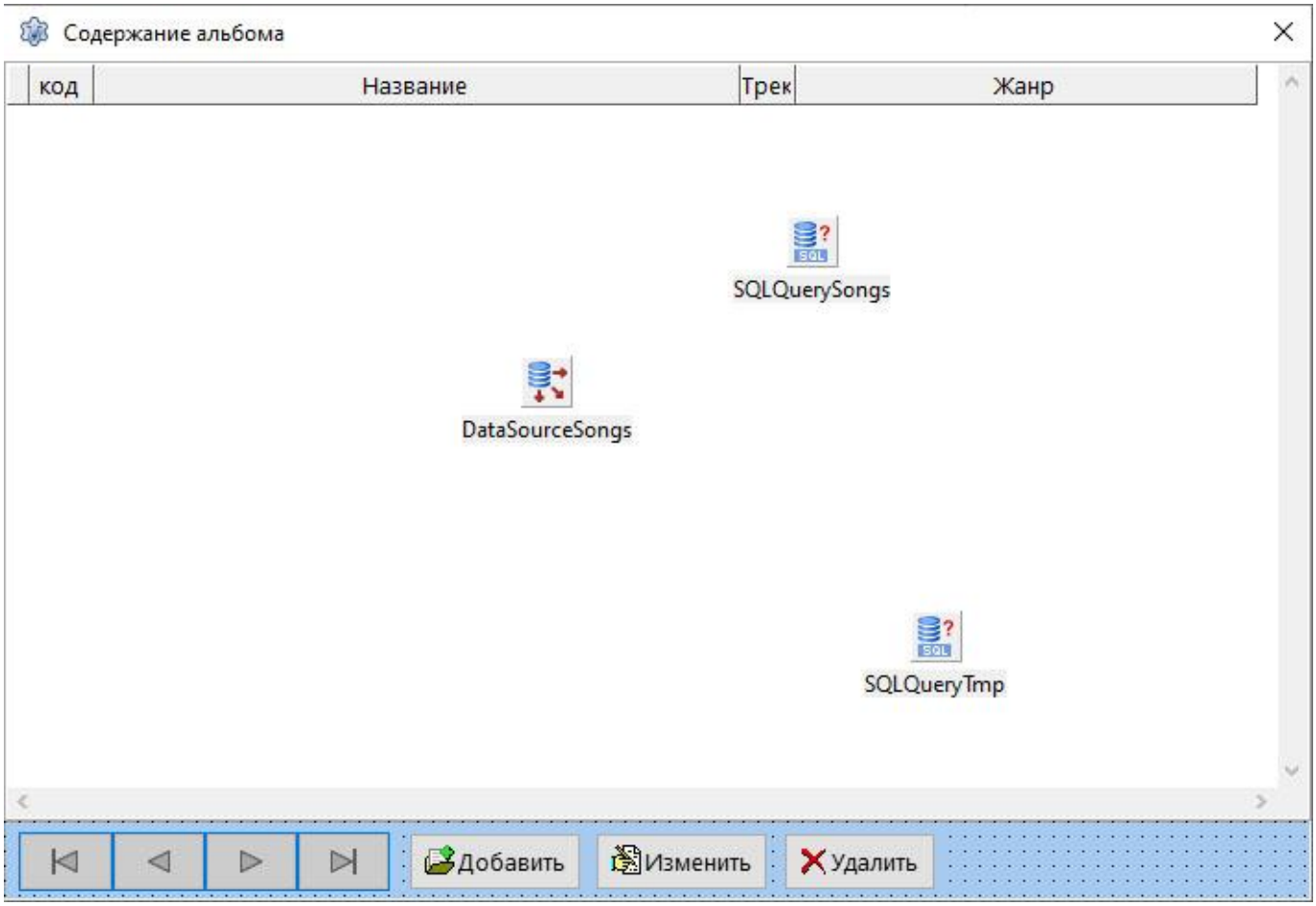
Здесь `Genre` и `GName` – общие переменные с будущей формой FrmTrack, т.е. их надо объявить глобальными либо в этом модуле, либо в Unit_Track. Они будут передаваться в модуль Unit_Track (будет описан ниже). Аналогично – и с `Artist`, `ArtistName`.

