**Лабораторная работа №1 (2 часа)**

**Тема: «Модуль данных»**

**Цель работы:** изучить способ создания модуля данных.

**Теоретические сведения**

**1. Доступ к базе данных**

Доступ к базе данных выполняется достаточно просто, с использованием богатого набора невизуальных компонентов работы с СУБД. Как правило, эти компоненты группируются в создаваемой программе в специальном модуле данных (TDataModule). Модуль данных представляет собой хранилище объектов, которое позволяет централизованно управлять их работой.

Создадим новый проект и добавим в него модуль данных командой File → New → Data Module. В свойство Name модуля данных поместите значение DM и сохраните модуль.

В модуле данных необходимо разместить компоненты, которые обеспечат доступ к самой БД. Для этого поместим в ранее созданный модуль данных новый компонент TIBDatabase, который расположен на панели InterBase. В свойстве DatabaseName надо указать полный путь поиска и имя файла базы данных, в свойстве Name – имя базы данных.

Работа с СУБД InterBase требует обязательной поддержки транзакций. Для этого поместим в модуль данных компонент TIBTransaction, ответственный за управление транзакциями. В его свойстве DefaultDatabase укажем созданный объект IBDataBase1. В свою очередь, в свойстве DefaultTransaction этого объекта надо указать транзакционный объект IBTransaction1, чтобы установить между ними связь.

Чтобы к БД можно было обращаться, свойству Connected надо присвоить значение True.

**2. Доступ к таблицам**

Теперь можно добавить в модуль данных компонент TIBTable, который является набором данных (НД). Он позволяют организовать доступ к конкретной таблице указанной базы данных. В свойство Database заносится значение IBDataBase1, в свойство TableName – имя таблицы (в этот момент InterBase может запросить имя пользователя и пароль для доступа к базе данных). В свойстве Name необходимо изменить имя IBTable1, используемое по умолчанию, на имя связанной с ним таблицы БД.

Чтобы можно было обращаться к этой таблице, необходимо поместить на форму ещё один компонент TDataSource (источник данных), в котором в свойстве DataSet указывается объект IBTable1. Теперь таблица из БД будет доступна в любой точке программы.

Чтобы к таблице можно было обращаться из программы, свойству Active надо присвоить значение True.

**3. Обработка подключения к базе данных**

При запуске программы пользователю каждый раз будет задаваться вопрос о его правах: имени и пароле. Это неудобно, поэтому лучше поместить нужные данные в программный код. Проще всего поступить так:

1. значение свойства LoginPromt (запрос прав пользователя) объекта IBDataBase1 установить равным False;

2. в свойстве Params указать две строки:

user\_name=SYSDBA

password=masterkey

Теперь при запуске программы диалоговое окно запроса параметров подключения появляться не будет.

**Задание:**

1. Создать модуль данных;

2. Добавить в модуль данных все таблицы БД.

3. Создать автоматическое подключение к БД.

4. Сохранить проект.

**Лабораторная работа № 2 (2 часа)**

**Тема: «Разработка формы»**

**Цели работы:** изучить компоненты TMainMenu, TDBGrid, TРanel, TbitBtn;

**Теоретические сведения**

**1. Использование компонента TMainMenu**

Откройте созданный проект. Создайте новую форму командой File → New → Form.

Измените следующие свойства пустой формы, заданные по умолчанию:

Caption - указывается заголовок формы (например, «База данных Аэропорт»;

Name – указывается имя формы (например, «Main»).

Поместите на форму компонент TMainMenu (компонент расположен на панели Standart).

**2. Использование компонентов TDBGrid, TРanel, TbitBtn**

Создайте новую форму. Задайте заголовок и имя формы.

Поместите на форму панель TPanel (компонент расположен на панели Standart), установите значение alBottom в свойстве Align. Очистите свойство Caption.

Поместите на форму компонент PageControl (панель Win32), установите значение alClient. Добавьте новую страницу (с помощью контекстного меню). Задайте имя страницы.

Чтобы пользователь программы мог менять высоту панелей, поместите на форму вешку разбивки TSplitte г (панель Additional), в ее свойство Align поместите значение alBottom, в свойство Beveled – True, и в свойство Height - 5.

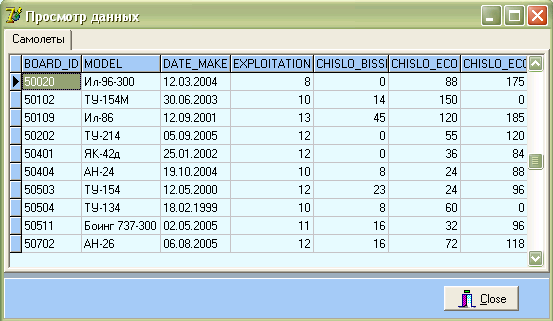
Поместите на верхнюю панель компонент TDBGrid (панель Data Controls) и в свойстве Align установите значение alClient.

Поместите на нижнею панель кнопку TBitBtn (вкладка Additional). В свойство Kind кнопки поместите значение bkClose.

С помощью команды File → Use Unit (Alt+F11) свяжите модуль главного окна с модулем данных.

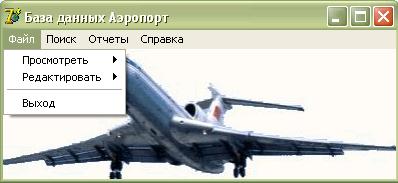
Для верхней сетки DBGridl раскройте список свойства DataSource и выберите в нем необходимый источник данных, сетка тут же наполнится данными из выбранной таблицы.

