Глава 10

Работа с базами данных

В этой главе...

- Терминология, используемая в программировании баз данных
- Использование компонента `TClientDataSet`
- Проектируем приложение для работы с базой данных DVD-дисков

В этой главе речь пойдет о том, как с помощью компонента `TClientDataSet` и компонентов, управляемыми данными (или, другими словами, информационных компонентов), быстро и легко создать небольшое приложение "Каталог DVD-дисков".

Мы поговорим также об использовании компонента `TActionManager` в связке с компонентами `TActionMainMenuBar` и `TActionToolBar` для создания современного интерфейса пользователя, пример которого показан на рис. 10.16.

Кое-что о базах данных

Для чего предназначен компьютер? Скорее всего, вы согласитесь с тем, что название "компьютер" не точно отражает сущность этого устройства. Слово *компьютер* позаимствовано из английского языка, в котором слово *computer* произошло от глагола *compute* — вычислять. Само собой разумеется, это устройство способно производить вычисления, однако главным его предназначением является хранение и получение данных. Эти данные могут представлять что угодно: номер телефона любимой тети, количество осадков, выпавших за данный период, курс валют и так далее. Так вот, практически любая порция информации хранится в какой-нибудь базе данных (БД); мы будем иметь дело с так называемыми *реляционными базами данных*. Помимо тривиальных значений, современные базы данных могут хранить изображения, звуки и видео.

Поскольку управление данными является очень важным делом, Delphi предлагает разработчикам специальные компоненты, которые можно найти на нескольких страницах окна Tool Palette (Палитра инструментов). Среди них наиболее важными являются компоненты страниц Data Access (Доступ к данным) и Data Controls (Элементы управления данными). На странице Data Access содержатся компоненты, позволяющие организовать связь с БД и заглянуть в ее содержимое. На странице Data Controls вы найдете компоненты, управляемые данными, которые умеют отображать и редактировать данные, хранящиеся в БД. Программа Delphi Database Explorer позволяет просматривать, создавать и редактировать доступные вам БД (если быть точнее, псевдонимы баз данных).

Для реализации поддержки БД Delphi использует механизм баз данных Borland Database Engine (BDE). С его помощью вы сможете с легкостью подключаться к удаленной базе данных Oracle на сетевом сервере так, будто бы она является таблицей Paradox, хранящейся на жестком диске вашего компьютера. (В целях простоты изложения материала в этой главе мы будем работать с локальной базой данных.) Delphi настолько умело уводит вас от мучительных подробностей, связанных с доступом к БД, что вам без особого труда удастся повысить уровень локальной БД до уровня клиент-сервер, изменив всего лишь несколько свойств.

Поговорим о терминах

Программисты, имеющие дело с базами данных, в процессе своей работы пользуются несколькими специфическими терминами, с которыми вам тоже придется познакомиться, иначе понять принципы работы любой базы данных вам будет очень тяжело. Ниже представлен "толковый словарь" терминов, связанных с базами данных.

Термин	Пояснение
Поле (field)	Наименьший элемент данных, находящийся в БД. Поле имеет тип и имя. Например, поле `Company Name` (Название фирмы) вероятнее всего будет иметь текстовый тип. Хранящиеся в поле данные могут быть представлены в виде строк, чисел, булевских значений, больших текстов (например, характеристики товара), изображений (например, фотографии товара) и тому подобного.
Запись (record)	Коллекция полей, в которых содержится связанная информация. Например, в записи `Customer` (Заказчик) может содержаться имя и адрес, а также уникальный идентификационный номер заказчика.
Таблица (table)	Коллекция записей, каждая из которых имеет одинаковую структуру поля. Когда таблица отображается в виде сетки, то каждая строка будет представлять собой одну запись, а каждый столбец — одно поле.
База данных (database)	Коллекция связанных между собой таблиц, обычно идентифицируемая каталогом, в котором содержатся таблицы, или *псевдонимом* (alias) базы данных.
Индекс (index)	В любой книге индекс помогает найти нужную тему. В контексте баз данных индекс помогает находить записи, имеющие определенное значение поля.
Первичный индекс (primary index)	Запрашиваемый многими функциями БД, первичный индекс управляет порядком, в котором отображаются записи БД. Поле первичного индекса должно быть уникальным; то есть никакие две записи не могут иметь одно и то же значение в этом поле.
Вторичный индекс (secondary index)	Более "свободный" тип индекса, который может быть основан на комбинации полей и который не обязательно должен быть уникальным.
Запрос (query)	Выражение на языке SQL (Structured Query Language — язык структурированных запросов), которое выбирает записи и поля из одной или нескольких таблиц. Запросы служат для получения информации *из* БД — например, это может быть список всех заказчиков, не оплативших счета.

За создание баз данных и обработку запросов, поступающих к ним, отвечают системы управления базами данных (СУБД). В настоящее время разработано много СУБД, среди которых наиболее известными являются Microsoft Access, Paradox, Oracle, dBase, InterBase и другие.

Разные СУБД по-разному хранят таблицы и поля БД. Например, в СУБД Paradox и InterBase каждой таблице отводится отдельный файл, а в Microsoft Access и InterBase в одном файле хранится несколько таблиц. В последнем случае база данных представляет собой имя файла, к которому можно обратиться по известному пути. А в системах типа клиент-сервер (например, Microsoft SQL Server) вся информация хранится на отдельном компьютере, и доступ к ней осуществляется с использованием языка SQL.

