ЛАБОРАТОРНАЯ РАБОТА №4

ОБЩИЕ ПРИНЦИПЫ РАБОТЫ С НАБОРАМИ ДАННЫХ

Цель:

1) Провести ознакомление с состояниями наборов данных.
2) Провести ознакомление с возможностями навигации по набору данных.
3) Провести ознакомление с возможностями внесения изменений в НД.
4) Рассмотреть возможности обработок ошибок при работе с данными в БД.
5) Рассмотреть особенности работы с мемо-полем и полем с графическим изображением.

Работа выполняется в приложении для работы с базой данных для своей предметной области (на примере базы данных для кафе).

Краткие теоретические сведения

1 Состояния наборов данных

НД могут находиться, например, в одном из следующих состояний (таблица 1).

Таблица 1 – Примеры состояний НД

Состояние	***Краткое описание***
Dslnactive	НД закрыт
dsBrowse	Состояние по умолчанию для открытого НД. Показывает, что записи просматриваются, но в данный момент не изменяются.
dsEdit	НД находится в состоянии редактирования текущей записи (после явно или неявно вызванного метода Edit).
Dslnsert	НД находится в состоянии добавления новой записи (после явно или неявно вызванного метода Insert или Append).
dsSetKey	НД находится в состоянии поиска записи по критерию, заданному методами FindKey, GotoKey,

	FindNearest или GotoNearest. По окончании поиска НД переходит в состояние dsBrowse.
dsCalcFields	Выполняется установление значений вычисляемых полей (по алгоритму, заданному в обработчике события OnCalcFields). В данном режиме изменения в НД вноситься не могут. После выхода из режима НД переходит в предыдущее состояние.
dsFilter	Обрабатывается фильтрация записей в НД при свойстве Filtered, установленном в True. Имеет место текущий вызов события OnFilterRecord для определения того, удовлетворяет ли текущая запись условию фильтрации, описанному в обработчике данного события. После выполнения события OnFilterRecord НД переводится в состояние dsBrowse.

Рассмотрим методы, которые могут переводить БД из одного состояния в другое.

1) **dsInactive → dsBrowse**
2) **dsBrowse → dsInactive**
3) **dsBrowse → dsEdit**
4) **dsEdit → dsBrowse**
5) **dsBrowse → dslnsert**
6) **dslnsert → dsBrowse**
7) **dsBrowse → dsSetKey**
8) **dsBrowse → dsFilter**

Получить текущее состояние НД можно, используя свойство **property State.**

Оно возвращает, например, следующие константы: dslnactive, dsBrowse, dsEdit, dslnsert, dsSetKey, dsCalcFields, dsFilter.

Реакция на изменение состояния набора данных.

Событие OnStateChange (компонент TDataSource) наступает всякий раз при изменении состояния НД.

2 Внесение изменений в НД

2.1 Изменение текущей записи

Чтобы изменить запись в НД, этот НД нужно перевести методом **Edit** из состояния dsBrowse в состояние dsEdit, затем произвести изменение значения одного или нескольких полей записи и использовать метод **Post** для запоминания измененной записи в НД. Post в данном случае при благополучном исходе переводит НД из состояния dsEdit в состояние dsBrowse.

Для отказа от запоминания измененной записи в НД используется метод **Cancel**. Он также переводит НД из состояния dsEdit в состояние dsBrowse.

Редактирование записи должно быть разрешено (свойство **property ReadOnly** должно быть установлено в false). Помимо этого, могут быть запрещены для корректировки отдельные поля записи (когда свойство ReadOnly соответствующих компонентов TField установлено в true).

Метод Edit может вызываться:

- программно;
- автоматически, когда пользователь в визуальном компоненте, связанном с НД, выполняет определенные действия. Вид этих действий завит от визуального компонента.

2.2 Добавление новой записи

Чтобы добавить новую запись в НД, нужно вызвать метод **Insert** или **Append** для перевода из состояния dsBrowse в состояние dsInsert. Затем производится присваивание значения одному или нескольким полям записи, после чего выполняется метод **Post** для запоминания новой записи в НД. Post при благополучном исходе переводит НД из состояния dsInsert в состояние dsBrowse.

Для отказа от запоминания новой записи в НД используется метод Cancel. Он также переводит НД из состояния dsInsert в состояние dsBrowse.

Метод procedure Insert

При добавлении записи изменение НД должно быть разрешено (свойство **property ReadOnly** должно быть установлено в false). Помимо этого, могут быть запрещены для корректировки отдельные поля записи (когда свойство ReadOnly соответствующих компонентов TField установлено в true). В этом случае в них нельзя ввести новые значения.

Метод Insert может вызываться:

- программно;
- автоматически, когда пользователь в визуальном компоненте, связанном с Н Д, предпринимает соответствующие действия. Для перехода в режим dsInsert в компоненте TDBGrid достаточно нажать на клавиатуре клавишу Insert или, находясь на последней записи НД, попытаться перейти на нижнюю, несуществующую запись. То же происходит при нажатии

соответствующей кнопки связанного с данным НД компонента TDBNavigator.

Метод procedure Append аналогичен методу Insert, но он добавляет запись в конец набора данных, в то время как Insert добавляет ее после текущей записи.

2.3 Запоминание изменений – метод procedure Post

Выполнение метода **Post** приводит к запоминанию изменений, сделанных в режиме добавления или изменения записи.

Если НД не находится в режиме dsInsert или dsEdit, то применение **Post** приводит к возбуждению исключительной ситуации.

Вызов Post зависит от способа, которым ранее был вызван метод Insert или Edit:

- программно;
- автоматически.

Вид этих действий зависит от визуального компонента, связанного с НД. Например, для компонента TDBGrid, связанного с набором данных, это - переход к другой записи. Для НД, управляемого компонентом TDBNavigator; это - нажатие соответствующей экранной клавиши. Реже изменения в наборе данных, автоматически переведенном в режим редактирования, запоминаются путем программного вызова метода Post.

Метод Post, независимо от того, вызывается он программно или автоматически, может завершиться неудачно. Причиной этого могут послужить неверные значения в соответствующих полях записи.

Например:

- поле обязательного заполнения (свойство Required в true у соответствующего компонента TField) содержит пустое значение;
- для таблицы БД, у которой определен уникальный ключ, возникла ситуация дублирования ключа (Key Violation), то есть ключевое поле (группа полей) данной записи содержит значение, которое уже хранится в поле (группе полей) в другой записи;
- не соответствие типов введенных значений с указанными типами полей;
- обработчики событий типа OnValidate (компонент TField) и BeforePostRecord обнаружили, что какое-либо поле содержит неверное значение, не удовлетворяющее некоторым условиям. В этом случае, программно возбуждается исключительная ситуация, которая подавляет выполнение Post.

В лучшем случае при возникновении препятствий для выполнения Post запись переводится в состояние, в котором НД находился до выполнения метода (dslnsert или dsEdit).

