2022.09.26

По следам статьи «Полноценный upper (или lower) в SQLite при работе с unicode» (автор, к сожалению, не известен)

Ссылка на статью: https://habr.com/ru/sandbox/98493/

Оглавление

1 Введение

При разработке ПО в среде Delphi зачастую возникает необходимость хранения данных в какой-либо локальной БД.

СУБД SQLite вполне подходит для этих целей, но в некоторых случаях проблема, обозначенная в указанной статье (см. выше), существенно усложняет ее использование.

Суть проблемы в том, что функции UPPER и LOWER в SQLite не умеет работать с кириллицей (под кириллицей в данном документе подразумевается русский алфавит).

В статье (см. выше) приведен конкретный, работающий вариант решения этой проблемы.

Воспользовавшись идеей и конкретным примером, приведенным в статье (см. выше), решил реализовать это применительно к среде Delphi (а если конкретно, то Delphi 10.2 Tokyo).

Позволю себе продублировать здесь фрагмент статьи (см. выше), где приведен соответствующий SQL-запрос (для **UPPER**):

```
select
  t.test_text
, upper(
    (
        WITH RECURSIVE
        under_name(test_text, char, level) as
            (select t.test_text, '', 0
            union
            select test_text,
coalesce(lu.u,substr(test_text,level,1)), under_name.level+1
            from under_name
            left join (
                select 'А' as u, 'а' as l union select 'Б' as u,
'б' as l union select 'В' as u, 'в' as l union select 'Г' as u, 'г'
as l union select 'Д' as u, 'д' as l union select 'Е' as u, 'е' as
l union select 'Ё' as u, 'ё' as l union select 'Ж' as u, 'ж' as l
union select 'З' as u, 'з' as l union select 'И' as u, 'и' as l
union select 'Й' as u, 'й' as l union select 'К' as u, 'к' as l
union select 'Л' as u, 'л' as l union select 'М' as u, 'м' as l
union select 'Н' as u, 'н' as l union select 'О' as u, 'о' as l
union select 'П' as u, 'п' as l union select 'Р' as u, 'р' as l
union select 'С' as u, 'с' as l union select 'Т' as u, 'т' as l
union select 'У' as u, 'у' as l union select 'Ф' as u, 'ф' as l
union select 'Х' as u, 'х' as l union select 'Ц' as u, 'ц' as l
union select 'Ч' as u, 'ч' as l union select 'Ш' as u, 'ш' as l
union select 'Щ' as u, 'щ' as l union select 'Ь' as u, 'ь' as l
union select 'Ы' as u, 'ы' as l union select 'Ъ' as u, 'ъ' as l
union select 'Э' as u, 'э' as l union select 'Ю' as u, 'ю' as l
union select 'Я' as u, 'я' as l
            ) lu on substr(test_text,level,1)=lu.l
```

```
            where level <= length(test_text)
        )
        select group_concat(char,'') from under_name
    )
) upper_text
from
(select 'Съешь ещё этих  мягких французских булок, да выпей же чаю.
Test.' test_text) t
```

В конечном итоге, хотелось получить что-то вроде такого (см. ниже).

Вместо этого (которое с кириллицей не работает):

```
S := S + 'upper(note) like '+#39+'%СЛОВО%'+#39;
```

Использовать вот это:

```
S := S + SQLite_Upper('note')+ ' like '+#39+'%СЛОВО%'+#39;
```

А еще лучше, что-то вроде этого:

```
procedure TfMain.n_ListWordsFilterCreateClick(Sender: TObject);
Var
  ListWords:TStrings; //Список слов (поисковых выражений)
  ListRes:TStrings;   //Результат
begin
  Application.ProcessMessages;
  ListWords:=TStringList.Create;
  ListRes:=TStringList.Create;
  TRY
    ListWords.Add('%ЦВЕТ%');
    ListWords.Add('%СКАЗ%');
    ListWords_Filter_Create('note', ListWords, ListRes, false);
    ShowMessage(ListRes.Text);
  FINALLY
    FreeAndNil(ListWords);
    FreeAndNil(ListRes);
  END;
end;
```

В результате, были разработаны несколько Delphi-функций (см. таблицу 1).

Таблица 1 – Список Delphi-функций

Функция	Назначение	Примечание
SQLite_Upper_Char	Формирование части SQL-запроса (UPPER) для одного символа кириллицы	Вспомогательная
SQLite_Upper	Формирование SQL-запроса (UPPER) для заданного поля таблицы БД	
WordFilter_Create	Формирование SQL-запроса (UPPER) для одного слова (поискового выражения) по значениям заданного поля таблицы БД	
ListWords_Filter_Create	Формирование SQL-запроса (UPPER) для списка заданных поисковых слов по значениям заданного поля таблицы БД	

Исходники этих функций приведены в этом документе.
А также, они есть в ZIP-архиве. Включая и тестовый пример.
Имя файла ZIP-архива: **`SQLite_Upper_Test_pas.zip`**

ВАЖНО!!!

1. Автор этого документа не ставил перед собой цель покорить этот мир красотой и оптимальностью примененных алгоритмов, а также – совершенством исходного кода.
2. Ясное дело, что эти исходники можно (да и нужно, пожалуй) использовать, как некий иллюстрирующий материал для своих собственных разработок.
3. Огромнейшее спасибо автору статьи (см. выше): https://habr.com/ru/sandbox/98493/

2 Функция SQLite_Upper_Char

Назначение этой вспомогательной функции: формирование части SQL-запроса для одной буквы кириллицы.

Например, для буквы **Ы**:

```
union select 'Ы' as u, 'ы' as l
```

2.1 Текст функции SQLite_Upper_Char

Входные параметры:

```
sChar:string; - соответствующий символ кириллицы;
sUnion:string; - должна быть пустая строка для буквы 'A' (для остальных:
'union');
```

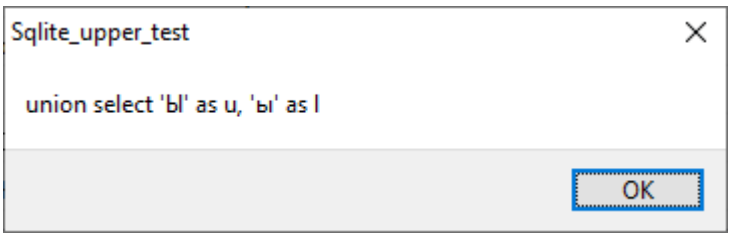
Возвращаемое значение: соответствующая часть SQL-запроса.

```
function SQLite_Upper_Char(sChar:string; sUnion:string='union'):string;
//Вспомогательная
begin
  Result:='';
  if length(sChar)=1 then begin
     sUnion:=trim(sUnion);
     Result:=sUnion+' select '+#39+AnsiUpperCase(sChar)+#39 + ' as u, ' +
#39+AnsiLowerCase(sChar)+#39+' as l';
  end;
end;
```

2.2 Пример

Сформировать часть SQL-запроса для буквы **Ы**.

```
ShowMessage(SQLite_Upper_Char('Ы','union'));
```



3 Функция SQLite_Upper

Назначение этой функции: формирование части SQL-запроса для одного поля таблицы БД.

Эта функция реализована в двух вариантах (overload):

- результат возвращается в виде списка TStrings;
- результат возвращается в виде строки String.

3.1 Текст функции SQLite_Upper (в варианте TStrings)

Входные параметры:

fn - имя (идентификатор) поля таблицы БД, где производится поиск;
ListRes – сформированная часть SQL-запроса;
tnAlias - алиас таблицы БД, к которой делается сформированный SQL-запрос;

Возвращаемое значение: кол-во строк в ListRes (или -1, если ошибка).

```
function SQLite_Upper(fn:string; ListRes:TStrings;
tnAlias:string='t9z'):integer;
//Формирование части SQL-запроса для одного поля таблицы БД.
//Идея взята отсюда:  https://habr.com/ru/sandbox/98493/
begin
  Result:=-1;
  if Assigned(ListRes) then begin
     ListRes.Clear;
     fn:=trim(fn);
     tnAlias:=trim(tnAlias);
     if tnAlias='' then tnAlias := 't9z';
     ListRes.Add('(');
     ListRes.Add('WITH RECURSIVE');
     ListRes.Add('under_name('+fn+', char, level) as');
     ListRes.Add('(');
     ListRes.Add('select '+tnAlias+'.'+fn+', '+#39+#39+', 0');
     ListRes.Add('union');
     ListRes.Add('select '+fn+', coalesce(lu.u,substr('+fn+',level,1)),
under_name.level+1');
     ListRes.Add('from under_name');
     ListRes.Add('left join ');
     ListRes.Add('(');
     ListRes.Add(SQLite_Upper_Char('A', ''));
     ListRes.Add(SQLite_Upper_Char('Б','union'));
     ListRes.Add(SQLite_Upper_Char('В','union'));
     ListRes.Add(SQLite_Upper_Char('Г','union'));
     ListRes.Add(SQLite_Upper_Char('Д','union'));
     ListRes.Add(SQLite_Upper_Char('Е','union'));
     ListRes.Add(SQLite_Upper_Char('Ё','union'));
     ListRes.Add(SQLite_Upper_Char('Ж','union'));
     ListRes.Add(SQLite_Upper_Char('З','union'));
     ListRes.Add(SQLite_Upper_Char('И','union'));
     ListRes.Add(SQLite_Upper_Char('Й','union'));
     ListRes.Add(SQLite_Upper_Char('К','union'));
     ListRes.Add(SQLite_Upper_Char('Л','union'));
     ListRes.Add(SQLite_Upper_Char('М','union'));
     ListRes.Add(SQLite_Upper_Char('Н','union'));
     ListRes.Add(SQLite_Upper_Char('О','union'));
     ListRes.Add(SQLite_Upper_Char('П','union'));
     ListRes.Add(SQLite_Upper_Char('Р','union'));
```

```
      ListRes.Add(SQLite_Upper_Char('С','union'));
      ListRes.Add(SQLite_Upper_Char('Т','union'));
      ListRes.Add(SQLite_Upper_Char('У','union'));
      ListRes.Add(SQLite_Upper_Char('Ф','union'));
      ListRes.Add(SQLite_Upper_Char('Х','union'));
      ListRes.Add(SQLite_Upper_Char('Ц','union'));
      ListRes.Add(SQLite_Upper_Char('Ч','union'));
      ListRes.Add(SQLite_Upper_Char('Ш','union'));
      ListRes.Add(SQLite_Upper_Char('Щ','union'));
      ListRes.Add(SQLite_Upper_Char('Ъ','union'));
      ListRes.Add(SQLite_Upper_Char('Ы','union'));
      ListRes.Add(SQLite_Upper_Char('Ь','union'));
      ListRes.Add(SQLite_Upper_Char('Э','union'));
      ListRes.Add(SQLite_Upper_Char('Ю','union'));
      ListRes.Add(SQLite_Upper_Char('Я','union'));
      ListRes.Add(')');
      ListRes.Add('lu on substr('+fn+',level,1)=lu.l');
      ListRes.Add('where level <= length('+fn+')');
      ListRes.Add(')');
      ListRes.Add('select group_concat(char,'+#39+''+#39+') from
 under_name');
      ListRes.Add(')');
      Result:=ListRes.Count;
   end;
 end;
```

3.2 Текст функции SQLite_Upper (в варианте String)

Входные параметры:

fn - имя (идентификатор) поля таблицы БД, где производится поиск

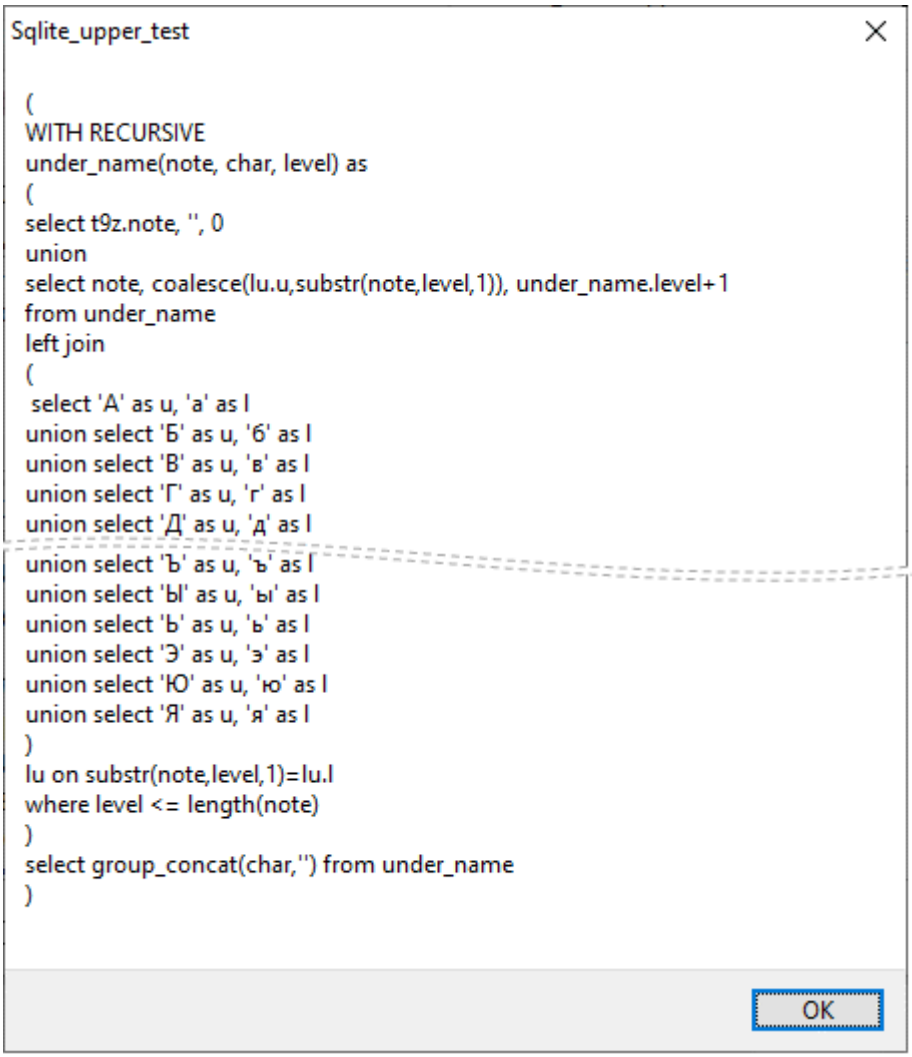
Возвращаемое значение: сформированная часть SQL-запроса (или пустая строка в случае ошибки).

```
 function SQLite_Upper(fn:string; tnAlias: string = 't9z'):string;
 //Формирование части SQL-запроса для одного поля таблицы БД.
 Var
   ListRes:TStrings;
 begin
   Result:='';
   ListRes := TStringList.Create;
   TRY
      if SQLite_Upper(fn, ListRes, tnAlias)>0 then begin
         Result:=trim(ListRes.Text);
      end;
   FINALLY
     FreeAndNil(ListRes);
   END;
 end;
```

3.3 Пример-1

Сформировать часть SQL-запроса для поля **note** таблицы БД с алиасом **t9z**. Результат должен быть сформирован в виде TStrings.

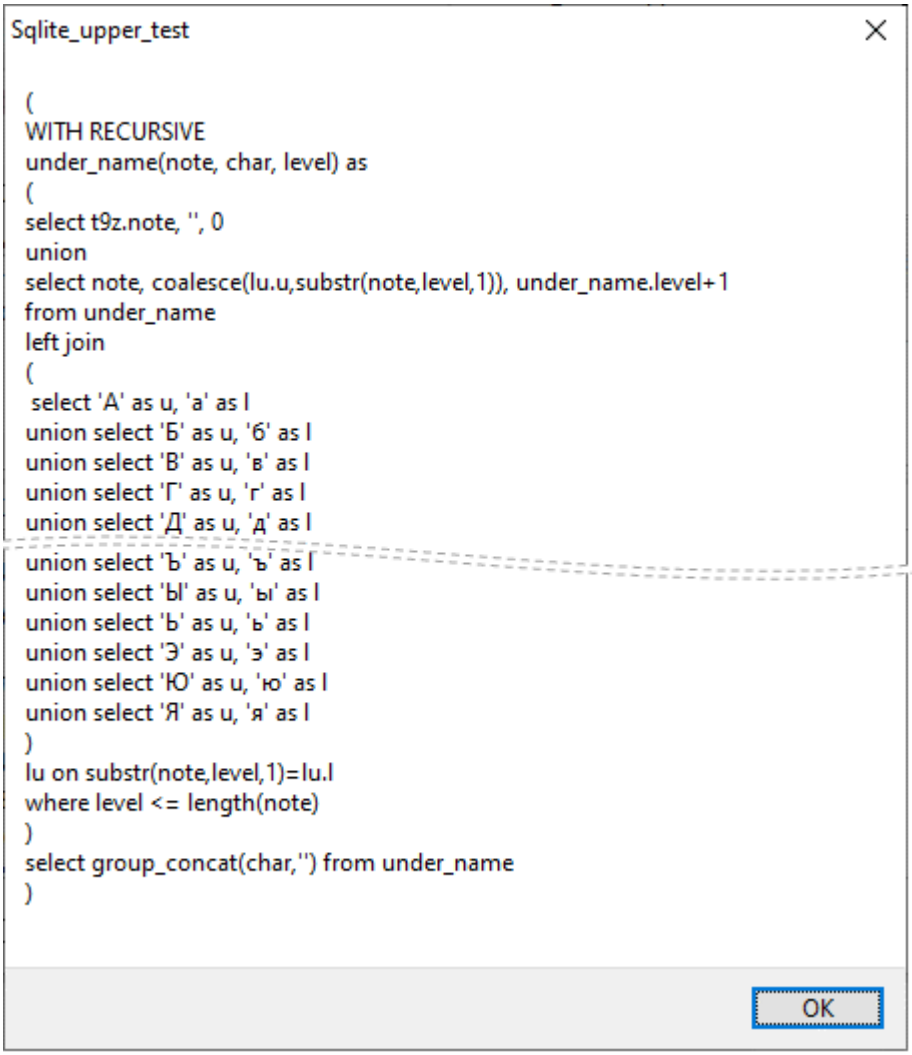
```
procedure TfMain.n_SQLiteUpperTStringsClick(Sender: TObject);
Var
  lSQL:TStrings;
begin
  Application.ProcessMessages;
  lSQL:=TStringList.Create;
  TRY
     SQLite_Upper('note', lSQL, 't9z');
     ShowMessage(lSQL.Text);
  FINALLY
     FreeAndNil(lSQL);
  END;
end;
```



3.4 Пример-2

Сформировать часть SQL-запроса для поля **note** таблицы БД с алиасом **t9z**.
Результат должен быть сформирован в виде String

```
procedure TfMain.n_SQLiteUpperStringClick(Sender: TObject);
Var
  Sx:string;
begin
  Sx:= SQLite_Upper('note', 't9z');
  ShowMessage(Sx);
end;
```



3.5 Пример-3

Сформировать часть SQL-запроса для поля **note** таблицы БД с алиасом **t9z**.

Применительно к **LIKE** для двух «слов» **%ЦВЕТ%** и **%СКАЗ%**, объединенных операцией **OR**.

Результат должен быть сформирован в виде String.

```
procedure TfMain.n_SQLiteUpperString2ORClick(Sender: TObject);
Var
  Sx:string;
begin
  Sx:='('+SQLite_Upper('note') + '  like
'+#39+'%ЦВЕТ%'+#39+')';
  Sx:=Sx+#10+#10+' OR '+#10+#10;
  Sx:=Sx+'('+SQLite_Upper('note') + '  like
'+#39+'%СКАЗ%'+#39+')';
  ShowMessage(Sx);
end;
```

Sqlite_upper_test

```
((
WITH RECURSIVE
under_name(note, char, level) as
(
select t9z.note, '', 0
union
select note, coalesce(lu.u,substr(note,level,1)), under_name.level+1
from under_name
left join
(
 select 'А' as u, 'а' as l
union select 'Б' as u, 'б' as l
union select 'В' as u, 'в' as l
union select 'Г' as u, 'г' as l
union select 'Ю' as u, 'ю' as l
union select 'Я' as u, 'я' as l
)
lu on substr(note,level,1)=lu.l
where level <= length(note)
)
select group_concat(char,'') from under_name
) like '%ЦВЕТ%')

OR

((
WITH RECURSIVE
under_name(note, char, level) as
(
select t9z.note, '', 0
union
select note, coalesce(lu.u,substr(note,level,1)), under_name.level+1
from under_name
left join
(
 select 'А' as u, 'а' as l
union select 'Б' as u, 'б' as l
union select 'В' as u, 'в' as l
union select 'Г' as u, 'г' as l
union select 'Ю' as u, 'ю' as l
union select 'Я' as u, 'я' as l
)
lu on substr(note,level,1)=lu.l
where level <= length(note)
)
select group_concat(char,'') from under_name
) like '%СКАЗ%')
```

4 Функция WordFilter_Create

Назначение этой функции: формирование SQL-запроса (UPPER) для одного слова (поискового выражения) по значениям заданного поля таблицы БД.

4.1 Текст функции WordFilter_Create

Входные параметры:

fn - имя (идентификатор) поля таблицы БД, где производится поиск;
sWord – слово (поисковое выражение);
NrVar – если =0, то «LIKE», если =1, то «=»;
tnAlias - алиас таблицы БД, к которой делается сформированный SQL-запрос;

Возвращаемое значение: сформированная часть SQL-запроса (или пустая строка в случае ошибки).

Предусмотрена обработка одинарных кавычек внутри значения sWord.

```
function WordFilter_Create(fn:string; sWord:string; NrVar:integer=0;
tnAlias: string = 't9z'):string;
//Сформировать фильтр для одного слова
begin
  Result:='';
  sWord:=trim(sWord);
  if sWord<>'' then begin
     fn:=trim(fn);
     if fn<>'' then begin
        Result:='(';
        sWord:=AnsiUpperCase(sWord);
        fn:=SQLite_Upper(fn, tnAlias); //вызов функции SQLite_Upper !!!
        sWord:=PrepareForSQL(sWord);   //обработка одинарных кавычек
        if (NrVar =0) then begin //LIKE
           Result:=Result+' '+fn + ' like '+#39+sWord+#39;
        end
        else begin // =
           Result:=Result+' '+fn + ' = '+#39+sWord+#39;
        end;
        Result:=Result+')';
     end;
  end;
end;
```

4.2 Пример-1

Сформировать часть SQL-запроса для поля **note** таблицы БД с алиасом **t9z**.
Слово (поисковое выражение): **%ЦВЕТ%** для «**LIKE**».

```
procedure TfMain.n_WordFilterCreateClick(Sender: TObject);
Var
  Sx:string;
begin
  Application.ProcessMessages;
  Sx:=WordFilter_Create('note', '%цВеТ%', 0); //Сформировать
для LIKE
  ShowMessage(Sx);
end;
```

```
Sqlite_upper_test


(( 
WITH RECURSIVE
under_name(note, char, level) as
(
select t9z.note, '', 0
union
select note, coalesce(lu.u,substr(note,level,1)), under_name.level+1
from under_name
left join
(
 select 'А' as u, 'а' as l
union select 'Б' as u, 'б' as l
union select 'В' as u, 'в' as l
union select 'Э' as u, 'э' as l
union select 'Ю' as u, 'ю' as l
union select 'Я' as u, 'я' as l
)
lu on substr(note,level,1)=lu.l
where level <= length(note)
)
select group_concat(char,'') from under_name
) like '%ЦВЕТ%')
```

4.3 Пример-2

Сформировать часть SQL-запроса для поля **note** таблицы БД с алиасом **t9z**.
Слово (поисковое выражение): **%ЦВЕТ%** для «=».

```
procedure TfMain.n_WordFilterCreateClick(Sender: TObject);
Var
  Sx:string;
begin
  Application.ProcessMessages;
  Sx:=WordFilter_Create('note', '%цВеТ%', 1); //Сформировать
для =
  ShowMessage(Sx);
end;
```

```
Sqlite_upper_test


( (
WITH RECURSIVE
under_name(note, char, level) as
(
select t9z.note, '', 0
union
select note, coalesce(lu.u,substr(note,level,1)), under_name.level+1
from under_name
left join
(
 select 'А' as u, 'а' as l
union select 'Б' as u, 'б' as l
union select 'В' as u, 'в' as l
union select 'Э' as u, 'э' as l
union select 'Ю' as u, 'ю' as l
union select 'Я' as u, 'я' as l
)
lu on substr(note,level,1)=lu.l
where level <= length(note)
)
select group_concat(char,'') from under_name
) = '%ЦВЕТ%')
```

5 Функция ListWords_Filter_Create

Назначение этой функции: формирование SQL-запроса (UPPER) для списка заданных поисковых слов по значениям заданного поля таблицы БД (LIKE)

5.1 Текст функции ListWords_Filter_Create

Входные параметры:

```
fn - имя (идентификатор) поля таблицы БД, где производится поиск
ListWords - список слов (поисковых выражений)
ListRes – сформированный текст SQL-запроса
nrAndOr - флаг. Если =TRUE, то все "слова" объединяются AND.
В противном случае - OR
tnAlias - алиас таблицы БД, к которой делается сформированный
SQL-запрос
```

Возвращаемое значение: `кол-во строк в ListRes.`

```
function ListWords_Filter_Create(fn:string; ListWords:TStrings;
ListRes:TStrings; nrAndOr:boolean=true; tnAlias: string = 't9z'):integer;
Var
  Sx, sWord:string;
  iWord:integer;
begin
  Result:=-1;
  if Assigned(ListRes) then begin
     ListRes.Clear;
     fn:=trim(fn);
     if fn<>'' then begin
        iWord:=-1;
        while iWord<(ListWords.Count-1) do
         begin
          iWord:=iWord+1;
          sWord:=trim(ListWords[iWord]);
          if sWord<>'' then begin
             Sx:=trim(WordFilter_Create(fn, sWord, 0)); //Используем LIKE
             if Sx<>'' then begin
                if nrAndOr then begin
                   if ListRes.Count>0 then ListRes.Add(' and ');
                end
                else begin
                   if ListRes.Count>0 then ListRes.Add(' or ');
                end;
                ListRes.Add(' '+Sx+' ');
             end;
          end;
        end;
     end;
     Result:=ListRes.Count
  end;
end;
```

5.2 Пример

Сформировать SQL-запроса для поля **note** таблицы БД с алиасом **t9z**.
Слова (поисковые выражения): **%ЦВЕТ%** и **%СКАЗ%** для «**LIKE**» (OR).

```
procedure TfMain.n_ListWordsFilterCreateClick(Sender: TObject);
Var
  ListWords:TStrings; //Список слов (поисковых выражений)
  ListRes:TStrings;   //Результат
begin
  Application.ProcessMessages;
  ListWords:=TStringList.Create;
  ListRes:=TStringList.Create;
  TRY
    ListWords.Add('%ЦВЕТ%');
    ListWords.Add('%СКАЗ%');
    ListWords_Filter_Create('note', ListWords, ListRes, false);
    ShowMessage(ListRes.Text);
  FINALLY
    FreeAndNil(ListWords);
    FreeAndNil(ListRes);
  END;
end;
```

Результат (в виде скрина) здесь не привожу («длинный» он)… ☺

6 Простая программка для тестирования разработанного функционала

Дополнительно к этому документу прилагаются исходные тексты как самих функций (проект в среде Delphi 10.2 Tokyo), так и программа тестирования (с исходниками, ясное дело).

Имя файла ZIP-архива: **SQLite_Upper_Test_pas.zip**

Скомпилированный EXE-модуль для тестирования (и пример БД) находится в папке: `\Win32\Debug\`

Имя EXE-модуля: `SQLite_Upper_Test.exe`.
Имя файла БД SQLite: `SQLite_Upper_Test.db`

Структура БД `SQLite_Upper_Test.db`:

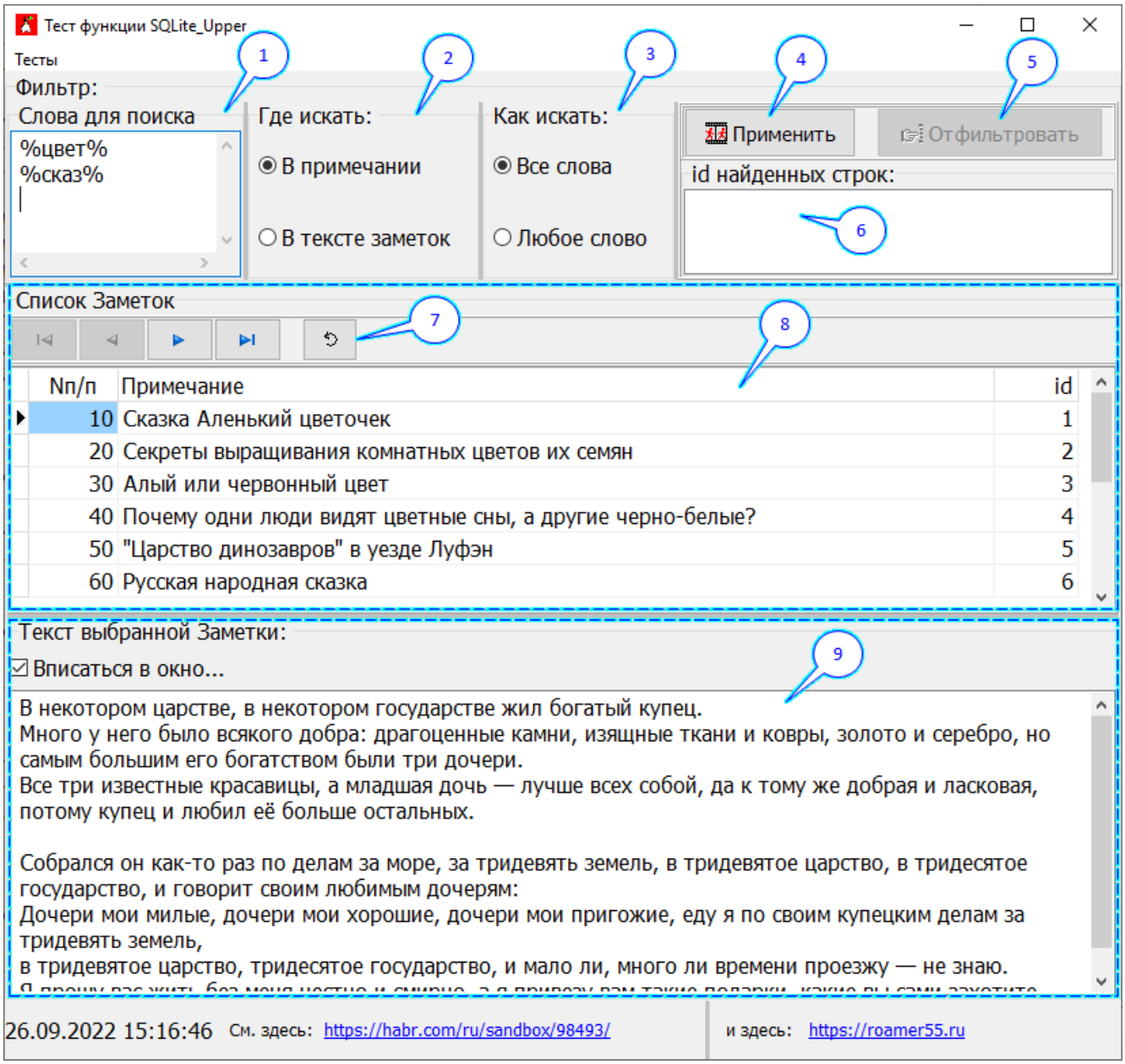
```
CREATE TABLE notes_list
(
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
npp integer,
note VARCHAR(512), -- примечание к заметке
info TEXT - текст заметки
);
CREATE INDEX npp_noteslist ON notes_list(npp ASC);
```

В таблице БД notes_list содержатся следующие строки:

notes_list

id	npp	note	info
1	10	Сказка Аленький цветочек	(WIDEMEMO)
2	20	Секреты выращивания комнатных цветов их семян	(WIDEMEMO)
3	30	Алый или червонный цвет	(WIDEMEMO)
4	40	Почему одни люди видят цветные сны, а другие черно-белые?	(WIDEMEMO)
5	50	"Царство динозавров" в уезде Луфэн	(WIDEMEMO)
6	60	Русская народная сказка	(WIDEMEMO)
7	70	Полноценный upper (или lower) в SQLite при работе с unicode	(WIDEMEMO)
8	80	SQLite - компактная встраиваемая СУБД	(WIDEMEMO)

На рисунке ниже приведен внешний вид главной формы программы.



Список выносок:

1 – элемент управления TMemo (список поисковых выражений);

2 – элемент управления TRadioGroup (выбор поля таблицы БД, где искать);

3 – элемент управления TRadioGroup (выбор операции AND или OR);

4 – кнопка, при нажатии на которую формируется соответствующий SQL-запрос, а также, производится выбора строк таблицы БД, соответствующих SQL-запросу (id найденных строк размещаются в TListBox, см. выноску-6);

5 – кнопка, при нажатии на которую производится фильтрация данных (см. выноску-8) в соответствии с SQL-запросом;

6 – список (TListBox), куда помещаются id найденных строк таблицы БД;

7 – кнопка, при нажатии на которую производится отображение всех строк в таблице БД (см. выноску-8);

8 – таблица (TDBGrid), где отображаются строки таблицы БД;

9 – содержимое поля `info` таблицы БД.

Можно, также, потестить каждую функцию по отдельности (см. примеры выше). Для этого используется главное меню приложения.