Вид окна к этому моменту:



**Задание**

1. Создайте меню своего приложения. Например:



2. Создать форму для просмотра данных из таблиц базы данных (основных, имеющих существенное значение при работе с приложением пользователю).

**Лабораторная работа № 3 (4 часа)**

**Тема: «Создание объектов-полей»**

**Цель работы:** создать объекты-поля.

**Теоретические сведения**

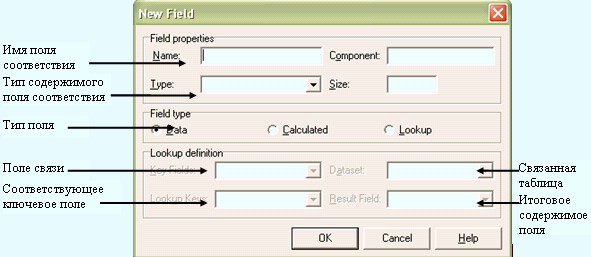
**1. Создание объектов-полей**

Создание объектов-полей для каждого, используемого в программе НД можно считать признаком хорошего стиля программирования, так как они упрощают доступ к данным и дают программисту дополнительные возможности.

При этом следует помнить, что если для НД создан хотя бы один объект-поле, соответствующий НД будет содержать значения только тех полей, для которых созданы объекты; и наоборот: если для НД не созданы объекты-поля, он содержит данные из всех полей соответствующей таблицы.

Перейдите на модуль данных. Двойным щелчком на компоненте IBTable1 (например, Airplains) откройте окно редактора полей, щелкните в окне редактора правой кнопкой мыши и в контекстном меню выберите команду Add all fields - окно редактора заполнится списком всех полей выбранной таблицы. Этот список означает, что теперь для каждого поля НД создан специальный объект со своими свойствами, методами и событиями. Если вы щелкнете на любом поле в окне редактора полей, то в окне инспектора объектов станут доступными свойства и события объекта-поля. По умолчанию Delphi присваивает объекту-полю имя, полученное сцеплением имени НД и имени поля.

Вновь двойным щелчком на компоненте IBTable1 (Airplains) откройте окно редактора полей и щелчком правой кнопкой мыши вызовите его контекстное меню. Выберите в меню команду New Field, чтобы открыть окно конструктора нового поля.



С помощью конструктора нового поля можно создать в НД новые поля одного из трех типов (тип определяется переключателями в группе Field type):

- Data - поля содержат произвольные данные;

- Calculated - поля содержат вычисляемые данные;

- Lookup - поля содержат данные из таблицы подстановки.

Поле первого типа будет отображаться в сетке пустой колонкой, которую можно заполнить в обработчике события OnGetText этого поля. Вычисляемое поле заполняется в обработчике события OnCalcFields набора данных. Для заполнения подстановочного поля данные берутся из нужного поля другого НД.

**2. Обработчик события OnGetText**

Откройте окно конструктора нового поля (для таблицы Rout). Создайте поле Marshrut типа Data, которое, например, будет содержать текст, составленный из трех подстановочных полей – Point\_otprav, Transit и Point\_prib.

Чтобы данные отображались в таком поле, воспользуемся обработчиком события OnGetText созданного объекта-поля Marshrut.

Перейдите к форме модуля данных, щелкните на поле Marshrut и откройте вкладку Events в окне инспектора объектов. Дважды щелкнув в правой части строки события OnGetText, напишите такой обработчик:

procedure TDM.RoutMarshrutGetText(Sender: TField; var Text: String; DisplayText: Boolean);

begin

if RoutTransit.Value = ' ' then

begin

Text := RoutPoint\_otprav.Value + RoutPoint\_prib.Value;

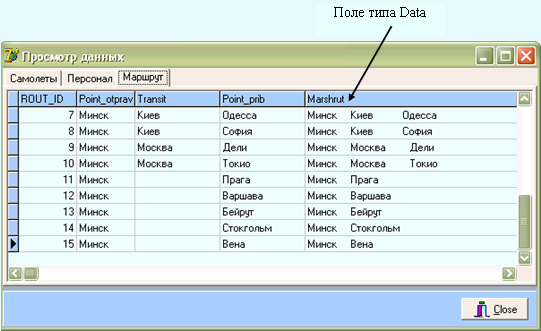
end

else

Text := RoutPoint\_otprav.Value + RoutTransit.Value + RoutPoint\_prib.Value;

end;

Результат такой процедуры во время выполнения программы будет выглядеть следующим образом:



Событие OnGetText возникает всякий раз, когда программа обращается к полю с целью отображения его содержимого в любом визуализирующем компоненте. Программист в обработчике этого события должен поместить нужную строку в переменную Text.

**3. Объекты для вычисляемых полей**

Вычисляемые поля не входят в состав конкретной таблицы БД, но могут быть присоединены к НД, связанному с этой таблицей. Они предназначены для отображения данных, которые вычисляются в ходе выполнения программы обычно с помощью значений из других полей той же записи. После присоединения вы­числяемого поля к НД оно становится во всем подобно обычным объектам-полям, связанным с реальными полями таблицы БД, за одним исключением: такие поля нельзя редактировать, а при вводе новой записи - помещать в них какое-либо значение.

Для создания вычисляемого поля нужно открыть окно New Field редактора полей и, заполнив строки Name, Type и Size, установить переключатель Calculated и закрыть окно щелчком на кнопке ОК. Заполнение вычисляемых полей осуществляется в обработчике события OnCalcFields набора данных.

