Сидоренко В.А., старший преподаватель
ФГБОУ ВО «КалмГУ»,
Гузенко А.С., обучающийся 3 курса
ФМФИТ ФГБОУ ВО «КалмГУ»,
г. Элиста, Российская Федерация

Особенности и проблемы применения среды программирования Lazarus в разработке локальных таблиц формата dBase

Аннотация

В статье показаны некоторые проблемы использования файлов от СУБД dBase в интегрированной среде разработки Lazarus и показаны пути их решения. Рассмотрены практические примеры с пояснениями причин таких проблем, а также даны рекомендации по работе с указанным типом таблиц. Данная статья будет полезна для разработчиков локальных баз данных, использующих среду программирования Lazarus.

Ключевые слова: Lazarus и DBF, dBase в Lazarus, проблемы DBF с Lazarus в Linux, Lazarus и индексы в Dbf, не редактируется таблица DBF в Lazarus.

Lazarus – это открытая свободно распространяемая среда программной разработки на языке Free Pascal [1]. Она была задумана и разработана сообществом программистов для использования в различных операционных системах, включая Windows и Linux, как замена популярному инструменту Delphi. И по этим причинам мы используем Lazarus в своём учебном процессе. Так же, как и Delphi, здесь имеется почти такой же удобный интерфейс с комплектом готовых компонентов. Основной набор компонентов, наиболее часто используемых в разработке интерфейса программы, практически такой же, как и в Delphi 7, например. Но в части работы с базами данных наблюдаются существенные отличия от того же Delphi 7, наиболее известного среди программистов, использующих Object Pascal в любой версии. Кстати, Free Pascal в основном ничем не отличается от Object Pascal [2] – скорее всего, назван так из-за своего свободного распространения.

При изучении баз данных в Lazarus приходится довольно часто сталкиваться с различными сложностями. Там, где в Delphi всё проходит в соответствии с обычной логикой, в Lazarus могут встречаться необъяснимые недоразумения. И что самое печальное, это то, что из-за невысокой популярности использования Lazarus в разработке баз данных профессиональными программистами, довольно трудно найти литературу по изучению работы Lazarus с базами данных – любых типов. Поэтому приходится применять здесь всё то, что известно из Delphi и самому перебирать знакомые приёмы и методы, подбирая из них то, что работает в Lazarus. И иногда получается так, что какие-то довольно привычные объекты или методы заменены в Lazarus на свои, не

очень удобные или имеющие какие-то ограничения, недостатки. В этой статье рассмотрим несколько таких примеров.

Итак, здесь рассматривается только один тип баз данных – доставшийся нам от dBase. Это хорошо известный формат, расширение файлов – DBF. Несмотря на некоторое пренебрежение им серьёзными профессионалами, он жив и продолжает широко использоваться – в основном в качестве файлов справочников, распространяемых официальными органами. Иногда в каких-то организациях требуется передавать информацию в вышестоящие инстанции именно в таких форматах таблиц. Поэтому не стоит пренебрегать изучением этого типа таблиц.

В Delphi для dbf-файлов применялась пара компонентов из вкладки BDE: Table и Query. Второй позволял составлять SQL-запросы к этой базе данных, а первый – работать без них, но с массой удобных и надёжных методов. И в принципе удобно было использовать оба объекта, ориентируясь по ситуации: если целесообразно было выполнить SQL-запрос к базе данных, то применялся объект (компонент) Query, но если удобнее было использовать Table без запроса, то обходились этим объектом (компонентом).

В Lazarus для подобных целей остался только один компонент Dbf – подобный Table, а Query не стали приспосабливать к dBase. Но даже и его несколько подсократили в возможностях. Можно, конечно вспомнить, что и Delphi использовал Table с dBase с некоторыми усечениями в сравнении с Paradox, например. Но в Lazarus эта тенденция ещё заметнее.

Создание таблиц (dbf-файлов) – здесь всё примерно так же, как и в Delphi. То есть, применяются те же приёмы, как если бы это был Paradox, а не dBase. Поэтому размер для целого числа в таблице задаётся автоматически какой-то свой – 9. Аналогично, для вещественного числа тоже задаются свои размеры – общая длина 18 знаков и 8 знаков после десятичной точки. Кстати, это касается и Delphi – даже ещё в большей степени (там даже не понять какие размеры). Если вам важно задавать свои размеры полей для чисел, то вам для этого надо разработать свои объекты и методы, которые бы позволили создать таблицу со строго заданными характеристиками. И это вполне возможно, зная описание стандарта dbf-формата. Но в Lazarus при создании таблицы не получается задать какую-то другую кодировку текста – она в любом случае будет UTF-8.

Проблемы использования индексов в Dbf

Создание индексов. Метод создания такой же – через AddIndex, результат получается аналогичный, даже при создании составного индекса, с учётом правил dBase. Отличие в том, что это делается только на открытой таблице.