2.4 Отмена сделанных изменений - метод procedure Cancel

Метод **Cancel** отменяет все изменения, сделанные в записи. Если НД находился в режиме добавления новой записи, запись в НД не добавляется. Если НД находился в режиме изменения записи, изменявшаяся запись в НД не записывается, и данные в ней остаются в том состоянии, в котором они находились до перехода в режим dsEdit. Сам НД переводится в режим dsBrowse.

Вызов Cancel зависит от способа, которым ранее был вызван метод Insert или Edit:

- программно;
- автоматически.

Cancel вызывается автоматически, если пользователь предпримет соответствующие действия, направленные на запоминание измененной записи в НД. Вид этих действий зависит от визуального компонента, связанного с НД. Например, для компонента TDBGrid, связанного с набором данных, это – нажатие клавиши Esc. Для НД, управляемого компонентом TDBNavigator, это - нажатие на соответствующей экранной клавиши компонента TDBNavigator.

2.5 Удаление записи

Удаление текущей записи в наборе данных реализуется методом **procedure Delete**.

Удаление записи может производиться:

- программно;
-автоматически, если это предусмотрено в том или ином компоненте.

В компоненте TDBGrid нажатие комбинации клавиш Ctrl + Del влечет за собой удаление записи, которое, в соответствии с опциями настроики TDBGrid, может выполняться как с запросом подтверждения, так и без него.

2**.6 Обработка ошибок смены состояний набора данных**

В случае неудачи при выполнении методов Insert, Edit, Delete и Post обработку ошибки можно реализовать в соответствующих обработчиках событий OnEditError (ошибки при выполнении Insert и Edit), OnDeleteError, (ошибки при выполнении Delete) и OnPostError (ошибки при выполнении Post).

Назначение параметров при этом:

DataSet - указатель на компонент, в котором произошла ошибка;

E - ссылка на объект-исключение;

Action - действие:

daFail - выполнение метода, вызвавшего ошибку, отменяется, выводит сообщение об ошибке;

daAbort - выполнение метода, вызвавшего ошибку, отменяется, сообщение об ошибке не выводится;

daRetry - метод, вызвавший ошибку, после выхода из обработчика выполняется заново; при этом в теле обработчика должна быть скорректирована при ошибке, иначе произойдет зацикливание программы.

2.7 Оценка изменения записи

Часто бывает необходимо знать, вносились ли в запись изменения в режимах dsInsert или dsEdit. Это актуально в тех случаях, когда внесение изменений в записи зависит от каких-либо условий, которые могут наступать или не наступать в разные моменты работы приложения.

Свойство НД **property Modified** автоматически устанавливается в true, если значение какого-либо поля записи НД было изменено в режимах dslnsert или dsEdit. Методы Post и Cancel переводят свойство в состояние False.

3 Навигация по набору данных

Понятие курсора набора данных. Под курсором набора данных понимается указатель текущей записи в конкретном наборе данных. Текущая запись - та запись, над которой в данный момент времени можно выполнять какие-либо операции (удаление, изменение, чтение значений, содержащихся в записи полей).

Существует 5 основных методов для изменения курсора НД (таблица 2).

Таблица 2 – Методы для изменения курсора НД

Метод	**Выполняемые действия**
procedure First;	Устанавливает курсор на первую запись в наборе данных.
procedure Last;	Устанавливает курсор на последнюю запись в наборе данных.
procedure Next;	Перемещает курсор на следующую запись в наборе данных относительно текущей записи.
procedure Prior;	Перемещает курсор на предыдущую запись в наборе данных относительно текущей записи.

Определение начала и конца набора данных

Свойство

property BOF возвращает true, если курсор установлен на первую запись в наборе данных.

Свойство

property EOF возвращает true, если курсор установлен на последнюю запись в наборе данных.

Может сложиться впечатление, что первая и последняя записи набора всегда фиксированы, что это физически первая и последняя записи в НД. Это неверно. Во-первых, как уже отмечалось выше, набор данных может содержать часть записей из таблицы БД. Поэтому набор данных - понятие логическое, а не физическое.

При изучении вопросов навигации по НД следует говорить прежде всего о логическом характере следования записей, поскольку физический характер их расположения в конкретном случае неизвестен.

Реакция на изменение курсора набора данных

Событие OnDataChange (компонент DataSource) возникает всякий раз при изменении курсора НД, т.е. при переходе к новой текущей записи. Это событие возникает, когда курсор НД уже находится на новой записи.

Событие происходит и в режимах dslnsert и dsEdit:

- при изменении какого-либо поля;
- при первом перемещении с измененного поля на другое поле.

Два события компонента типа "набор данных" также происходят при переходе к новой записи:

- **BeforeScroll -** событие наступает перед переходом на другую запись в наборе данных;
- **AfterScroll -** событие наступает после перехода на другую запись в наборе данных.

Свойство набора данных property RecordCount возвращает текущее число записей в НД в виде целого числа.

Свойство набора данных property RecNo возвращает порядковый номер текущей записи в виде целого числа.

Задание 1. Внесение изменений в наборы данных, работа с состояниями наборов данных, навигация по набору данных.

- Перейдите на форму fmList.

- На нашей форме находятся компоненты DBGrid1 и DBNavigator1, которые позволяют проводить изменения в наборах данных, проводить навигацию по наборам данных.

- Проделаем изменения наборов данных и навигацию программно.

- С закладки Win 32 разместите на форме fmList компонент панели инструментов ToolBar1. На нем будем размещать наши кнопки для изменения данных и навигации данных.

- Установите у ToolBar1 свойство ShowCaptions в true – чтобы была возможность для кнопок панели инструментов показывать текст на кнопках.

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption В начало;

- событие OnClick:

DataSource1->DataSet->First();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Назад;

- событие OnClick:

DataSource1->DataSet->Prior();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Вперед;

- событие OnClick:

DataSource1->DataSet->Next();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption В конец;

- событие OnClick:

DataSource1->DataSet->Last();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New separator – чтобы отделить кнопки навигации.

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Добавить;

- событие OnClick:

DataSource1->DataSet->Insert();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Удалить;

- событие OnClick:

```
if (Application->MessageBox("Удалить текущую запись?",
"Предупреждение", MB_YESNO) == IDYES)
    DataSource1->DataSet->Delete();
```

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Изменить;

- событие OnClick:

DataSource1->DataSet->Edit();

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New separator – чтобы отделить кнопки добавления, удаления, редактирования.

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Сохранить;

- событие OnClick:

```
if ((DataSource1->DataSet->State ==dsEdit) ||
(DataSource1->DataSet->State ==dsInsert))
      DataSource1->DataSet->Post();
 else
     Application->MessageBox("Не было изменений", "Внимание",
MB_ICONINFORMATION);
```

- Нажмите на ToolBar1 правую кнопку мыши и выберете команду New button.

- Для созданной кнопки укажите:

- свойство Caption Отменить;

- событие OnClick:

```
DataSource1->DataSet->Cancel();
```

- Нажмите F9.

- Посмотрите, как отрабатывают запрограммированные кнопки.

