Министерство образования и науки, молодежи и спорта Украины
Севастопольский национальный технический университет

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторная работа №3

«Основы разработки приложений для работы с базами данных в среде *C++Builder*»

по дисциплине

«**Персональные компьютеры в вычислительной практике**»

для студентов дневной и заочной формы обучения

по направлению 6.050901 — Радиотехника.

Севастополь
2013 г.

УДК.519.72

Методические указания «Лабораторная работа №3 Основы разработки приложений для работы с базами данных в среде *C++Builder* по дисциплине «Персональные компьютеры в вычислительной практике» для студентов дневной и заочной форм обучения по направлению 6.050901 — Радиотехника / Сост. А.В. Лукьянчиков — Севастополь: Изд-во СевНТУ, 2013. — 22 с.

Методические указания рассмотрены и утверждены на заседании кафедры радиотехники и телекоммуникаций (протокол № _ от __ ____ 2013 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент: канд. техн. наук, доцент Савочкин А.А.

Ответственный за выпуск: заведующий кафедрой радиотехники и телекоммуникаций, доктор техн. наук, профессор Гимпилевич Ю.Б.,

3. ЛАБОРАТОРНАЯ РАБОТА № 3
ОСНОВЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ В СРЕДЕ *C++BUILDER*

3.1. Цель работы: На примере простейщего приложения разобраться в принципах разработки приложений баз данных в среде *C++ Builder.*

3.2. Теоретические сведения

С точки зрения пользователя, база данных — это программа, которая обеспечивает работу с информацией. При запуске такой программы на экране, как правило, появляется таблица, просматривая которую можно найти нужные сведения. Если система позволяет, то пользователь может внести изменения в базу данных, например, добавить новую информацию или удалить ненужную.

С точки зрения программиста, база данных — это набор файлов, в которых находится информация. Разрабатывая базу данных для пользователя, программист создает программу, которая обеспечивает работу с файлами данных.

В состав *C++ Builder* включены компоненты, используя которые программист может создавать программы работы с файлами данных в форматах *dBase*, *Microsoft Access*, *Infomix* и *Oracle* и др.

База данных — это набор, совокупность файлов, в которых находится информация. Программная система (приложение), обеспечивающая работу с базой данных (файлами данных) называется системой управления базой данных (СУБД). Следует обратить внимание, что вместо термина СУБД часто используется термин база данных, при этом файлы данных и СУБД рассматриваются как единое целое.

В зависимости от расположения программы, которая использует данные, и самих данных, а также от способа разделения данных между несколькими пользователями различают локальные и удаленные базы данных. Данные локальной базы данных (файлы данных) локализованы, т. е. находятся на одном устройстве, в качестве которого может выступать диск компьютера или сетевой диск (диск другого компьютера, работающего в сети). Локальные базы данных не обеспечивают одновременный доступ к информации нескольким пользователям. Для обеспечения разделения данных (доступа к данным) между несколькими пользователями (программами, работающими на одном или разных компьютерах) в локальных базах данных используется метод, получивший название "блокировка файлов". Суть этого метода заключается в том, что пока данные используются одним пользователем, другой пользователь не может работать с этими данными, т. е. данные для него закрыты, заблокированы. Несомненным достоинством локальной базы данных является высокая скорость доступа к информации. Приложения работы с локальной базой данных и саму базу данных часто размещают на одном компьютере. *dBase*, *Paradox*, *FoxPro* и *Microsoft Access* — это локальные базы данных.

Удаленные базы данных строятся по технологии "клиент-сервер". Программа работы с удаленной базой данных состоит из двух частей: клиентской и серверной. Клиентская часть программы работает на компьютере пользователя и обеспечивает взаимодействие с серверной программой посредством запросов, передаваемых на удаленный компьютер (сервер), обеспечивая тем самым доступ к данным. Серверная часть программы, работающая на удаленном компьютере, принимает запросы, выполняет их и пересылает данные клиентской программе. Программа, работающая на удаленном сервере, проектируется так, чтобы обеспечить одновременный доступ к базе данных нескольким пользователям. При этом для обеспечения доступа к данным вместо механизма блокировки файлов используют механизм транзакций, Транзакция — это последовательность действий, которая должна быть обязательно выполнена над данными перед тем, как они будут переданы. В случае обнаружения ошибки во время выполнения любого из действий вся последовательность действий, составляющая транзакцию, повторяется снова. Таким образом, механизм транзакций обеспечивает защиту от аппаратных сбоев. Он также обеспечивает

возможность многопользовательского доступа к данным. *Oracle*, *Infomix*, *Microsoft SQL Server* и *InterBase* — это удаленные базы данных.

База данных — это набор однородной и, как правило, упорядоченной по некоторому критерию информации. База данных может быть представлена в "бумажном" или в "компьютерном" виде.

Типичным примером "бумажной" базы данных является каталог библиотеки — набор бумажных карточек, содержащих информацию о книгах. Информация в этой базе однородная (содержит сведения только о книгах) и упорядоченная (карточки расставлены, например, в алфавитном порядке фамилий авторов). Другими примерами "бумажной" базы данных являются телефонный справочник и расписание движения поездов. Компьютерная база данных представляет собой файл (или набор связанных файлов), содержащий информацию, который часто называют файлом данных. Файл данных состоит из записей. Каждая запись содержит информацию об одном экземпляре. Например, каждая запись базы данных "Ежедневник" содержит информацию только об одном экземпляре — запланированном мероприятии или задаче.

Записи состоят из полей. Каждое поле содержит информацию об одной характеристике экземпляра. Например, запись базы данных "Ежедневник" может состоять из полей: "Задача", "Дата" и "Примечание "Задача", "Дата" и "Примечание " — это имена полей. Содержимое полей характеризует конкретную задачу.

Следует обратить внимание, что каждая запись состоит из одинаковых полей. Некоторые поля могут быть не заполнены, однако они все равно присутствуют в записи.

На бумаге базу данных удобно представить в виде таблицы. Каждая строка таблицы соответствует записи, а ячейка таблицы — полю. При этом заголовок столбца таблицы — это имя поля, а номер строки таблицы — номер записи.

Информацию компьютерных баз данных обычно выводят на экран в виде таблиц. Поэтому часто вместо словосочетания "файл данных" используют словосочетание "таблица данных" или просто "таблица".

Разрабатывая программу работы с базой данных, программист не знает, на каком диске и в каком каталоге будут находиться файлы базы данных во время ее использования. Например, пользователь может поместить базу данных в один из каталогов диска C:, *D*: или на сетевой диск. Поэтому возникает проблема передачи в программу информации о месте нахождения файлов базы данных.

В *C++ Builder* проблема передачи в программу информации о месте нахождения файлов базы данных решается путем использования псевдонима базы данных. Псевдоним (*Alias*) — это имя, поставленное в соответствие реальному, полному имени каталога базы данных. Например, псевдонимом каталога *C:\data\Petersburg* может быть имя *Peterburg*. Программа работы с базой данных для доступа к данным использует не реальное имя каталога, а псевдоним. Псевдоним базы данных можно создать при помощи утилиты *BDE Administrator*. Информация о всех зарегистрированных в системе псевдонимах хранится в специальном файле.

Обычно для доступа и манипулирования данными используется соответствующая СУБД. Однако часто возникает необходимость получить доступ к информации, которая находится в базе данных, из прикладной программы. Решить эту задачу можно при помощи компонентов доступа к данным.

C++ Builder предоставляет в распоряжение программиста компоненты, используя которые можно построить приложение, обеспечивающее работу практически с любой базой данных.

Компоненты доступа к данным находятся во вкладках *BDE*. *Data Access*, *ADO* и *InterBase*. Компоненты вкладок *BDE* и *Data Access* для доступа к данным используют процессор баз данных *Borland Database Engine (BDE)*, реализованный в виде набора динамических библиотек и драйверов. Компоненты вкладки *ADO* для доступа к данным используют разработанную *Microsoft* технологию *ADO* (*ActiveX Data Object — ADO*). Компоненты вкладки *InterBase* обеспечивают непосредственный доступ к данным *InterBase*.