Напишите такой обработчик события OnCalcFields:

procedure TDM.AirplainsCalcFields(DataSet: TDataSet);

begin

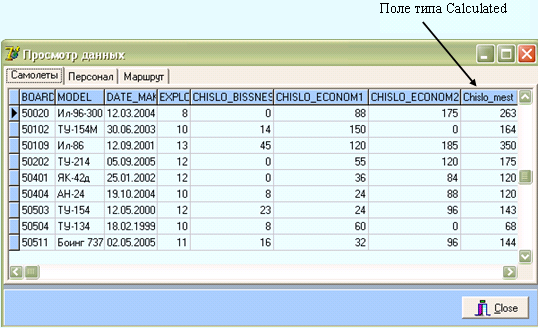
dm.AirplainsChislo\_mest.Value := dm.AirplainsCHISLO\_BISSNES.Value +

dm.AirplainsCHISLO\_ECONOM1.Value +

dm.AirplainsCHISLO\_ECONOM2.Value;

end;

Результат такой процедуры во время выполнения программы будет выглядеть следующим образом:



**4. Подстановочные поля**

Подстановочные поля создаются с целью упрощения понимания пользователем данных таблиц, т.е. вместо ID поля можно показать в сетке таблицы его значение, взятое из другого НД.

Для этого необходимо создать новое поле, указав тип – Lookup. В поле Key Fields указывается поле связи из исходной таблицы, в Dataset – выбрать таблицу, из которой будут взяты данные, в Lookup Keys – ключевое поле, по которому устанавливается связь между таблицами, в Result Field – результирующее поле, значения которого будут отображаться в создаваемом.

**Задание**

1. Создать объекты-поля для всех таблиц.

2. Создать необходимые подстановочные, вычисляемые поля и поля с произвольными данными.

**Лабораторная работа № 4 (1 час)**

**Тема: «Создание объектов-столбцов. Поля»**

**Цель работы:**

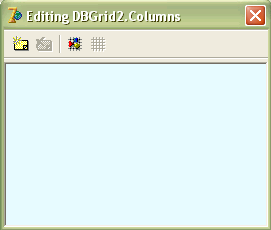
1. создать объекты-столбцы.

2. изучить свойства объектов-полей.

**Теоретические сведения**

**1. Создание объектов-столбцов**

Объекты-столбцы - специальные компоненты, облегчающие управление отображением данных. С их помощью можно изменять шрифт и цвет колонок, формировать их заголовки и т. д. Для создания объектов-столбцов нужно вызвать редактор столбцов (дважды щелкните мышью на сетке DBGrid).



Чтобы создать объект-столбец, нужно щелкнуть на кнопку Add New редактора столбцов или нажать клавишу Ins. Выделив (щелчком) появившийся в окне компонент ТColumn, мы с помощью инспектора объектов можем менять его свойства.

Раскройте список свойства FieldName и выберите поле, которое необходимо видеть в сетке DBGrid. Раскройте список вложенных свойств сложного свойства Title и в его вложенное свойство Caption введите заголовок столбца.

**2. Проверка правильности введенного в поле значения**

Для контроля за правильностью вводимых в поле значений можно использовать события OnValidate или OnSetText объекта-поля. Оба события наступают после изменения значения поля, но до его запоминания в таблице.

В обработчике OnValidate в случае обнаружения неверного значения программа должна предотвратить его запоминание, создав исключительную ситуацию или обратившись к глобальной процедуре Abort. Например, поле EXPLOITATION содержит данные по сроке эксплуатации самолета и, естественно, не может иметь отрицательных значений. Для контроля над этим можно написать такой обработчик:

procedure TDM.AirplainsEXPLOITATIONValidate(Sender: TField);

begin

if dm.AirplainsEXPLOITATION.Value < 0 then

begin

ShowMessage ('Значение не может быть отрицательным!');

Abort

end

end;

Следует учесть, что, отказавшись от запоминания неверного значения в таблице, программа, тем не менее, оставляет НД в состоянии редактирования. Это означает, что пользователь не может покинуть это поле до тех пор, пока не введет в него правильное значение. Если бы мы решили защитить поле от любого изменения с помощью показанного ниже обработчика, то обнаружили бы, что при попытке изменить значение поля программа попросту «зацикливается», не давая пользователю покинуть редактируемое поле:

procedure TDM.AirplainsEXPLOITATIONValidate(Sender: TField);

begin

ShowMessage ('Значение не должно изменяться!');

Abort

end;

В данном случае использовать событие OnSetText значительно разумнее, так как в нем можно просто игнорировать неверные значения и не создавать исключительную ситуацию. Дело в том, что в этом обработчике программа должна присвоить полю новое значение или оставить поле без изменения:

procedure TDM.AirplainsEXPLOITATIONSetText(Sender: TField; const Text: String);

begin

if StrToInt (Text)<0 then

ShowMessage ('Значение не может быть отрицательным!');

else

dm.AirplainsCHISLO\_ECONOM1.Value:=StrToInt(Text);

end;

Обратите внимание: можно не бояться того, что при преобразовании текста в значение с плавающей запятой может возникнуть исключение. При вводе значения в поле какого-то типа, кроме символьного, программа автоматически отвергает любые символы, не соответствующие символьному представлению этого поля.

**Задание**

1. Создать объекты-столбцы для таблиц из лабораторной работы 2.

2. Осуществить проверку правильности вводимых в поле значений.

3. Самостоятельно изучить Свойства, Методы.

**Лабораторная работа № 5 (3 часа)**

**Тема: «Навигация. Фильтрация записей. Сортировка. Поиск записей в НД»**

**Цели работы:**

1. изучить способы навигации по наборам данных.

2. сортировка данных.

3. фильтрация записей в НД: свойство Filter.

4. рассмотреть принципы работы методов Locate и Lookup.