Задание 2. Добавление и редактирование данных через отдельную форму

При добавлении и редактировании информации о сотрудниках будем показывать отдельные формы для добавления и редактирования информации о сотруднике.

- Добавьте в проект новую форму File/New/Form.

- Свойство Name у новой форму установите в fmSotrudniki для работы с данными о сотруднике.

- Сохраните форму в проект. Для этого нажмите File/Save all и в диалоговом окне сохраните новую форму в той же папке, что и другие модули проекта. При сохранении укажите новому модулю имя Sotrudniki.cpp.

- Перейдите на форму fmSotrudniki.

- Перейдите на вкладку DataAccess. Найдите на данной вкладке компонент DataSource и разместите его объект (DataSource1) на форму fmSotrudniki

- Для DataSource1 установите свойство Name в dsSotrudniki – так мы обозначим его привязку к набору данных DMMain->ADOT_Sotrudniki (для таблицы Сотрудники).

- Наши наборы данных находятся на модуле данных в модуле DM. Данный файл DM.h необходимо подключить в форме fmSotrudniki.

Для этого, находясь на форме fmSotrudniki, выберете в меню File/Include Unit Hdr…(или нажмите Alt+F11). В диалоговом окне Use Unit выберете модуль DM.

После этого в модуле Sotrudniki.cpp появится запись #include "DM.h"

Или можно было просто в модуле fmSotrudniki.cpp прописать #include "DM.h".

-Установите у dsSotrudniki свойство DataSet в набор данных DMMain->ADOT_Sotrudniki.

- с закладки Standart разместите на форме fmSotrudniki компонент Label1. Для Label1 установите свойство Caption в Код сотрудника.

- Рядом с Label1 разместите с закладки Data Controls компонент DBText1. Для него установите свойства:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Код_сотрудника.

- Под Label1 разместите с закладки Standart на форме fmSotrudniki компонент Label2. Для Label2 установите свойство Caption в Фамилия сотрудника.

- Рядом с Label2 разместите с закладки Data Controls компонент DBEdit1. Для него установите свойства:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Фамилия.

- Под Label2 разместите с закладки Standart на форме fmSotrudniki компонент Label3. Для Label3 установите свойство Caption в Имя сотрудника.

- Рядом с Label3 разместите с закладки Data Controls компонент DBEdit2. Для него установите свойства:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Имя.

- Под Label3 разместите с закладки Standart на форме fmSotrudniki компонент Label4. Для Label4 установите свойство Caption в Отчество сотрудника.

- Рядом с Label4 разместите с закладки Data Controls компонент DBEdit3. Для него установите свойства:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Отчество.

- Под Label4 разместите с закладки Data Controls компонент DBRadioGroup1.

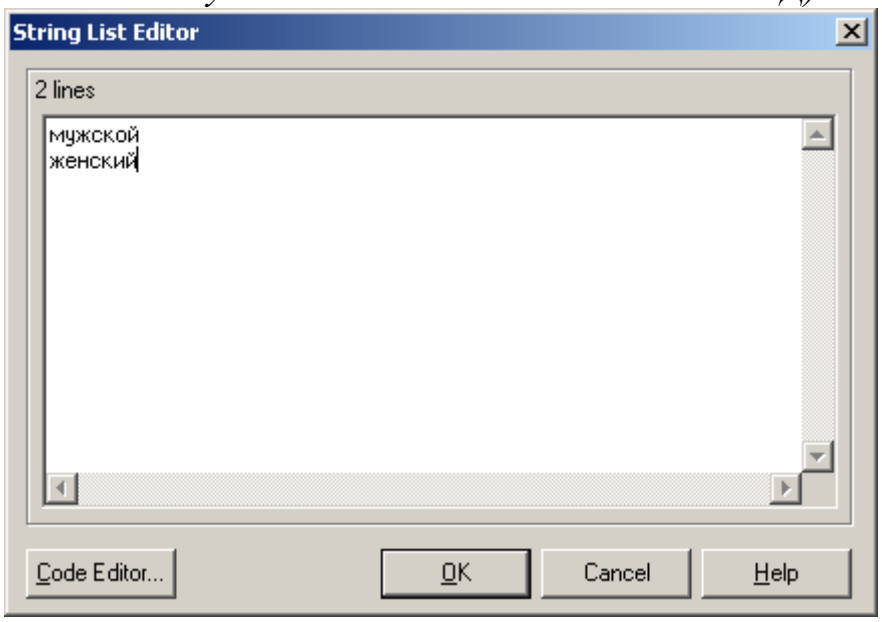
- Для компонента DBRadioGroup1 установите:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Пол;

- свойство Caption в Пол сотрудника;

- свойство Items пропишите в мужской и женский (так как только такие значения допустимы в данной колонке в нашей БД):



- свойство Columns в 2 – будем отображать в 2 колонки.

- Под компонентом DBRadioGroup1 разместите с закладки Standart на форме fmSotrudniki компонент Label5. Для Label5 установите свойство Caption в Должность.

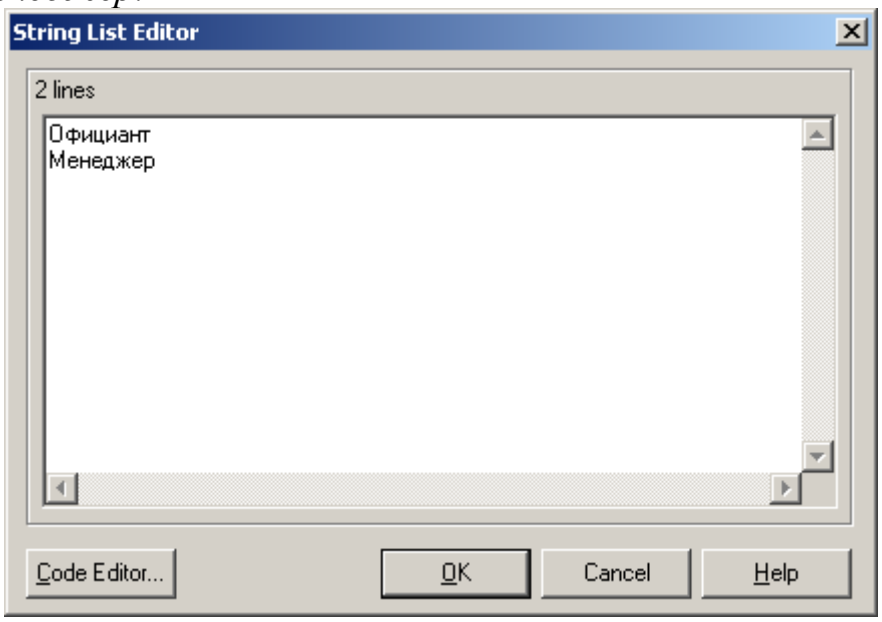
- Рядом с компонентом Label5 с закладки закладки Data Controls разместите компонент DBComboBox1.

- Для компонента DBComboBox1 установите:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Должность;

- свойство Items пропишите пока несколько значений, затем будем его формировать автоматически. Например, впишите значения Официант, Менеджер:



- Под компонентом DBComboBox1 разместите с закладки Standart на форме fmSotrudniki компонент Label6. Для Label6 установите свойство Caption в Семейное положение.