Добавим ещё обработку закрытия формы:

```
procedure TFrmGenres.FormClose(Sender: TObject; var CloseAction: TCloseAction);
begin
  Panel1.SetFocus;
  SQLQuery1.Close;
  CloseAction:= caHide;
end;
```

Содержание альбома

Содержание альбома – это список треков альбома, который вызывается кнопкой «Треки» на главной форме. Для этого создадим новую форму:



Соответствующий модуль назовём Unit_Songs, а форму – FrmSongs.

Естественно, следует подключить к модулю два созданных ранее и один будущий модуль. И объявим глобальную для модуля переменную `Album` (см. фрагмент кода ниже).

```
implementation
{$R *.lfm}

uses Unit_Main, Unit_Track, Unit_Genres;

var Album:integer;
```

Соответственно, не забываем и в главном модуле сделать взаимное подключение в **uses**.

DB-компоненты настроим в Инспекторе объектов как обычно, в том числе:

```
SQLQuerySongs.Database = Form1.SQLite3Connection1
SQLQuerySongs.DataSource = Form1.DataSource2
SQLQueryTmp.Database = Form1.SQLite3Connection1
```

Для `SQLQuerySongs` следует написать SQL-запрос:

```
SELECT * FROM Songs
LEFT JOIN Genres ON Genre = Genres.ID
WHERE Album = :ID
```

После этого запроса у `SQLQuerySongs` следует проверить наличие параметра ID в Инспекторе объектов. И проверить его свойство `DataType = ftInteger`. Если параметр не появился, то его надо создать.

Список заполняется в момент открытия формы, т.е. в событии OnShow:

```
procedure TFrmSongs.FormShow(Sender: TObject);
begin
  SQLQuerySongs.Open;
  Album:= Form1.SQLQuery2.Fields[0].AsInteger;
  Caption:= ' Содержание альбома: ' + Form1.SQLQuery2.Fields[1].AsString
end;
```

Поскольку запрос сложный, то редактировать его в `SQLQuerySongs` не получится. Для манипуляций с содержанием альбома потребуется дополнительный компонент `SQLQueryTmp` и отдельная форма информации о треке, которую будем вызывать соответствующими кнопками «Добавить» и «Изменить».

```
procedure TFrmSongs.BtnAddClick(Sender: TObject);
var k:integer;
begin
  FrmTrack.Tag:= 0;
  k:= FrmTrack.ShowModal;
  if k <> 1 then exit;
  DataSourceSongs.DataSet.DisableControls; // замораживаем изменения в DBGrid
  SQLQuerySongs.Close;
  SQLQueryTmp.SQL.Clear;
  inc(ID_Song);
  SQLQueryTmp.SQL.Add('INSERT INTO Songs');
  SQLQueryTmp.SQL.Add(format('VALUES (%d, %d, ''%s'', %d, %d)',
      [ID_Song, Track, SName, Album, Genre]));
  SQLQueryTmp.ExecSQL;
  Forml.SQLTransactionl.Commit;
  SQLQuerySongs.Open;
  SQLQuerySongs.Last;
  DataSourceSongs.DataSet.EnableControls; // показываем изменения в DBGrid
end;

procedure TFrmSongs.BtnEditClick(Sender: TObject);
var k,ID:integer;
  s:string;
begin
  FrmTrack.Tag:= 1;
  k:= FrmTrack.ShowModal;
  if k <> 1 then exit;
  ID:= SQLQuerySongs.Fields[0].AsInteger;
  SQLQueryTmp.SQL.Clear;
  s:= format('SET Track=%d, NAME=''%s'', Genre=%d', [Track, SName, Genre]);
  SQLQueryTmp.SQL.Add('UPDATE Songs');
  SQLQueryTmp.SQL.Add(s);
  SQLQueryTmp.SQL.Add('WHERE ID = ' + IntToStr(ID));
  SQLQuerySongs.Close;
  SQLQueryTmp.ExecSQL;
  Forml.SQLTransactionl.Commit;
  SQLQuerySongs.Open;
  SQLQuerySongs.Locate('ID', ID, []);
end;
```

Здесь используется форма FrmTrack из модуля Unit_Track, но пока это ещё не создано, соответствующие строки надо бы закомментировать, чтобы Lazarus не отвлекался на ошибки.

