17

Доступ к базам данных из VB-кода. Microsoft DAO

Поскольку Visual Basic не является языком баз данных[1], для доступа к данным следует использовать предназначенные для этого объекты (их свойства и методы). В этой главе рассматривается модель объектов доступа к данным: Data Access Objects (DAO).

Microsoft DAO позволяет работать с базами данных из VB- и приложений, поддерживающих Visual Basic for Applications. Объекты DAO подразделяются на те, которые отражают структуру базы данных, и на те, которые являются данными и позволяют взаимодействовать со многими типами баз данных из любого VB-приложения. При помощи объектов DAO можно:

- создавать базы данных, изменять их структуру;
- извлекать, добавлять, удалять и обновлять данные, хранимые в таблицах баз данных;
- подключаться к базам данных на удаленных серверах и разрабатывать клиент-серверные приложения.

DAO позволяет осуществлять доступ к базам данных различных форматов, которые можно разбить на три категории:

- Формат Microsoft Jet — все базы данных, построенные на основе ядра баз данных Microsoft Jet (включая созданные с использованием Microsoft Access, Microsoft Visual Basic, Microsoft Visual C++, Microsoft Excel и т.д.).
- Формат, поддерживаемый устанавливаемыми драйверами ISAM, которые обеспечивают доступ к внешним базам данных через DAO или Microsoft Jet. Драйверы ISAM имеются для таких форматов, как Microsoft FoxPro, dBASE, Microsoft Excel и других приложений Microsoft Office.
- Источники данных ODBC (например, Microsoft SQL Server). Для операций с источниками данных ODBC модель DAO можно использовать через Microsoft Jet или ODBCDirect.

Технологию DAO для операций с источниками данных ODBC можно использовать двумя способами: посредством Microsoft Jet [если необходимы уникальные возможности этого ядра, например, создание и модификация объектов, или объединение данных из баз различных форматов] или ODBCDirect [если нужно выполнять запросы или процедуры, хранимые на сетевом

[1] Языки баз данных, например FoxPro, содержат команды (и функции), которые позволяют непосредственно обрабатывать информацию, содержащуюся в таблицах баз данных. Например, в языке FoxPro команда APPEND добавляет новую запись в конец текущей базы данных (база данных состоит из единственной таблицы и сохраняется в одном файле), а команда DELETE помечает на удаление указанный диапазон записей. В языках баз данных имеются команды, позволяющие создавать таблицы, изменять их структуру, индексировать записи таблицы, а также устанавливать связи между таблицами.

В языках, не являющихся языками баз данных, подобные функции выполняют объекты, разрабатываемые для связи кода с информацией, хранимой в базе данных. Методы и свойства таких объектов очень похожи (наименованиями и назначением) на соответствующие команды в языках баз данных.

сервере, или если клиентскому приложению требуются специфические возможности ODBC, например, пакетное обновление или асинхронные запросы]. Поскольку же посредством ODBCDirect доступны не все возможности DAO, в Microsoft DAO предусмотрена поддержка ODBC ядром Microsoft Jet. Таким образом, для работы с источниками данных ODBC можно использовать и Microsoft Jet, и ODBCDirect или и то, и другое.

На вершине DAO-моделей находится единственный объект **DBEngine**. Объект **DBEngine** создавать не нужно. Даже если он не используется явно, программа все равно обращается ко всем другим объектам модели DAO посредством объекта **DBEngine**.

Коллекции объекта DBEngine

Объект **DBEngine** содержит две коллекции объектов (рис. 17.1): **Errors** — коллекция самых последних ошибок базы данных, возникших во время текущего сеанса; **Workspaces** — коллекция определенных рабочих пространств.

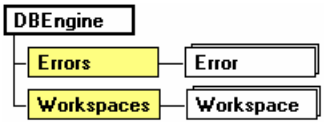

Рис. 17.1
Объектная модель DAO

Коллекция Errors

Коллекция **Errors** содержит все сохраненные **Error**-объекты, каждый из которых связан с единственной операцией, имеющей отношение к DAO.

Любая операция, включающая DAO-объекты, может генерировать одну или несколько ошибок. При возникновении ошибки один или несколько **Error**-объектов помещаются в **Errors**-коллекцию **DBEngine**-объекта. Если ошибку генерирует другая DAO-операция, **Errors**-коллекция освобождается и новый набор **Error**-объектов помещается в **Errors**-коллекцию. Объект с самым большим номером в **Errors**-коллекции (**DBEngine.Errors.Count - 1**) соответствует ошибке, о которой «сообщил» объект **Err** Microsoft Visual Basic (или VBA).

DAO-операции, которые не генерируют ошибку, не влияют на **Errors**-коллекцию. **Errors**-коллекция не поддерживает методы **Append** и **Delete**.

Набор объектов **Error** коллекции **Errors** описывает одну ошибку. Первый **Error**-объект — это ошибка самого низкого уровня, второй — следующего более высокого уровня и так далее. Например, если ODBC-ошибка возникает во время попытки открытия **Recordset**-объекта, первый error-объект содержит ODBC-ошибку самого низкого уровня; последующие error-объекты содержат ODBC-ошибки, возвращенные различными звеньями ODBC. Последний **Error**-объект содержит DAO-ошибку, указывающую, что объект не мог быть открыт.

Перечисление специфических ошибок в **Errors**-коллекции дает возможность пользовательским программам обработки ошибок более точно определить причину и источник происхождения ошибки и предпринять соответствующие меры для исправления.

Если вы используете ключевое слово **New** для создания объекта, который является причиной ошибки до или во время помещения в **Errors**-коллекцию, коллекция не содержит error-информации об этом объекте, потому что новый объект не связан с **DBEngine**-объектом. Однако error-информация будет помещена в VB-объект (и VBA-объект) **Err**.

Коллекция **Errors** имеет свойство **Count** (тип — **Integer**), возвращающее число объектов коллекции, и метод **Refresh** который, обновляет объекты в коллекции для отражения ее текущего состояния. Поскольку члены коллекции начинаются с 0, следует всегда кодировать циклы, начинающиеся с 0-го члена и заканчивающиеся значением свойства **Count** - 1. Если вы хотите выполнить цикл для членов коллекции без проверки свойства **Count**, следует использовать команду **For Each...Next**. Свойство **Count** никогда не устанавливается в значение **Null**. Если значением свойства **Count** является 0, в коллекции нет объектов.

Объект Error

Объект **Error** содержит информацию об ошибках доступа к данным, каждая из которых имеет отношение к единственной операции, включающей DAO.

В обоих рабочих пространствах — **Microsoft Jet** и **ODBCDirect** (которые рассматриваются далее в этой главе) — вы можете читать свойства объекта **Error** для получения специфической информации о каждой ошибке, включая:

- Свойство **Description**, содержащее текст сообщения об ошибке, которое отобразится на экране, если ошибка не будет перехвачена.

- Свойство **Number**, содержащее целочисленный код (**Long**) ошибки-константы (error constant).

- Свойство **Source**, которое идентифицирует объект, вызвавший ошибку (raised the error). Это особенно полезно, когда у вас имеются несколько объектов **Error** в **Errors**-коллекции, отслеживающих запрос к источнику данных **ODBC data source**.

- Свойства **HelpFile** и **HelpContext**, которые указывают соответствующий Microsoft Windows Help-файл и Help-раздел, соответственно, (если имеется) для данной ошибки.

Ваш VB-код обработки ошибок должен проверять **Errors**-коллекцию каждый раз, когда вы ожидаете ошибку доступа к данным. Если вы пишете централизованный обработчик ошибок, тестируйте VB-объект **Err** для определения того, является ли достоверной информация об ошибках в **Errors**-коллекции. Если свойство **Number** последнего элемента **Errors**-коллекции и код (value) объекта **Err** совпадают, вы можете использовать серию операторов **Select Case** для идентификации отдельной DAO-ошибки или ошибок.

Свойство Description объектов Error и Err

Свойство возвращает строку описания, связанную с ошибкой. Это свойство умолчания для объекта **Error**. Возвращаемое значение имеет тип **String**, который описывает ошибку. Свойство **Description** включает краткое описание ошибки. Используйте это свойство для предупреждения пользователя об ошибке, которую вы не можете или не хотите обрабатывать.

Свойства HelpContext, HelpFile объекта Err

Свойство **HelpContext** возвращает контекстный ID как переменную типа **Long** для раздела в Help-файле. Свойство **HelpFile** возвращает строку (**String**), которая представляет полный путь к Help-файлу.

Если вы определяете Help-файл в **HelpFile**, вы можете использовать свойство **HelpContext** для автоматического отображения Help-раздела, который оно определяет.

Обязательно следует писать процедуры в приложениях для обработки типичных ошибок. При программировании с использованием Err-объекта, вы можете использовать справочную информацию, предоставляемую Help-файлом этого объекта для повышения качества вашей обработки ошибок или для отображения значимого сообщения для вашего пользователя, если ошибку нельзя исправить.

Свойства объектов Error и Err

Свойство **Number** объектов **Error** и **Err** возвращает числовое значение, определяющее ошибку. Возвращаемое значение типа **Long** представляет номер ошибки. Вы можете использовать свойство **Number** для определения возникшей ошибки.

Свойство **Source** объектов **Error** и **Err** возвращает имя объекта или приложения, которое первоначально сгенерировало ошибку. Возвращаемое значение типа **String** представляет объект или приложение, которое сгенерировало ошибку. Значение свойства **Source** является обычно именем класса (class) объекта или программным ID. Вы можете использовать **Source** для предоставления вашим пользователям информации, когда ваш код не может обработать ошибку, сгенерированную в каком-либо объекте другого приложения.

Например, если вы обращаетесь к Microsoft Excel и приложение генерирует ошибку деления на нуль («Division by zero»), Microsoft Excel устанавливает **Error.Number** в код Microsoft Excel для этой ошибки и устанавливает свойство **Source** в **"Excel.Application"**. Заметим, что если ошибка генерируется в каком-либо другом объекте, вызываемом Microsoft Excel, Microsoft Excel перехватывает эту ошибку и устанавливает **Error.Number** в код Microsoft Excel. Однако свойства другого объекта **Error** (включая **Source**) сохранят значения такими, как они были установлены объектом, который сгенерировал ошибку. Свойство **Source** всегда содержит имя объекта, который первоначально сгенерировал ошибку.

На основе всей документации по ошибкам вы можете написать код, который будет обрабатывать ошибку соответствующим образом. Если ваш обработчик ошибок «потерпит неудачу», вы можете использовать информацию объекта **Error** для описания ошибки, используя свойство **Source** и другие **Error**-свойства, чтобы дать пользователю информацию о том, какой объект первоначально вызвал появление ошибки, описание ошибки и так далее.

Процедура, отображающая в окне **MsgBox** свойства элементов коллекции **Errors** и представленная в листинге 17.1, будет одинаково работать и в среде Visual Basic, и в VBA.

Листинг 17.1 Просмотр свойств элементов коллекции Errors

```
1:   Private Sub ErrorTest1()
2:
3:       On Error GoTo ErrorHandler
4:
5:       Set dbsTest = OpenDatabase("LostDatabase")
6:
7:       Exit Sub
8:
9:   ErrorHandler:
10:      For i = 0 To Errors.Count - 1
11:          MsgBox Errors(i).Number & vbCr & _
12:                 Errors(i).Description & vbCr & _
13:                 Errors(i).Source, , "Тестирование Error"
14:
15:      Next
16:
17:  End Sub
```

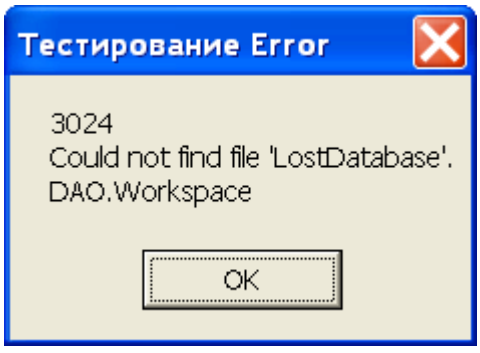

Рис. 17.2
Результат работы процедуры
ErrorTest1

Здесь намеренно выполняется попытка открыть несуществующую базу данных. Кроме отображения свойств элементов коллекции **Errors**, отображаются свойства объекта **Err**, которых немного больше. Обратите внимание на строки 4–5 диалогового окна **MsgBox** (рис. 17.3). Первая из них представляет **HelpContext** справочного файла, в вторая — полное имя этого файла, в котором имеется более полная информация о возникшей ошибке, чем в свойстве **Description**.