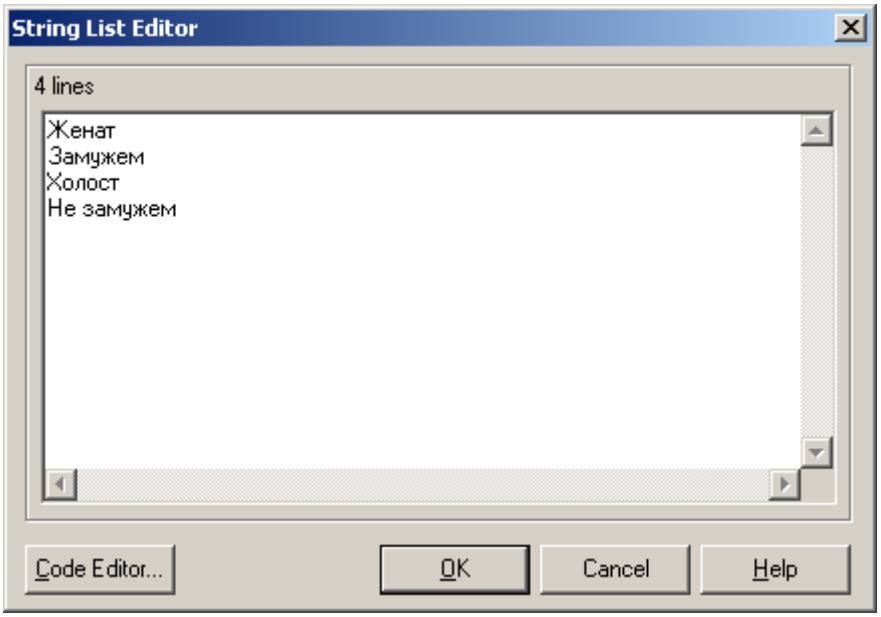
- Рядом с компонентом Label6 с закладки закладки Data Controls разместите компонент DBListBox1.

- Для компонента DBListBox1 установите:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Семейное_положение;

- свойство Items пропишите допустимы для данного поля в нашей БД значения:



- Под компонентом DBListBox1 разместите с закладки Standart на форме fmSotrudniki компонент Label7. Для Label7 установите свойство Caption в Дети.

- Рядом с компонентом Label7 с закладки закладки Data Controls разместите компонент DBCheckBox1.

- Для компонента DBCheckBox1 установите:

- свойство DataSource (источник данных) в dsSotrudniki;

- свойство DataField (имя поля) в Дети;

- свойство Caption в Дети есть;

- свойство ValueChecked (значение при нажатой галочке) пропишите в Дети есть;

- свойство ValueUnChecked (значение при отжатой галочке) пропишите в Детей нет;

- значения Дети есть и Детей нет в нашей БД являются допустимыми для поля Дети.

- Под компонентом DBCheckBox1с закладки Standart разместите компонент Button1.

- Для Button1 установите:

- свойство Caption в Сохранить;

- свойство ModalResult установите из списка в mrOk – при этом по нажатии данной кнопки форма fmSotrudniki будет автоматически закрываться;

- свойство Default установите в true – в этом случае нажатие данной кнопки будет происходить и по клавише Enter.

- Рядом с Button1 с закладки Standart разместите компонент Button2.

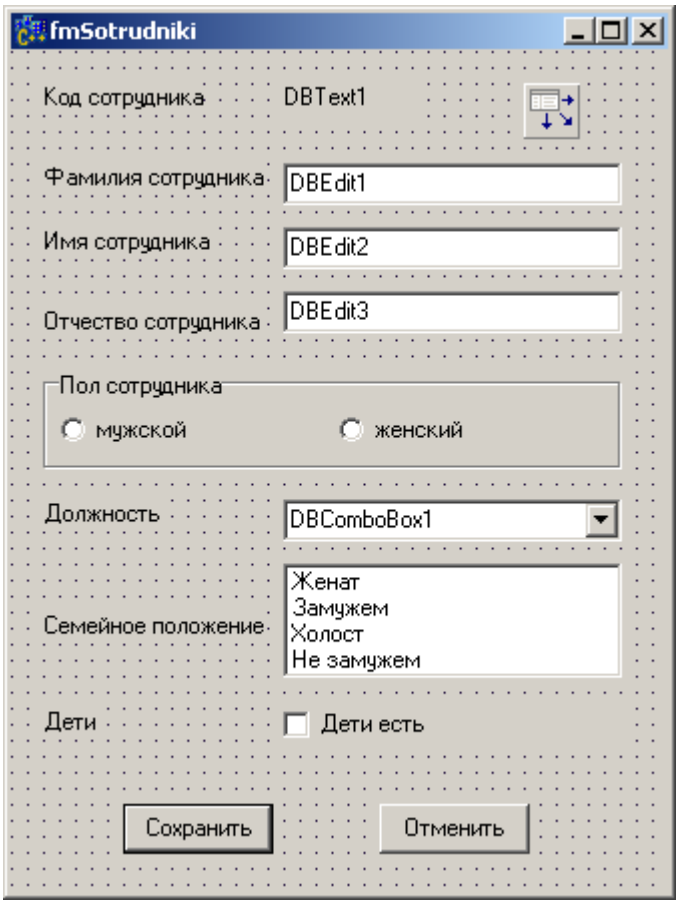
- Для Button2 установите:

- свойство Caption в Отменить;

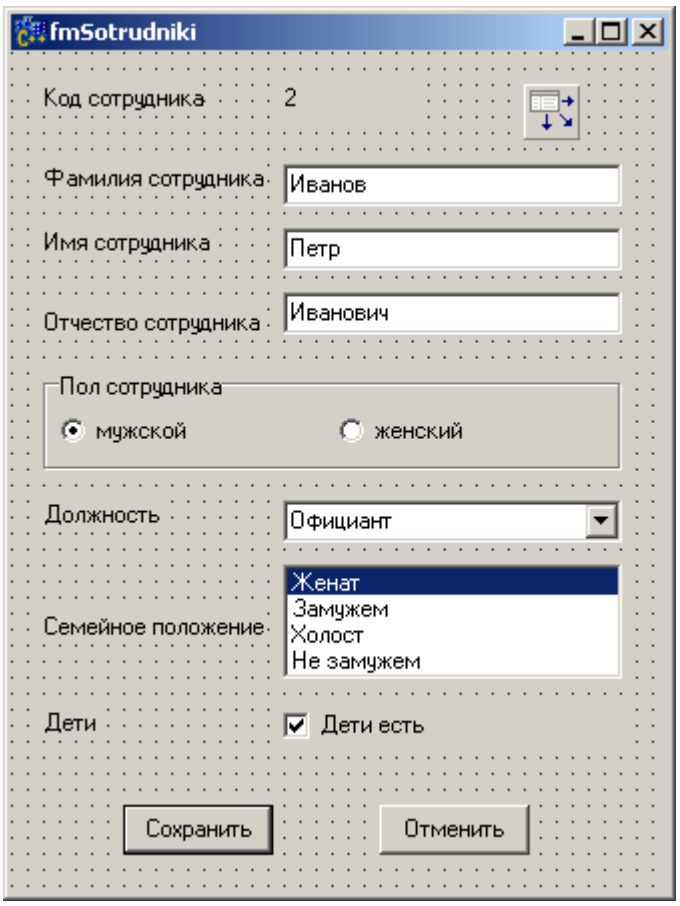
- свойство ModalResult установите из списка в mrCancel – при этом по нажатии данной кнопки форма fmSotrudniki будет автоматически закрываться;

- свойство Cancel установите в true – в этом случае нажатие данной кнопки будет происходить и по клавише Esc.