При закрытии формы используем стандартные действия:

```
procedure TFrmSongs.FormClose(Sender: TObject; var CloseAction: TCloseAction);
begin
  SQLQuerySongs.Close;
  CloseAction:= caHide;
end;
```

Рассмотрим подробнее процесс удаления записи. Удалять трек будем через SQL-запрос в `SQLQueryTmp`. Но мы для наглядности используем DBGrid, который связан не с `SQLQueryTmp`, а с `SQLQuerySongs`. Поэтому

придётся после удаления перезагружать `SQLQuerySongs` и желательно как-то курсор установить на место старой записи, как и делается в приличных программах. То есть, перед удалением надо выяснить ID соседней записи, как перед удаляемой записью, так и после неё, с учётом того, что удаляемая запись может быть и крайней в списке. Поэтому запоминаем ID удаляемой записи и соседней (объявим её как ID1).

Итак, для удаления трека используем следующую процедуру:

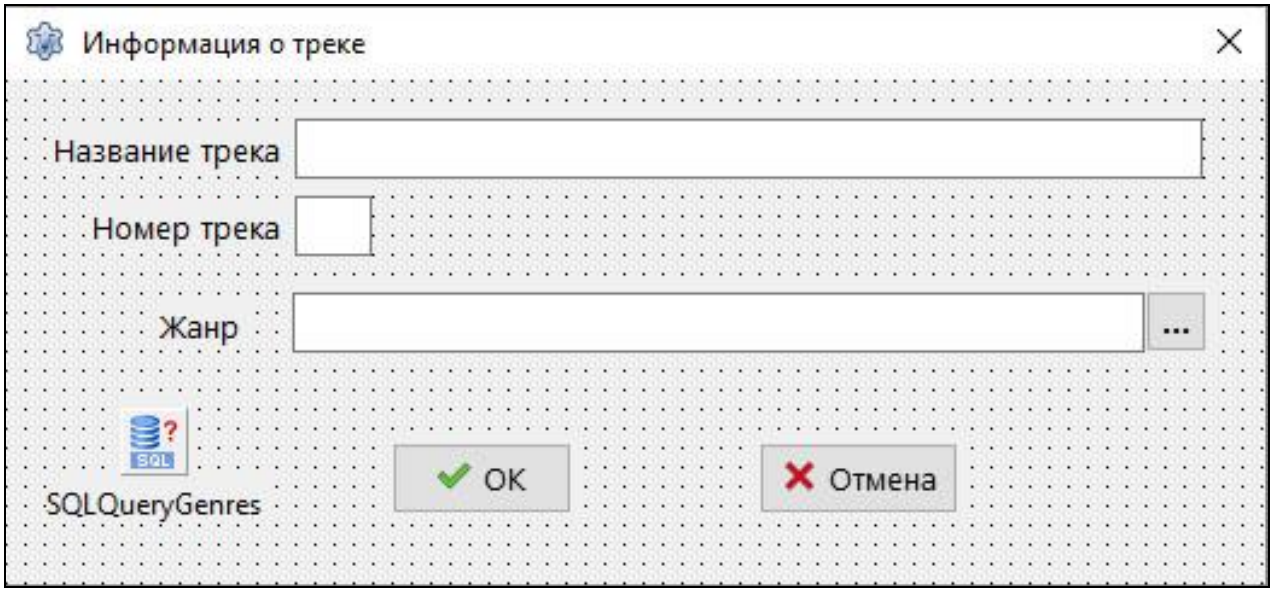
```
procedure TFrmSongs.BtnDelClick(Sender: TObject);
var ID,ID1:integer;
begin
  if SQLQuerySongs.RecordCount > 0 then
    if Application.MessageBox('Удалить запись?', PChar(Caption),
       MB_ICONQUESTION + MB_YESNO) = mrYES then
    begin
      ID:= SQLQuerySongs.Fields[0].AsInteger;
      DataSourceSongs.DataSet.DisableControls;
      SQLQuerySongs.Next;
      if not SQLQuerySongs.EOF then
        ID1:= SQLQuerySongs.Fields[0].AsInteger
      else begin
        SQLQuerySongs.Prior;
        ID1:= SQLQuerySongs.Fields[0].AsInteger
      end;
      SQLQueryTmp.SQL.Clear;
      SQLQueryTmp.SQL.Add('DELETE FROM Songs');
      SQLQueryTmp.SQL.Add('WHERE ID = ' + IntToStr(ID));
      SQLQuerySongs.Close;
      SQLQueryTmp.ExecSQL;
      Form1.SQLTransaction1.Commit;
      SQLQuerySongs.Open;
      SQLQuerySongs.Locate('ID', ID1, []);
      DataSourceSongs.DataSet.EnableControls;
    end;
end;
```