Наиболее универсальным механизмом доступа к базам данных является механизм, реализованный на основе *BDE*. Драйверы, входящие в состав *BDE*, обеспечивают доступ как к локальным базам данных (*Paradox*, *Access*, *dBASE*), так и к удаленным серверам баз данных

(*Microsoft SQL Server*, *Oracle)*. Набор драйверов, включенных в *BDE*, определяется вариантом *C++ Builder*.

3.3. Методические рекомендации к выполнению работы

3.3.1. Создание базы данных

Процесс создания базы данных рассмотрим на примере. Создадим локальную базу данных "Ежедневник", которая представляет собой одну-единственную таблицу в формате *Paradox*. Для этого воспользуемся поставляемой вместе с *C++ Builder* утилитой *Database Desktop*.

Запустить *Database Desktop* можно из *C++ Builder*, выбрав в меню *Tools* команду *Database Desktop*, или из *Windows* (команда Пуск | Программы | *C++Builder\Database Desktop*).

Процесс создания базы данных состоит из двух шагов: сначала надо создать псевдоним базы данных, затем — таблицу (в общем случае — несколько таблиц). Псевдоним (*Alias*) определяет расположение таблиц базы данных и используется для доступа к ним.

Для того чтобы создать псевдоним, надо:

В меню *Tools* выбрать команду *Alias Manager*.

В появившемся диалоговом окне *Alias Manager* щелкнуть на кнопке *New*.

Ввести в поле *Database alias* псевдоним создаваемой базы данных — например, *organizer*. Ввести в поле *Path* путь к файлам таблиц базы данных (таблицы будут созданы на следующем шаге). Щелкнуть на кнопке *Keep New* (рис. 3.1). Теперь можно приступить к созданию таблицы.

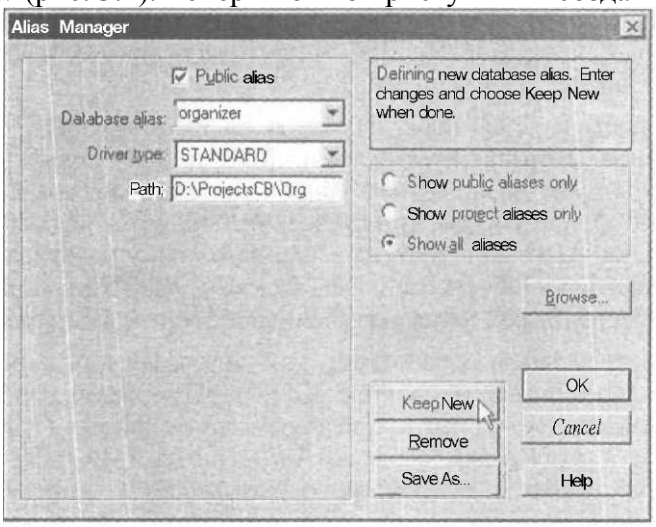

Рисунок 3.1 — Создание псевдонима базы данных

Чтобы создать таблицу, надо в меню *File* выбрать команду *New* | *Table* (рис. 3.2), затем в появившемся диалоговом окне *Create Table* — тип таблицы (рис. 3.3).

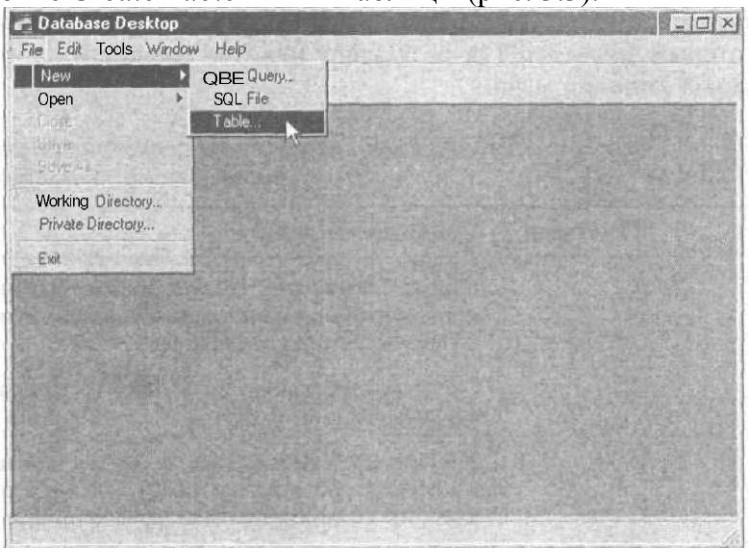

Рисунок 3.2 — Начало работы над новой таблицей

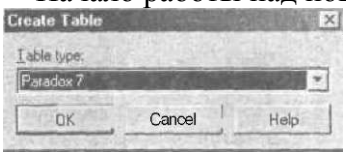

Рис. 3.3 — В списке *Table type* надо выбрать тип создаваемой таблицы (файла данных)

В результате выполнения перечисленных выше действий открывается окно *Create Table*, в котором надо определить структуру таблицы — задать имена полей базы данных и указать их гип и размер (рис. 3.4).

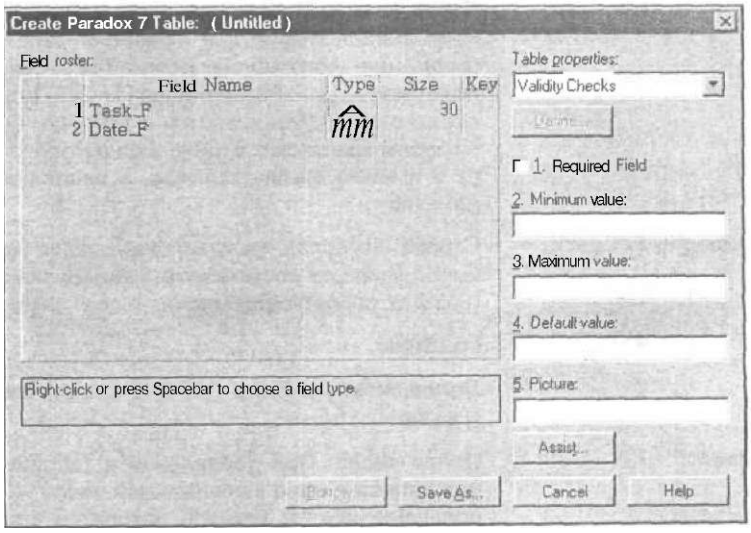

Рисунок 3.4 — В диалоговом окне *Create Table* надо задать структуру таблицы создаваемой базы данных

Записи базы данных "Ежедневник" состоят из двух полей: *Task_F* и *Date_F*. Поле *Task_F* (символьного типа) содержит название задачи (мероприятия), поле *Date_F* (типа *Date*) — дату, не позднее которой задача должна быть выполнена (дату проведения мероприятия).

Имена полей вводят в столбец *Field Name*, тип — в столбец *Type*. При записи имени поля можно использовать латинские буквы и цифры. При этом следует учитывать, что имя поля не должно совпадать ни с одним из ключевых слов языка *SQL* (таких, например, как *WHEN* или *SELECT*). Тип поля определяет тип данных, которые могут быть помещены в поле. Задается тип поля при помощи одной из приведенных в табл. 3.1 констант. Константа определяющая тип поля, может быть введена с клавиатуры или выбором в списке, который появляется в результате нажатия клавиши "пробел" или щелчка правой кнопкой мыши.