Использование индексов

Здесь различия существенные – вплоть до того, что практически невозможно использовать индексы в некоторых важных приёмах поиска записей. Особенно это касается составных индексов. Для простого индекса здесь только замена функции FindKey на SearchKey с изменением аргументов.

Если в FindKey в Delphi допускается использовать составной ключ – правда, только в Paradox – то здесь только одно значение ключевого поля и можно указать тип поиска TSearchKeyType – например, stEqual для точного совпадения. Это второй аргумент функции SearchKey.

С составным индексом всё сложнее, и тем более с русским текстом. Рассмотрим примеры.

```
with dbf1.FieldDefs do begin
  Clear;
  Add('Code', ftInteger);
  Add('Fam', ftString, 30);
  Add('Im', ftString, 30);
  Add('Ot', ftString, 30);
  Add('DR', ftDate);
  Add('Shtraf', ftFloat); // ftCurrency - нет в Lazarus
end;

dbf1.CreateTable;
dbf1.Open;
dbf1.AddIndex('iCode', 'Code', []);
dbf1.AddIndex('iFID', 'Fam+Im+DToS(DR)', [ixExpression]);
dbf1.AddIndex('iIm', 'Im', []);
```

Здесь создали таблицу и три индекса. Поработаем с такой таблицей, заполним её и посмотрим, как работают индексы. Переключение на какой-нибудь индекс означает сортировку по соответствующему ключу.

Form1

CODE	FAM	IM	OT	DR	SHTRAF
1	Петров	Иван	Кузьмич	13.01.1980	1200,50 р
2	Иванов	Пётр	Ильич	23.02.1978	1000,00 р
3	Сидоров	Семён	Семёнович	31.05.1981	2000,00 р
4	Stalone	Silvester		11.09.1948	
5	Canneyt	Mark		12.12.1960	
6	Gates	William		10.10.1955	

Рис.1. Таблица в исходном состоянии

Переключимся на индекс iIm, т.е. включим сортировку по именам:

Form1

CODE	FAM	IM	OT	DR	SHTRAF
3	Сидоров	Семён	Семёнович	31.05.1981	2000,00 р
1	Петров	Иван	Кузьмич	13.01.1980	1200,50 р
2	Иванов	Пётр	Ильич	23.02.1978	1000,00 р
5	Canneyt	Mark		12.12.1960	
4	Stalone	Silvester		11.09.1948	
6	Gates	William		10.10.1955	

Рис.2. Таблица в состоянии сортировки по именам

Это трудно назвать упорядочиванием по именам. Мы ожидаем, что сначала английские имена выстроятся по алфавиту, затем – русские. И если английские здесь придерживаются своего алфавита, то русские – нет.

Посмотрим, может быть, хоть поиск по имени работает.

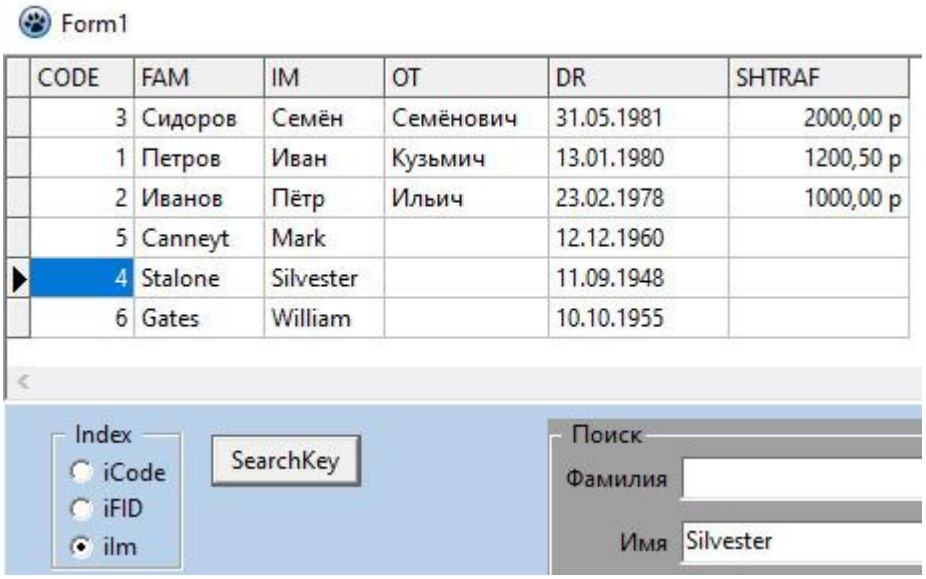

Рис.3. Таблица в состоянии поиска по именам (SearchKey)

Этого можно было и ожидать: по английским словам поиск работает, а по русским – нет никакой реакции, так что показать нечего.