Здесь также используем

```
DataSourceSongs.DataSet.DisableControls;
DataSourceSongs.DataSet.EnableControls;
```

Информация о треке

Выше мы отмечалось, что потребуется и такая форма с модулем для работы с содержанием альбома. Определим её вид:



Первые два поля здесь – это компоненты LabeledEdit, третий – Edit, хотя ничего не мешает здесь задать его тоже через LabeledEdit. Рядом кнопка SpeedButton1 – для вызова списка жанров. Внизу две кнопки типа BitBtn. Первая настроена на тип «OK», а вторая – на «Cancel». Компонент SQLQueryGenres нужен только для поиска по ссылке в Songs названия жанра в таблице жанров.

Естественно, не забываем указывать в Инспекторе объектов

```
SQLQueryGenres.Database = Form1.SQLite3Connection1
```

Поля заполняются в событии формы `OnShow`. При этом используем свойство формы `Tag` для работы формы в двух вариантах – при добавке новой записи (`Tag=0`) и при корректировке записи (`Tag=1`).

```
procedure TFrmTrack.FormShow(Sender: TObject);
begin
  if Tag = 0 then begin     // добавка новой записи
    LabeledEdit1.Text:= '';
    LabeledEdit2.Text:= '';
    Edit1.Text:= '';
    Genre:= 0;
    Track:= 0;
  end
  else begin
    SName:= FrmSongs.SQLQuerySongs.FieldByName('Name').AsString;
    LabeledEdit1.Text:= SName;
    Track:= FrmSongs.SQLQuerySongs.FieldByName('Track').AsInteger;
    LabeledEdit2.Text:= IntToStr(Track);
    Genre:= FrmSongs.SQLQuerySongs.FieldByName('Genre').AsInteger;
    SQLQueryGenres.Close;
    SQLQueryGenres.SQL.Clear;
    SQLQueryGenres.SQL.Add('SELECT Name FROM Genres');
    SQLQueryGenres.SQL.Add('WHERE ID=' + IntToStr(Genre));
    SQLQueryGenres.Open;
    Edit1.Text:= SQLQueryGenres.Fields[0].AsString;
    SQLQueryGenres.Close;
  end;
  LabeledEdit1.SetFocus;
end;
```

Для вызова списка жанров используем кнопку SpeedButton1:

```
procedure TFrmTrack.SpeedButton1Click(Sender: TObject);
var k: integer;
begin
  FrmGenres.Tag:= 1;
  k:= FrmGenres.ShowModal;
  if k <> 1 then exit;
  Edit1.Text:= GName;
end;
```

По кнопке «ОК» форма должна закрыться, но при этом желательно проверить правильность заполнения данных, и если есть какие-то замечания, то отменить закрытие формы. Это возможно реализовать в событии формы `OnCloseQuery` (закрытие с запросом), которое происходит перед `OnClose`. Но в обработчике этого события есть возможность отменить Close – это логический параметр `CanClose`.

```
procedure TFrmTrack.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
const
  ErrTrack = #10'Укажите правильно номер трека';
var
  msg: string;
begin      // проверка полноты заполнения данных

  if (LabeledEdit1.Text = '') and
     (LabeledEdit2.Text = '') and
     (Edit1.Text = '') then
  begin
    Genre:= 0;
    Track:= 0;
    CanClose:= True;
    exit
  end;
  msg:= '';
  SName:= Trim(LabeledEdit1.Text);
  if SName = '' then
    msg:= 'Не задано название трека';
  try
    Track:= StrToInt(LabeledEdit2.Text);
    if Track = 0 then msg:= msg + ErrTrack;
  Except
    msg:= msg + ErrTrack;;
  end;
  if msg = '' then CanClose:= True
  else begin
    if msg[1] = #10 then Delete(msg,1,1);
    Application.MessageBox(PChar(msg), PChar(Caption), MB_ICONSTOP);
    CanClose:= False
  end;
end;
```

По кнопке «Отменить» следует отменить изменения, и это можно сделать очень просто:

```
procedure TFrmTrack.BitBtn2Click(Sender: TObject);
begin
  FormShow(nil);  // возврат к начальному состоянию
end;
```

После этого можно либо закрыть форму, либо работать с ней заново. Это можно настроить самостоятельно – как кому удобно.