**Теоретические сведения**

**1. Сортировка записей**

Сортировка записей в таблице осуществляется с помощью индексов. Если необходимо отсортировать данные не по индексу, то во время программы можно подключить вторичный индекс, что позволит показывать записи в различном порядке в зависимости от действий пользователя.

Подключение к таблице вторичного индекса осуществляется присвоением нового значения свойству IndexName. Для клиент-серверных СУБД вместо свойства IndexName обычно используется свойство IndexFieldNames, позволяющее указывать список названий полей, описывающих индекс, через «;».

Свойства IndexName и IndexFieldNames не допускают совместного использования. При внесении значения в одно из этих свойств другое свойство автоматически очищается.

**Пример.** Отсортировать таблицу Member\_crews по ФИО.

Для этого можно воспользоваться контекстным меню, для пункта меню «ФИО» которого можно написать следующий обработчик:

procedure TForm31.N1Click (Sender: TObject);

begin

dm.Member\_crews.IndexFieldNames := 'FIO\_MEMBER';

end;

**2. Навигация по НД**

Под курсором набора данных понимается указатель текущей записи в конкретном НД. Текущая запись – это та запись, над которой в данный момент времени можно выполнять какие либо операции (удаление, изменение, чтение значений, содержащихся в полях записи).

С помощью методов First, Last, Next, Prior, MoveBy, FindFast, FindLast, FindNext, FindPrior можно изменять положение курсора и, следовательно, выбирать нужную запись.

Для выполнения действий по последовательному перебору записей, начиная от некоторой стартовой записи и до конца НД, используют цикл while…do или repeat…until.

Например, в случае перемещения в НД от начала к концу можно написать следующий код:

with Table1 do

begin

First;

while not EOF do

begin

{какие-либо действия над очередной записью}

Next;

end;

end;

Если использовать repeat…until то:

with Table1 do

begin

First;

repeat

{какие-либо действия над очередной записью}

until not FindNext;

end;

**Пример.** Вывести названия должностей в алфавитном порядке.

Добавьте на форму компонент ComboBox, в котором будут отображаться данные (названия должностей) из таблицы Post. Для этого необходимо написать следующий обработчик:

procedure TForm31.FormCreate(Sender: TObject);

begin

with dm.Post do

begin

First;

while not EOF do

begin

ComboBox1.Items.Add(dm.PostPost\_name.Value);

Next;

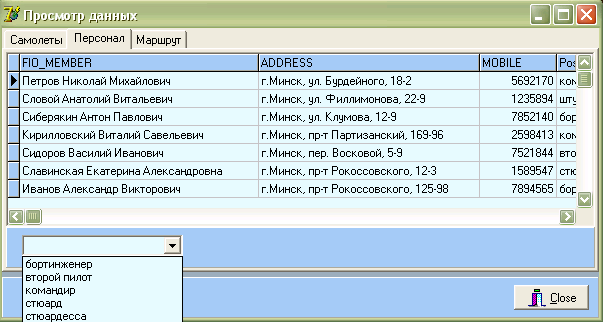
end;

end;

ComboBox1.Sorted:= true;

end;

Результат выполнения процедуры:



**3. Свойство Filter**

Просматривать таблицы, содержащие большое число записей, неудобно. Часто требуется выделить только группу записей, выбранную по некоторому критерию. Для этого набор данных надо отфильтровать по заданному условию. Такое условие записывается в свойство Filter набора данных в виде текстовой строки и содержит логическое выражение, в котором используются поля текущего набора данных, константы, логические операции и операции отношения.

Чтобы разрешить фильтрацию значений, надо свойство набора данных Filtered установит равным True.

**Пример.** Отфильтровать сведения о членах экипажа по должности.

Для этого можно написать следующий обработчик:

procedure TForm31.ComboBox1Select(Sender: TObject);

var x, y:string;

begin

with dm.Post do

begin

DM.Post.First;

x:=combobox1.Text;

while not(EOF) do

begin

if (dm.PostPost\_name.Value=x)

then

y:=IntToStr(dm.PostPost\_id.Value);

DM.Post.Next;

end;

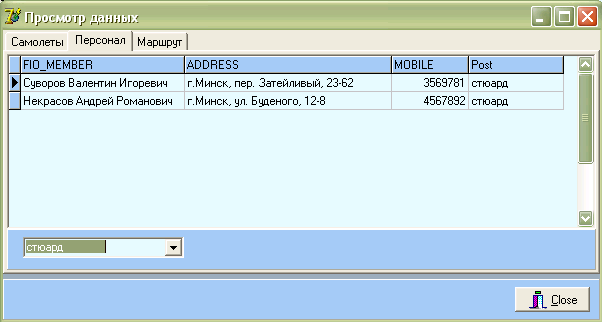
end;

dm.Member\_crews.Filter := 'Post\_id = '+QuotedStr(y)+'';

dm.Member\_crews.Filtered := true;

end;

Результат выполнения процедуры:



**4. Поиск записей в НД**

**Метод Locate** ищет первую запись, удовлетворяющую критерию поиска, и если такая запись найдена, делает ее текущей. В этом случае в качестве результата возвращается значение True. Если запись не найдена, возвращается значение False и курсор не меняет своего положения.

function Locate (const KeyFields: String;

const KeyValues: Variant;

Options: TLocaleOptions):Boolean

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

Критерии поиска задаются в вариантном массиве KeyValues так, что i-e значение в KeyValues ставиться в соответствие i-му полю в KeyFields.

Параметр Options указывает, необязательные значения режимов поиска.

- IoCaseInsensitivite – поиск ведется без учета возможной разницы регистра букв в текстовых полях и в критерии поиска.