Таблица 3.1 — Тип поля определяет тип информации, которая может в нем находиться

Тип поля	Константа	Содержимое поля
Alpha	*A*	Строка символов. Максимальная длина строки определяется характеристикой ***Size*,** значения которой находятся.в диапазоне 1—255
Number	*N*	Число из диапазона 10*e*–307…10*e*308 *с* пятнадцатью значащими цифрами
Money	$	Число в денежном формате. Цифры числа делятся на группы при помощи разделителя групп разрядов. Так же выводится знак денежной единицы
Short	*S*	Целое число из диапазона от -32767 до 32767

Long Integer	*I*	Целое число из диапазона от –2 147 483 648 до 2 147483 647
Date	*D*	Дата
Time	*T*	Время, отсчитываемое от полуночи, выраженное в миллисекундах
Timestamp	*@*	Время и дата
Memo	*M*	Строка символов произвольной длины. Поле типа *Memo* используется для хранения текстовой информации, которая не может быть сохранена в поле типа *Alpha*. Размер поля 1…2401 определяет, сколько символов хранится в таблице. Остальные символы хранятся в файле, имя которого совпадает с именем файла таблицы, а расширение файла — *mb*
Formated Memo	*F*	Строка символов произвольной длины (как у типа *Memo*). Имеется возможность указать тип и размер шрифта, способ оформления и цвет символов
Graphic	*G*	Графика
Logical	*L*	Логическое значение "истина" (*true*) или "ложь" (*false*)
Autoincrement	+	Целое число. При добавлении в таблицу очередной записи в поле записывается число на единицу большее, чем то, которое находится в соответствующем поле последней добавленной записи
Bytes	*Y*	Двоичные данные. Поле этого типа используется для хранения данных, которые не могут быть интерпретированы *Database Desktop*
Binary	B	Двоичные данные. Поле этого типа используется для хранения данных, которые не могут быть интерпретированы *Database Desktop*. Как и данные типа *Memo*, эти данные не находятся в файле таблицы. Поля типа *Binary*, как правило, содержат аудиоданные

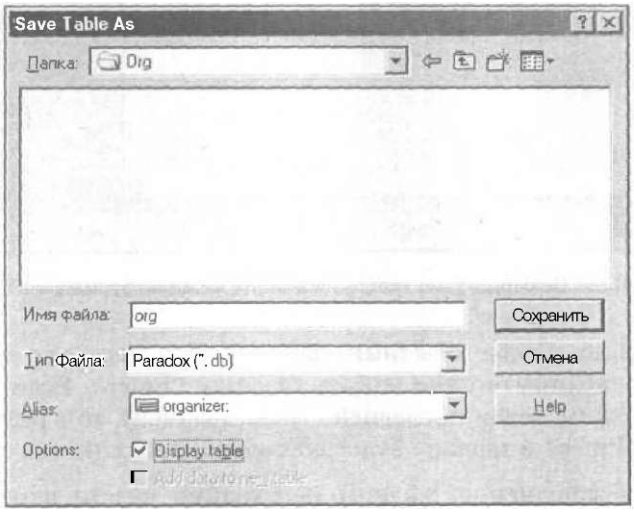

Рисунок 3.5 — Сохранение таблицы базы данных

После того как будут определены все поля, надо щелкнуть на кнопке *Save As*. На экране появится диалоговое окно *Save Table As* (рис. 3.5). В нем нужно выбрать (в списке *Alias*) псевдоним базы данных, элементом которой является сохраняемая таблица, в поле Имя файла ввести имя файла

таблицы, установить переключатель *Display table* и щелкнуть на кнопке Сохранить. В результате в указанном каталоге (псевдоним связан с конкретным каталогом локального или сетевого диска) будет создан файл таблицы и на экране появится диалоговое окно *Table* (рис. 3.6), в котором можно ввести данные в только что созданную таблицу (базу данных). Следует обратить внимание, что по умолчанию *Database Desktop* открывает таблицы в режиме просмотра, и для того чтобы внести изменения в таблицу (добавить, удалить или изменить запись), необходимо, выбрав в меню *Table* команду *Edit* (или нажав клавишу <*F*8>), активизировать режим редактирования таблицы.

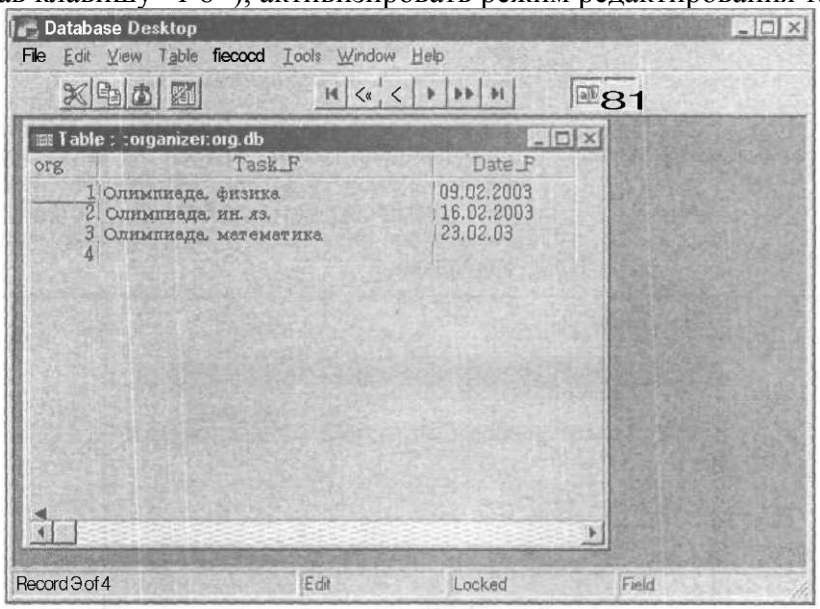

Рисунок 3.6 — *Database Desktop* можно использовать для ввода информации в базу данных

Данные в таблицу вводят обычным образом. Для перехода к следующему полю (столбцу таблицы) нужно нажать клавишу <*Enter*>. Если текущее поле является последним полем последней строки (записи), то в результате нажатия клавиши <*Enter*> в таблицу будет добавлена строка (новая запись).

Если во время заполнения таблицы необходимо внести изменения в уже заполненное поле, то надо, используя клавиши перемещения курсора, выбрать это поле и нажать клавишу <*F*2>.

изменить запись), необходимо, выбрав в меню *Table* команду *Edit* (или нажав клавишу <*F*8>), активизировать режим редактирования таблицы.

Если при вводе данных в таблицу буквы русского алфавита отображаются неверно, то надо изменить шрифт, который используется для отображения данных. Для этого нужно в меню *Edit* выбрать команду *Preferences*, затем, в появившемся диалоговом окне во вкладке *General* щелкнуть на кнопке *Change*. В результате этих действий откроется диалоговое окно *Change Font* (рис. 3.7), в котором надо выбрать русифицированный шрифт *TrueType*. Следует обратить внимание, что в *Microsoft Windows* 2000 (*Microsoft Windows XP*) используются шрифты типа *Open Type*, в то время как программа *Database Desktop* ориентирована на работу со шрифтами *TrueType*. Поэтому в списке шрифтов нужно выбрать русифицированный шрифт именно *TrueType*. После выбора шрифта необходимо завершить работу с *Database*

Desktop, т. к. внесенные в конфигурацию изменения будут действительны только после перезапуска утилиты.

Рисунок 3.7 — Для правильного отображения данных в *Database Desktop* нужно выбрать русифицированный шрифт *TrueType*

3.3.2. Создание программы

Доступ к базе данных обеспечивают компоненты *Database*, *Table*, *Query* и *DataSource*. Значки этих компонентов находятся на вкладках *Data Access* и *BDE* (рис. 3.8).

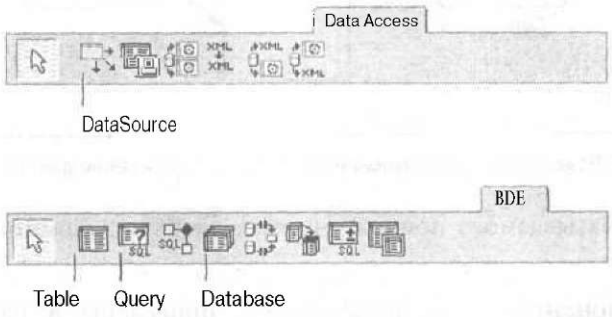

Рисунок 3.8 — Компоненты вкладок *Data Access* и *BDE* обеспечивают доступ к данным

Компонент *Database* представляет базу данных как единое целое, т. е. как совокупность таблиц, а компонент *Table* — как одну из таблиц базы данных. Компонент *DataSource* (источник данных) обеспечивает связь между компонентом отображения-редактирования данных (например, компонент *DBGrid*) и источником данных, в качестве которого может выступать таблица (компонент *Table*) или результат выполнения *SQL*-запроса к таблице (компонент *Query*). Компонент *DataSource* позволяет оперативно выбирать источники данных, использовать один и тот же компонент (например, *DBGrid*) для отображения всей таблицы (базы данных) или только результата выполнения *SQL*-запроса к этой таблице. Компоненты доступа к данным обращаются к базе данных не напрямую, а через процессор баз данных - *Borland Database Engine* (*BDE*).