Поиск исполнителя

Можно было ожидать, что подойдёт такой запрос:

```
SQLQuery1.SQL.Text:=
'SELECT * FROM Artists
   WHERE UPPER(NAME) LIKE ''%' + AnsiUpperCase(s) + '%'';
```

где `s` – строка поиска:

```
s:= InputBox(PChar(Caption), 'Имя исполнителя', '');
```

Но функция UPPER() не работает с русским текстом, поэтому, чтобы игнорировать регистр в русских буквах, приходится использовать событие `SQLQuery1.OnFilterRecord`:

```
procedure TForm1.SQLQuery1FilterRecord(DataSet: TDataSet;
    var Accept: Boolean);
begin
  Accept:= pos(SearchName,
    AnsiUpperCase(DataSet.FieldByName('Name').AsString)) > 0
end;
```

`SearchName` – глобальная для этого модуля переменная и задаётся в начале работы поиска в следующей процедуре:

```
procedure TForm1.MenuSearchArtistClick(Sender: TObject);
begin
  SearchName:= InputBox(PChar(Caption), 'Имя исполнителя', '');
  if SearchName = '' then begin
    SQLQuery1.Filtered:= False;
    exit
  end;
  SQLQuery1.Close;  // иначе повторно не сработает !
  SearchName:= AnsiUpperCase(SearchName);
  SQLQuery1.Filtered:= True;
  SQLQuery1.Open;
end;
```

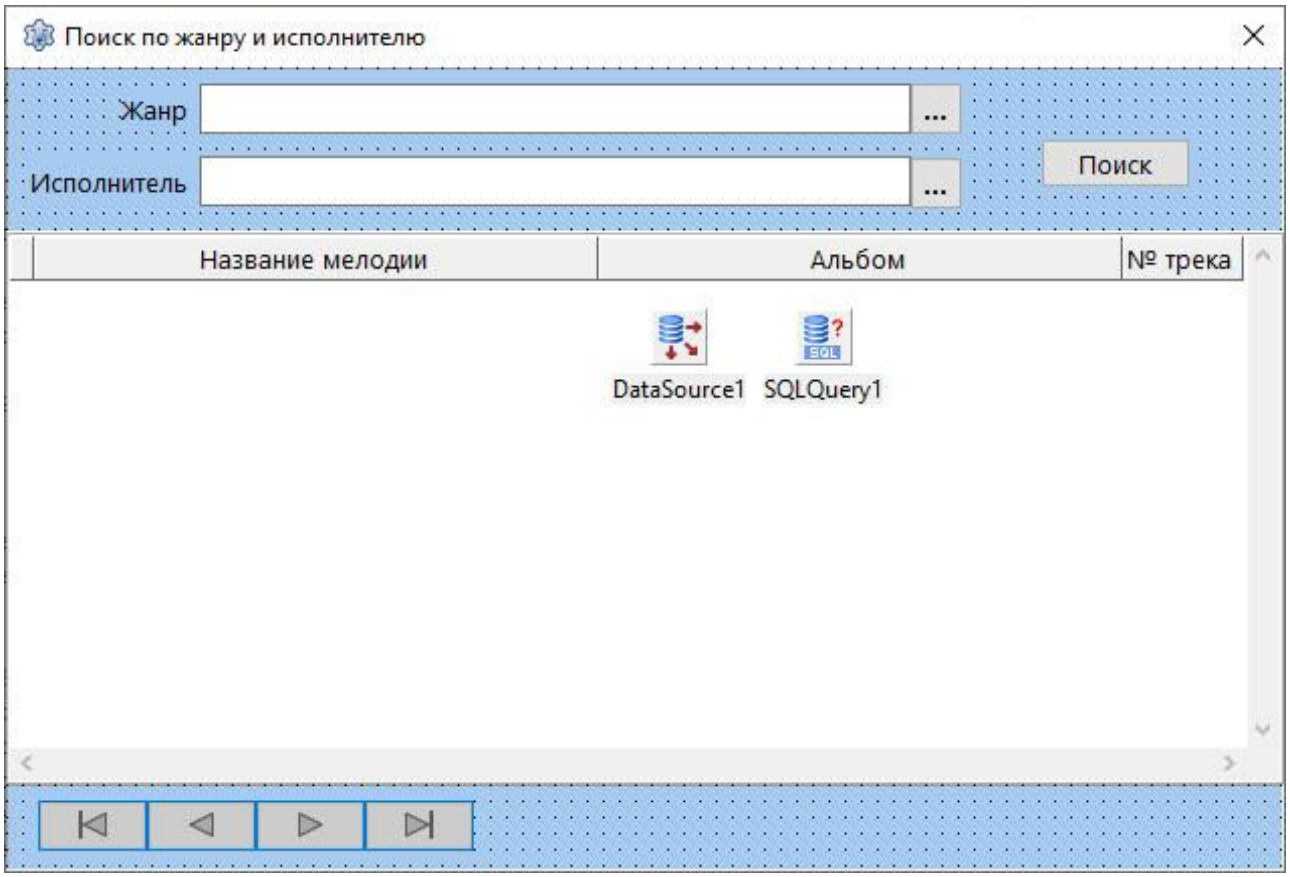
Здесь `SQLQuery1.Filtered:= True` включает работу события `SQLQuery1.OnFilterRecord`.