Листинг 17.2 Просмотр свойств элементов коллекции Errors и объекта Err

```
 1:   Private Sub ErrorTest2()
 2:
 3:       On Error GoTo ErrorHandler
 4:
 5:       Set dbsTest = OpenDatabase("LostDatabase")
 6:
 7:       Exit Sub
 8:
 9:   ErrorHandler:
 10:      For i = 0 To Errors.Count - 1
 11:          MsgBox Errors(i).Number & vbCr & _
 12:                 Errors(i).Description & vbCr & _
 13:                 Errors(i).Source, , "Тестирование Error"
 14:
 15:      Next
 16:
 17:  MsgBox Err.Number & vbCr & _
 18:                 Err.Description & vbCr & _
 19:                 Err.Source & vbCr & _
 20:                 Err.HelpContext & vbCr & _
 21:                 Err.HelpFile & vbCr & _
 22:                 Err.LastDllError _
 23:                 , , "Тестирование Err"
 24:
 25:  End Sub
```

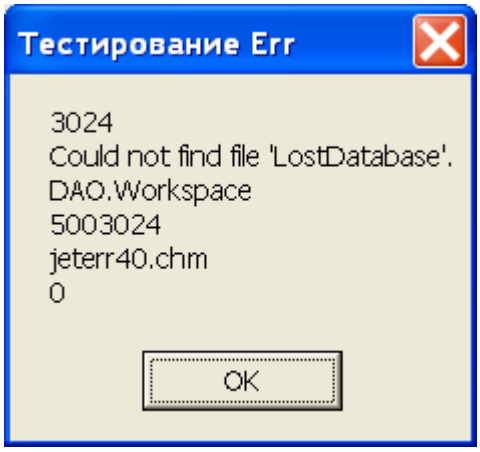

Рис. 17.3
Второе окно, выдаваемое процедурой
ErrorTest2

Файл **jetterr.chm**, указанный в окне на рис. 17.3, — это справочный файл, содержащий список сообщений об ошибках Microsoft Jet (рис. 17.4).

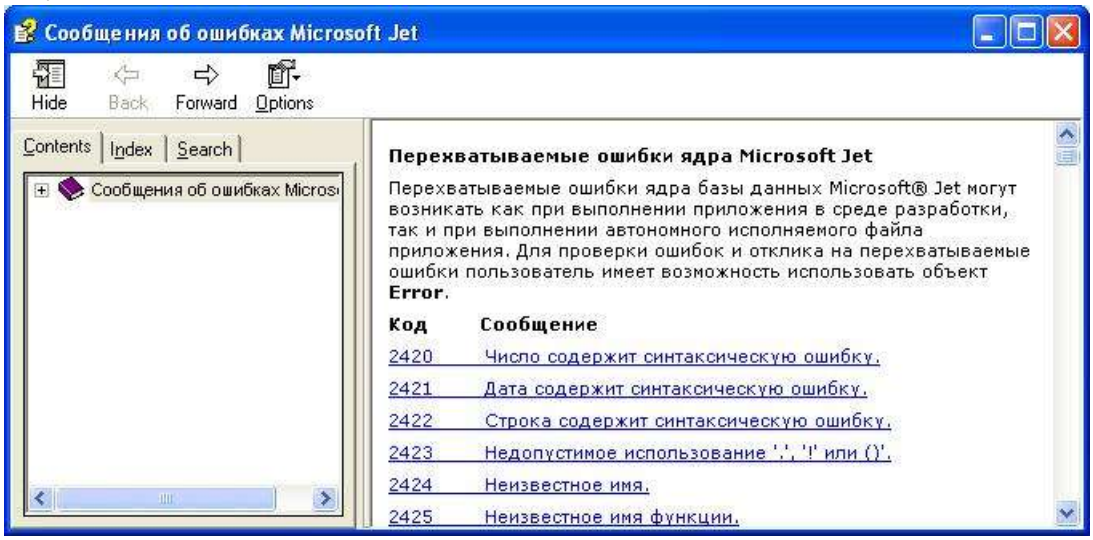

Рис. 17.4
Справочный файл, содержащий список сообщений об ошибках Microsoft Jet

На рис. 17.5 приведено описание ошибки с номером 3024, которая инициируется кодами листингов 17.1 и 17.2.

Рис. 17.5
Описание ошибки с номером 3024, которая инициируется кодами листингов 17.1 и 17.2

Это описание, конечно, «интереснее» получить, не просматривая весь файл, а, например, прямо из кода при возникновении ошибки, как это показано в коде листинга 17.3. Здесь к аргументам **`MsgBox`** был добавлен аргумент *buttons* (константа **`vbMsgBoxHelpButton`** для отображения кнопки **Help**), *helpfile* (Err.HelpFile — свойство объекта **`Err`**) и *context* (Err.HelpContext — свойство объекта **`Err`**).

Листинг 17.3 Просмотр свойств элементов объекта Err

```
 1:   Private Sub ErrorTest3()
 2:       On Error GoTo ErrorHandler
 3:
 4:       Set dbsTest = OpenDatabase("LostDatabase")
 5:
 6:       Exit Sub
 7:
 8:   ErrorHandler:
 9:   MsgBox Err.Number & vbCr & _
 10:                 Err.Description & vbCr & _
 11:                 Err.Source & vbCr & _
 12:                 Err.HelpContext & vbCr & _
 13:                 Err.HelpFile & vbCr & _
 14:                 Err.LastDllError, vbMsgBoxHelpButton, _
 15:                 "Тестирование Err", Err.HelpFile, Err.HelpContext
 16:
 17:  End Sub
```

Если при выдаче на экран диалогового окна **MsgBox** нажать клавишу F1 или щелкнуть на кнопке **Help**, то полная справочная информация о возникшей ошибке будет отображена в справочном окне для Microsoft Visual Basic (без необходимости поиска и запуска файла **jeterr.chm**).

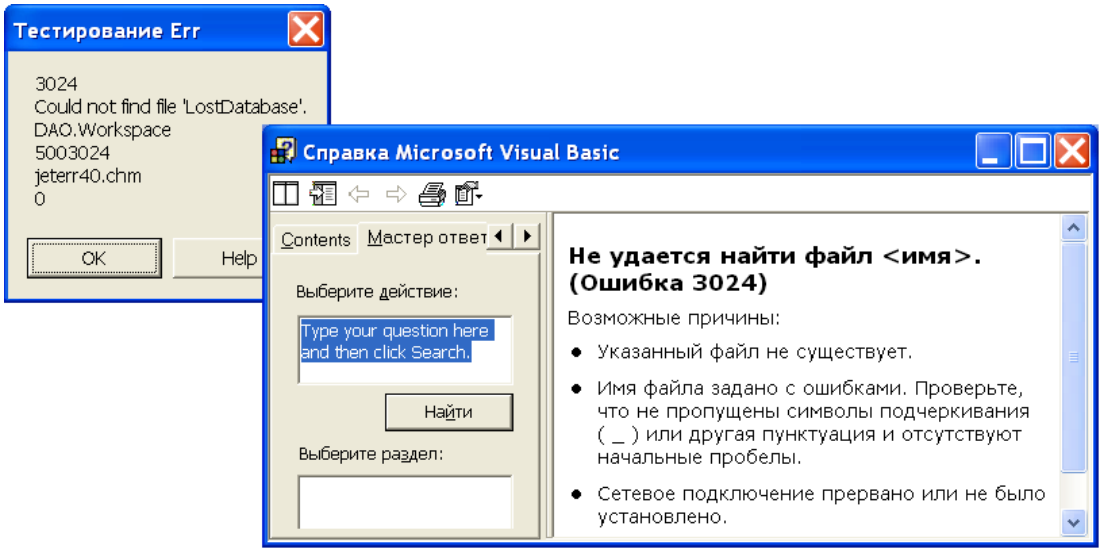

Рис. 17.6
Код листинга 17.3 позволяет больше узнать о возникшей ошибке

Рабочие пространства DAO

DAO поддерживает две различные среды баз данных (databases enivironments) или два типа рабочего пространства:

- **Microsoft Jet workspace** — обеспечивает доступ к данным в базах Microsoft Jet, ODBC-источниках через Microsoft Jet и в базах в формате устанавливаемых драйверов ISAM, таких как Microsoft FoxPro, dBASE, Paradox, Microsoft Excel, Microsoft Exchange и Outlook, Lotus 1-2-3 и др. DAO-модель **Microsoft Jet workspace** представлена на рис. 17.7[2].

- **ODBCDirect workspace** — обеспечивает доступ к данным посредством ODBC, без загрузки ядра (engine) базы данных Microsoft Jet. DAO-модель **ODBCDirect workspace** представлена на рис. 17.8.

[2] На рисунках 16.7 и 16.8 затененные прямоугольники представляют коллекции, а незатененные — объекты.

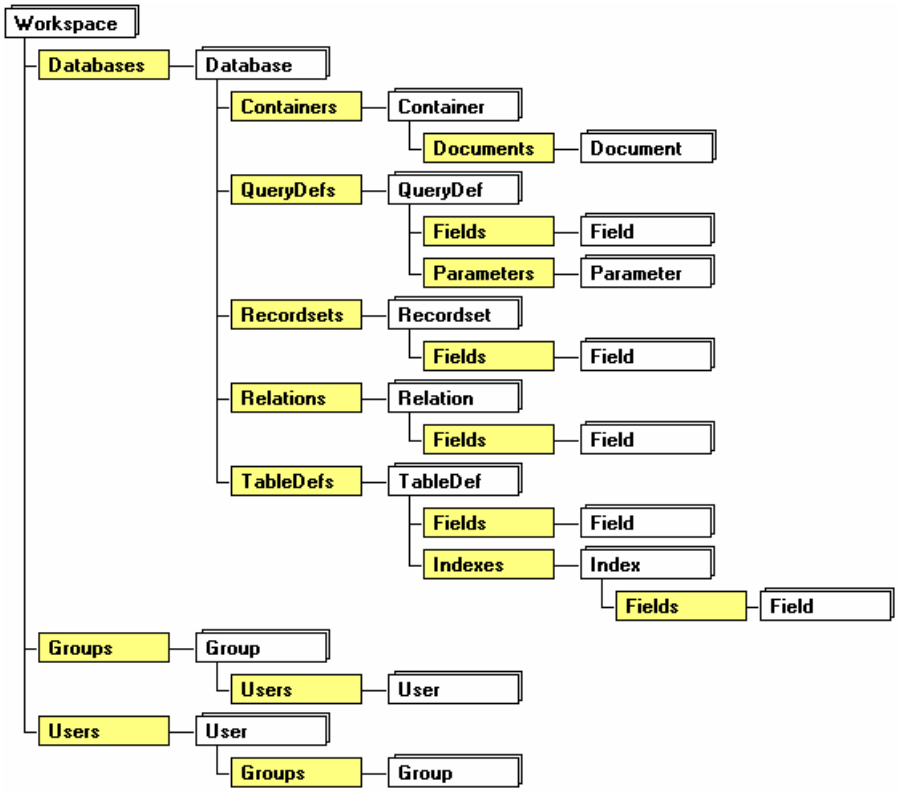

Рис. 17.7
Объектная модель **Microsoft Jet workspace**

Рабочее пространство **Microsoft Jet** следует использовать при работе с базами Microsft Jet (.mdb–файлами) и другими настольными (desktop) ISAM-базами или, если необходимы преимущества некоторых уникальных возможностей системы Microsoft Jet, таких как объединение данных из нескольких баз с различными форматами.

Рабочее пространство **ODBCDirect** удобно, когда необходимо выполнить запрос или *хранимую процедуру* (*stored procedure*) на удаленном сервере, таком как *Microsoft SQL Server*. Вам могут понадобиться некоторые преимущества ODBC, например, *пакетное обновление* (*batch update*), выполнение *асинхронного запроса* (*asynchronous query*).