- IoPartialKey – запись считается удовлетворяющей условию поиска, если она содержит часть поискового контекста; например, удовлетворяющим контексту "Диа" будут признаны записи со значениями в поле поиска "Диалект", "Диалог-МИФИ", "ДиаСофт" и т.д.

Метод Locate производит поиск по любому полю независимо от того входит оно в состав какого-либо индекса или нет. Если поле (поля) поиска входят в какой-либо индекс, Locate автоматически использует его, что существенно увеличивает скорость поиска.

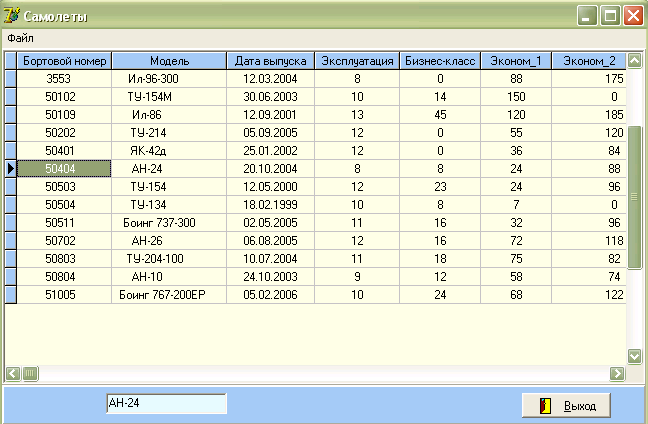
**Пример.** Добавьте на форму своего проекта компонент Edit. Добавьте этому компоненту обработчик события OnChange и в нем напишите следующее:

procedure TForm2.Edit2Change (Sender: TObject);

begin

if dm.Airplains.Locate ('MODEL', Edit2.Text, []) then DBGrid1.SetFocus;

end;



Теперь при воде в компонент нужной модели самолета программа отыщет в наборе Airplains первую запись с названием такой модели.

**Метод Lookup** находит запись, удовлетворяющую условию поиска, но не делает ее текущей, а возвращает значения некоторых ее полей. Независимо от результата поиска записи указатель текущей записи в НД не изменяется. В отличие от метода Locate метод Lookup осуществляет поиск только на точное соответствие критерию поиска значения поля поиска записи.

function (const KeyFields: String;

const KeyValues: Variant;

ResultFields: String): Variant

В параметре KeyFields указывается список полей, по которым необходимо осуществить поиск (если задается значение по нескольким полям, соседние поля разделяются точкой с запятой).

Параметр KeyValues определяет поисковые значения полей, список которых содержится в параметре KeyFields.

В параметре ResultFields перечисляются поля, значения которых требуется получить в случае успешного поиска. Тип результата поиска – Variant или вариантный массив.

Если имеется несколько поисковых полей, каждому i-му полю в KeyFields ставиться в соответствие i-e значение в KeyValues. Если поиск ведется по одному полю, его поисковое значение можно указывать в качестве KeyValues непосредственно; в случае нескольких полей их необходимо приводить к типу вариантного массива при помощи функции преобразования VarArrayOf.

Как и в методе Locate, в качестве поисковых полей можно указывать поля как входящие в какой-либо индекс, так и не входящие в него. Если в результате поиска запись не найдена, метод Lookup возвращает Null.

Если поиск оказался успешным, Lookup возвращает из найденной записи значения полей, список которых содержит ResultFields. При этом размерность результата зависит от того, сколько результирующих полей указано в ResultFields.

**Пример.** Добавьте следующие компоненты: TEdit, 2 компонента TLabel, TButton – в свойстве Caption наберите слово «Найти».

Добавьте обработчик события OnClick и напишите в нем следующее:

procedure TForm2.Button1Click (Sender: TObject);

var LookupResult:variant;

begin

LookupResult := dm.Airplains.Lookup ('BOARD\_ID', Edit2.Text, 'MODEL;DATE\_MAKE');

if VarType (LookupResult) = varNull then

ShowMessage ('Нет самолета с номером ' + Edit2.Text)

else if VarType (LookupResult) = varEmpty then

ShowMessage ('Поиск не произведен')

else if VarIsArray (LookupResult) then

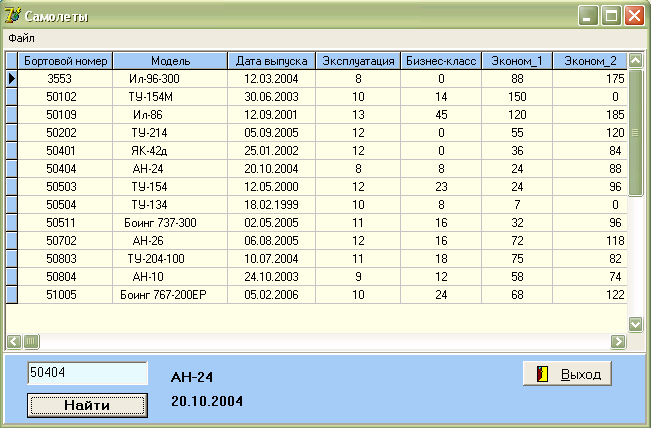
begin

Label1.Caption:= LookupResult[0];

Label2.Caption:= LookupResult[1];

end;

end;



**Задание:**

1. Реализовать сортировку данных.

2. Реализовать алгоритмы навигации для прохода по НД от конца к началу.

3. Придумать несколько примеров на фильтрацию записей.

4. Продемонстрировать работу методов Locate и Lookup.