Результат будет выведен на главной форме.

Чтобы отменить фильтр, надо вызвать поиск с пустой строкой `SearchName`.

Поиск по жанру и исполнителю

Для реализации этого пункта создадим новую форму:



Соответствующий модуль назовём SearchGenre, а форму – FrmSearchGenre

Естественно, следует подключить к модулю два созданных ранее:

```
uses Unit_Main, Unit_Genres;
```

Соответственно, не забываем и в главном модуле сделать взаимное **uses**.

DB-компоненты настроим в Инспекторе объектов как обычно, в том числе:

```
SQLQuery1.Database = Form1.SQLite3Connection1
```

Для того, чтобы удобно было настраивать `DBGrid1`, зададим в `SQLQuery1` предварительный SQL-запрос:

```
SELECT S.Name, A.Name, Track FROM Songs S, Albums A
WHERE S.Album = A.ID
AND S.Genre = 8
AND A.Artist = 3
```

Числа 8 и 3 здесь значения не имеют – запрос формальный, только для того, чтобы проявились поля для работы со столбцами.

Здесь на верхней панели два компонента LabeledEdit и два SpeedButton.

SpeedButton1 – для вызова списка жанров:

```
procedure TFrmSearchGenre.SpeedButton1Click(Sender: TObject);
var k: integer;
begin
  FrmGenres.Tag:= 1; // вызов списка жанров
  k:= FrmGenres.ShowModal;
  if k <> 1 then exit;
  LabeledEdit1.Text:= GName;
end;
```

SpeedButton2 – для вызова списка исполнителей:

```
procedure TFrmSearchGenre.SpeedButton2Click(Sender: TObject);
var k: integer;
begin
  FrmGenres.Tag:= 2; // вызов списка исполнителей
  k:= FrmGenres.ShowModal;
  if k <> 1 then exit;
  LabeledEdit2.Text:= ArtistName;
end;
```