Понятно, что дальнейшие эксперименты с индексацией и поискам по русским словам бесполезны.

Тогда посмотрим, может быть, с английским текстом удастся хоть что-то сделать. Итак, проверим поиск по составному индексу iFID для анлийского текста. Пока смотрим, что получается в состоянии этого индекса:

Рис.4. Таблица в состоянии сортировки по Фамилии, имени и дате рождения

Опять видим, что английские строки упорядочены, а русские – нет. Причём русские строки проникли внутрь английского списка.

Для поиска или выборки диапазона в Delphi для dbf-таблиц есть и работают некоторые методы компонента Table. А что же здесь? Как оказалось, что для dbf комбинацию SetKey; GotoKey и GotoNearest Lazarus не знает, то есть ни английский, ни русский текст не будет работать. Такая же участь постигла и комбинацию SetRangeStart; SetRangeEnd и ApplyRange для dbf – её

тоже Lazarus не знает. Это то, что работало в Delphi. Что же тогда в Lazarus придумать похожее? Остаётся метод

```
SetRange(LowRange:variant; HighRange:variant);
```

Оказалось, что его можно применить – но с некоторыми нюансами. Дело в том, что для задания границ диапазона такой способ (как в Delphi) тоже не работает:

```
v1:= VarArrayOf([Fam, Im, DR]);
```

Здесь надо прямо смотреть, как образована индексная таблица в mdx-файле. А она строилась совсем просто – по той комбинации, какая и была задана, т.е. ключевые слова образовывались как сумма: `Fam+Im+DToS(DR)`. И поскольку в файле в данном случае под фамилию и имя отводилось по 30 байт, то так и надо использовать в строке поиска 68 байт (30+30+8):

```
v1:= LongName(Fam1, 30) + LongName(Im1, 30) + DTOS(DR1);
v2:= LongName(Fam2, 30) + LongName(Im2, 30) + DTOS(DR2);
SetRange(v1, v2);
```

Здесь LongName – это функция, которая в нашем случае удлиняет строку до 30 байт. DTOS – функция преобразования даты в тот формат, который применяется внутри самого dbf-файла, т.е. YYYYMMDD:

```
function LongName(Name:string; len:integer):string;
var
  i,n : integer;
begin
  Result:= Name;
  n:= length(Name); // длина в байтах
  for i:=1 to len-n do Result:= Result + ' ';
end;

function DTOS(D:string):string;  // => дата в формате dBase
begin
  Result:= Copy(D, 7,4) + Copy(D, 4, 2) + Copy(D, 1, 2);
end;
```

В таком варианте получаем следующий результат:

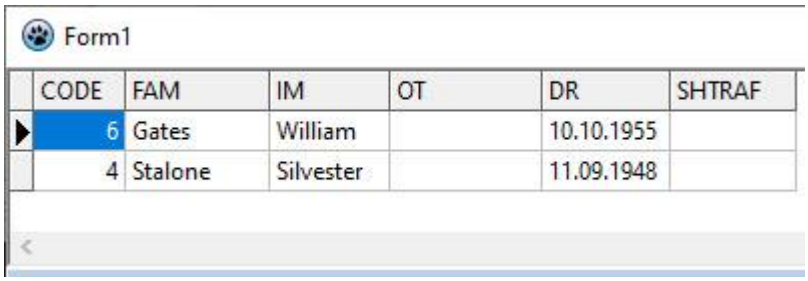

Рис.5. Таблица в результате применения метода `SetRange` (от Gates до Stalone)

Но вторая попытка, например, для фрагмента от Canneyt до Gates даёт нам следующую, в общем-то ожидаемую, картину:

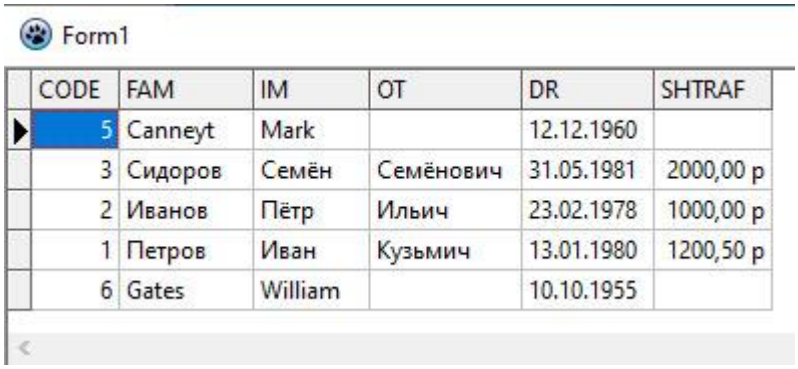

Рис.6. Таблица в результате применения метода `SetRange` (от Canneyt до Gates)

Поскольку список так и был неудачно упорядочен согласно индексу iFID, то все записи, какие были от Canneyt до Gates, также попали в эту выборку по диапазону. И мы видим, что это русский текст, который непредсказуемо попадает в любое место среди упорядоченных английских слов. Поэтому методом можно пользоваться, если текст только английский, точнее, если в ключах нет русского текста.

Что же тогда остаётся для русского текста – какие способы выборки или поиска? С индексами – ничего, а без индексов – то же самое, что и в Delphi: событие OnFilterRecord или свойство Filter для объекта Dbf. Функции Locate и Lookup тоже работают.

Не редактируется таблица DBF в Lazarus

Рассмотрим ещё один интересный вопрос. DBF-таблица создана в Lazarus под Windows, там с ней поработали, потом её перенесли в Linux – без каких-либо изменений. Программа работает, как и прежде, таблица загружается, никаких проблем не видно. Но попытки редактировать таблицу ни к чему не приводят. Если мы пытаемся это делать с помощью DBGrid, то никаких замечаний от Lazarus нет, а таблицу невозможно изменить. При этом в программе абсолютно ничего не меняли. Что же произошло? Где, в чём проблема, как понять и исправить? Оказывается, это довольно известный вопрос, в Интернете множество сообщений от удивлённых людей – наверное, тысячи – с одним и тем же: почему вдруг таблица перестала редактироваться. При различных вариантах попыток как-то внести изменения в таблицу иногда Lazarus выдаёт идею, что таблица в статусе Read Only. Но никто это свойство таблице не указывал, там по-прежнему ReadOnly=false. Вопрос этот задали тысячи раз, а ответа в Интернете практически не найти. По крайней мере, мы его там не видели.

Один способ решения – это удалить файл, созданный в Windows, и создать заново в Linux. И это работает. Но как быть, если в файле уже много информации и её не хочется терять? Но главное – в чём же причина такого странного поведения таблицы? И как её устранить? Оказалось, что причина в одной мелочи, в одном байте, который находится в заголовке (Head) таблицы и отвечает за номер языковой кодировки. Казалось бы, причём здесь язык? Но

вот такой коварный Lazarus. В Linux файл создаётся с кодировкой 88, а в Windows – почему-то с номером 38 (который Delphi причисляет к DOS866). И таблицу с этим номером Lazarus в Linux наделяет неким скрытым статусом ReadOnly. Многие другие номера кодовых таблиц Lazarus в Linux воспринимает нормально, а именно этот ему почему-то не нравится. И главное – он не сообщает, что кодировка ему не подходит (хотя ведь странно – всё же показывается правильно). В Windows же и 88 хорошо воспринимается – там по этому поводу никаких проблем. Получается, что проблема довольно неочевидна и понять такую нелогичную причину оказалось непросто. Поэтому и ответов в Интернете не видно.

Так что самое надёжное решение – это перед подключением таблицы проверить её номер кодовой странице и в случае, если он не 88, исправить на 88. При этом в Windows эта операция ничего не испортит, всё тоже будет нормально работать. Так что эту функцию можно включать всюду.

```
// изменить кодовую страницу в *.dbf
function SetLangCode(FileName:string; Code:byte):boolean;
var
  FS: TFileStream;
  b: byte;
begin
  Result:=true;
  b:=0;
  try
    FS:= TFileStream.Create(FileName, fmOpenReadWrite);
    FS.Seek(29, soFromBeginning);
    FS.ReadBuffer(b,1);
    if b <> Code then begin
      b:= Code;
      FS.Seek(-1, soFromCurrent);
      FS.WriteBuffer(b,1);
    end;
    FS.Free;
  except
    Result:= false;
  end;
end;
```

Довольно просто и понятно. Первый аргумент – это название файла, а второй – номер кодовой страницы.

Пример применения функции:

```
Dbf1.TableName:= FWriters;
Dbf2.TableName:= FWorks;
SetLangCode(FWriters, 88);
SetLangCode(FWorks, 88);
```

Список использованной литературы

1. Википедия: Lazarus https://ru.wikipedia.org/wiki/Lazarus
2. Алексеев Е. Р., Чеснокова О. В., Кучер Т. В. Free Pascal и Lazarus: Учебник по программированию. – М.: Альт Линукс, ДМК Пресс, 2010. 440 с.
3. Архангельский А.Я. Программирование в Delphi 7. – М.: ООО Бином-Пресс, 2003 г.–1152 с.
4. M. van Canneyt, M. Gartner, S.Heinig, F.Monteiro de Cavalho, I.Ouedraogo. Lazarus, the Complete Guide. Blaise Pascal Magazine, 2011. 735 с.

Сидоренко В.А.,
Гузенко А.С.

2024