Вид формы fmSotrudniki может иметь вид (при свойстве Active у DMMain->ADOT_Sotrudniki в значении false):



Вид формы fmSotrudniki может иметь вид (при свойстве Active у DMMain->ADOT_Sotrudniki в значении true):



- Теперь необходимо определить показ данной формы. Данная форма будет показываться при добавлении информации о сотруднике и редактировании информации о сотруднике в момент работы формы fmList.

- Перейдите на форму fmList.

- Подключите на форме fmList форму fmSotrudniki.

Для этого, находясь на форме fmList, выберете в меню File/Include Unit Hdr…(или нажмите Alt+F11). В диалоговом окне Use Unit выберете модуль Sotrudniki.

После этого в модуле List.cpp появится запись #include "Sotrudniki.h"

Или можно было просто в модуле List.cpp прописать #include "Sotrudniki.h"

- Измените обработчик нажатия кнопки Добавить на панели инструментов на форме fmList. Для этого в событии OnClick для данной кнопки укажите:

```
DataSource1->DataSet->Insert();
 if (DataSource1->DataSet==DMMain->ADOT_Sotrudniki)
        {
              fmSotrudniki->Caption="Добавление сведений о сотруднике";
              fmSotrudniki->ShowModal();
        }
```

- Измените обработчик нажатия кнопки Изменить на панели инструментов на форме fmList. Для этого в событии OnClick для данной кнопки укажите:

```
DataSource1->DataSet->Edit();
 if (DataSource1->DataSet==DMMain->ADOT_Sotrudniki)
        {
              fmSotrudniki->Caption="Редактирование сведений о сотруднике";
              fmSotrudniki->ShowModal();
        }
```

- Перейдите на форму fmSotrudniki.

- Укажите у формы fmSotrudniki событии OnClose (срабатывает при закрытии формы):

```
 if (ModalResult==mrOk)
DMMain->ADOT_Sotrudniki->Post();
else
DMMain->ADOT_Sotrudniki->Cancel();
```

Задание 3. Обработка ошибок при работе с данными в приложении

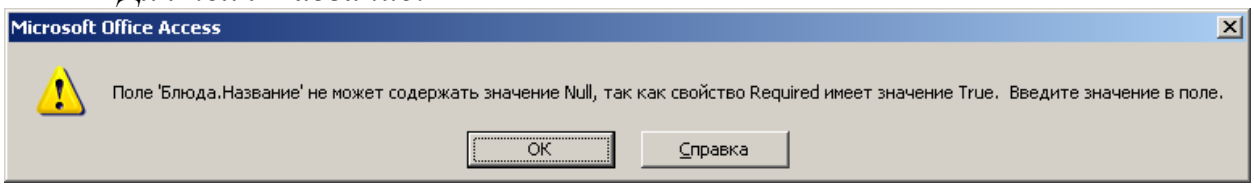
Задача обеспечения целостности данных в БД (на таблицы, поля, ссылочная целостность) обеспечивается СУБД. В случае нарушения целостности СУБД генерирует соответствующее сообщение и реакцию (см. лабораторную работу №1).

Рассмотрим примеры обработок ошибок при работе с данными в приложении.

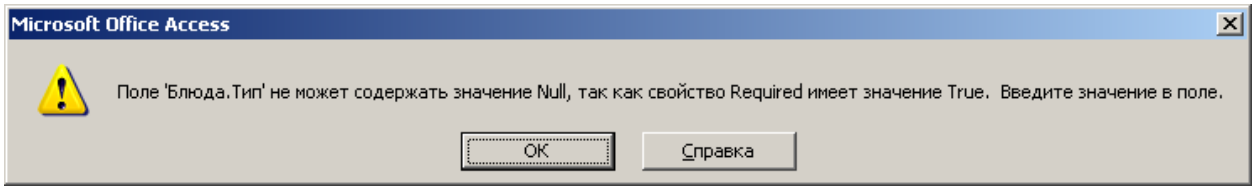
3.1 Работа с данными по блюдам

3.1.1 Для таблицы Блюда поля Название, Тип, Цена_продажи должны быть обязательно заполнены пользователем. Если пользователь оставит их пустыми, то СУБД сгенерирует ошибки:

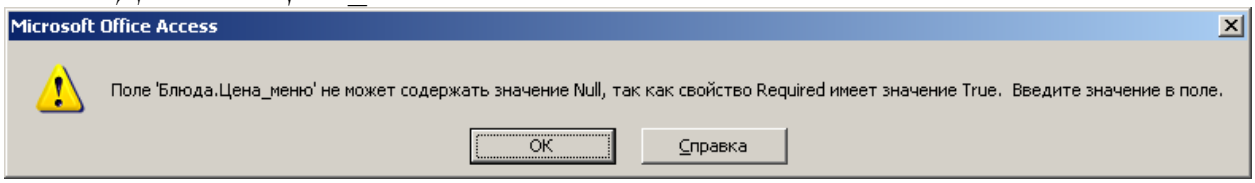
- Для поля Название:



- Для поля Тип:



- Для поля Цена_меню:



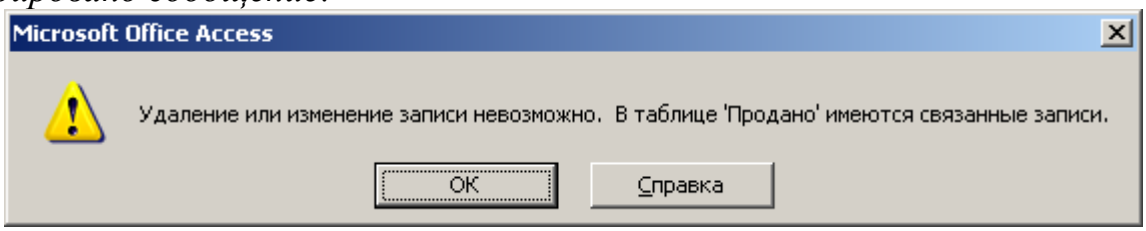
- Чтобы пользователю объяснить более понятно совершенную им ошибку можно в приложении заблокировать ошибку СУБД и передать свой обработчик.