Ядро *BDE* образуют динамические библиотеки, реализующие механизмы обмена данными и управления запросами. В состав *BDE* включены драйверы, обеспечивающие работу с файлами данных форматов *Paradox*, *dBase*, *FoxPro*. Имеется также механизм подключения драйверов *ODBC*. Доступ к данным *SQL* серверов обеспечивает отдельная система драйверов — *SQL Links*. С их помощью можно получить доступ к базам данных *Oracle*, *Sysbase* и *Interbase*.

Механизм взаимодействия компонента отображения-редактирования данных (*DBGrid*) С Данными (*Table* ИЛИ *Query*) Через компонент *DataSource* показан на рис. 3.9.

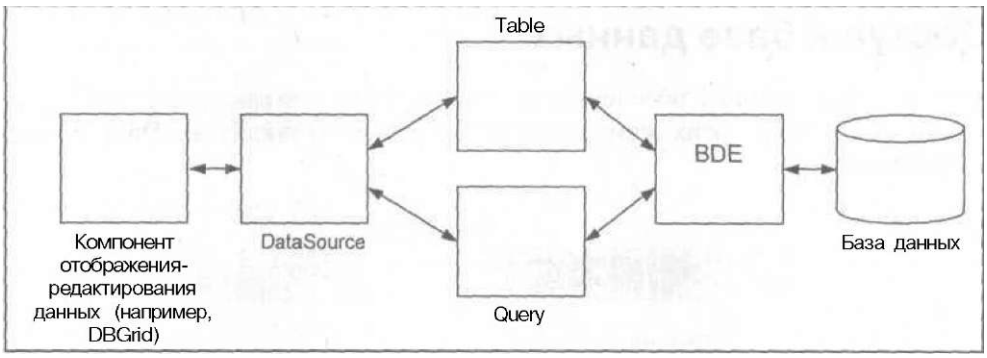

Рисунок 3.9 — Взаимодействие компонентов доступа-отображения данных и *BDE*

В форму разрабатываемого приложения надо добавить компоненты *Table* и *DataSource*.

Свойства компонентов *Table* и *DataSource* приведены в табл. 3.2 и 3.3. Свойства перечислены в том порядке, в котором рекомендуется устанавливать их значения.

Значения свойств *OatabaseName* и *TableName* задаются путем выбора из списков. В списке *DatabaseName* перечислены все зарегистрированные на данном компьютере псевдонимы, а в списке *TabieName* — имена файлов таблиц, которые находятся в соответствующем псевдониму каталоге.

Таблица 3.2 — Свойства компонента *Table*

Свойство	Определяет
DatabaseName	Имя базы данных, частью которой является таблица (файл данных), для доступа к которой используется компонент. В качествезначения свойства следует использовать псевдоним базы данньх
TableName	Имя файла данных (таблицы данных), для доступа к которому используется компонент
TableType	тип таблицы. Таблица может быть набором данньх в формате *Paradox* (*ttParadox*), *dBase* (*ttDBase*), *FoxPro* (*ttFoxPro*) или другого типа. По умолчанию значение свойства равно *ttDefault* — это означает, что тип таблицы будет определен на основе информации, которая находится в файле таблицы
Active	Признак активизации файла данных (таблицы). В результате присваивания свойству значения *true* файл таблицы будет открыт

Таблица 3.3 — Свойства компонента *DataSource*

Свойство	Определяет
Name	Имя компонента. Используется для доступа к свойст вам компонента
DataSet	Компонент, представляющий входные данные (таблица или запрос)

Свойство *DataSet* компонента *DataSource* обеспечивает возможность выбора источника данных, а также связь между компонентом, представляющим данные (таблица или запрос), и компонентом отображения данных. Например, большая база данных может быть организована как набор таблиц одинаковой структуры. В этом случае в приложении работы с базой данных каждой

таблице будет соответствовать свой компонент *Table*, а выбор конкретной таблицы можно осуществить установкой значения свойства *DataSet*.

Компоненты доступа к базе данных являются невизуальными и во время работы программы на форме не видны. Поэтому их можно поместить в любую точку формы (рис. 3.10).

Значения свойств компонентов *Table* 1 и *DataSourcel* приложения "Ежедневник" приведены в табл. 3.4 и 3.5.

Таблица 3.4 — Значения свойств компонента *Tdblel*

Свойство	Значение
Name	*Tablel*
DatabaseName	*organizer*
TableName	*org.db*
Active	*false*

Таблица 3.5 — Значения свойств компонента *DataSource*1

Свойство	Значение
Name	*DataSourcel*
DataSet	*Tablel*

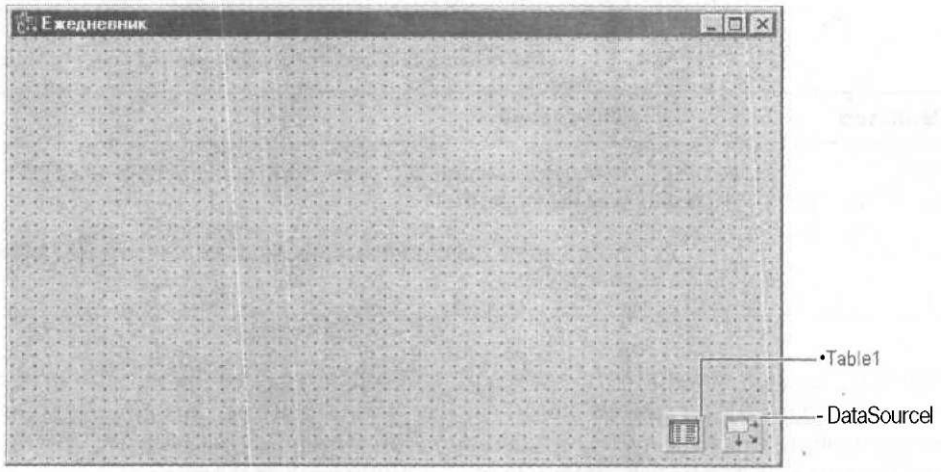

Рисунок 3.10 — Форма после добавления компонентов *Table* и *DataSource*

Отображение данных

Пользователь может просматривать базу данных в режиме формы или в режиме таблицы. В режиме формы можно видеть только одну запись, а в режиме таблицы — несколько записей одновременно. Часто эти два режима комбинируют. Краткая информация (содержимое некоторых ключевых полей) выводится в табличной форме, а при необходимости увидеть содержимое всех полей выполняется переключение в режим формы.

Компоненты, обеспечивающие отображение и редактирование полей записей базы данных, находятся на вкладке *Data Controls* (рис. 3.11).

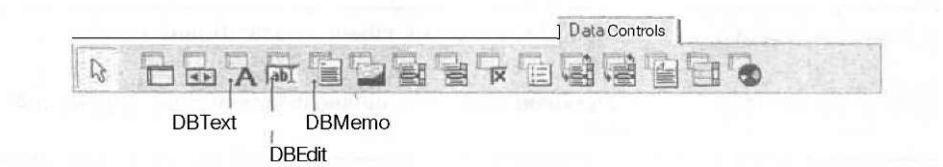

Рисунок 3.11 — Компоненты отображения и редактирования полей

Компонент *DBText* обеспечивает отображение содержимого отдельного поля, а компоненты *DBEdit* и *DBMsmo* — отображение и редактирование. В табл. 3.6 перечислены некоторые свойства этих компонентов. Свойства перечислены в том порядке, в котором следует устанавливать их значения.