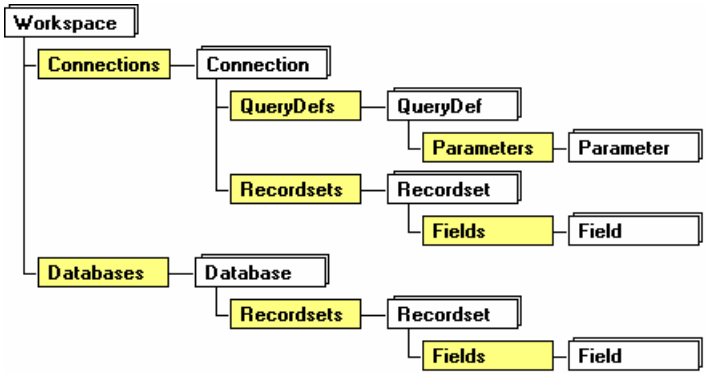

Рис. 17.8
Объектная модель **ODBCDirect workspace**

DAO имеет 17 различных типов объектов. Каждый DAO-объект (кроме **`DBEngine`**) имеет соответствующую коллекцию, которая включает все существующие объекты этого типа. Например, коллекция **`Recordsets`** содержит все открытые **`Recordset`**–объекты. Каждая коллекция принадлежит объекту, который расположен на предыдущем уровне в иерархической DAO-структуре. Например, объект **`Recordset`** «владеет» коллекцией **`Fields`**. Большинство DAO-объектов имеют коллекции и свойства по умолчанию. Например, коллекцией по умолча-

нию объекта **Recordset** является **Fields**, а свойством по умолчанию объекта **Field** является **Value**.

Свойства объекта DBEngine

Чтобы успешно работать с объектом **DBEngine**, необходимо хорошо знать его свойства и методы. **DBEngine** имеет следующие свойства:

DefaultType	Тип — **Long**. Тип рабочего пространства (**Microsoft Jet** или **ODBCDirect**), который будет использоваться при создании Workspace–объекта. По умолчанию свойство устанавливается равным значению константы **dbUseJet** (стандартным является пространство **Microsoft Jet**). Создавая рабочее пространство явно, можно указать значение этого свойства при помощи аргумента метода **CreateWorkspace**, используя константу **dbUseJet** или **dbUseODBC** (для **ODBCDirect**).
DefaultUser	Тип — **String**. Имя пользователя, используемое при создании рабочего пространства по умолчанию. Строка может быть длиной от 1 до 20 символов в рабочем пространстве **Microsoft Jet** и любой длины в рабочем пространстве **ODBCDirect**. Она может включать алфавитно-цифровые символы, пробелы и любые символы, кроме: **"** (двойные кавычки), / (прямой слэш), \ (обратный слэш), **[ ]** (квадратные скобки), **:** (двоеточие), | (символ конвейеризации), < (знак меньше чем), > (знак больше чем), + (знак плюс), = (знак равенства), **;** (точка с запятой), **,** (запятая), **?** (знак вопроса), ***** (звездочка), «ведущих» пробелов и управляющих символов (ASCII 00 – ASCII 31). По умолчанию свойство **DefaultUser** устанавливается равным строке "admin". Имя пользователя обычно «чувствительно» к регистру.
DefaultPassword	Тип — **String**. Пароль, используемый при создании рабочего пространства по умолчанию. **DefaultPassword** — строка длиной до 14 символов в рабочем пространстве **Microsoft Jet** и любой длины в рабочем пространстве **ODBCDirect**. Может включать любые символы, кроме ASCII 0. Значение по умолчанию — строка нулевой длины (""). Пароль «чувствителен» к регистру.
IniPath	Информация о ключе в Windows Registry, который содержит значения для **Microsoft Jet database engine** (только для рабочего пространства Microsoft Jet). Ядро **Microsoft Jet** можно конфигурировать с использованием Windows Registry. Реестр можно использовать для установки таких параметров, как настраиваемые ISAM DLL. Чтобы эта опция имела какой-либо эффект, необходимо установить свойство **IniPath** перед тем, как ваше приложение активизирует любой другой DAO-код. Для инициализации параметров некоторых настраиваемых ISAM-драйверов можно также использовать Windows Registry.
LoginTimeout	Количество секунд, через которое выдается ошибка при неудачной попытке подключения к ODBC-базе данных.
SystemDB	Путь для текущего местонахождения файла с информацией рабочей группы (только для рабочего пространства **Microsoft Jet**).
Version	Рабочее пространство **Microsoft Jet**: для **DBEngine**-объекта возвращается используемая версия DAO. Для объекта **Database** возвращается версия Jet, в которой был создан mdb-файл. Рабочее пространство **ODBCDirect**: для **DBEngine**-объекта возвращается используемая версия DAO. Для объекта **Database** возвращается версия используемого драйвера ODBC.

Можно использовать простую программу для считывания некоторых свойств этого объекта **DBEngine**:

```
Sub DBEngineTest()
    MsgBox DBEngine.DefaultType     'возвращает 2
```

```
    MsgBox DBEngine.LoginTimeout    'возвращает 20
    MsgBox DBEngine.Version         'возвращает 3.6
End Sub
```

Этот код будет работать и в VB-, и VBA-модуле. Для тестирования его в VBA-модуле просто создайте новый модуль в редакторе VB и, поместив весь код в этот модуль, щелкните кнопку **Run Sub/UserForm** (курсор мыши должен находиться на какой-либо строке кода). Если делать это не в Access (в Excel, Word, PowerPoint и т.д.), то необходимо в окне **References** установить ссылку на **Microsoft DAO 3.6 Object Library.**

Для тестирования процедуры в VB необходимо запускать ее из кода событийной процедуры. Поэтому предварительно нужно создать форму и событийную процедуру, например, загрузки этой формы. Ссылка на **Microsoft DAO 3.6 Object Library** также необходима.

Для указания используемого типа рабочего пространства (**Microsoft Jet** или **ODBCDirect**) можно либо непосредственно установить свойство **DefaultType** объекта **DBEngine,** либо использовать метод **CreateWorkspace**. Следующие операторы непосредственно задают свойство **DefaultType** объекта **DBEngine**:

```
DBEngine.DefailtType = dbUseJet
DBEngine.DefailtType = dbUseODBC
```

В этих операторах используются DAO-константы **dbUseJet** и **dbUseODBC**, которые предназначены для указания типа рабочего пространства.

Как видно из рис. 17.7 и 17.8, объекты **Workspace** обеих моделей имеют коллекции открытых баз данных и предоставляют механизмы работы с таблицами базы. Объект **Workspace** можно не включать в коллекцию **Workspaces**, если он используется только в той процедуре (подпрограмме), в которой и создан. В рабочем пространстве **Microsoft Jet** для доступа к данным базы Microsoft Jet, источникам, требующим драйверов ISAM, и ODBC-источникам можно использовать DAO совместно с ядром Microsoft Jet. В рабочем пространстве **ODBCDirect** для доступа к источникам данных ODBC допускается использование DAO без обработки запросов ядром Microsoft Jet.

Использование DAO-модели для Microsoft Jet

Рабочее пространство **Microsoft Jet** включает объекты (рис. 17.7), определяющие структуру базы данных: **TableDef**, **Field**, **Index**, **QueryDef**, **Parameter** и **Relation**. Здесь же находятся объекты, предназначенные для манипуляций с данными, например, **Recordset**. Для защиты данных предназначены такие объекты, как **User**, **Group**, **Container** и **Document**.

Для создания рабочего пространства следует использовать метод **CreateWorkspace** объекта **DBEngine** со следующим синтаксисом:

Синтаксис

Set workspace = CreateWorkspace(*name*, *user*, *password* [, *type*])

Аргумент *name* — строка с уникальным наименованием нового Workspace-объекта. Аргумент *user* — строка с именем владельца Workspace-объекта. Аргумент *password* — строка с паролем Workspace-объекта (до 14-ти символов). Необязательный аргумент *type* — значение одной из констант **dbUseJet** или **dbUseODBC** (по умолчанию — **dbUseJet**).

В следующих операторах для указания используемого типа рабочего пространства применяется метод **CreateWorkspace**:

```
Dim wrkJet As Workspace
Set wrkJet = CreateWorkspace("JetWorkspace", "admin", "")
```

Объект Database

Объект **Database** в модели DAO представляет открытую базу данных. Для открытия базы данных и получения ссылки на представляющий ее объект **Database** во всех приложениях (кроме Microsoft Access) следует использовать метод **OpenDatabase** объекта **BDEngine** или **Workspace** (на них можно не ссылаться для открытия базы в стандартном рабочем наборе). Синтаксис метода **OpenDatabase** следующий[3]:

Синтаксис

```
Set database = [workspace.]OpenDatabase(dbname [, options] [, read-only] [, connect])
```

Здесь *database* — объектная переменная, ссылающаяся на **Database**-объект; *workspace* — объектная переменная, представляющая существующее рабочее пространство (по умолчанию — стандартное пространство). Единственный обязательный аргумент *dbname* — строка с именем существующего файла базы данных Microsoft Jet (mdb-файла) или DSN (data source name) ODBC-источника данных.

Необязательный аргумент *options* устанавливает некоторые опции для базы данных:

- Для пространства **Microsoft Jet** можно использовать значения констант **True** (открывает базу для единоличного использования) и **False** (открывает базу для коллективного использования) — значение по умолчанию.

- Для пространства **ODBCDirect** можно использовать значения констант **dbDriverNoPrompt** [ODBC Driver Manager[4] использует строку соединения (connection string)[5], предоставляемую аргументами *dbname* и *connect*. Если информации будет недостаточно, возникнет runtime-ошибка], **dbDriverPrompt** [ODBC Driver Manager отображает диалоговое окно **ODBC Data Sources**, которое содержит любую релевантную информацию, предоставляемую аргументами *dbname* или *connect*. Строка соединения состоит из DSN, которое пользователь выбирает посредством диалоговых окон. Если пользователь не указывает DSN, используется DSN по умолчанию.], **dbDriverComplete** (значение по умолчанию) [Если аргументы *connect* и *dbname* включают всю необходимую для соединения информацию, ODBC Driver Manager использует строку в *connect*. Иначе он действует, как в случае, когда указывается константа **dbDriverPrompt**] и **dbDriverCompleteRequired** [оказывает действие, подобное действию константы **dbDriverComplete**, за исключением того, что ODBC-драйвер запрещает запросы любых данных, которые не требуются для выполнения соединения].

Аргумент *read-only* — может принимать значение константы **True** для доступа «только на чтение» и — **False** (по умолчанию) для разрешения вносить изменения в базу. Аргумент

[3] Приведен полный синтаксис метода — не только для Microsoft Jet.

[4] Приложение, которое управляет соединениями между источниками данных, доступными посредством ODBC, и драверами, используемыми для доступа.

[5] Строка, используемая для определения источника данных внешней базы и обычно назначаемая свойству **Connect** объектов **QueryDef**, **TableDef**, **Connection** или **Database** или аргументу метода **OpenDatabase**.

connect — строковое выражение для определения информации о подключении, включая пароль.

Объект **BDEngine** (или **Workspace**) позволяет не только открыть существующую базу данных Microsoft Jet, но и создать новую с использованием метода **CreateDatabase** со следующим синтаксисом:

Синтаксис

```
Set database = [workspace.]CreateDatabase (dbname, locale [, options])
```

Здесь *database* — объектная переменная, ссылающаяся на **Database**-объект; *workspace* — объектная переменная, представляющая существующее рабочее пространство (по умолчанию — стандартное пространство). Аргумент *dbname* — строка с (полным) именем создаваемого файла базы данных Microsoft Jet (mdb-файла). Аргумент *locale* — строковое выражение, которое определяет порядок сортировки по умолчанию текстовых значений содаваемой базы данных; в качестве значений этого аргумента следует использовать константы **dbLangGeneral**, **dbLangArabic**, **dbLangCyrillic** и другие (см. справочную систему). Этот же аргумент можно использовать для задания пароля для создаваемой базы данных, конкатенируя значение необходимой константы со строкой вида "`;pwd=NewPassword`", например:

```
dbLangCyrillic & ";pwd=NewPassword"
```

Необязательный аргумент *options* — константа (или комбинация констант), определяющая дополнительные свойства создаваемой базы данных. В качестве констант могут использоваться: **dbEncrypt** (база данных с шифрованными данными), **dbVersion10** (используется формат файлов версии Microsoft Jet database engine 1.0), **dbVersion11** (используется формат файлов версии Microsoft Jet database engine 1.1), **dbVersion20** (используется формат файлов версии Microsoft Jet database engine 2.0), **dbVersion30** (значение по умолчанию, используется формат файлов версии Microsoft Jet database engine 1.0 , совместимый с версией 3.5).