Для этого на модуле данных DMMain выделите компонент ADOT_Dishes. Для него в событии OnPostError (ошибка в момент сохранения записи) пропишите:

```
if ( ( (ADOT_Dishes->FieldByName("Название")->IsNull) ||
(ADOT_Dishes->FieldByName("Название")->AsString==""))
    || ((ADOT_Dishes->FieldByName("Тип")->IsNull) || (ADOT_Dishes-
>FieldByName("Тип")->AsString==""))
    ||  (ADOT_Dishes->FieldByName("Цена_меню")->IsNull))
    {
        Application->MessageBox("Необходимо обязательно в блюдах
указать Название, Тип и Цена_меню","Ошибка", MB_ICONERROR);
        Action=daAbort; // Запрещаем вывод ошибки СУБД
    }
```

3.1.2 Для таблицы Блюда, так как у нас не установлен в БД механизм каскадного удаления, то нельзя удалить блюдо, если оно указано в Продано.

При попытки удаления блюда со связанными записями, будет сгенерировано сообщение:



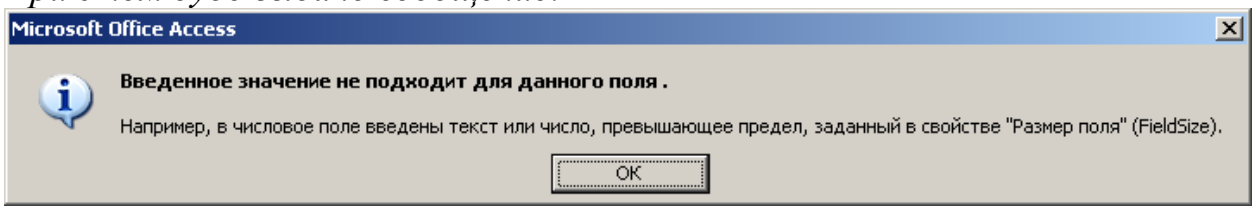
Если необходимо подменить данное сообщение, необходимо в событии OnDeleteError у ADOT_Dishes прописать:

```
Application->MessageBox("Нельзя удалить данное блюда, так как оно
указано в продажах. Сначала удалите продажи данного блюда", "Ошибка
удаления", MB_ICONERROR);
```

Action=daAbort;

3.1.3 Для таблицы блюда пользователь может ввести не тот тип данных в поля Цена_меню, Выход. Например, строку в числовой формат.

При этом буде выдано сообщение:



В данном случае в нашем приложении данную ошибку оставим отработке СУБД, так как в нашей программе пользователь работает с данными Цена_Меню и Выход по блюдам через DBGrid, который блокирует ввод строковых значений в числовые поля.

3.2 Аналогично проделайте обработку ограничений в приложении для таблицы Сотрудники:

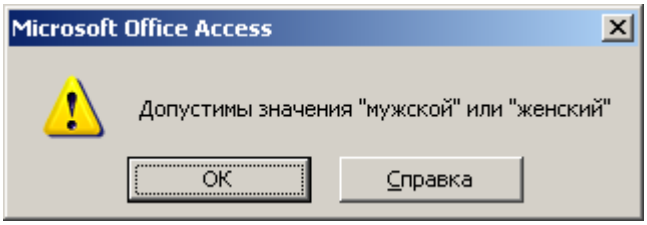
- обязательные поля Фамилия, Имя, Должность;

- нельзя удалять сотрудника, если есть связанные продажи.

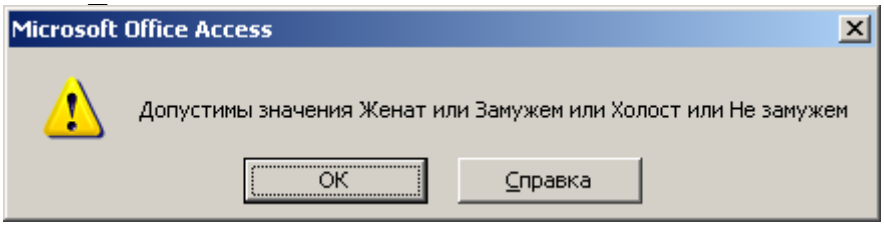
Примечание:

На момент сохранения в таблице Сотрудники могут быть еще ошибки для полей со значениями из списка:

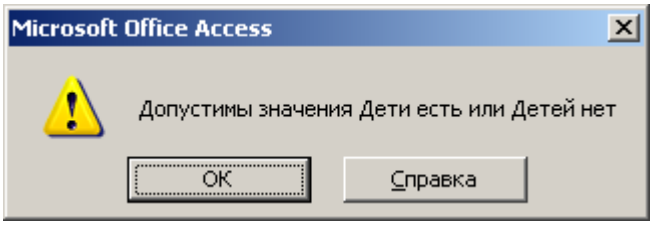
- Пол:



- Семейное_положение:

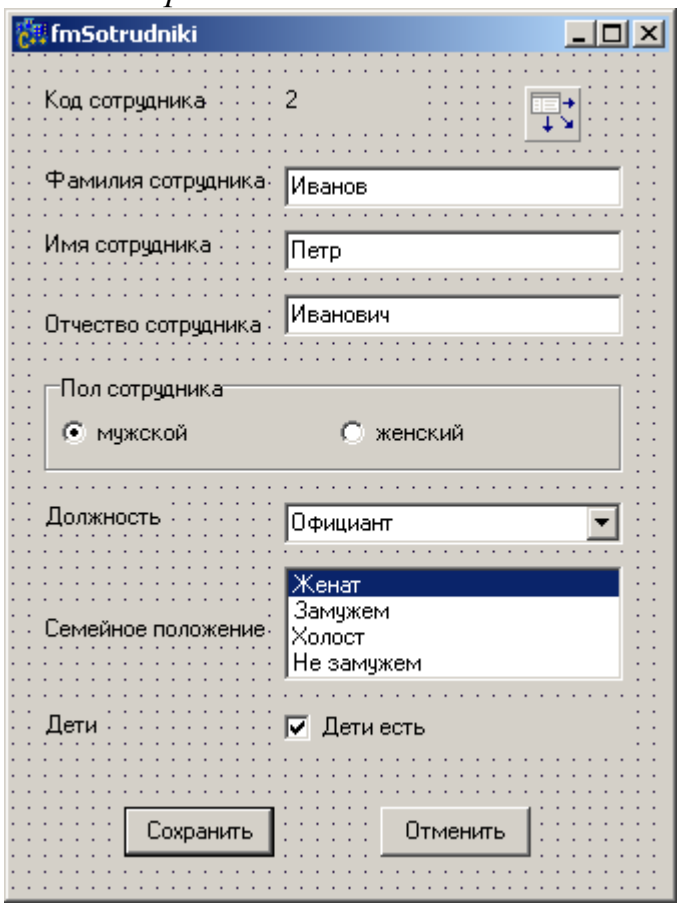


- Дети:



Оставим обработку данных сообщений на уровне СУБД, так как в нашей программе пользователь не может написать в данные поля значения, кроме допустимых значений, так как мы запретили ему работу с данными

по сотрудникам через DBGrid, а в отдельной форме для редактирования и добавления данных о сотруднике пользователь через специальные компоненты заносит только предоставленные из списка данные:



3.3 Обработайте в приложении ограничения таблицы Продажи:
- обязательное поле Код_сотрудника;
- нельзя удалять продажу, если у нее есть проданные товары.

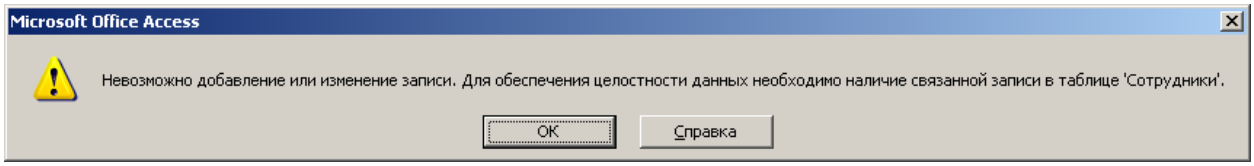
Примечание1:

Пользователь может в поле Дата вводить дату не верно. В этом случае необходимо на событии EditError(ошибка редактирования) у ADOT_Sales прописать:

Application->MessageBox("Не верно вводите дату","Ошибка", MB_ICONERROR);
Action=daAbort;

Примечание 2:

В поле Код_сотрудника по обеспечению целостности должно быть введено значение, существующее в поле Код_сотрудника таблицы Сотрудники. Если это будет нарушено, то будет сообщение:

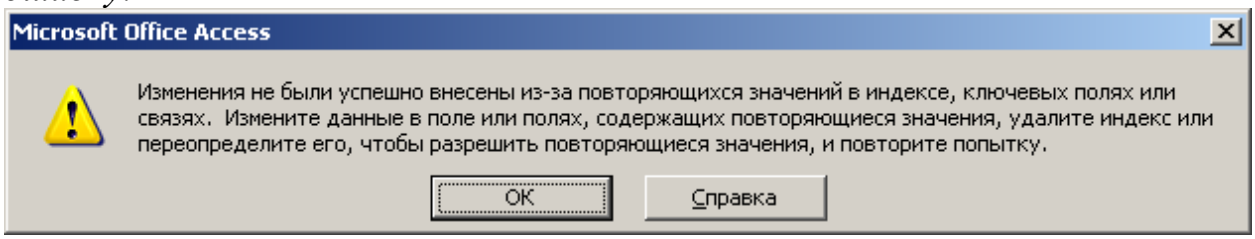


Данное сообщение оставим на уровне СУБД. Кроме того, в нашем приложении пользователь в продажах сам не пишет Код_сотрудника, а выбирает из подстановочного поля (задание 3 лабораторной работы №3), чем мы тоже способствуем поддержке ссылочной целостности.

3.4 Обработайте в приложения ограничения для таблицы Продано:

- обязательные поля Номер_продажи, Код_блюда, Цена, Количество;

- значения в полях Номер_продажи и Код_блюда совместно уникальны (так как первичный ключ), т.е. в одной и той же продаже одно и то же блюда дважды не указывается. В случае нарушения СУБД сгенерирует ошибку:



Для упрощения обработки ограничений для таблицы Продано в приложении будем в независимости от вида ограничения на событие OnPostError (ошибка при сохранении) сообщать пользователю об ошибке. Для этого в событии OnPostError у ADOT_DisheSale пропишите:

```
Application->MessageBox("Ошибка сохранения данных","Ошибка",
MB_ICONERROR);
        Action=daAbort;
```

3.5 При работе в форме fmSales может возникать ошибка, когда значение записи для продажи не сохранено, а уже начата работа с данными проданных блюд.

Для этого перейдите на модуль данных DMMain.

- Выделите компонент ADOT_DisheSale. Для в него в событии BeforeInsert (перед добавлением записи) пропишите:

```
if ((ADOT_Sales->State == dsInsert) || (ADOT_Sales->State == dsEdit))
  ADOT_Sales->Post();
```

- Данный обработчик события ADOT_DisheSaleBeforeInsert необходимо указать и на событие BeforeEdit (перед редактированием) для ADOT_DisheSale. Для этого в событии BeforeEdit из списка выберете ADOT_DisheSaleBeforeInsert.

Аналогично можно обрабатывать другие ограничения БД.

Задание 4. Особенности работы с мемо-полем и полем с графическим изображением

Задание 4.1. Работа с MEMO полями

Для сохранения в НД Состава блюда мы используем компонент TDBMemo.

Элемент управления TDBMemo является потомком TMemo и является простым текстовым редактором, адаптированным для работы с текстовым представлением BLOB поля набора данных.

Возможности TDBMemo при работе с текстом аналогичны его родителю TMemo – выделение и редактирование текста, работа с буфером обмена, поддержка TEditActions. Следует обратить внимание на методы TMemo.Lines.LoadFromFile() и TMemo.Lines.SaveToFile(), которые позволяют загружать из файла и сохранять в файл содержимое редактора (и соответствующего поля). Особенность – перед вызовом TMemo.Lines.LoadFromFile() нужно убедиться, что набор данных находиться в режиме редактирования.

- Разместите панель инструментов (TToolBar) с закладки Win32 для работы с DBMemo, разместив ее над DBMemo.

- Для того, чтобы панель инструментов показывала всплывающие подсказки установите свойство ShowHint в true.

- Для того, чтобы панель инструментов показывала заголовки кнопок установите свойство ShowCaption в true.

- Для открытия и сохранения файлов используйте с закладки Dialogs компоненты OpenDialog1 – для открытия файлов, SaveDialog1 для сохранения файлов.

Кнопка Копировать:

- На панель инструментов добавьте кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonCopyMemo.

Для ToolButtonCopyMemo укажите:

- свойство Caption в Копировать.

-Свойство Hint в Копировать.

- Событие OnClick:

```
DBMemo1->CopyToClipboard();
```

Кнопка Вставить:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonPasteMemo.

Для ToolButtonPasteMemo укажите:

- свойство Caption в Вставить.

-Свойство Hint в Вставить.

- Событие OnClick:

DBMemo1->PasteFromClipboard();

- Разметите на форме fmList с закладки Dialogs компонент OpenDialog1 – стандартный диалог открытия файлов.

- Разметите на форме fmList с закладки Dialogs компонент SaveDialog1 – стандартный диалог сохранения файлов.

Основными свойствами объектов типа TOpenDialog и TsaveDialog являются (таблица 1.3).

Таблица 4.1- Основные свойствами объектов типа TOpenDialog и TSaveDialog

Свойство	Тип	Комментарии
DefaultExt	String	Задает расширение файла по умолчанию Оно будет добавлено к введенному пользователем имени файла, если оно указано без расширения.
FileName	String	Имя выбранного/сохраненного файла
Filter	String	В этом свойстве указывается список типов видимых в диалоге файлов
FilterIndex	Integer	Номер фильтра при открытии диалога
Options	TOpenOptions TSaveOptions	В этом свойстве указываются различные параметры диалога.
Title	String	Текст заголовка диалога выбора файлов

Для того чтобы выдать диалоги на экран, необходимо использовать их метод Execute, возвращающий логическое значение, которое равно True, если пользователь выбрал или сохранил файл.

Для задания фильтра в компонентах выбора и сохранения файлов можно вызвать редактор свойства Filter, нажав кнопку с многоточием справа отзначения в инспекторе объектов. В нем имеется таблица с двумя

колонками : в левой помещается описание варианта фильтра, а в правый – сам фильтр.

Кнопка Открыть:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonOpenMemo:.

Для ToolButtonOpenMemo укажите:

- свойство Caption в Открыть.

-Свойство Hint в Открыть.

- Событие OnClick:

```
try
    {
        if (OpenDialog1->Execute())
        DMMain->ADOT_Dishes->Edit();
        DBMemo1->Lines->LoadFromFile(OpenDialog1->FileName);
    }
    catch(...)
    {
        Application->MessageBoxA("Не тот тип файлов","Ошибка",
MB_ICONERROR);
    }
```

Кнопка Сохранить:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonSaveMemo.

Для ToolButtonSaveMemo укажите:

- Свойство Caption в Сохранить.

-Свойство Hint в Сохранить.

- Событие OnClick:

```
if (SaveDialog1->Execute())
    DBMemo1->Lines->SaveToFile(SaveDialog1->FileName);
```

Задание 4.2. Работа с графическим объектом

Для сохранения в НД графического изображения мы используем компонент TDBImage.

Для изменения картинки можно использовать два способа.

Первый заключен в использовании буфера обмена (clipboard).

TDBImage имеет специальные методы для копирования картинки, которая сейчас содержится в поле назначенного набора данных (и отображается) в системный буфер обмена Windows.

Это методы TDBImage->**CopyToClipboard** и TDBImage->**CutToClipboard**.

Второй метод отличается тем, что после сохранения картинки в буфере содержимое TDBImage и соответствующего поля стирается.

Картинка сохраняется в формате Bitmap и может быть вставлена из буфера во многие Windows программы (Word, Image Editor. Paint, Excel).

Также есть метод для вставки картинки, которая сейчас содержится в системном буфере в TDBImage и поле назначенное ему. Это метод **PasteFromClipboard**.

Все эти методы можно вызывать программно – например, назначив событие на кнопку или TAction. Также за ними уже зарезервированы комбинации клавиш, которые вы может использовать без какого либо дополнительного кодироваия (таблица 4.1).

Таблица 4.1 - Комбинации клавиш

Метод	**Клавиши**	**Действие**
CopyToClipboard	Ctrl+C Ctrl+Insert	Скопировать картинку в буфер обмена.
CutToClipboard	Ctrl+X Shift+Delete	«Вырезать» картинку в буфер обмена.
PasteFromClipboard	Ctrl+V Shift+Insert	Скопировать картинку из буфера обмена.

Второй способ использование методов свойства TDBImage->Picture.

Используйте метод TDBImage->Picture->SaveToFile(FileName) для сохранения картинки в файле формата *.bmp.

И метод TDBImage->Picture->LoadFromFile(FileName) для загрузки файла в TDBImage и текущее поле.

Один нюанс при использовании последнего метода: набор данных должен быть в режиме редактирования. Проверяйте это условие перед вызовом LoadFromFile и если это не так – переводите набор данных в режим редактирования принудительно: например TDBImage.DataSource.DataSet.Edit.

Все вышесказанное в одинаковой степени касается и простого элемента управления TImage, т.к. TDBImage является его потомком в иерархии классов VCL/CLX. Но при использовании простого TImage Вам придется самостоятельно отслеживать изменение состояния набора данных и нужного поля.

- Перейдите на форму fmList.

- На компонент GroupBox1 с закладки Win32 разместите компонент ToolBar.

- Для того, чтобы панель инструментов показывала всплывающие подсказки установите свойство ShowHint в true.

- Для того, чтобы панель инструментов показывала заголовки кнопок установите свойство ShowCaption в true.

На данной панели инструментов разместим кнопки для копирования, вставки фотографии блюда, для открытия фото из файла и сохранения в файл.

Кнопка Копировать:

- На панель инструментов добавьте кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonCopyImage.

Для ToolButtonCopyImage укажите:

- Свойство Caption в Копировать.

-Свойство Hint в Копировать.

- Событие OnClick:

DBImage1->CopyToClipboard(); //Копировать

Кнопка Вставить:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonPasteImage.

Для ToolButtonPasteImage укажите:

- Свойство Caption в Вставить.

-Свойство Hint в Вставить.

- Событие OnClick:

DBImage1->PasteFromClipboard(); //Вставить

Для кнопок Открыть и Сохранить стандартные диалоги Открытия и Сохранения графических файлов.

- Разметите на форме fmList с закладки Dialogs компонент OpenPictureDialog1 – стандартный диалог открытия графических файлов.

- Разметите на форме fmList с закладки Dialogs компонент SavePictureDialog1 – стандартный диалог сохранения графических файлов.

Для того чтобы выдать диалоги на экран, необходимо использовать их метод Execute, возвращающий логическое значение, которое равно True, если пользователь выбрал или сохранил файл.

Для задания фильтра в компонентах выбора и сохранения файлов можно вызвать редактор свойства Filter, нажав кнопку с многоточием справа отзначения в инспекторе объектов. В нем имеется таблица с двумя колонками : в левой помещается описание варианта фильтра, а в правый – сам фильтр.

Кнопка Открыть:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonOpenImage.

Для ToolButtonOpenImage укажите:

- Свойство Caption в Открыть.

-Свойство Hint в Открыть.

- Событие OnClick:

```
try
    {
        if (OpenPictureDialog1->Execute())
        DMMain->ADOT_Dishes->Edit();
        DBImage1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
    }
    catch(...)
    {
        Application->MessageBoxA("Не тот тип файлов","Ошибка", MB_ICONERROR);
    }
```

Кнопка Сохранить:

- На панель инструментов добавьте еще кнопку путем нажатия правой кнопки мыши на панели и выбора команды New Button). Свойство Name укажите в ToolButtonSaveImage.

Для ToolButtonSaveImage укажите:

- Свойство Caption в Сохранить.

-Свойство Hint в Сохранить.

- Событие OnClick:

```
if (SavePictureDialog1->Execute())
    DBImage1->Picture->SaveToFile(SavePictureDialog1->FileName);
```

Примечание:

Примеры картинок для блюд можете взять из папки Photo_Dishes. Задайте данные картинки некоторым блюдам в Вашей БД.

Задание 5. Оформление отчета к лабораторной работе

Оформить отчёт со следующим содержанием:

1. Титульный лист.

2. Цель работы.
3. Постановка задачи.
4. Краткая теория и ход выполнения заданий.
5. Описание результатов.

Приведите скриншоты полученной программы и листинг программы.
Покажите обработку ошибок в приложении.

6. Заключение (выводы).