Таблица 3.6 — Свойства компонентов *DBText*, *DBEdit* и овмето

Свойство	Определяет
DataSource	Источник данных (компонент *Table* или *Query*)
DataField	Поле записи, для отображения или редактирования которого исполь зуется компонент

Для обеспечения просмотра базы данных в режиме таблицы используется компонент *DBGrid*. Свойства компонента *DBGridl* определяют вид таблицы и действия, которые могут быть выполнены над данными во время работы программы. В табл. 3.7 перечислены некоторые свойства компонента *DBGrid*.

Таблица 3.7 — Свойства компонента *DBGrid*

Свойство	Определяет
DataSource	Источник данных (компонент *Table* или *Query*)
Columns	Отображаемая информация (поля записей)
Options. *dgTitles*	Разрешает вывод строки заголовка столбцов
Options.*dglndicator*	Разрешает вывод колонки индикатора. Во время рабо ты с базой данных текущая запись помечается в колон ке индикатора треугольником, новая запись — звездоч кой, редактируемая — специальным значком
Options.*dgColumnResize*	Разрешает менять во время работы программы ширину колонок таблицы
Options, *.dgColLines*	Разрешает выводить линии, разделяющие колонки таблицы
Options, *.dgRowLines*	Разрешает выводить линии, разделяющие строки таблицы

В диалоговом окне программы "Ежедневник" данные отображаются в режиме таблицы. Поэтому в форму надо добавить компонент *DBGrid*1 и установить значения его свойств в соответствии с табл. 3.8.

Таблица 3.8 — Значения свойств компонента *DBGridi*

Свойство	Значение
DataSource	*DataSourcel*

Как было сказано ранее, свойство *Columns* компонента *DBGrid* определяет поля, содержимое которых будет отображено в таблице *DBGrid*. Свойство *columns* является сложным свойством и представляет собой массив элементов типа *TColumn*. Свойства элементов массива определяют поля, содержимое которых будет в таблице, а так же вид колонок (табл. 3.9).

Таблица 3.9 — Свойства объекта *TColumn*

Свойство	Определяет
FieldName	Поле, содержимое которого отображается в колонке
width	Ширину колонки в пикселах
Font	Шрифт, используемый для вывода текста в ячейках колонки
Color	Цвет фона колонки
Alignment	Способ выравнивания текста в ячейках колонки. Текст может быть выровнен по левому краю (*taLeftJustify*), по центру (*taCenter*) или по правому краю (*-Bight*

	Justify)
Title.Caption	Заголовок колонки. Значением по умолчанию является имя поля записи
Title.Alignment	Способ выравнивания заголовка колонки. Заголовок может быть выровнен по левому краю (*taLeftJustify*), по центру (*taCenter*) или по правому краю (*t aRight Justify*)
Title. Color	Цвет фона заголовка колонки
Title, *Font*	Шрифт заголовка колонки

По умолчанию компонент *DBGrid* содержит одну колонку. Чтобы добавить в компонент *DBGrid* еще одну колонку, надо в окне *Object Inspector* выбрать свойство компонента *DBGrid*, щелкнуть на кнопке с тремя точками, а

затем в появившемся окне — на кнопке *Add New* (рис. 3.12). После

этого, используя *Object Inspector*, надо установить значения свойств элементов массива *columns*.

Выбрать настраиваемую колонку (ее свойства отражаются в окне *Object Inspector*) можно в окне *Editing* или в окне *Object TreeView*.

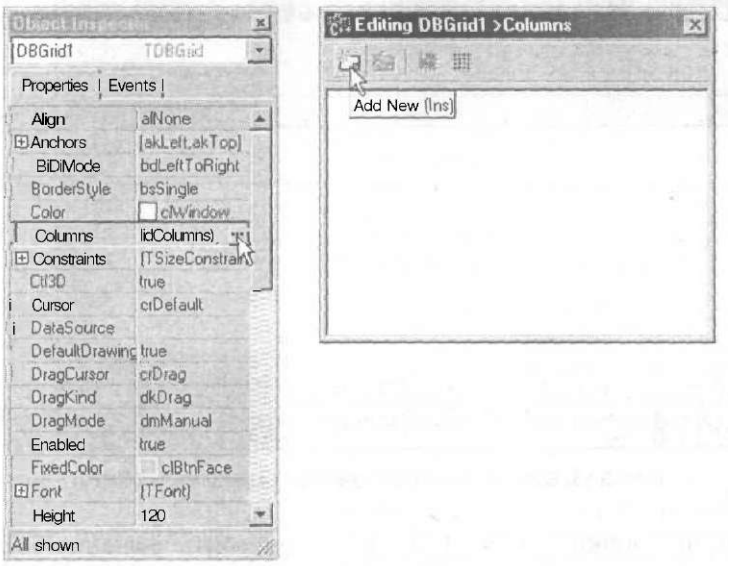

Рисунок 3.12 — Чтобы добавить колонку в компонент *DBGrid*, щелкните в строке *Coiums* на кнопке с тремя точками, затем — на кнопке *Add New*

В простейшем случае для каждой колонки достаточно установить значение свойства *FieldName*, которое определяет поле, содержимое которого отображается в колонке, а также значение свойства *Title.caption*, определяющее заголовок колонки. В табл. 3.10 приведены значения свойств компонента *DBGridi*, а на рис. — вид формы после настройки компонента.

Таблица 3.10 — Значения свойств компонента *DBGrid*1

Свойство	**Значение**
Columns[0].*FieldName*	*Date_F*
Columns[0].*TitleCaption*	Когда
Columns[0].*Title.Font.Style.Italic*	*true*
Свойство	**Значение**
Columns.[1].*FieldName*	*Task_F*
Columns.[1].*TitleCaption*	Что

Columns.[1]..*Title.Font.Style.Italic*	*true*

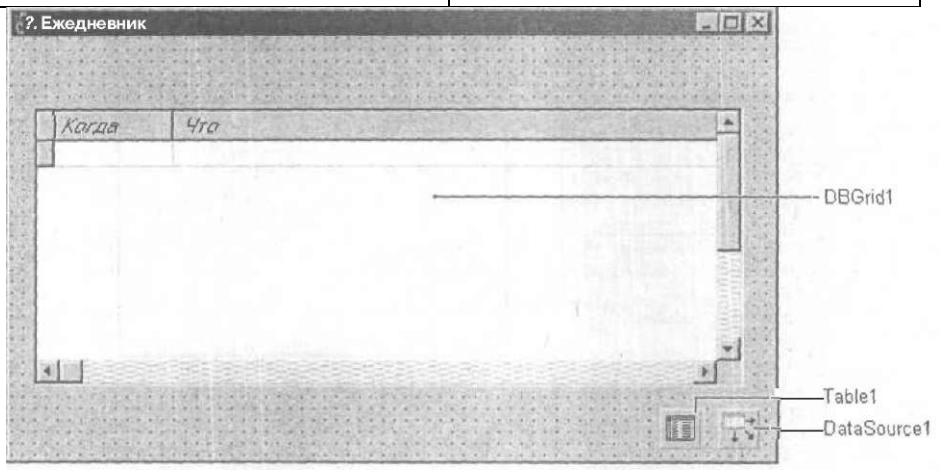

Рисунок 3.13 — Вид формы после настройки компонента *DBGrid*

Если после настройки компонента *DBGrid* присвоить значение *true* свойству *Active* компонента *Tablel*, то в поле компонента *DBGrid* будет выведено содержимое базы данных.

Манипулирование данными

Для того чтобы пользователь мог не только просматривать базу данных (решение этой задачи в рассматриваемой программе обеспечивает компонент *DBGrid*), но и редактировать ее, в форму приложения надо добавить компонент *DBNavigator*, значок которого находится на вкладке *Data Controls* (рис. 3.14). Компонент *DBNavigator* (рис. 315) представляет собой набор командных кнопок, обеспечивающих перемещение указателя текущей записи к следующей, предыдущей, первой или последней записи базы данных, а также добавление в базу данных новой записи и удаление текущей записи.

Табл. 3.11 содержит описания действий, которые выполняются в результате щелчка на соответствующей кнопке компонента *DBNavigator*. Свойства компонента *DBNavigator* перечислены в табл. 3.12.