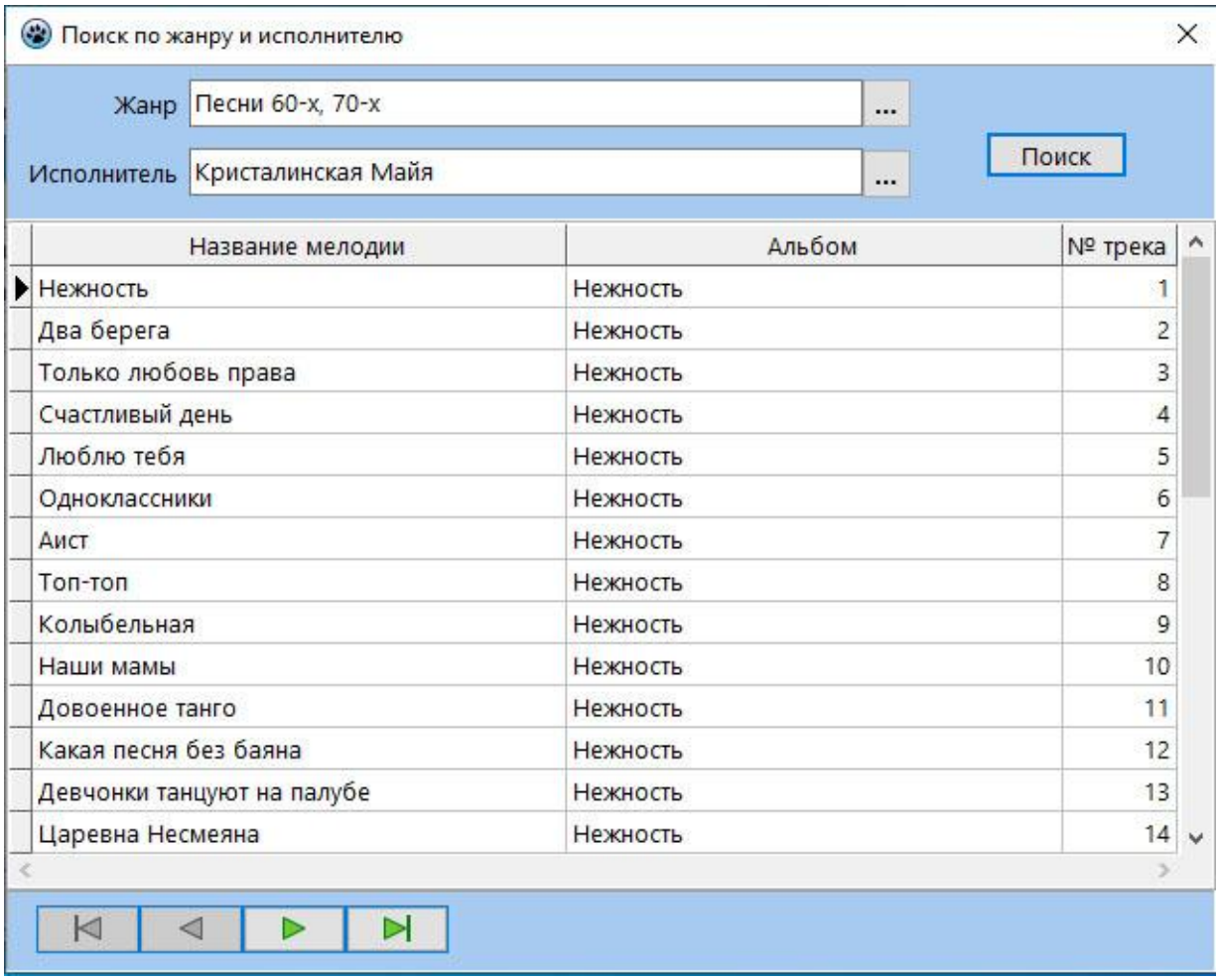
После того, как пользователь укажет жанр и/или исполнителя, кнопкой «Поиск» запускается соответствующий запрос:

```
procedure TFrmSearchGenre.Button1Click(Sender: TObject);
begin
  SQLQuery1.Close;
  if (Artist = 0) and (Genre = 0) then  exit;
  SQLQuery1.SQL.Clear;
  SQLQuery1.SQL.Add('SELECT S.Name, A.Name, Track FROM Songs S, Albums A');
  SQLQuery1.SQL.Add('WHERE S.Album = A.ID');
  if Genre > 0 then
    SQLQuery1.SQL.Add('AND S.Genre = ' + IntToStr(Genre));
  if Artist > 0 then
    SQLQuery1.SQL.Add('AND A.Artist = ' + IntToStr(Artist));
  SQLQuery1.Open;
end;
```

При закрытии формы следует закрыть и полученный набор данных:

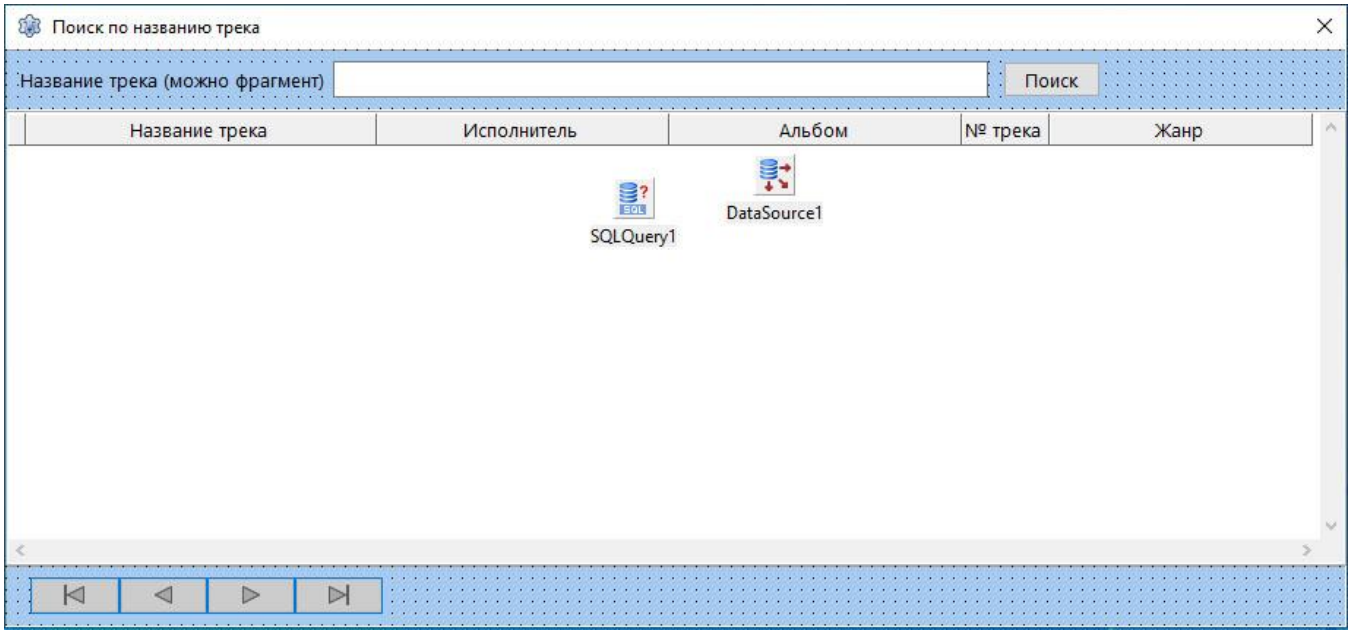
```
procedure TFrmSearchGenre.FormClose(Sender: TObject;
  var CloseAction: TCloseAction);
begin
  SQLQuery1.Close;
  CloseAction:= caHide;
end;
```

Результат работы поиска:



Поиск по названию трека

Для реализации этого пункта создадим новую форму:



Соответствующий модуль назовём SearchTrack, а форму – FrmSearchTrack

Естественно, следует подключить главный модуль:

```
uses Unit_Main;
```

DB-компоненты настроим в Инспекторе объектов как обычно.

В SQLQuery1 зададим сложный SQL-запрос (в Инспекторе объектов):

```
SELECT S.Name, Ar.Name, Al.Name, Track, G.Name
FROM Songs S, Albums Al, Artists Ar
LEFT JOIN Genres G On S.Genre = G.ID
WHERE S.Album = Al.ID
AND Al.Artist = Ar.ID
```

Строку поиска `SName` объявим глобальной для данного модуля (см. фрагмент кода ниже).

```
implementation

{$R *.lfm}
uses Unit_Main;

var SName:string;
```

Поскольку мы планируем работать с русским текстом в любом регистре и с фрагментом строки, то не обойтись без события SQLQuery1.OnFilterRecord:

```
procedure TFrmSearchTrack.SQLQuery1FilterRecord(DataSet: TDataSet;
  var Accept: Boolean);
begin
  Accept:=
    pos(SName, AnsiUpperCase(DataSet.Fields[0].AsString)) > 0;
end;
```

Строку поиска `SName` считываем с LabeledEdit1 в обработке события Button1.OnClick. Здесь же перезагружается список с новым фильтром:

```
procedure TFrmSearchTrack.Button1Click(Sender: TObject);
begin
  SName:= AnsiUpperCase(Trim(LabeledEdit1.Text));
  SQLQuery1.Close;  // надо закрыть !!
  SQLQuery1.Filtered:= True;
  SQLQuery1.Open;
end;
```

При закрытии формы – стандартные действия:

```
procedure TFrmSearchTrack.FormClose(Sender: TObject;
  var CloseAction: TCloseAction);
begin
  SQLQuery1.Close;
  CloseAction:= caHide;
end;
```

Результат работы поиска по названию трека: