**Лабораторная работа №6**

# ПОИСК, ФИЛЬТРАЦИЯ ЗАПИСЕЙ В НД

**Цель работы:**

- изучение различных методов поиска, фильтрации и сортировок записей в наборах данных;
- применение изученных методов в приложении (на примере БД для Кафе).

## 1 Фильтрация записей в НД

Существует множество способов фильтрации записей в наборах данных.

Рассмотрим некоторые общие для наборов данных TTable и TQuery средства фильтрации записей в НД.

Помимо описываемых ниже средств, для фильтрации данных может применяться секция WHERE оператора SELECT языка SQL (см. лабораторную работу №5).

### 1.1 Свойство Filtered

**property bool Filtered = {read=FFiltered, write=SetFiltered, default=0};**

Свойство Filtered, установленное в true, инициирует фильтрацию, условие которой записано или в обработчике события OnFilterRecord, или содержится как строковое значение в свойстве Filter.

Если установлены разные условия фильтрации и в событии OnFilterRecord, и в свойстве Filter, выполняются оба.

Установка Filtered в false приведет к отмене фильтрации, условия которой указаны в событии OnFilterRecord или (и) свойстве Filter.

Отмена одного из этих условий фильтрации не приводит к отмене другою способа фильтрации.

### 1.2 Событие OnFilterRecord

Событие OnFilterRecord возникает, когда свойство Filtered устанавливается в true.

Обработчик события OnFilterRecord имеет два параметра: имя фильтруемого набора данных и bool &Accept, указывающий условия фильтрации записей в НД.

В отфильтрованный НД включаются только те записи, для которых параметр Accept имеет значение true.

В условие фильтрации могут входить любые поля НД, в том числе не входящие в текущий индекс, а также не входящие ни в один индекс. Возможность фильтрации НД по не индексным полям, а также полям, не входящим в текущий индекс, выгодно отличает способ фильтрации с использованием события OnFilterRecord и свойства Filtered .

Однако следует помнить о том, что при указании условий фильтрации НД в обработчике OnFilterRecord, в нем последовательно перебираются все записи таблицы БД при анализе их на предмет соответствия условию фильтрации.Это делает использование OnFilterRecord предпочтительным для небольших объемов записей и сильно ограничивает применение данного способа фильтрации при больших объемах данных.

Всякий раз, когда приложение обрабатывает событие OnFilterRecord, НД переводится из состояния dsBrowse в состояние dsFilter. Это предотвращает модификацию НД во время фильтрации. После завершения текущего вызова обработчика события OnFilterRecord, НД переводится в состояние dsBrowse.

Методы FindFirst, FindLast, FindNext, FindPrior также используют свойство OnFilterRecord, когда выполняют навигацию по НД.

### 1.3 Свойство Filter

**property AnsiString Filter = {read=FFilterText, write=SetFilterText};**

Свойство Filter позволяет указать условия фильтрации. В этом случае НД будет отфильтрован, как только его свойство Filtered станет равным true.

Синтаксис похож на синтаксис предложения WHERE SQL-оператора SELECT с тем исключением, что имена переменных программы указывать нельзя, можно указывать имена полей и литералы (явно заданные значения).

Можно применять операторы отношения:

| | |
|---|---|
| < | Меньше чем |
| > | Больше чем |
| >= | Больше или равно |
| <= | Меньше или равно |
| = | Равно |
| <> | Не равно |

а также использовать логические операторы AND, NOT и OR.

Строку фильтрации можно ввести во время выполнения.

Однако при этом нужно следить, чтобы введенная строка соответствовала требованиям, предъявляемым к синтаксису строки Filter.

Другим способом мог бы быть обработчик, считывающий значения фильтрации и преобразующий их к формату строки Filter.

### 1.4 Навигация в неотфильтрованном НД между записями, удовлетворяющими фильтру

**Методы FindFirst, FindLast, FindNext, FindPrior** позволяют перемещаться в не отфильтрованном НД (у которого Filtered = false) между записями удовлетворяющими условию фильтрации. Условие фильтрации задается событием OnFilterRecord или (и) свойством Filter. Действие данных методов таково: они кратковременно переводят НД в отфильтрованное состояние (Filtered - true) без визуализации этой фильтрации в TDBGrid или другом подобном компоненте, находят соответствующую запись и переводят НД в не отфильтрованное состояние (Filtered - false).

Если искомая запись найдена, данные методы возвращают true в противном случае - false.

### Задание 1. Фильтрация данных

*- Перейдите на форму fmList.*

*- С закладки Win32 разместите панель инструментов ToolBar. Свойство Name задайте ей в ToolBarFind.*

*- Разместите на панели инструментов ToolBarFind компоненты:*

*- с закладки Standart компонент CheckBox1. Свойство Name для него укажите в CheckBoxFilter, свойство Caption у CheckBoxFilter задайте в Фильтровать.*

*- с закладки Standart рядом с CheckBoxFilter разместите на панели ToolBarFind компонент Edit1. Свойство Name задайте ему в EditFilter, свойство Text очистите.*

*Будем проводить фильтрацию по нажатию галочки на CheckBoxFilter, по значениям, заданным в EditFilter.*

***Фильтрацию будем производить по выбранному пользователем в программе в DBGrid1 столбцу (свойство DBGrid1->SelectedField->FieldName возвращает строкой имя выбранного пользователем столбца в DBGrid1). Результат фильтрации отображается в DBGrid1.***

*Для фильтрации по нажатию галочки запишите в событии OnClick у CheckBoxFilter:*

```
if (CheckBoxFilter->Checked) // Нажата галочка
  {
     if (EditFilter->Text=="")
```

```
            {
                Application->MessageBox("Вы ничего не задали", "Внимание",
MB_ICONINFORMATION);
                CheckBoxFilter->Checked=false;
             }
            else
            {
                try
                {
                DataSource1->DataSet->Filter=DBGrid1->SelectedField-
>FieldName+"="'+EditFilter->Text+"'";
                ShowMessage(DataSource1->DataSet->Filter);
                DataSource1->DataSet->Filtered=true;
                }
                catch(...)
                {
                Application->MessageBox("Ошибка фильтрации
данных","Ошибка", MB_ICONERROR);
                CheckBoxFilter->Checked=false;
                EditFilter->Text="";
                }
            }
          }
           else
           {
           DataSource1->DataSet->Filtered=false;
           EditFilter->Text="";
           }
         if ((DataSource1->DataSet->RecordCount==0) && (DataSource1-
>DataSet->Filtered))
        {
            Application->MessageBox("Нет таких значений", "Сообщение",
MB_ICONINFORMATION);
            CheckBoxFilter->Checked=false;
            EditFilter->Text="";
        }
```

**Примечание:**

Пример фильтрации при помощи запроса см. в лабораторной работе №5.

## 2 Обзор методов поиска записей в НД

Существует множество способов поиска записей в наборах данных.

Рассмотрим некоторые общие для наборов данных TADOTable и TADOQuery средства поиска записей в НД. Кроме перечисленных для компонента TADOTable можно воспользоваться методами Find; GoToKey, FindNearest, GoToNearest.

### 2.1 Метод Locate

**function  Locate(const AnsiString KeyFields, const System::Variant &KeyValues, TLocateOptions Options);**

Метод Locate ищет первую запись, удовлетворяющую критерию поиска, и ели такая запись найдена, делает ее текущей. В этом случае в качестве результата возвращается true. Если поиск был неуспешен, возвращается false.

Параметры:

Список *KeyFields* указывает поле или несколько полей, по которым ведется поиск, в виде строкового выражения. В случае нескольких поисковых полей их названия разделяются точкой с запятой.

Критерии поиска задаются в вариантном массиве *Key Values* так, что i-е значение в Key Values ставится в соответствие i-му полю в KeyFields. В случае поиска по одному полю в Key Values указывается одно значение.

*Options* позволяет указать необязательные значения режимов поиска:

***loCaseInsensitive*** - поиск ведется без учета высоты букв, т.е. если в Key Values указано 'принтер', а в некоторой записи в данном поле встретилось 'Принтер' или 'ПРИНТЕР', запись считается удовлетворяющей условию поиска;

***loPartialKey*** - запись считается удовлетворяющей условию поиска, если она содержит часть поискового контекста; например, удовлетворяющими контексту «Ма» будут признаны записи с значениями в искомом поле «Машин», «Макаров» т.д.

Locate производит поиск по любому полю. Поле или поля, по которым производится поиск, могут не только не входить в текущий индексе, но и не быть индексными вообще.

В случае, если поля поиска входят в какой-либо индекс, Locate использует этот индекс при поиске. Если искомые поля входят в несколько индексов, то трудно сказать, какой из них будет использован. Соответственно, трудно предсказать, какая запись из множества записей, удовлетворяющих критерию поиска, будет сделана текущей - особенно в случае, если поиск ведется не по текущему индексу.

### 2.2 Использование методов FindFirst, FindLast, FindNext, FindPrior

Известно, что набор данных может быть отфильтрован с использованием свойства Filtered. Условие фильтрации задается свойством

Filter или описывается в обработчике события OnFilterRecord. Свойство Filtered указывает, выполнять ли фильтрацию (значение true) или нет (значение false). В этом случае в НД показываются все записи, а не только удовлетворяющие условию фильтрации.

В неотфильтрованном в данный момент НД можно обеспечить навигацию только между теми записями, которые удовлетворяют условию фильтрации (оно в текущий момент, когда свойство Filtered — false, не действует).

Для этой цели используются методы *FindFirst*, *FindLast*, *FindNext*, *FindPrior*.

Условие фильтрации можно сделать совпадающим с условием поиска, указанным в параметре KeyValues метода Locate. При этом поиск с помощью указанных методов имеет довольно большое преимущество перед поиском с помощью Locate: если в Locate можно указывать только значения, то в условии фильтрации можно указывать логические условия.

В случае, если искомая запись найдена, данные методы возвращают true, в противном случае - false.

**function FindFirst** - переходит на первую запись, удовлетворяющую фильтру;

**function FindLast** - переходит на последнюю запись, удовлетворяющую фильтру;

**function FindNext** - переходит на следующую запись, удовлетворяющую фильтру;

**function FindPrior** - переходит на предыдущую запись, удовлетворяющую фильтру;

**property Found** - возвращает true, если последнее обращение к одному из методов FindFirst, FindLast, FindNext, FindPrior привело к нахождению нужной записи.

### 2.3 Метод Lookup

**function Lookup(const AnsiString KeyFields, const Variant &KeyValues, const AnsiString ResultFields);**

Метод Lookup находит запись, удовлетворяющую условию, но не делает ее текущей, а возвращает значения некоторых полей этой записи. Тип результата - Variant или вариантный массив. Независимо от успеха поиска записи, указатель текущей записи в НД не изменятся.

Lookup осуществляет поиск только на точное соответствие критерия поиска и значения полей записи. Такой режим, как loPartialKey метода Locate (поиск по частичному соответствию значений), отсутствует.

Параметры:

В *KeyFields* указывается список полей, по которым необходимо осуществить поиск. При наличии в этом списке более чем одного поля, соседние поля разделяются точкой с запятой.

*KeyValues* указывает поисковые значения полей, список которых содержится в KeyFields. Если имеется несколько поисковых полей каждому i-му полю в списке KeyFields ставится в соответствие i-ое значение в списке KeyValues. При наличии одного поля, его поисковое значение можно указывать в качестве KeyValues непосредственно; в случае нескольких полей - их необходимо приводить к типу вариантного массива при помощи *VarArrayOf.*

В качестве поисковых полей можно указывать поля как входящие в какой-либо индекс, так и не входящие в него; тип текущего индекса не имеет значения.

В противном случае Lookup возвращает из этой записи значения полей, список которых указан в ResultFields. При этом размерность результата зависит от того, сколько результирующих полей указано в ResultFields:

- указано одно поле - результатом будет значение соответствующего типа или Null, если поле в найденной записи содержит пустое значение;
- указано несколько полей - результатом будет вариантный массив, число элементов в котором меньше или равно числу результирующих полей; меньше потому, что некоторые поля найденной записи могут содержать пустые значения.

## Задание 2. Поиск данных

### Задание 2.1

*- Перейдите на форму fmList.*

*- Рядом со строкой ввода EditFilterc с палитры Additional разместите кнопку BitBtn1 для поиска данных. Для нее укажите:*

*- свойство Name в btnFind;*

*- через свойство Glyph у btnFind в инспекторе объектов задайте соответствующую картинку для кнопки поиска;*

*- свойство Caption у btnFind задайте в Поиск.*

***Поиск будем производить по выбранному пользователем в программе в DBGrid1 столбцу (свойство DBGrid1->SelectedField->FieldName возвращает строкой имя выбранного пользователем столбца***

***в DBGrid1). Результат поиска отображается в DBGrid1. Условие для поиска задаются в строке вводе EditFilter.***

*На обработчик события нажатия мышкой OnClick на кнопке панели инструментов btnFind запишите:*

```
if (EditFilter->Text=="")
    {
        Application->MessageBox("Вы ничего не задали", "Внимание",
MB_ICONINFORMATION);
        exit;
    }
    else
    {
    try
    {
    if (DataSource1->DataSet->Locate(DBGrid1->SelectedField-
>FieldName, EditFilter->Text, TLocateOptions() << loCaseInsensitive <<
loPartialKey ))
    DBGrid1->SetFocus();
    else
    {
    ShowMessage("Значение "+EditFilter->Text+" в поле "+DBGrid1-
>SelectedField->DisplayLabel+" не найдено!");
    EditFilter->Text="";
    EditFilter->SetFocus();
    }
    }
    catch(...)
    {
    Application->MessageBox("Ошибка поиска","Ошибка",
MB_ICONERROR);
    EditFilter->Text="";
    }
    }
```

*Для отработки поиска по нажатию Enter на кнопке btnFind укажите для btnFind свойство Default в true.*

*Для того, чтобы строка ввода на момент показа формы для работы со справочниками была пустой допишите в событии OnShow (на момент показа формы) у fmList:*

```
EditFilter->Text="";
```

## *Задание 2.2*

*Будем для продаж осуществлять ввод данных о продажах через отдельную группу при нажатии кнопок Добавить и Редактировать.*

*- Перейдите на форму fmSales.*

*- На панель инструментов для продаж рядом с DBNavigatorSales разместите с палитры Additional кнопку BitBtn для добавления продаж. Для нее укажите:*

*- свойство Name в btnAdd;*

*- через свойство Glyph у btnAdd в инспекторе объектов задайте соответствующую картинку для кнопки добавить продажу;*

*- свойство Caption у btnAdd задайте в Добавить продажу.*

*- На панель инструментов для продаж рядом с btnAdd разместите с палитры Additional кнопку BitBtn для редактирования продаж. Для нее укажите:*

*- свойство Name в btnEdit;*

*- через свойство Glyph у btnEdit в инспекторе объектов задайте соответствующую картинку для кнопки редактировать продажу;*

*- свойство Caption у btnEdit задайте в Редактировать продажу.*

*- Добавьте на fmSales с палитры Standart компонент GroupBox. Укажите для него свойство Name в GroupBoxSales, свойство Caption в Данные о продаже.*

*- Разместите на GroupBoxSales с палитры Standart компонент Label. Свойство Name укажите в LabelNumber, свойство Caption укажите в Номер продажи.*

*- Разместите на GroupBoxSales рядом с LabelNumber с палитры DataCotrols компонент DBText для вывода номера продажи. Свойство Name для него укажите в DBTextNumber.*

*- Для DBTextNumber укажите свойство DataSourse в dsSales, свойство DataField в Номер_продажи.*

*- Разместите на GroupBoxSales с палитры Standart под LabelNumber компонент Label. Свойство Name укажите в LabelDate, свойство Caption укажите в Дата продажи.*

*- Разместите на GroupBoxSales рядом с LabelDate с палитры DataCotrols компонент DBEdit для даты продажи. Свойство Name для него укажите в DBEditDate.*

*- Для DBEditDate укажите свойство DataSourse в dsSales, свойство DataField в Дата.*

*- Разместите на GroupBoxSales с палитры Standart под LabelDate компонент Label. Свойство Name укажите в LabelSotrudnik, свойство Caption укажите в Сотрудник.*

*- Разместите на GroupBoxSales рядом с LabelSotrudnik с палитры DataCotrols компонент DBEdit для сотрудника. Свойство Name для него укажите в DBEditSotrudnik.*

*- Для DBEditSotrudnik укажите свойство DataSourse в dsSales, свойство DataField в Сотрудник.*

*- Разместите на GroupBoxSales с палитры Standart рядом с DBEditSotrudnik компонент Button. Свойство Name укажите в ButtonSotrudnik, свойство Caption укажите в Выбрать сотрудника.*

*- Разместите на GroupBoxSales с палитры Standart под ButtonSotrudnik компонент Button. Свойство Name укажите в ButtonBack, свойство Caption укажите в «Вернуться к списку продаж».*

*- Перейдите на форму fmList. Разместите на панели рядом с кнопкой поиска btnFind с палитры Additional разместите кнопку BitBtn для выбора сотрудников. Для нее укажите:*

- *свойство Name в btnVibor;*
- *через свойство Glyph у btnVibor в инспекторе объектов задайте соответствующую картинку для кнопки выбора сотрудника;*
- *свойство Caption у btnVibor задайте в Выбрать сотрудника;*
- *свойство Visible задайте в false;*
- *свойство ModalResult в mrOk.*

*- Перейдите на форму fmSales.*

*- В событии OnClick у btnAdd пропишите:*

```
DBGridSales->Visible = false;
GroupBoxSales->Align = alTop;
GroupBoxSales->Visible = true;
DMMain->ADOT_Sales->Insert();
if (DMMain->ADOT_Sales->FieldByName("Код_сотрудника")->IsNull)
ButtonBack->Enabled = false;
GroupBoxDisheSale->Visible = false;
```

*- В событии OnClick у btnEdit пропишите:*

```
DBGridSales->Visible = false;
GroupBoxSales->Align = alTop;
GroupBoxSales->Visible=true;
DMMain->ADOT_Sales->Edit();
ButtonBack->Enabled=true;
GroupBoxDisheSale->Visible=true;
```

*- В событии OnClick у ButtonSotrudnik пропишите обработчик выбора сотрудника из формы для сотрудников:*

```
fmList-> btnVibor->Visible=true;
fmList->DataSource1->DataSet=DMMain->ADOT_Sotrudniki;
if (DMMain->ADOT_Sales->State==dsInsert)
    DMMain->ADOT_Sotrudniki->First();
else
    DMMain->ADOT_Sotrudniki->Locate("Код_сотрудника",
DMMain->ADOT_Sales->FieldByName("Код_сотрудника")->Value,
TLocateOptions() << loCaseInsensitive << loPartialKey );
if (fmList->ShowModal() == mrOk)
{
    DMMain->ADOT_Sales->Edit();
    DMMain->ADOT_Sales->FieldByName("Код_сотрудника")-
>Value=DMMain->ADOT_Sotrudniki->FieldByName("Код_сотрудника")-
>Value;
    DMMain->ADOT_Sales->Post();
    fmList-> btnVibor->Visible=false;
 }
 ButtonBack->Enabled=true;
 GroupBoxDisheSale->Visible=true;
```

*- В событии по закрытию формы OnClose у fmSales пропишите:*

```
DBGridSales->Visible = true;
DBGridSales->Align = alTop;
GroupBoxSales->Visible = false;
GroupBoxDisheSale->Visible=true;
 if ((DMMain->ADOT_Sales->State==dsInsert)||(DMMain-
>ADOT_Sales->State==dsEdit))
{
    if (Application->MessageBox("Сохранить изменения в
продаже?", "Вопрос", MB_YESNO)==IDYES)
    DMMain->ADOT_Sales->Post();
    else
    DMMain->ADOT_Sales->Cancel();
 }
```

*- В событии OnClick у кнопки ButtonBack пропишите вызов обработчика закрытия формы:*

```
FormClose(Sender, caNone);
```

*- Создадим для удобства кнопку поиска по номеру продажи.*

*- На панель инструментов для продаж рядом с btnEdit разместите с палитры Additional кнопку BitBtn для поиска продаж. Для нее укажите:*

*- свойство Name в btnFindSale;*

*- через свойство Glyph у btnFindSale в инспекторе объектов задайте соответствующую картинку для кнопки поиска продажи;*

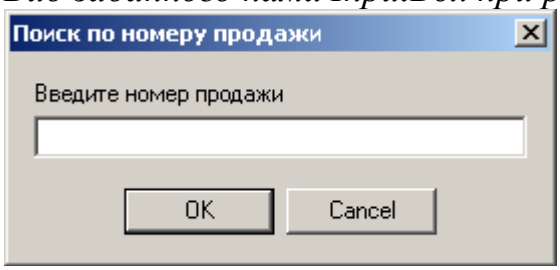
*- свойство Caption у btnFindSale задайте в Поиск продажи.*

*- В событии OnClick у btnFindSale пропишите обработчик поиска продажи:*

```
AnsiString s;
s=InputBox("Поиск по номеру продажи","Введите номер продажи","");
if (s=="") ShowMessage("Вы ничего не ввели");
else
if(!DMMain->ADOT_Sales->Locate("Номер_продажи", s, TLocateOptions() << loCaseInsensitive << loPartialKey ))
ShowMessage("Таких продаж нет!");
```

*Где InputBox – диалог со строкой ввода, введенное в строке ввода значение хранит в виде AnsiString.*

*Вид заданного нами InputBox при работе приложения:*




***Примечание:***

*Реализация поиска и фильтра через запрос см. лабораторную работу №5.*