Рис. 3.14 — Значок компонента *DBNavigator* находится на вкладке *Data Controls*

Рисунок 3.15 — Компонент *DBNavigator*

Таблица 3.11 — Кнопки компонента *DBNavigator*

Кнопка		Обозначение	Действие
	К первой	*nbFirst*	Указатель текущей записи перемещается к первой записи файла данных
	К предыдущей	*nbPrior*	Указатель текущей записи перемещается к предыдущей записи файла данных
	К следующей	*nbNext*	Указатель текущей записи перемещается к следующей записи файла данных
	К последней	*nbLast*	Указатель текущей записи перемещается к последней записи

			файла данных
	Добавить	*nblnsert*	В файл данных добавляется новая запись
	Удалить	*nbDelete*	Удаляется текущая запись файла данных
	Редактировани е	*nbEdit*	Активизирует режим редактирования текущей записи
	Сохранить	*nbPost*	Изменения, внесенные в текущую запись, записываются в файл данных
	Отменить	*Cancel*	Отменяет внесенные в текущую запись изменения
	Обновить	*nbRefresh*	Записывает внесенные изменения в файл

Таблица 3.12 — Свойства компонента *DSNavigator*

Свойство	Определяет
DataSource	Компонент, являющийся источником данных. В качестве источника данных может выступать база данных (компонент *Database*), таблица (компонент *Table*) или результат выполнения запроса (компонент *Query*)
VisibleButtons	Видимые командные кнопки

Следует обратить внимание на свойство *VisibleButtons*. Оно позволяет скрыть некоторые кнопки компонента *DBNavigator* и тем самым запретить выполнение соответствующих операций над файлом данных. Например, присвоив значение *false* свойству *VisibleButtoris.nbDelete*, можно скрыть кнопку *nbDeiete* и тем самым запретить удаление записей.

На рис. 3.16 приведен вид формы приложения "Ежедневник" после добавления Компонента *DBNavigator*. свойству *DataSource* Компонента *DBNavigatorl* следует присвоить значение *Table*1.

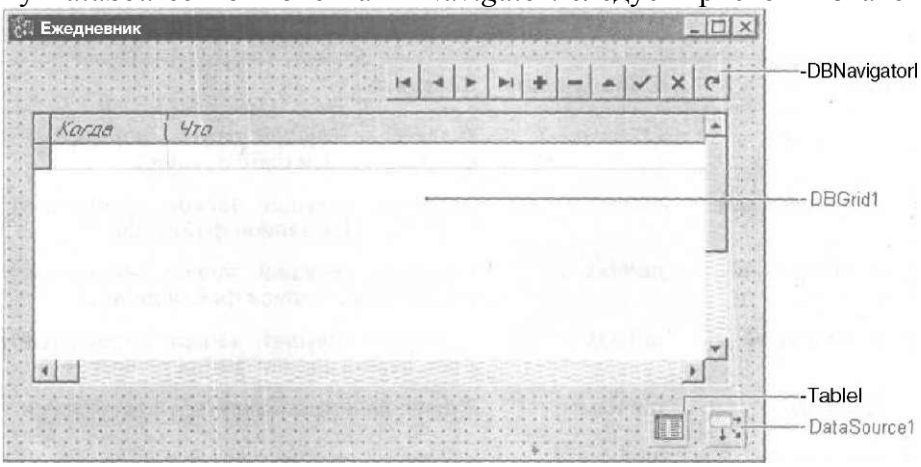

Рисунок 3.16 — Форма приложения после добавления компонента *DBNavigator*

После этого программу можно откомпилировать и запустить. Следует обратить внимание, что для того чтобы после запуска программы в окне появилась информация или, если база данных пустая, можно было вводить новую информацию, свойство *Active* таблицы-источника данных должно иметь значение *true*.

Работа с базой данных, представленной в виде таблицы, во многом похожа на работу с электронной таблицей *Microsoft Excel*. Используя клавиши перемещения курсора вверх и вниз, а также клавиши листания текста страницами (<*Page Up*> и <*Page Down*>), можно, перемещаясь от строки к строке, просматривать записи базы данных. Нажав клавишу <*Ins*>, можно добавить запись, а нажав клавишу <*Del*> — удалить. Для того чтобы внести изменения в поле записи, нужно, используя клавиши перемещения курсора влево и вправо, выбрать необходимое поле и нажать клавишу <*F*2>.

Выбор информации из базы данных

При работе с базой данных пользователя, как правило, интересует не все ее содержимое, а некоторая конкретная информация. Найти нужные сведения можно последовательным просмотром записей. Однако такой способ поиска неудобен и малоэффективен.

Большинство систем управления базами данных позволяют выполнять выборку нужной информации путем выполнения запросов. Пользователь формулирует запрос, указывая критерий, которому должна удовлетворять интересующая его информация, а система выводит записи, удовлетворяющие запросу.

Для выборки из базы данных записей, удовлетворяющих некоторому критерию, предназначен компонент *Query* (рис. 3.17).

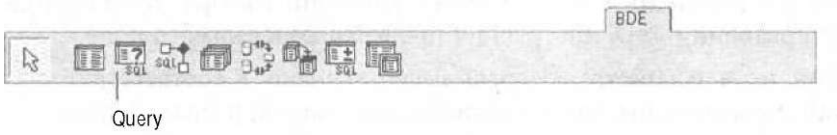

Рисунок 3.17 — Компонент *Query*

Компонент *Query*, как и компонент *Table*, представляет собой записи базы данных, но в отличие от последнего он представляет не всю базу данных (все записи), а только ее часть — записи, удовлетворяющие критерию запроса.

В табл. 3.13 перечислены некоторые свойства компонента *Query*.

Таблица 3.13 — Свойства компонента *Query*

Свойство	Определяет
Name	Имя компонента. Используется компонентом *DataSource* для связи результата выполнения запроса (набора записей) с компонентом, обеспечивающим просмотр записей, например *DBGrid*
SQL	Записанный на языке *SQL* запрос к базе данных (к таблице)
Active	При присвоении свойству значения *true* активизируется процесс выполнения запроса
RecordCount	Количество записей, удовлетворяющих критерию запроса

Для того чтобы во время разработки программы задать, какая информация должна быть выделена из базы данных, в свойство *SQL* надо записать запрос — команду на языке *SQL* (*Structured Query Language*, язык структурированных запросов).

В общем виде *SQL*-запрос на выборку данных из базы данных (таблицы) выглядит так:

SELECT СписокПслей *FROM* Таблица *WHERE* (Критерий)

ORDER BY СписокПолей где:

SELECT — команда "выбрать из таблицы записи и вывести содержимое полей, имена которых указаны в списке";

FROM — параметр команды, который определяет имя таблицы, из которой нужно сделать выборку;

WHERE — параметр, который задает критерий выбора. В простейшем случае критерий — это инструкция проверки содержимого поля;

ORDER BY — параметр, который задает условие, в соответствии с которым будут упорядочены записи, удовлетворяющие критерию запроса.

Например,запрос

SELECT Date_F, *Task_F FROM* ':*organizer*:*org.db*' *WHERE* (*Date_F* = '09.02.2003') *ORDER BY Date_F*

обеспечивает выборку записей из базы данных *organizer* (из таблицы у которых в поле *Date_F* находится текст 09.02.2003, т. е. формирует список мероприятий, назначенных на 9 февраля 2003 года.

Другой пример. Запрос

SELECT Date_F, *Task_F FROM* = ', *organ*! *zer*; *org*. *db* ! *WHERE*

(*Date_F* >='10.02.2003') *AND* (*Date_F* <= '16.02.2003 ') *ORDER BY Date_F*

формирует список дел, назначенных на неделю (с 10 по 16 февраля 2003 года).

Запрос может быть сформирован и записан в свойство *SQL* компонента *Query* во время разработки формы или во время работы программы.

Для записи запроса в свойство *SQL* во время разработки формы используется редактор списка строк (рис. 3.18) окно которого открывается в результате щелчка на кнопке с тремя точками в строке свойства *SQL* (в окне *Object Inspector*).