**Лабораторная работа 6 (2 часа)**

**Тема: «Добавление, редактирование и удаление записей»**

**Цель работы:** изучить способы добавления, редактирования и удаления записей.

**Теоретические сведения**

**1. Добавление записей**

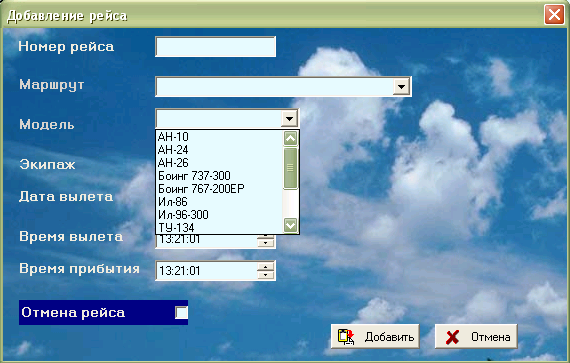
Организовать добавление новых записей в таблицу с помощью элемента TDBGrid довольно сложно. Это связано с тем, что в таблицах могут храниться значения ключей из других таблиц. Добавлять такие значения вручную, не зная, какому названию соответствует значение, бессмысленно.

В системе Delphi предусмотрена возможность связывания разных таблиц по ключевым полям, но она обычно используется на этапе просмотра и редактирования данных.

Добавление новых записей проще осуществлять в отдельных формах программы.

Для создания такой формы можно использовать следующие компоненты: Label, Edit, ComboBox, CheckBox, RadioButton, RadioGroup, ListBox и т.д.

Форма для заполнения таблицы «Рейсы» может выглядеть следующим образом:



Данные о маршруте берутся из таблицы «Маршрут», о модели самолета – из таблицы «Самолеты», о экипаже – из таблицы «Состав экипажа». Получается, если вводить эти сведения по кодам (ID данных полей), то обычный пользователь не сможет запомнить такое количество кодов. А открывать и работать одновременно с несколькими таблицами будет не удобно.

Для добавления данных в таблицу, связанную с несколькими, необходимо написать программный код, в котором можно программно определять ID полей связанных таблиц.

Процесс записи данных из полей формы в таблицу:

DataModule1.Table1.Insert – добавляет пустую запись в набор данных Table1;

DataModule1.Table1.Fields[1].AsString := Edit1.Text – выбирается значение из поля формы;

DataModule1.Table1.Post – метод, который записывает изменения в БД.

**Пример.** Добавить в таблицу «Trip» новую запись о рейсе:

procedure TForm4.BitBtn2Click (Sender: TObject);

begin

DM.Trip.Insert;

DM.Trip.Fields[0].AsInteger:= StrToInt(LabeledEdit1.Text);

DM.Trip.Fields[1].AsInteger:= StrToInt(LabeledEdit2.Text);

DM.Trip.Fields[2].AsString:= LabeledEdit3.Text;

DM.Trip.Fields[3].AsString:= ComboBox4.Text;

DM.Trip.Fields[4].AsDateTime:= dateTimePicker1.Date;

DM.Trip.Fields[5].AsDateTime:= dateTimePicker2.Time;

DM.Trip.Fields[6].AsDateTime:= dateTimePicker3.Time;

if CheckBox1.Checked then

DM.Trip.Fields[7].AsString:= 'да'

else DM.Trip.Fields[7].AsString:= 'нет';

DM.Trip.Post;

end;

**2. Изменение записей**

Для модификации данных используется метод Edit:

DataModule1.Table1. Edit

**Пример.** Изменить запись о рейсе:

procedure TForm4.BitBtn3Click(Sender: TObject);

begin

DM.Trip.Edit;

DM.Trip.Fields[0].AsInteger:= StrToInt(LabeledEdit1.Text);

DM.Trip.Fields[1].AsInteger:= StrToInt(LabeledEdit2.Text);

DM.Trip.Fields[2].AsString:= LabeledEdit3.Text;

DM.Trip.Fields[3].AsString:= ComboBox4.Text;

DM.Trip.Fields[4].AsDateTime:= dateTimePicker1.Date;

DM.Trip.Fields[5].AsDateTime:= dateTimePicker2.Time;

DM.Trip.Fields[6].AsDateTime:= dateTimePicker3.Time;

if CheckBox1.Checked then

DM.Trip.Fields[7].AsString:= 'да'

else DM.Trip.Fields[7].AsString:= 'нет';

DM.Trip.Post;

end;

**3. Удаление записей**

Для удаления данных используется метод Delete:

DataModule1.Table1. Delete

**Задание**

1. Организовать добавление, редактирование и удаление данных с помощью контекстного меню.

2. Создать одну форму для добавления и модификации данных таблицы, связанной с несколькими (2-3).

2. Организовать удаление данных из БД.

**Лабораторная работа 7 (2часа)**

**Тема: «Компонент TQuery»**

**Цель работы:** изучить особенности компонента TQuery.

**Теоретические сведения.**

Элемент TQuery позволяет получать доступ к одной или более таблиц из базы данных, используя SQL-запросы. Также его можно использовать для работы с различными серверами баз данных (Sybase, SQL Server, Oracle, Informix, DB2, и InterBase), с локальными таблицами (Paradox, InterBase, dBASE, Access и FoxPro), а также БД, которые совместимы с ODBC. Компонент TQuery удобно использовать для одновременного доступа к строкам и столбцам нескольких таблиц.

Компонент TQuery может выполнять не более одного SQL-запроса одновременно. Синтаксис самого запроса зависит от используемой базы данных, которые в своем большинстве совместимы со стандартом SQL-92.

**Пример.**

Поместите на модуль данных элемент TQuery и DataSource. В свойстве SQL элемента TQuery напишите необходимый запрос.

Создайте новую форму и поместите на нее элемент TDBGrid. Для обеспечения вывода значений таблиц в размещенный на форме элемент DBGrid следует присвоить значение DataSource созданного запроса свойству DataSource.

Также можно организовать выполнение данного запроса по нажатию кнопки.

Для этого разместим на форме компонент TMemo и кнопку Button, при нажатию на которую будет совершаться отсылка запроса элементу TQuery. Для этого нужно добавить обработчик события OnClick и в него следующие строки:

procedure TForm32.Button1Click (Sender: TObject);

begin

dm.Q\_Post.SQL := memo1.Lines;

dm.Q\_Post.Active := true;

dm.Q\_Post.Open;

end;

После написания текста запроса в поле Memo, по нажатию кнопки будет выполнен данный запрос.

**Задание:**

1. Продемонстрировать работу компонента TQuery.

**Лабораторная работа №8 (2 часа)**

**Тема: «Создание отчётов. Технология Quick Report»**

**Цели работы:**

1. установка пакета компонентов Qreport;

2. принципы построения отчёта.

**Теоретические сведения**

В большинстве случаев данные, которые хранятся в таблицах БД, необходимо тем или иным способом опубликовать, создавая так называемые отчёты, В Delphi имеется набор компонентов, предназначенных для решения этой задачи. Чтобы ознакомиться с их списком, можно взглянуть на вкладку QReport, где находится порядка двадцати компонентов. По умолчанию, данные компоненты не устанавливаются. Если вы хотите работать с технологией Quick Report, то вам необходимо установить пакет DCLQRT70.BPL. Для этого выберите команду Component → Install Packages, щёлкните на кнопке Add и укажите этот файл, который находится в папки BIN каталога размещения Delphi.

Центральным компонентом при построении отчёта является компонент TQuickRep, определяющий свойства отчёта в целом. Этот компонент всегда помещается на отдельную форму, образуя с её помощью визуальную среду конструирования отчёта.

Создадим отчёт, который будет вызываться с помощью кнопки Печать, расположенной на главной форме приложения. Наш отчёт будет представлять собой товарную накладную на отпуск книг со склада оптового поставщика. Основные этапы работы, необходимые для создания отчёта в Delphi

Выполнить:

1. Создайте в проекте пустую форму, расположите на ней компонент TQuickRep, измените имя формы на fmPrint и сохраните в файле fmPrintU. Эта форма не будет создаваться автоматически в момент старта программы, поэтому после её присоединения к проекту выберите команду Project -> Options и перенесите её в список доступных форм Avialable forms.

2. С помощью команды File → Use unit свяжите новую форму с модулем данных, после чего поместите в свойство DataSet компонента QuickReport значение DM.Move. Следует заметить, что это свойство всегда должно быть заполнено и указывать на ту таблицу, которая будет содержать основную информацию.

3. Поместите на отчёт полосу TQRBand. Эта полоса по умолчанию имеет тип rbTitle, то есть появляется всего один раз в самом начале отчёта. Используем её для размещения заголовка накладной, для чего установим её высоту Height=99 и поместим на неё следующие комопненты:

TQRLabel со свойством Caption = Товарная накладная №, Тор = 8, Left = 176, Font.Size = 16, Font.Style.fsBold = True;

TQRDBText со свойствами DataSet = DM.Nakls, DataField = NakllD, Top = 8, Left = 424, Font.Size = 16, Font.Style.fsBold = True;

TQRDBText : Caption = от, Top = 8, Left = 472, Font.Size = 16, Font.Style.fsBold =true

TQRDBText: DataSet = DM.Nakls, DataField = NDate, Top = 8, Left = 504, Font.Size = 16, Font.Style.fsBold = True;

TQRLabel: Caption = Отправитель, Top = 40, Left =16, Font.Size =12, Font.Style.fsBold = True;

TQRLabel: Caption = Получатель, Top = 64, Left = 16, Font.Size = 12, Font.Style.fsBold = True;

TQRDBText: DataSet = DM.Nakls, DataField = Firm, Top = 64, Left = 120, Font.Size =10, Font.Style.fsUnderline = True.

4. Чтобы увидеть отчёт в окне предварительного просмотра, щёлкните на компоненте QuickRepl правой кнопкой мыши и выберите в контекстном меню команду Preview.

5. Теперь необходимо поместить в отчёт заголовки его колонок. Для этого используется компонент TQRBand со свойством BandType = rbColumHeader. Кроме того, установите для компонента свойства Font.Size = 10 и Font.Style.fsBold = True, а также поместите значение True в свойства Frame.DrawBottom, Frame.DrawLeft, Frame.DrawRight и Frame.DrawTop, чтобы полоса очерчивалась прямоугольной рамкой;

6. Разместите на полосе четыре компонента TQRLabel со следующими свойствами:

QRLabeB: Caption = Название, автор, издательство, Left =163, Тор =10;

QRLabel4: Caption = Количество, Left = 499, Top =10, AutoSize = False, Width = 66, Alignment = taCenter;

QRLabel5: Caption = Цена, Left = 567, Top = 10, AutoSize = False, Width = 66, Alignment = taCenter;

QRLabel6: Caption = Сумма, Left = 632, Top = 10, AutoSize = False, Width = 66, Alignment = taCenter.

7. Чтобы в отчёте появился связанный с накладной список книг, поместите в него компонент TQRSubDetail со значениями свойств DataSet = DM.Move, Font.Size = 8, Height = 15, Frame.DrawLeft = True, Frame.DrawRight = True, Frame.DrawBottom = True;

8. На полосе TQRSubDetail поместите четыре компонента TQRDBText для отображения содержимого четырёх колонок отчёта:

QRDBText4: AutoSize = False, DataSet = DM.Move, DataField = Name, Frame.DrawRight = True, Height = 15, Left = 0, Top = 0, Width = 498;

QRDBText5: AutoSize = False, DataSet = DM.Move, DataField = MQuan, Frame.DrawRight = True, Height = 15, Left = 499, Top = 0, Width = 70, Alignment = taCenter;

QRDBText6: AutoSize = False, DataSet = DM.Move, DataField = MPrice, Frame.DrawRight = True, Height = 15, Left = 567, Top = 0, Width = 70, Alignment = taCenter;

QRDBText7: AutoSize = False, DataSet = DM.Move, DataField = Summa, Frame.DrawRight = True, Height = 15, Left = 638, Top = 0, Width = 70, Alignment = taCenter;

9. Для размещения в отчёте итоговой информации используется полоса RQRBand типа rbSummary. Поместите на эту полосу:

TQRLabel: Тор = 10, Left = 10, Caption = Итого:;

TQRSysData: Top = 10, Left = 56, Data = qrsDetailCount;

TQRLabel: Top =10, Left =10, Caption = Наименование на сумму;

TQRDBText: Top = 10, Left = 248, DataSet = DM.Nakls, DataField = NSum.

10. Чтобы вызвать отчёт в программе, создайте обработчик нажатия копки "Печать", как показано ниже:

begin

fmPrint := TfmPrint.Create(Self);

fmPrint.QuickRepl .Preview;

fmPrint.Free; end;

**Задание:**

Привести пример создания отчета.

**Лабораторная работа №9 (2 часа)**

**Тема: «Создание отчетов. Rave-проектировщик»**

**Цель работы:** изучать возможности создания отчетов Rave-проектировщиком.

**Теоретические сведения:**

Отчеты используются в самых разных программах. Особенно активно они формируются во всевозможных приложениях, связанных с автоматизацией делопроизводства, когда документы хранятся в электроном виде в базе данных, но их требуется регулярно выводить на печать. Отчеты в системе Delphi 7 ориентированы в первую очередь на печать информации из таблиц баз данных.

В системе Delphi 7 отчет - это виртуальный образ бумажного листа, который в дальнейшем без изменений воспроизводится на принтере. В седьмой версии Delphi появился набор компонентов Rave Reports (панель Rave), который заменил морально устаревший набор QReports. Эти компоненты позволяют подготовить произвольное число виртуальных страниц отчетов в простом визуальном редакторе. Компоненты набора обладают множеством характеристик, что позволяет детально настроиться на возможности конкретного принтера.

Для создания отчета необходимо на форму добавить компоненты ТТаЫе и TRvDataSetConnection. В свойстве DataSet указывается название объекта-таблицы.

Для проектирования внешнего вида потребуется вызвать Rave-проектировщик командой Tools>Rave Designer. Окно проектировщика состоит из четырех основных частей. В первой части расположены кнопки управления и панели компонентов. В центре можно видеть проектируемый отчет. В левой части находятся редактор свойств текущего объекта, в правой разработчику доступен Просмотрщик объектов.

Дайте команду file → New Data Object (Файл → Новый объект данных) и в окне типов связи выберем строку Direct Data View (Прямой обзор данных).

В следующем окне будет предложено выбрать соединения с СУБД, доступные на данный момент в среде. В нашем случае это единственная связь TRvDataSetConnection 1. Теперь структуру установленной связи можно посмотреть в правой части Rave-редактора.

Для подготовки табличной формы отчета необходимо, выполнить команду Tools → Report Wizards → Simple Table и выбрать в диалоговом окне объекта DataViewl. В следующем окне будет предложено отметить поля таблицы, которые желательно видеть в отчете.

На следующем этапе можно будет отсортировать, ввести название отчета, указать границы печатаемой страницы и задать шрифты различных разделов. После нажатия кнопки Generate отчет будет создан и размещен в центре виртуальной страницы.

В одном проекте может быть несколько различных отчетов (их модно просмотреть в правом окне в элементе Report Library).

Теперь создайте шаблон отчета, для этого необходимо сохранить в подходящей папке. Файл проекта получит расширение .rav .

Вернемся к форме, добавьте на нее компонент TRvProject. Свойство Project File настройте на подготовленный файл отчета.

Поместите на форму кнопку и в обработчике нажатия запишите следующий оператор:

RvProject.ExecuteReport ('MyReport');

В нем происходит обращение к методу ExecuteReport компонента TRvProject, ио которому строится и вызывается для просмотра и печати указываемый отчет.

Визуальный Rave-проектировщик

В ходе проектирования отчета разработчику доступны наборы компонентов следующих панелей:

- Drawing - графические элементы оформления;

- Bar Code - штрих-кодовые графические элементы;

- Standard - стандартные элементы отчета многостраничные подписи, картинки, средства настройки шрифтов;

- Report- компоненты, обеспечивающие связь отчета с базами данных;

- Zoon-средства масштабирования виртуальных страниц;

- Color- цветовые настройки отчета;

- Lines- средства построения линий;

- Fills- элементы заполнения фона;

- Fonts-средства настройки шрифтов различных частей отчета;

- Alignment- компоненты, позволяющие гибко выравнивать части отчета. Rave-компоненты:

Компоненты связи с источниками данных:

- TrvCustomConnection - поставляет в отчет данные из обычных файлов или массивов в памяти программы;

- TRvDataSetConnection - поставляет в отчет данные, получаемые от таблиц BDE; TrvQuerySetConnection - поставляет в отчет данные, получаемые от запроса BDE.

**Задание:**

Привести пример создания отчета.