Database-объект имеет свойства и методы, которые позволяют манипулировать данными открытой базы. Например, для любого типа базы данных можно использовать метод **Execute** для выполнения запроса-действия (action query)[6] или задавать свойство **Connect** для установления связи с ODBC-источником данных. Для ограничения времени ожидания выполнения запроса для ODBC-источника данных можно использовать свойство **QueryTimeout**.

Метод **OpenRecordset** позволяет выполнить запрос на выбор данных (select-запрос)[7] и создает **Recordset**-объект.

С базой данных Microsoft Jet (mdb-файл) можно также использовать такие методы и свойства (и коллекции) для манипулирования **Database**-объектами, как:

- методы **CreateTableDef** и **CreateRelation** для создания таблиц и связей;
- метод **CreateProperty** для определения новых **Database**-свойств;
- метод **CreateQueryDef** для создания постоянного (persistent) или временного определения запроса;

[6] Запрос позволяет извлекать данные или изменять их в базе данных.

[7] Запрос, который «отвечает на вопрос», какие данные хранятся в базе данных, и возвращает **Recordset**-объект без изменения данных. После нахождения **Recordset**-данных их можно проверять и изменять в основных таблицах. В отличие от этих запросов запросы-действия могут изменять данные, но не возвращают наборов данных.

- методы **MakeReplica**, **Synchronize** и **PopulatePartial** для создания и синхронизации полных или частичных реплик[8] базы данных;

- свойство **CollatingOrder** для установления алфавитного порядка сортировки полей, содержащих символьные данные.

В рабочем пространстве **ODBCDirect** можно использовать:

- Свойство **Connection** для получения ссылки на объект **Connection**, который соответствует **Database**-объекту.

Объект Recordset

Объект **Recordset** представляет набор записей в базе данных и позволяет манипулировать данными базы на уровне записи. В DAO имеется пять типов **Recordset**-объектов:

- *Table-type Recordset* (табличный) — (только в рабочих пространствах **Microsoft Jet**) представление (в коде) базисной таблицы в базе данных. В рабочем пространстве **Microsoft Jet** объект **Recordset** этого типа можно использовать для добавления, удаления и обновления записей в таблице. Свойство **RecordCount** табличного объекта **Recordset** возвращает число записей в таблице.

- *Dynaset-type Recordset* (динамический набор) — результат запроса по одной или нескольким таблицам. Этот набор записей позволяет добавлять, модифицировать и удалять записи из входящей в него таблицы или таблиц. Объект **Recordset** этого типа можно создавать в рабочих пространствах **Microsoft Jet** и **ODBCDirect**.

- *Snapshot-type Recordset* (статический набор) — результат запроса. Набор содержит значения всех полей, указанных в запросе. Данные в наборе нельзя модифицировать, но объект **Recordset** такого типа требует меньше ресурсов, чем динамический набор.

- *Forward-only-type Recordset* (статический набор с последовательным доступом) — результат запроса. Объект **Recordset** этого типа похож на **Recordset** статического набора, за исключением того, что по его записям можно «продвигаться» только в направлении от начала к концу набора.

- *Dynamic-type Recordset* (динамический) — (только в рабочих пространствах **ODBCDirect**) результат запроса из одной или более таблиц.

Объект **Recordset** создается методом **OpenRecordset** объекта **Database**. Синтаксис метода следующий:

Синтаксис

Set *recordset* = *object*.OpenRecordset (*source* [, *type*] [, *options*] [, *lockedits*])

Здесь *recordset* — объектная переменная, представляющая открываемый объект **Recordset**; *object* — объектная переменная, представляющая уже созданный объект, используемый для создания нового **Recordset**-объекта; *source* — строка, определяющая источник записей [имя таблицы или запроса, SQL-инструкция, возвращающая записи; для табличного объекта

---

[8] Копия базы данных, включающая ее таблицы, запросы, формы, отчеты, макросы и модули.

Recordset в базе данных Microsoft Jet допускается указание только имени таблицы в mdb-файле].

Необязательный аргумент *type* указывает тип объекта **Recordset** и может принимать значение одной из констант:

- **dbOpenTable** — для открытия **Recordset**-объекта типа «табличный»;
- **dbOpenDynaset** — для открытия **Recordset**-объекта типа «динамический набор»;
- **dbOpenSnapshot** — для открытия **Recordset**-объекта типа «статический набор»;
- **dbOpenForwardOnly** — для открытия **Recordset**-объекта типа «статический набор с последовательным доступом»;
- **dbOpenDynamic** — для открытия **Recordset**-объекта типа «динамический».

Необязательный аргумент *options* определяет характеристики нового **Recordset**-объекта и может принимать значение одной из констант:

- **dbAppendOnly** — позволяет добавлять новые записи в набор, но не разрешает редактирование и удаление существующих записей (только для типа «динамический набор» Microsoft Jet);
- **dbSQLPassThrough** — передает SQL-инструкцию ODBC-источнику, доступному посредством Microsoft Jet (только для типа «статический набор» Microsoft Jet);
- **dbSeeChanges**— генерирует runtime-ошибку, если один пользователь изменяет данные, которые редактируются другим пользователем (только для типа «динамический набор» Microsoft Jet);
- **dbDenyWrite** — не разрешает другим пользователям модифицировать и добавлять записи (только для **Recordset**-объекта Microsoft Jet);
- **dbDenyRead** — не разрешает другим пользователям считывать данные из таблицы (только для типа «табличный» Microsoft Jet);
- **dbForwardOnly** — создает статический набор с последовательным доступом (только для **Recordset**-объекта типа «статический набор» Microsoft Jet). Необходим только для обратной совместимости; при этом в качестве аргумента *type* нужно использовать значение константы **dbOpenForwardOnly**;
- **dbReadOnly** — не допускает внесение изменений в **Recordset**-объект (только для Microsoft Jet). Присвоение значения константы **dbReadOnly** аргументу *lockedits* заменяет эту константу, которая предназначена только для обратной совместимости;
- **dbRunAsync** — запускает асинхронный запрос (только для рабочего пространства **ODBCDirect**);

- **dbExecDirect** — выполняет запрос, пропуская **SQLPrepare** и вызывая непосредственно **SQLExecDirect** (только для **ODBCDirect**). Следует использовать эту опцию только для **Recordset**-объекта, не основанного на запросе с параметрами[9];

- **dbInconsistent** — позволяет выполнять несогласованные обновления (только для **Recordset**-объекта типа «динамический набор» и «статический набор» Microsoft Jet);

- **dbConsistent** — позволяет выполнять только согласованные обновления (только для **Recordset**-объекта типа «динамический набор» и «статический набор» Microsoft Jet).

Необязательный аргумент *lockedits* устанавливает тип блокировки данных и может принимать значение одной из констант:

- **dbReadOnly** — запрещает пользователям вносить изменения в **Recordset**-объект (только для рабочего пространства **ODBCDirect**). Можно использовать **dbReadOnly** либо в аргументе *options*, либо в *lockedits*, но не в обоих. В противном случае возникает runtime-ошибка;

- **dbPessimistic** — использует «пессимистичекую» блокировку для определения порядка внесений изменений в **Recordset** в многопользовательской среде. Страница[10], содержащая редактируемую запись, блокируется сразу при использовании метода **Edit** (по умолчанию для рабочих пространств **Microsoft Jet**);

- **dbOptimistic** — использует «оптимистичекую» блокировку[11] для определения порядка внесений изменений в многопользовательской среде. Страница, содержащая редактируемую запись, не блокируется, пока не используется метод **Update**;

- **dbOptimisticValue** — использует «оптимистичекий» параллелизм, основывающийся на значениях строк (только для рабочего пространства **ODBCDirect**);

- **dbOptimisticBatch** — позволяет выполнять «оптимистическое» обновление пакетов[12] (только для рабочего пространства **ODBCDirect**).

---

[9] Запрос, который требует предоставления одного или нескольких значений-критериев перед выполнением запроса.

[10] Часть базы данных, в которой хранятся записи. В зависимости от размера записей страница может содержать более одной записи. В базах данных Microsoft Jet (mdb-файлы) страница имеет длину 2048 (2K) байтов.

[11] Тип блокировки, при которой страница данных, содержащая одну или несколько записей, включая редактируемую запись, недоступна другим пользователям, только пока запись обновляется методом **Update**, но доступна между вызовами методов **Edit** и **Update**. «Оптимистическая» блокировка используется при доступе к ODBC-базам данных или, когда свойство **LockEdit** объекта **Recordset** равно значению константы **False**.

[12] Модель курсора для клиентов, которые работают с курсорами, но не блокируют сервер или выдают обновление по одной строке. Вместо этого клиент обновляет несколько строк, которые буферизуются локально, а затем использует пакетное обновление. Эта модель курсора дает возможность пользователю отменять соединение с сервером и вновь устанавливать соединение с тем же или даже другим сервером.

Для работы с объектом **Recordset** необходимо знать его свойства и методы. Некоторые из них приведены в следующей таблице.

Метод	**Описание**
AddNew	Добавляет новую (пустую) запись. После внесения в поля записи данных следует вызвать метод **Update**.
Close	Закрывает **Recordset**-объект.
Delete	Удаляет текущую запись (которая была достигнута методами **MoveFirst**, **MoveLast**, **MoveNext**, **MovePrevious**, **FindFirst**, **FindLast**, **FindNext**, **FindPrevious**, **Seek**) в **Recordset**-объекте.
Edit	Устанавливает режим редактирования текущей записи **Recordset**-объекта. После редактирования записи следует вызвать метод **Update**.
MoveFirst, **MoveLast**, **MoveNext**, **MovePrevious**	Устанавливает в качестве текущей первую, последнюю, следующую или предыдущую запись **Recordset**-объекта, соответственно.
FindFirst, **FindLast**, **FindNext**, **FindPrevious**	Устанавливает в качестве текущей соответственно первую, последнюю, следующую или предыдущую запись, удовлетворяющую заданным условиям (в виде текстовой строки с условным выражением).
Seek	Устанавливает в качестве текущей запись, удовлетворяющую некоторым условиям, в индексированном **Recordset**-объекте типа «табличный».
Update	Сохраняет результаты методов **AddNew** и **Edit**.
CancelUpdate	Отменяет все изменения в **Recordset**-объекте, выполненные посредством методов **AddNew** и **Edit**.
Свойство	**Описание**
BOF, (**EOF**)	Возвращает значение константы **True**, если указатель текущей записи находится перед первой записью (после последней записи) набора.
NoMatch	Возвращает значение константы **True**, если необходимая запись найдена.
RecordCount	Возвращает количество записей в **Recordset**-объекте.

Для получения подробной информации о методах и свойствах объекта **Recordset** следует обратиться к справочной системе.

Пример подключения VBA-кода к базе данных

Рассмотрим пример использования объекта **Recordset** в Excel-приложении. Ранее нами была создана база данных и заполнены ее таблицы в среде Access. Воспользуемся таблицей **Товары** этой базы для вывода информации в диалоговом Excel-окне, которое в режиме разработки представлено на рис. 17.9.

Рис. 17.9
Форма в режиме разработки для тестирования методов объектов **Database** и **Recordset**

Несмотря на то, что форма довольно простая, приведем, как обычно, ее элементы управления в следующей таблице:

Тип элемента	Свойство, которое изменено (используется в коде)	Значение	Примечание
UserForm	Name	UserForm1	Имя главной формы, на которое можно ссылаться в коде.
	Caption	Подключение к базе данных из VBA-кода	Заголовок формы.
Frame	Name	Frame1	
	Caption	Данные набора	Текст – заголовок.
ListBox	Name	ListBox1	Имя, на которое можно ссылаться в коде.
CommandButton	Name	CmdLoad	
	Caption	Загрузить данные	Заголовок для кнопки.
CommandButton	Name	CmdExit	
	Caption	Выход	Заголовок для кнопки.
	Cancel	True	Клавиша Esq также вызовет процедуру CmdExit_Click.

Код модуля этой формы, включающий две событийные процедуры, приведен в листинге 17.4.

Листинг 17.4 Открытие существующей базы данных и считывание ее записей в элемент управления формы

```
1:  Private Sub CmdLoad_Click()
2:
3:      Dim my_base As Database          'DAO-объект: база данных
4:      Dim rstNwind As Recordset        'DAO- объект: набор данных
5:
6:      'открыть базу данных:
7:      Set my_base = OpenDatabase("f:\фирма.mdb")
8:
9:      'открыть набор:
10:      Set rstNwind = my_base.OpenRecordset("Товары", dbOpenDynaset)
11:
12:     If rstNwind.RecordCount > 0 Then
13:
14:         rstNwind.MoveFirst      'перейти к началу набора
15:
16:         'до конца набора добавлять записи в список:
17:         Do While Not rstNwind.EOF
18:             ListBox1.AddItem rstNwind.Fields(1)
```

```
19:             rstNwind.MoveNext
20:         Loop
21:     End If
22:
23: End Sub
24:
25: Private Sub CmdExit_Click()
26:         Unload Me
27: End Sub
```

В строках 1–23 описана событийная процедура для кнопки с заголовком «**Загрузить данные**». В строках 3–4 описываются переменные **my_base** и **rstNwind**, одна из которых имеет тип **Database**, другая — **Recordset**. Код в строке 7 использует метод **OpenDatabase** объекта **DBEngine**, чтобы открыть существующую базу данных. Код в строке 10 открывает **Recordset**-объект **rstNwind** с записями таблицы **Товары**.

В строке 12 проверяется условие наличия записей в наборе **rstNwind**. Пока не было обращения к записям набора, свойство **RecordCount** не содержит точного количества записей в наборе **rstNwind**, но, по крайней мере, его значение будет больше нуля, если в таблице **Товары** имеется хотя бы одна запись. Если записи в таблице **Товары** имеются, то выполняются строки кода 14–20: метод **MoveFirst** устанавливает текущей первую запись набора, далее в цикле считываются записи второго столбца таблицы с использованием коллекции **Fields** — полей набора, которым соответствуют кортежи отношения.

В строке 17 находится заголовок цикла **Do While**, который в качестве условия окончания цикла использует значение свойства **EOF** набора **rstNwind**. В строке 19 применяется метод **MoveNext** объекта **Recordset** для перемещения к следующей записи набора. Результат работы процедуры **CmdLoad_Click** приведен на рис. DAO.4.

В строках 25–27 описана событийная процедура кнопки с заголовком «**Выход**», которая просто завершает работу приложения.

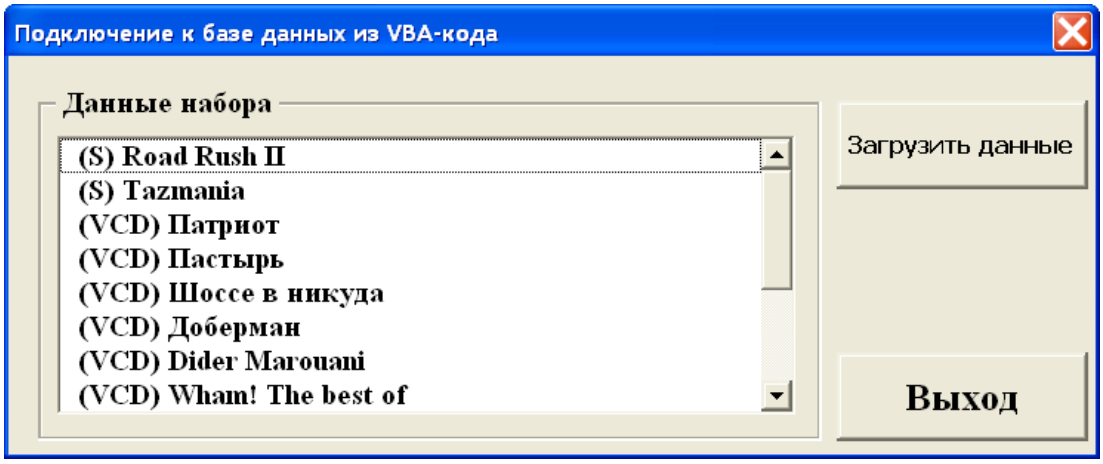

Рис. 17.10
Диалоговое окно в режиме исполнения для тестирования методов объектов **Database** и **Recordset**

Рассмотрим подробнее строку 10 кода листинга 17.4, которая связывает с **Recordset**-объектом **rstNwind** набор записей таблицы **Товары** из mdb-базы данных:

```
10:  Set rstNwind = my_base.OpenRecordset("Товары", dbOpenTable)
```

В этом операторе используется константа **dbOpenTable** для выбора **Recordset**-объекта **rstNwind** типа «табличный». В результате в набор попали все поля таблицы (все кортежи отношения) **Товары**, хотя нам, на самом деле, нужны были только наименования товаров. По этой причине в строке 18 при извлечении наименований из набора используется оператор, в котором указан номер второго поля набора:

```
18:   ListBox1.AddItem rstNwind.Fields(1)
```

Если данные всей таблицы **Товары** действительно не нужны, то набор **rstNwind** можно открыть другим оператором:

```
Set rstNwind = my_base.OpenRecordset("SELECT НаименованиеТовара FROM Товары", _
                                     dbOpenDynaset)
```

И поскольку теперь коллекция **`Fields`** содержит только один элемент типа **`Field`**, оператор в строке 18 должен иметь вид:

```
ListBox1.AddItem rstNwind.Fields(0)
```

Что изменилось в вызове метода **`OpenRecordset`**? В качестве первого аргумента используется строка с SQL-запросом, поэтому переменная **rstNwind** ссылается на набор, состоящий из таблицы с одним столбцом. Но важно в данном случае то, что мы, вообще-то, можем использовать не только этот простой QSL-запрос, а любой — какой только нам будет нужен.

Изменим немного рассматриваемую форму: добавим к форме текстовое окно для ввода строки и вместо элемента **`ListBox`** поместим на форму элемент **`MSFlexGrid`**, который позволяет выводить несколько столбцов таблицы (рис. 17.11). При этом таблица со свойствами элементов управления формы будет иметь следующий вид:

Тип элемента	Свойство, которое изменено (используется в коде)	Значение	Примечание
UserForm	Name	UserForm1	Имя главной формы, на которое можно ссылаться в коде.
	Caption	Подключение к базе данных из VBA-кода	Заголовок формы.
Frame	Name	Frame1	Имя, на которое можно ссылаться в коде.
	Caption	Данные набора	Текст – заголовок.
MSFlexGrid	Name	MSFlexGrid1	Имя, на которое можно ссылаться в коде.
CommandButton	Name	CmdLoad	
	Caption	Загрузить список	Заголовок для кнопки.
CommandButton	Name	CmdExit	
	Caption	Выход	Заголовок для кнопки.
	Cancel	True	Клавиша Esq также вызовет процедуру CmdExit_Click.
Label	Name	Label1	Метка для TextBox1.
	Caption	SQL-запрос	
TextBox	Name	TextBox1	Текстовое окно для ввода строки с SQL-запросом.
	MultiLine	True	Для размещения многострочных данных.

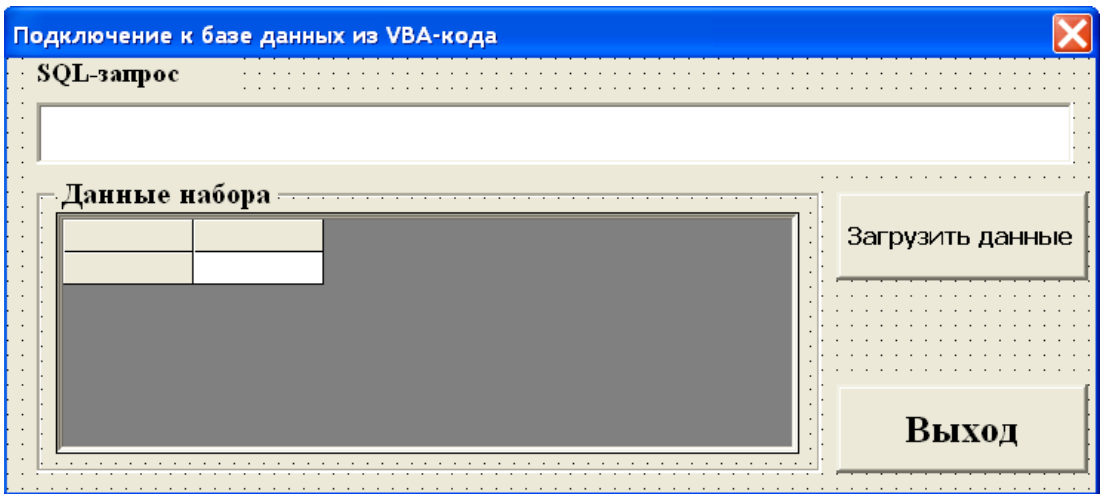

Рис. 17.11
Диалоговое окно в режиме разработки для тестирования методов объектов **Database** и **Recordset**

С учетом свойств элемента **MSFlexGrid** событийная процедура кнопки с заголовком «Загрузить данные» может выглядеть так, как приведено в листинге 17.5.

Листинг 17.5 Открытие существующей базы данных и считывание ее записей в элемент управления формы MSFlexGrid

```
1:  Private Sub CmdLoad_Click()
2:
3:      Dim my_base As Database          ' DAO-объект: база данных
4:      Dim rstNwind As Recordset        ' DAO-объект: набор данных
5:      Dim SQLstr As String
6:
7:      ' открыть базу данных:
8:      Set my_base = OpenDatabase("f:\фирма.mdb")
9:
10:     ' ввести SQL-запрос:
11:     SQLstr = TextBox1.Text
12:
13:     ' открыть набор:
14:     Set rstNwind = my_base.OpenRecordset(SQLstr, dbOpenDynaset)
15:
16:     If rstNwind.RecordCount > 0 Then
17:
18:         rstNwind.MoveFirst      ' перейти к началу набора
19:
20:         ' задать количество строк:
21:         MSFlexGrid1.Cols = rstNwind.Fields.Count + 1
22:
23:         ' установить ширину первых колонок:
24:         MSFlexGrid1.ColWidth(0) = 1
25:         If MSFlexGrid1.Cols > 1 Then MSFlexGrid1.ColWidth(1) = 1500
26:         If MSFlexGrid1.Cols > 2 Then MSFlexGrid1.ColWidth(2) = 2000
27:
28:         ' до конца набора добавлять записи в таблицу:
29:         Do While Not rstNwind.EOF
30:
31:             'сформировать элемент добавления к таблице:
32:             newElement = ""
33:             For i = 0 To rstNwind.Fields.Count - 1
34:                 newElement = newElement & Chr(9) & rstNwind.Fields(i)
35:             Next
36:
37:             'добавить элемент к таблице:
38:             MSFlexGrid1.AddItem newElement
39:
40:             rstNwind.MoveNext
41:         Loop
42:     End If
43:
44: End Sub
```

В строке 11 приведенного кода считывается, возможно, строка с SQL-запросом, который затем используется в строке 14 в качестве аргумента метода **OpenRecordset**. В строке 21 задается свойство **Cols** (количество столбцов должно соответствовать количеству столбцов **Recordset**-объекта) объекта **MSFlexGrid1**. В строках 24–26 задается ширина первых трех столбцов объекта **MSFlexGrid1**. Эта часть кода существенно зависит от выводимых данных, поэтому ее трудно сделать универсальной.

В строках 32–35 формируется строка для добавления данных в объект **MSFlexGrid1**. Эта строка состоит из значений полей набора **rstNwind** (**rstNwind.Fields.Count** — количество полей в наборе), разделяемых символом табуляции. В строке 38 к объекту **MSFlexGrid1** добавляется очередная строка с данными.

Конечно, код процедуры **CmdLoad_Click** должен иметь обработчик прерываний для обработки ошибок, но для простоты кода этого не сделано, так как чаще всего такие естественные для разработчика элементы программного кода затеняют главную цель того или иного примера.

На рис. 17.12 приведено диалоговое окно данного приложения, в котором в текстовое поле введена строка **“Товары”**, т.е. здесь не используется SQL-инструкция, а просто указывается имя таблицы. В результате работы процедуры **CmdLoad_Click** в объекте **MSFlexGrid1** отображаются все поля таблицы.

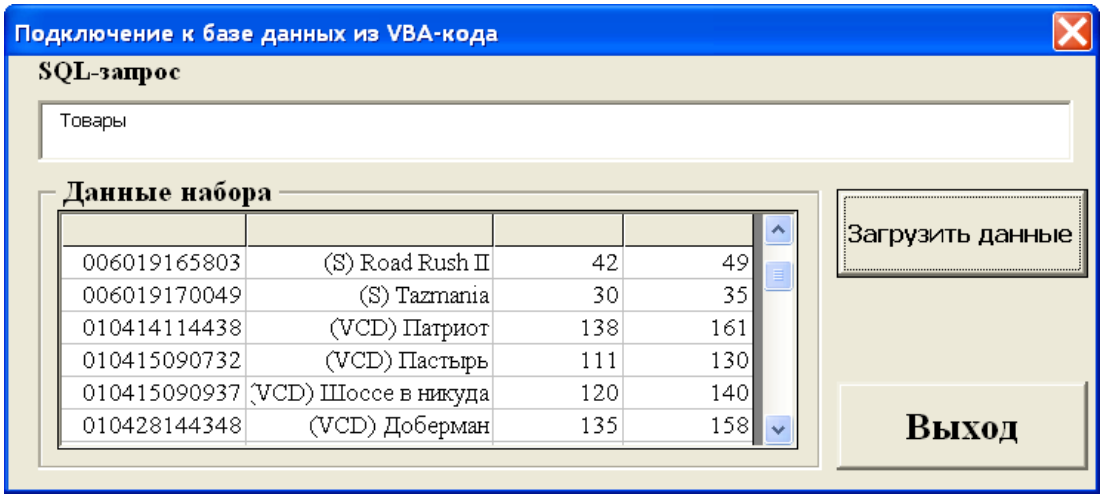

Рис. 17.12

Диалоговое окно отображает все поля таблицы
Товары

На рис. 17.13 приведен результат работы процедуры **CmdLoad_Click**, код которой использует (из текстового окна **TextBox1**) следующий SQL_запрос:

```
SELECT a.НаимТовара, b.Количество FROM Товары a, Запасы b WHERE
a.КодТовара=b.КодТовара
```

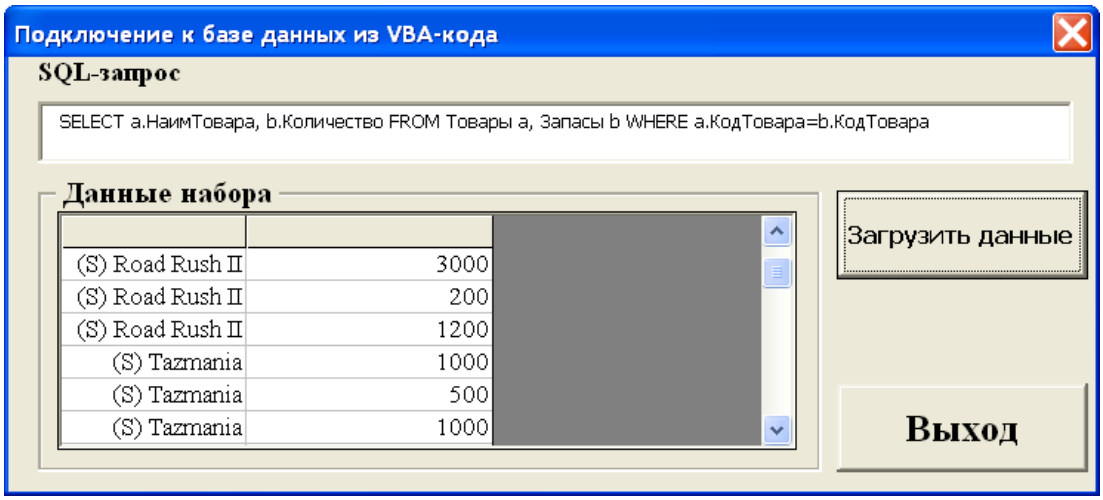

Рис. 17.13

Диалоговое окно отображает поля, полученные в
наборе в результате SQL-запроса

На рис. 17.14 приведен результат работы процедуры **CmdLoad_Click**, код которой использует (из текстового окна **TextBox1**) следующий SQL_запрос:

SELECT c.НаимПодразделения, SUM(b.Количество) FROM Запасы b, Подразделения c WHERE b.КодПодразделения=c.КодПодразделения GROUP BY c.НаимПодразделения

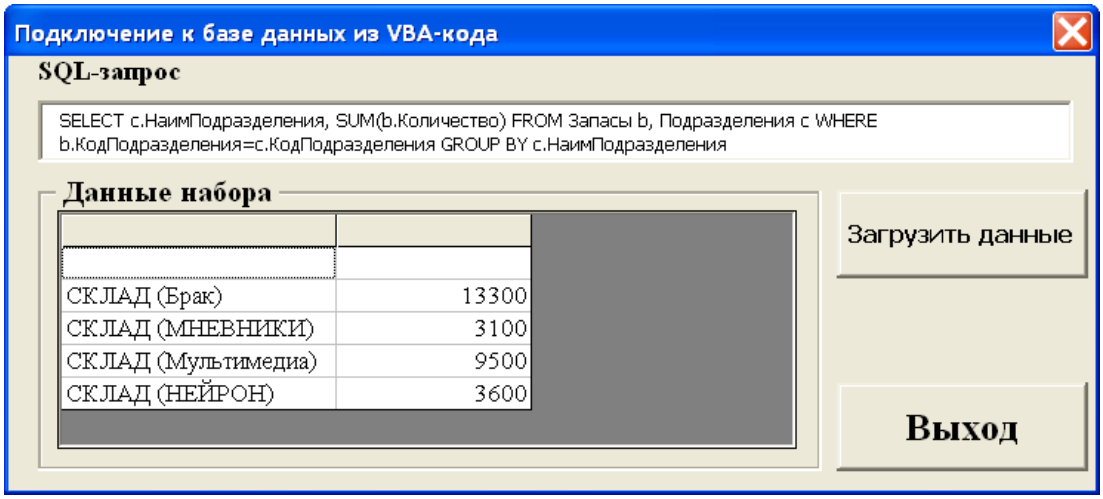

Рис. 17.14

Диалоговое окно отображает поля, полученные в наборе в результате SQL-запроса

Пример создания базы данных из VBA-кода

Рассмотрим пример создания базы данных из кода VBA. Создавать базы данных из кода приходится в основном для каких-либо временных таблиц или для данных, которые существенно связаны с датами. Например, если создается база данных с накладными некоторой производственно-торговой фирмы, то есть смысл накладные для каждого отчетного периода (например, месяца) хранить в отдельных mdb-файлах. Относительно небольшой размер файла за период будет способствовать увеличению скорости обработки информации, а хранение данных в нескольких файлах поможет решить задачи надежности — при порче файла текущего периода информация, хранящаяся в файлах с данными предыдущих периодов (быть может, даже защищенных от записи), не пострадает.

Создавать базы данных из кода можно также при настройке какой-либо системы для использования конкретным пользователем. Пользователь сам мог бы определять и наименование базы данных, и характеристики необходимых полей, хотя, конечно, работал бы при этом в ограничивающем его возможности диалоге, существенно зависящем от специфики системы.

В листинге 17.6 приведен код простой процедуры, которая создает файл с именем **test_mdb.mdb** и таблицу **Телефоны** (эту базу данных, конечно, необязательно содавать из кода) с тремя полями.

Листинг 17.6 Создание базы данных из VBA-кода

```
1  Sub CreateDatabaseTest()
2  'создает базу данных test_mdb.mdb с таблицей Телефоны
3
4     Dim myDatabase As Database
5     Dim myTable As TableDef
6     Dim myField As Field
7
8     'создание базы данных:
9     On Error GoTo NotCreateBase
10     Set myDatabase = DBEngine.CreateDatabase("test_mdb", dbLangGeneral)
11     MsgBox "База данных test_mdb.mdb создана"
12
13     'создание таблицы базы данных с тремя полями:
14     On Error GoTo NotCreateTable
15
```

```
16    Set myTable = New TableDef
17
18    With myTable
19       .Fields.Append .CreateField("Фамилия", dbText)
20       .Fields.Append .CreateField("Имя", dbText)
21       .Fields.Append .CreateField("Телефон", dbText)
22    End With
23
24    'присвоить имя таблице:
25    myTable.Name = "Телефоны"
26    myDatabase.TableDefs.Append myTable
27
28    Set myDatabase = Nothing
29
30    MsgBox "Таблица создана"
31    Exit Sub
32
33  NotCreateBase:
34    MsgBox "У нас проблема с созданием базы данных test_mdb.mdb"
35    Exit Sub
36
37  NotCreateTable:
38    MsgBox "У нас проблема с созданием таблицы"
39
40  End Sub
```

В строках 4–6 описываются переменные типа **`Database`** (база данных), **`TableDef`** (определение таблицы) и **`Field`** (поле таблицы), место которых в модели DAO показано на рис. 17.7. Эти типы данных доступны в VBA-коде, если в окне **References** (меню **Tools | References**) установлен флажок **Microsoft DAO 3.6 Object Library**.

Замечание

Для проверки того, поддерживается ли VBA-системой тот или иной тип данных, не следует торопиться набирать этот тип с клавиатуры. Лучше подождать после набора ключевого слова As, пока VB-редактор предложит выбрать поддерживаемый тип данных из всплывающего списка. Полезно также набирать наименования свойств и методов на одном регисте — если VB-редактор не изменяет отдельные символы в таких наименованиях, значит (из-за ошибки при вводе) они ему неизвестны.

В строке 9 «включен» обработчик ошибок для сигнализации о том, что по каким-то причинам mdb-файл не создан. В строке 10 находится оператор, создающий новую базу данных при помощи метода **`CreateDatabase`** объекта **`DBEngine`**. Если обработчик ошибки создания базы не запускается, значит, база создана, можно выполнять код дальше. При возникновении ошибки выполнение кода продолжается с оператора в строке 34: перед завершением процедуры выдается сообщение.

В строке 14 устанавливается новый обработчик ошибок. В строке 16 создается новый объект типа **`TableDef`**. В строках 18–22 для нового объекта создаются три текстовых поля. В строке 25 новая таблица получает имя, а в строке 26 созданная таблица добавляется к базе данных.

Если вы устанавливали на своем компьютере ERWin и создавали базу данных, рассматриваемую в главе «Типы данных, константы и переменные», при помощи этой системы, то теперь вам может стать частично понятным сгенерированный для создания базы данных код.

Доступ к источникам данных ODBC

При доступе к данным ODBC рабочее пространство **Microsoft Jet** обеспечивает следующие (некоторые) возможности, отсутствующие в рабочем пространстве **ODBCDirect**:

- *Обновляемые объединения*: возможность модификации данных в **Recodrset**-объектах, представляющих объединения нескольких таблиц.

- *Поддержка подключенных таблиц*: возможность хранить в локальной базе данных Microsoft Jet постоянные связи с сервером баз данных с возможностями кэширования информации о структуре, полях и индексах подключенной таблицы в локальной базе данных.

- *Поддержка методов поиска*: доступность методов **FindFirst**, **FindNext**, **FindPrevious** и **FindLast** объекта **Recodrset**.

- *Нестандартные свойства:* возможность добавлять новые свойства к существующим объектам.

- *Перекрестные запросы*: возможность использовать SQL-оператор TRANSFORM для создания перекрестных запросов.

- *Доступ к неоднородным данным*: возможность работать с удаленными данными, данными mdb-файлов и данными, доступными посредством устанавливаемых драйверов ISAM. Возможность объединять данные из разных источников.

- *Язык определения данных (Data Definition Language, DDL)*: возможность модифицировать структуру базы данных.

Технология **ODBCDirect** дает возможность осуществления доступа к удаленным данным посредством модели DAO поверх интерфейса ODBC API, что позволяет устанавливать соединения, создавать курсоры и выполнять сложные процедуры при минимальном использовании ресурсов локального компьютера, минуя ядро Microcoft Jet. **ODBCDirect** имеет следующие преимущества:

- *Прямой доступ*: VBA-приложение может непосредственно обращаться к источникам данных ODBC, что позволяет повысит производительность, снизить сетевой трафик, переложить на сервер бо́льшую часть нагрузки по обработке данных.

- *Улучшенный доступ к специфическим возможностям сервера*: приложение получает доступ к функциям сервера баз данных, недоступным через **Microsoft Jet**. Например, можно задавать входные параметры для хранимых (на сервере) процедур и контролировать возвращаемые значения.

- *Асинхронные запросы*: после выдачи запроса, можно выполнять любой программный код, не дожидаясь результата запроса, время от времени проверяя, не завершен ли запрос.

- *Пакетное обновление с нежесткой блокировкой (batch optimistic updating)*: приложение может локально кэшировать изменения объекта **Recordset** и передавать их на сервер единым пакетом.

- *Выполнение хранимых процедур*: приложение может обрабатывать возвращаемые значения и выходные параметры хранимых процедур.

Перед использованием ODBC необходимо зарегистрировать источник данных ODBC, информация о котором сохраняется в реестре и становится доступной всем приложениям. Зарегистрировать источник данных можно с помощью диспетчера источников данных ODBC или из VB-кода. В качестве примера зарегистрируем как источник ODBC базу данных **f:\фирма.mdb**, для работы с которой использовалась форма, представленная на рис. 17.11–17.14.

Щелкните (один раз или дважды, в зависимости от настройки вашей системы) значок (рис. 17.15) диспетчера источников данных ODBC.

Рис. 17.15
Значок диспетчера источников данных ODBC

Возможно, окно диспетчера источников данных ODBC будет похоже на представленное на рис. 17.16.

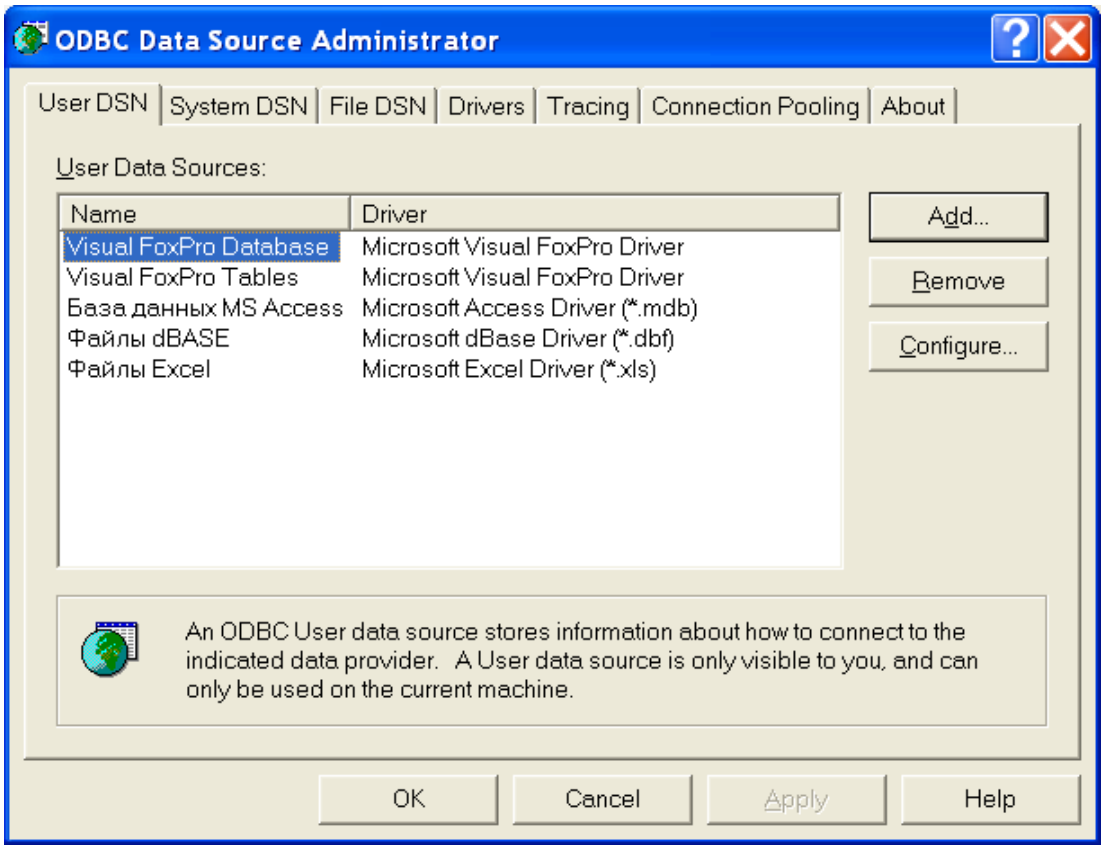

Рис. 17.16

Возможно, окно диспетчера источников данных
ODBC будет похоже на это окно

Щелкните кнопку **Add** (добавить) для добавления нового источника и в списке **Select a driver for which you want to set up a data source** (рис. 17.17) окна **Create New Data Source** выберите **Microsoft Access Driver (*.mdb)**. Затем щелкните кнопку **Finish**.

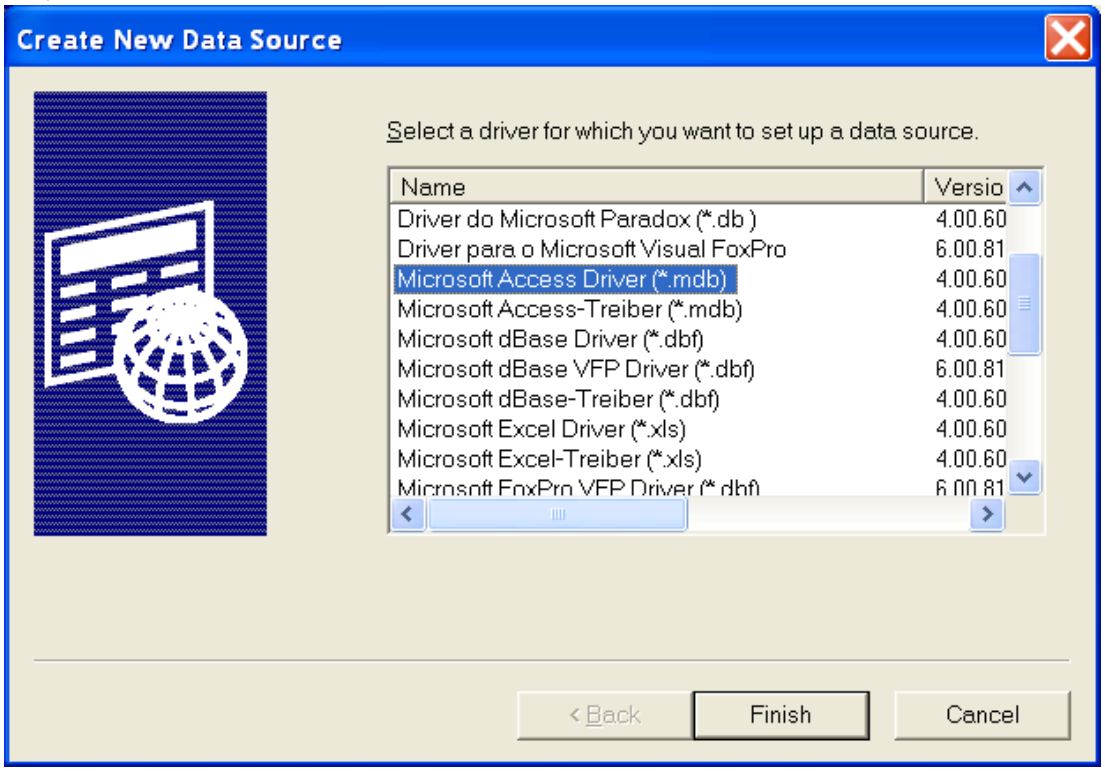

Рис. 17.17

В списке **Select a driver for which you want to set up a data source** выберите **Microsoft Access Driver (*.mdb)**

В текстовом поле **Data Source Name** (имя источника данных) окна **ODBC Microsoft Access Setup** (**Установка драйвера ODBC для Microsoft Access**) введите имя нового источника (то, на что из VB-кода мы будем ссылаться как на DSN), например, **Тестирование ФИРМА.MDB** (рис.17.18). Для выбора базы данных для ODBC-источника щелкните кнопку **Select**.

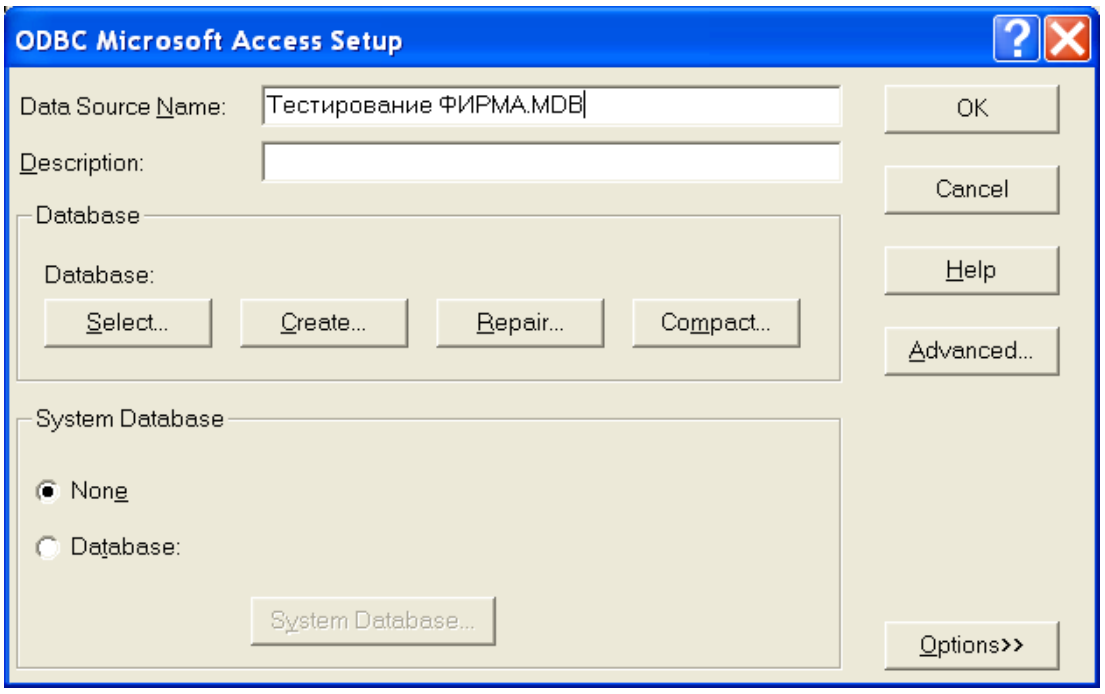

Рис. 17.18

В текстовом поле **Data Source Name** окна **ODBC Microsoft Access Setup** введите имя нового источ-

В диалоговом окне **Select Database** очень просто указать базу данных — рис. 17.19.

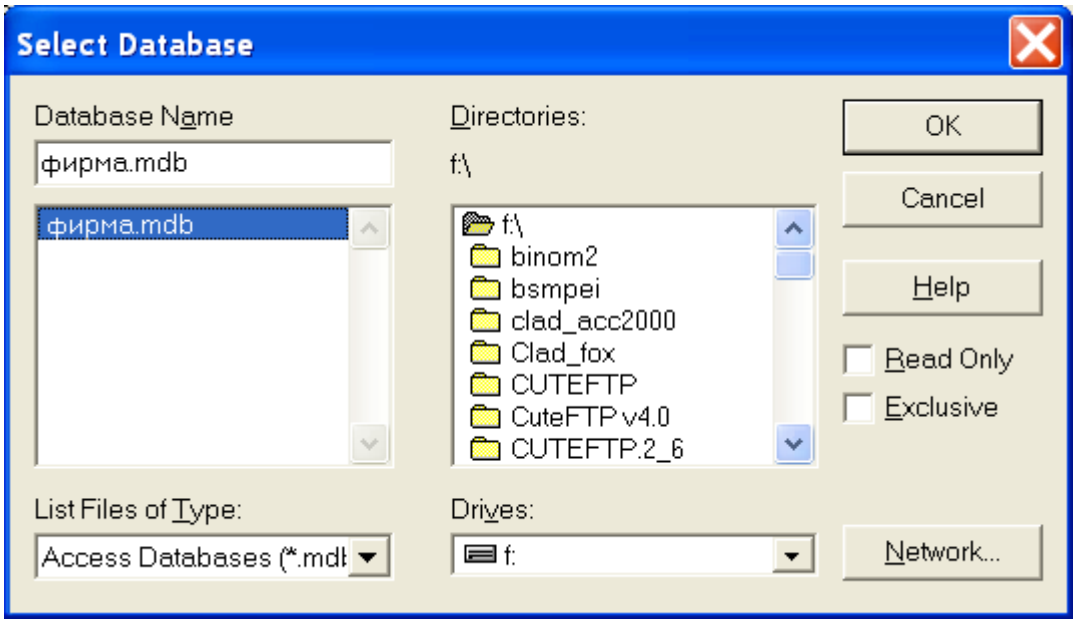

Рис. 17.19

В диалоговом окне **Select Database** очень просто
указать базу данных

В результате в окне **ODBC Microsoft Access Setup** можно будет увидеть выбранный mdb-файл (рис. 17.20), а в окне **ODBC Data Source Administrator** (рис. 17.21) появляется новый источник ODBC-данных — **Тестирование ФИРМА.MDB**.

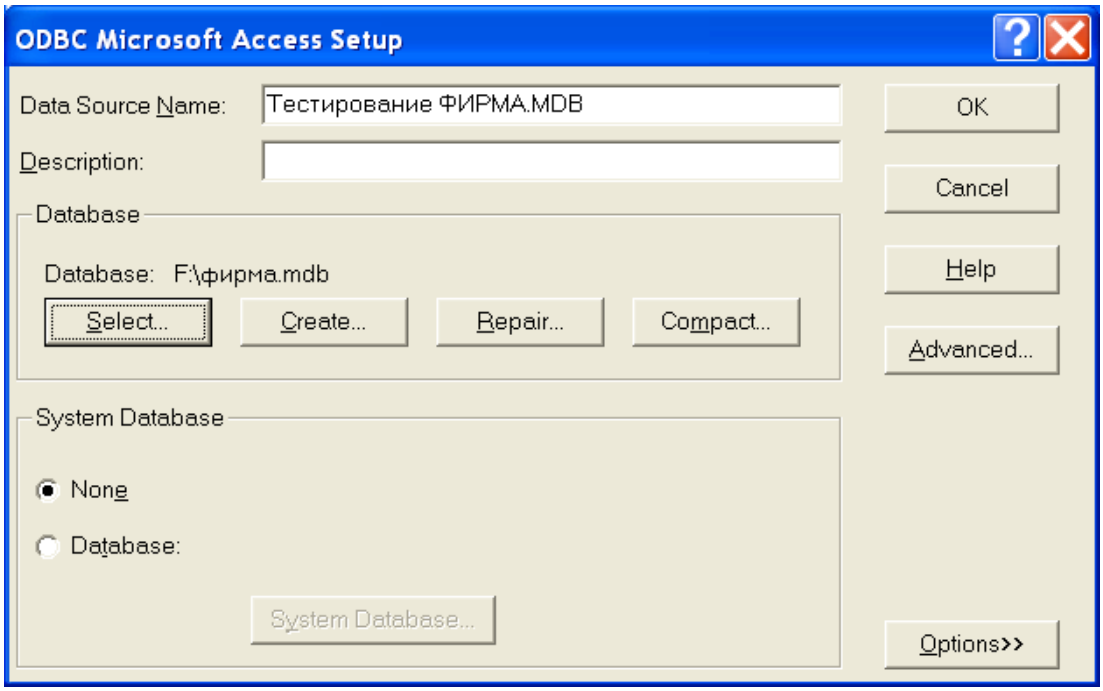

Рис. 17.20

В диалоговом окне **ODBC Microsoft Access Setup** должна быть именно такая информация

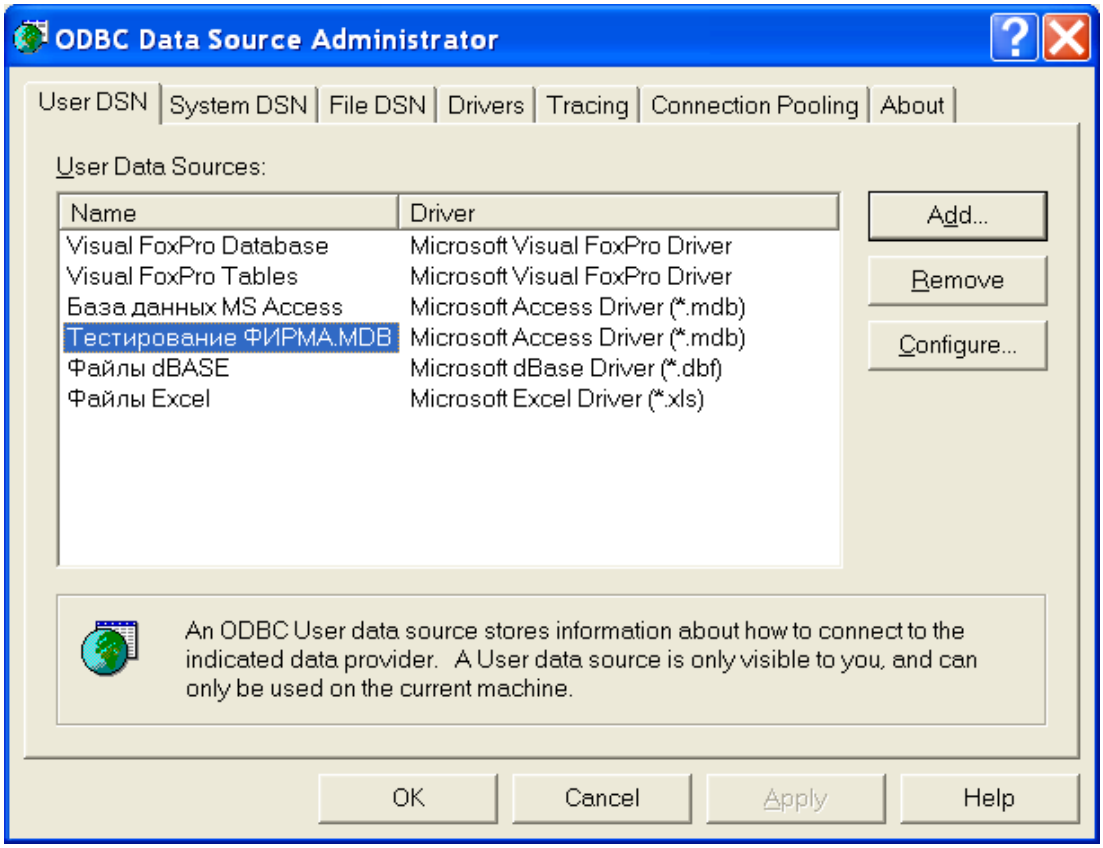

Рис. 17.21

В окне **ODBC Data Source Administrator** появляется новый источник ODBC-данных — **Тестирование ФИРМА.MDB**

Использование DAO-модели для ODBCDirect

Модель объектов рабочено пространства **ODBCDirect** включает подмножество объектов рабочего пространства **Microsoft Jet** и объект **`Connection`** (см. рис. 17.8). Объект **`Workspace`**, содержащий коллекцию **`Connections`**, представляет рабочее пространство **ODBCDirect** и может быть включен в коллекцию рабочих пространств **`Workspaces`**.

Объект **`Connection`** является элементом коллекции **`Connections`** (все открытые объекты **`Connection`**), представляет соединение с источником ODBC в рабочем пространстве **ODBCDirect** и предоставляет следующие возможности для доступа к ODBC-данным:

- *Асинхронное подключение*. Программа может асинхронно подключаться к источнику данных ODBC, не ожидая установления, но впоследствии проверяя результат выполнения соединения.

- *Асинхронные запросы*. Программа может асинхронно выполнять запросы к ODBC-источнику, не ожидая выполнения длительных запросов, но впоследствии проверяя результат выполнения запроса.

- *Объекты QueryDef*. Для работы с данными ODBC-источника можно определять объекты **`QueryDef`**, представляющие запросы.

Для создания объекта **Connection** следует использовать метод **OpenConnection** объекта **Workspace** с синтаксисом:

Синтаксис

Set *connection* = [*workspace*.]OpenConnection (*dbname* [, *options*] [, *readonly*] [, *connect*])

Здесь *connection* — объектная переменная типа **Connection**, которая будет ссылаться на новое соединение. Необязательная переменная *workspace* — переменная типа **Workspace**, ссылающаяся на рабочее пространство, которое будет содержать новое соединение.

Аргумент *dbname* — строковое выражение, определяющее имя зарегистрированного источника данных.

Необязательный аргумент *options* определяет, следует ли и когда выдавать запрос на подключение, следует ли устанавливать асинхронное соединение. Для этого аргумента предусмотрены следующие константы:

- **dbDriverNoPrompt** — ODBC Driver Manager (диспетчер драйверов ODBC) использует строку подключения, исходя из содержимого аргументов *dbname* и *connect*.
- **dbDriverPrompt** — ODBC Driver Manager выводит на экран диалоговое окно **ODBC Data Sources**, которое отображает любую релевантную информацию, содержащуюся в *dbname* или *connect*. *Строка подключения* (connection string) определяется DSN, выбранным пользователем в этом окне.
- **dbDriverComplete** — (значение по умолчанию) Если аргумент *connect* включает всю необходимую информацию для выполнения соединения, ODBC Driver Manager использует в качестве строки соединения значение аргумента *connect*. В противном случае поведение диспетчера аналогично поведению при использовании константы **dbDriverPrompt**.
- **dbDriverCompleteRequired** — Использование этой константы имеет такой же результат, как и при использовании **dbDriverComplete**, но ODBC-драйвер блокирует запросы любой информации, которая не требуется для завершения подключения.
- **dbRunAsync** — Выполнять метод асинхронно. Может использоваться с любой другой *options*-константой.

Необязательный аргумент *readonly* (тип **Boolean**) определяет режим доступа: при значении **True** — доступ только на чтение; при **False** (значение по умолчанию) — на чтение и запись.

Необязательный аргумент *connect* — строка подключения, содержащая параметры для диспетчера драйверов ODBC. Она может включать имя пользователя, пароль, имя базы данных по умолчанию и имя источника данных, переопределяющее значение аргумента *dbname*. Строка подключения начинается с «ODBC;» и содержит последовательность именованных параметров, необходимых драйверу для доступа к данным. Состав именованных параметров зависит от источника данных, но как минимум, требуются идентификатор пользователя (UID), пароль (PWD) и имя источника данных (DSN). Если аргумент не указывается, значения UID и/или PWD берутся из свойств **UserName** и **Password** объекта **Workspace**.

Для полученного в предыдущем разделе ODBC-источника данных с именем **Тестирование ФИРМА.MDB** событийная процедура **CmdLoad_Click** из листинга 17.5 может быть переписана, как показано в листинге 17.7.

Листинг 17.7 Открытие существующей базы данных и считывание ее записей в элемент управления формы MSFlexGrid

```
Private Sub CmdLoad_Click()

    Dim my_wrk As Workspace
    Dim my_cnn As Connection
    Dim ConnectStr As String
    Dim rstNwind As Recordset
    Dim SQLstr As String
    Dim newElement As String, i As Integer

    ConnectStr = "ODBC;DSN=Тестирование ФИРМА.MDB;UID=;PWD="

    'ввести SQL-запрос:
    SQLstr = TextBox1.Text

    Set my_wrk = DBEngine.CreateWorkspace("NewODBCDirect", "sa", "", dbUseODBC)
    Set my_cnn = my_wrk.OpenConnection("Тестирование ФИРМА.MDB ", _
                                       dbDriverNoPrompt, False, ConnectStr)
    'открыть набор:
    Set rstNwind = my_cnn.OpenRecordset(SQLstr, dbOpenDynaset)

    rstNwind.MoveLast      'перейти в конец набора для инициализации RecordCount
    If rstNwind.RecordCount > 0 Then

        rstNwind.MoveFirst       'перейти к началу набора

       'задать количество строк:
       MSFlexGrid1.Cols = rstNwind.Fields.Count + 1

       'установить ширину первых колонок
       MSFlexGrid1.ColWidth(0) = 1
       If MSFlexGrid1.Cols > 1 Then MSFlexGrid1.ColWidth(1) = 1500
       If MSFlexGrid1.Cols > 2 Then MSFlexGrid1.ColWidth(2) = 2000

       'до конца набора добавлять записи в список
       Do While Not rstNwind.EOF

          'сформировать элемент добавления:
          newElement = ""
          For i = 0 To rstNwind.Fields.Count - 1
             newElement = newElement & vbTab & rstNwind.Fields(i)
          Next

          'добавить элемент к таблице:
          MSFlexGrid1.AddItem newElement

          rstNwind.MoveNext
       Loop
    End If
End Sub
```

Этот пример приведен только для демонстрации использования ODBC-источника данных. Аналогичным образом можно в качестве источника данных ODBC использовать файл в dbf-формате.