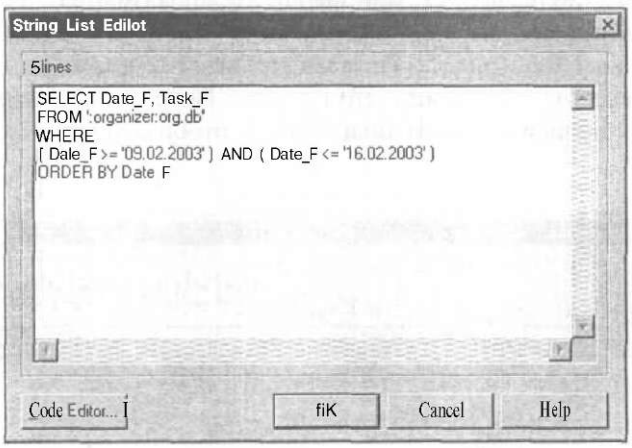

Рисунок 3.18 — Ввод *SQL*-запроса во время разработки формы приложения

Сформировать запрос во время работы профаммы можно при помощи метода *Add*, применив его к свойству *SQL* компонента *Query*.

Ниже приведен фрагмент кода, который формирует запрос (т. е. записывает текст запроса в свойство *SQL* компонента *Query*) на выбор информации из таблицы *org* базы данных *organizer*. Предполагается, что строковая переменная *today* (ТИП *Ansistring*) содержит дату В формате

Forml->Queryl->SQL->Add("*SELECT Date_F*, *Task_F*");

Forml->Queryl->SQL->Add("*FROM* ' :*organizer*:*org.db*'");

Forml->Queryl->SQL->Add("*WHERE* (*Date_F* = + *today* + '")");

Forml->Queryi->SQL->Add("*ORDER BY Date_F*");

Если запрос записан в свойство *SQL* компонента *Query* во время разработки формы приложения, то во время работы профаммы критерий запроса можно изменить простой заменой соответствующей строки текста запроса.

Например, для запроса:

SELECT Date_F, *Task_F FROM organizer*:*org.db*' *WHERE* (*Date_F*= '09.02.2003') *ORDER BY Date_F*

инструкция замены критерия выглядит так:

щелчка на кнопке с тремя точками в строке свойства *SQL* (в окне *Object Inspector*).

Queryl->SQL->Strings [3] = " (*Date_F* = '" + *tomorrow* + "')";

Следует обратить внимание на то, что свойство *SQL* является структурой типа *TStrings*, в которой строки нумеруются с нуля.

Для того чтобы пользователь мог выбирать информацию из базы данных, в форму разрабатываемого приложения надо добавить кнопки Сегодня, Завтра, Эта неделя и Все (рис. 3.19). Назначение этих кнопок очевидно. Также в форму добавлены два компонента *Label*. Поле *Label* 1 используется для отображения текущей даты. В поле *Label*2 отображается режим просмотра базы данных.

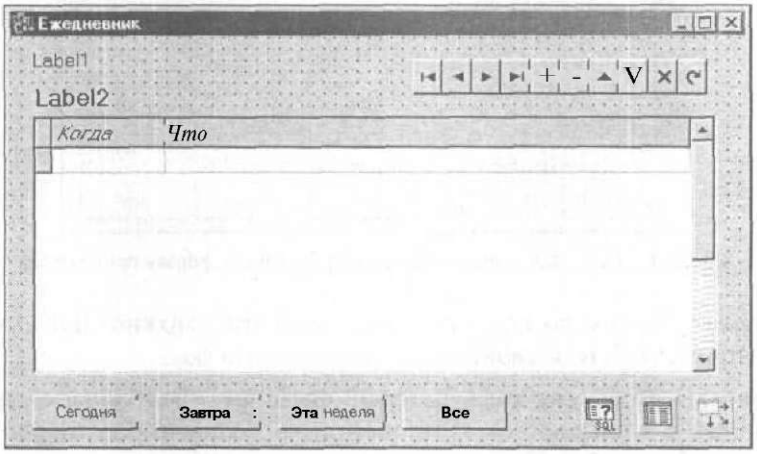

Рисунок 3.19 — Окончательный вид формы

Функции обработки события *click* на кнопках Сегодня, Завтра и Эта неделя приведены в листинге 3.1. Каждая из этих функций изменяет соответствующим образом сформированный во время разработки формы *SQL*-запрос. Для получения текущей даты функции обращаются к стандартной функции *NOW*, которая возвращает текущую дату и время. Преобразование даты в строку СИМВОЛОВ выполняет стандартная ФУНКЦИЯ *FormatDateTime*.

Листинг 3.1. Обработка события *Click* на кнопках Сегодня, Завтра и Эта неделя

```
// Щелчок на кнопке Сегодня
void __fastcall TForm1::ButtonlCiick (TObject *Sender)
{
AnsiString today = FormatDateTime("dd/mm/yyyy",Now());
Forml->Label2->Caption = "Сегодня";
//изменить критерий запроса
Queryl->SQL->Strings [3] = " (Date_F = '" + today + "')";
// выполнить запрос Forml->Queryl->Open ( ) ;
Forml->DataSourcel->DataSet = Forml->Queryl;
if ( ! Forml->Query1->RecordCount) {
ShowMessage («На сегодня никаких дел не запланировано!»);
}
}
// щелчок на юношей Завтра
void __fastcall TFoml::Button2Click (TObject *Sender)
{
AnsiString tomorrow = FormatDateTime ("dd/mm/yyyy", Now() +1);
Forml->Label2->Caption = "Завтра";
// изменить критерий запроса
Queryl->SQL->Strings[3] = " (Date_F = '" + tomorrow + "')";
// выполнить запрос Forml->Queryl->Open () ;
Forml->DataSourcel->DataSet = Forml->Queryl;
if ( ! Forml->Queryl->RecordCount) {
```

```
ShowMessage ("На завтра никаких дел не запланировано!");
}
)
// щелчок на кнопке Эта неделя
void __fastcall TForml::Button3Click (TObject *Sender)
{
// от текущего дня до конца недели (до воскресенья) TDateTime Present, EndOfWeek;
Label2->Caption = "На этой неделе";
Present= Now (); // Wow — возвращает текущую дату
// для доступа к StartOfWeek, EndOfWeekr YearOf и WeekOf
// надо подключить DateUtils.hpp (см. директивы #include)
EndOfWeek = StartOfAWeek(YearOf(Present),WeekOf(Present)+1);
Queryl->SQL->Strings[3] =
"(Date F >= ' " + FormatDateTime ("dd/mm/vyyy",  Present) +" ') AND "+
"(Date F < ' " + FormatDateTime ( "dd/mm/уууу".. EndOfWeek) +" ' ) " ;
Queryl->Open() ;
if ( Queryl->RecordCount) {
DataSourcel->DataSet = Fonru->Queryl;
}
else
ShowMessage {"На эту неделю никаких дел не запланировано.");
)
```

В результате щелчка на кнопке Все в диалоговом окне программы должно быть выведено все содержимое базы данных. Базу данных представляет компонент *Table* 1. Поэтому функция обработки события *click* на кнопке Все просто "переключает" источник данных на таблицу (листинг 3.2).

Листинг 3.2. Обработка события на кнопке Все

```
// Щелчок на кнопке Все
void __fastcall TForml ::Button4Click(TObject *Sender) {
// установить: источник данных — таблица
// таким образом, отобажается вся БД
Forml->DataSourcel->DataSet = Forml->Tablel;
Label2->Caption = "Все, что намечено сделать";
}
```

Программа "Ежедневник" спроектирована таким образом, что при каждом ее запуске в диалоговом окне выводится текущая дата и список дел, запланированных на этот и ближайшие дни. Вывод даты и названия дня недели в поле *Label* выполняет функция обработки события *onActivate* (ее текст приведен в листинге 3.3). Эта же функция формирует критерий запроса к базе данных, обеспечивающий вывод списка задач, решение которых запланировано на сегодня (в день запуска программы) и на завтра. Если программа запускается в пятницу, субботу или воскресенье, то завтрашним днем считается понедельник. Такой подход позволяет сделать упреждающее напоминание, ведь, возможно, что пользователь не включит компьютер в выходные дни.

Листинг 3.3. Функция обработки события *OnActivate*

```
AnsiString stDay[7] = ("воскресенье","понедельник","вторник", "среда",
"четверг","пятница","суббота"};
AnsiString stMonth[12] = {"января","февраля","марта", "апреля", "мая", "июня"', "июля",
"августа","сентября","октября", "ноября","декабря"};
// активизация формы
void __fastcall TForml::FormActivate(TObject *Sender)
{
TDateTime Today, // сегодня
```

```
NextDay; // следующий день (не обязательно завтра)
Word Year, Month, Day; // год, месяц, день
Today = Now ();
DecodeDate(Today, Year, Month, Day);
Labell->Caption = "Сегодня " + IntToStr(Day) + " " +
stMonth[Month-l] + " " +
IntToStr(Year) + " года, " +
stDay[DayOfWeek(Today) -1] ;
Label2->Caption = "Сегодня и ближайшие дни";

// вычислим следующий день
// если сегодня пятница, то, чтобы не забыть,
// что запланировано на понедельник, считаем, что следующий
// день - понедельник
switch ( DayOfWeek(Today)) {
case 6 : NextDay = Today + 3; break; // сегодня пятница
case 7 : NextDay = Today + 2; break; // сегодня суббота
default : NextDay = Today + 1; break;
}
// запрос к базе данных; есть ли дела, запланированные
// на сегодня и на следующий день
Queryl->SQL->Strings[3] =
"(Date_F >= '"+ FormatDateTime("dd/mm/yyyy",Today)+"') AND " +
"(Date_F <= '"+ FormatDateTime("dd/mm/yyyy",NextDay)+"')";
Queryl->0pen();
DataSourcel->DataSet = Forml->Queryl;
if ( ! Queryl->RecordCount)
(
ShowMessage("На сегодня и ближайшие дни никаких дел
не запланировано.");
}
}
```

Использование псевдонима для доступа к базе данных обеспечивает независимость программы от размещения данных в системе, позволяет размещать программу работы с данными и базу данных на разных дисках компьютера, в том числе и на сетевом. Вместе с тем для локальных баз данных типичным решением является размещение базы данных в отдельном подкаталоге того каталога, в котором находится программа работы с базой данных. Таким образом, программа работы с базой данных "знает", где находятся данные. При таком подходе можно отказаться от создания псевдонима при помощи *Database Desktop* и возложить задачу создания псевдонима на программу работы с базой данных. Очевидно, что такой подход облегчает администрирование базы данных.

В качестве иллюстрации сказанного в листинге 3.4 приведен вариант реализации функции *onActivate*, которая создает псевдоним для базы данных *organizer*. Предполагается, что база данных находится в подкаталоге *DATA* того каталога, в котором находится выполняемый файл программы. Непосредственное создание псевдонима выполняет функция которой в качестве параметра передается псевдоним и соответствующий ему каталог. Так как во время разработки программы нельзя знать, в каком каталоге будет размещена программа работы с базой данных и, следовательно, подкаталог базы данных, имя каталога определяется во время работы программы путем обращения к функциям и *ExtractFilePatch*. Значение первой — полное имя выполняемого файла программы, второй - путь к этому файлу. Таким образом, процедуре *AddStandardAlias* передается полное имя каталога базы данных.

Листинг 3.5 Создание псевдонима во время работы программы

```
void __fastcall TForml::ForraActivate (TObject *Serider) {
TDateTime Today, // сегодня
NextDay; // следующим день (не обязательно завтра)
Word Year, Month, Day; // год, месяц, день
Today = Now();
DecodeDate(Today, Year, Month, Day);
Labell->Caption = "Сегодня " + IntToStr (Day) + " " +
stMonth[Month-1] + “ ” +
IntToStr(Year) + “ года, ” +
stDay[DayOfWeek(Today) -1] ;
Label2->Caption = "Сегодня и ближайшие дни";
// вычислим следующий день
// если сегодня пятница, то, чтобы не забыть,
что запланировано на понедельник, считаем, что следующий
// день — понедельник
switch ( DayOfWeek(Today)) {
case 6 : NextDay = Today + 3; break; // сегодня пятница
case 7 : NextDay = Today +2; break; // сегодня суббота
default : NextDay = Today +1; break;
}
#define DIN_ALIAS // псевдоним доступа к БД создается динамически
// если псевдоним создан при помощи Database Desktop
// или BDE Administratorf директиву #define DIN_ALIAS надо удалить
#ifdef DIN_ALIAS // псевдоним создается динамически
// создадим псевдоним для доступа к БД
Session->ConfigMode = cmSession;
Session->AddStandardAlias("organizer",
ExtractFilePath(ParamStr(0))+"DATA\\",
"PARADOX"); // база данных "Ежедневник" - // в формате Paradox
#endif
Foml->Tablel->Active = true; // открыть таблицу
// запрос к базе данных: есть ли дела, запланированные
// на сегодня и завтра
Queryl->SQL->Strings[3] =
"(Date_F >=•'"+ FormatDateTime("dd/mm/yyyy",Today)+"') AND " +
"(Date_F <= '"+ FormatDateTime("dd/mm/yyyy",NextDay)+"')";
Queryl->0pen();
DataSourcel->DataSet = Forml->Queryl;
if ( ! Queryl->RecordCount)
{
ShowMessage("На сегодня и ближайшие дни никаких дел
не запланировано.");
}
}
```

3.4. Индивидуальные задания

Согласно последней цифре зачетной книжке необходимо написать программы «Телефонная книга» или «Ежедневник» табл. 3.14.

Таблица 3.14 — Индивидуальное задание

M	Программа

Четная	«Ежедневеик»
Нечетная	«Телефонная книга»

Общие требования к функционалу:

— Отображать содержимое БД в виде таблицы;

— Возможность редактирования, добавления, удаления записи БД;

— Все данные хранить в БД *Paradox* доступ к данным осуществлять с помощью технологии *BDE*.

— Возможность автоматического создания *Alias* для доступа к БД, если на целевой машине не создан *Alias*.

Программа «ежедневник» должна обеспечивать следующий функционал:

— Таблица должна содержать следующие поля: «№ задания»; «Дата задания»; «Содержимое задания»;

— Возможность фильтрации отображаемых заданий «Сегодня», «Завтра», «Эта неделя», «Все»;

— При каждом запуске программы в диалоговом окне должно выводиться текущая дата и список дел, запланированных на этот и ближайшие дни.

Программа «Телефонная книга» должна обеспечивать следующий функционал:

— Таблица должна содержать следующие поля: «№ записи»; «ФИО»; «дом. телефон»; «моб. телефон»; «*E-mail*».

— Возможность фильтрации отображаемых записей по полям: «ФИО»; «дом. телефон»; «моб. телефон»; «*E-mail*».

3.5. Содержание отчета

Отчет должен содержать:

1. Титульный лист;
2. Цель работы;
3. Описание и алгоритм работы программы;
4. Структуру БД;
5. Текст программы и результаты разработки программы;
6. Результаты работы программы;
7. Выводы.

Перед защитой отчета студент должен продемонстрировать работу программы. Минимальное количество записей — 30.

3.6. Контрольные вопросы

3.6.1. Что представляет собой псевдоним БД и как он создается?

3.6.2. Как создать таблицу с помощью программы *Database Desktop*? Какие другие средства для создания таблиц можно использовать?

3.6.3. Какие компоненты используются для связи таблиц БД с компонентами визуализации и управления данными *DBGrid*, *DBEdit*?

3.6.4. Как связать компоненты *Table* и *Query* с нужной таблицей БД?

3.6.5. Как связать компонент *DBNavigator* с нужной таблицей БД?

3.6.6. Что такое первичные и вторичные индексы для таблицы и как их создать?

3.6.7. Что такое *SQL* и из каких частей он состоит?

3.6.8. Что означают следующие *SQL*-запросы:

а) *SELECT name*, *projectname FROM emploee*, *project WHERE empno=team_leader*;

б) *SELECT name*, *salary FROM emploee*, *prohibit WHERE salary*>2900;

3.6.9. Как использовать *SQL*-запрос в *C++Builder* ?