Идея использования псевдонимов заслуживает отдельного внимания. Когда-то, во времена заселения Америки, многие переселенцы придумывали себе псевдонимы, чтобы избежать правосудия. Применительно к базам данных псевдонимы имеют более "мирное" назначение. Дело в том, что конкретные свойства баз данных отличаются друг от друга, и пользователю придется очень туго, если ему нужно будет указывать в приложении каждый каталог, файл, сервер и тому подобное. Псевдонимы как раз и служат для облегчения работы пользователей. Псевдоним содержит все сведения, необходимые для получения доступа к конкретной БД. Эти сведения сообщаются только один раз во время создания псевдонима. После этого приложению, которому понадобится информация из какой-то БД, нужно использовать этот псевдоним.

Теперь, после знакомства с терминологией, мы сразу же перейдем к практическим занятиям. Сначала мы рассмотрим пример применения компонента `TCliendDataSet`, а затем создадим приложение, с помощью которого вы сможете заведовать домашней коллекцией DVD-дисков.

Компонент TClientDataSet

Компонент `TClientDataSet` (его можно найти на странице Data Access) идеально подходит для создания автономных приложений баз данных, поскольку он характеризуется перечисленными ниже особенностями.

- ✓ Он хранит табличные данные в оперативной памяти, что приводит к существенному ускорению выполнения всех операций.
- ✓ Он может сохранять данные в форматах CDS (двоичные файлы, размер которых меньше XML-файлов) и XML.
- ✓ Вместе со своим приложением вы должны распространять только динамически подключаемую библиотеку (DLL) `MIDAS.DLL` (370 Кбайт).
- ✓ Вы можете скомпоновать весь компонент в исполняемый файл, включив модуль `MidasLib` в список `uses` приложения (если вы сделаете это, размер исполняемого файла увеличится на 252 Кбайт, зато вам не придется распространять вместе со своим приложением DLL-библиотеку `MIDAS.DLL`).

Чтобы просто оценить, насколько легко создать простое приложение баз данных с помощью компонента `TClientDataSet`, мы сейчас займемся разработкой приложения, которое будет отображать на форме данные, взятые из файла-примера `clients.xml`.

Чтобы создать это приложение, вы должны сначала перенести на форму компоненты `TClientDataSet` и `TDataSource`, которые находятся на странице Data Access.

Компонент TClientDataSet будет необходим для доступа к XML-файлу, а компонент TDataSource — для отображения информации из БД на форме в элементах, управляемыми данными.

Чтобы данные из файла clients.xml отобразились на форме, этот файл нужно назначить свойству FileName (Имя файла) компонента TClientDataSet. Чтобы выбрать файл clients.xml, щелкните на кнопке с многоточием (...) справа от свойства FileName в окне Object Inspector (Инспектор объектов), после чего на экране появится стандартное диалоговое окно Open (Открыть).

По умолчанию файл clients.xml (а также другие образцы баз данных) хранится в каталоге X:\Program Files\Common Files\Borland Shared\Data (следует отметить, что специально для данного примера использовалась нестандартная локализованная версия этого файла; X — это конкретная буква диска).

Прежде чем добавлять информационные элементы управления, которые будут отображать данные на форме, вы должны указать компоненту TDataSource, из какой БД считывать данные, необходимые для информационных элементов управления. Чтобы сделать это, назначьте компонент TClientDataSet свойству DataSet (Набор данных) компонента TDataSource.

Теперь перенесите на форму компоненты TDBGrid, TDBImage и TDBNavigator, которые находятся на странице Data Controls, как показано на рис. 10.1.

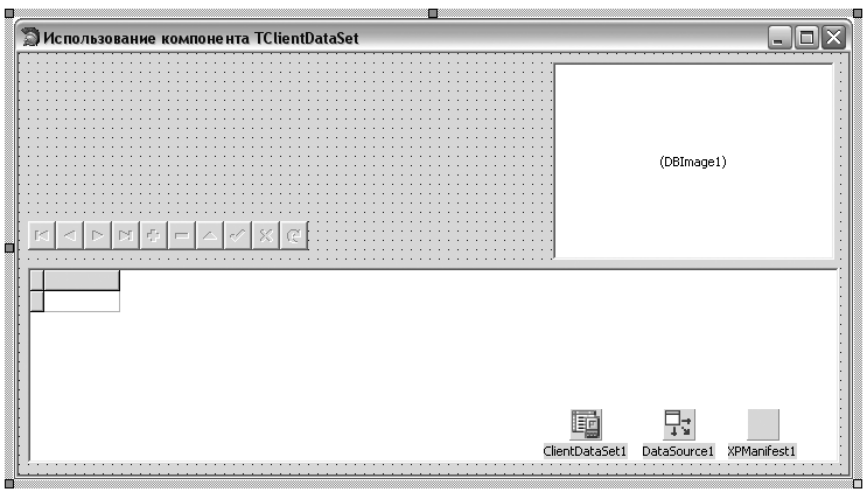


Рис. 10.1. Некоторые информационные элементы управления

Компонент TDBGrid предназначен для отображения всех записей компонента TClientDataSet, компонент TDBNavigator нужен для того, чтобы пользователь мог просматривать базу данных, а компонент TDBImage используется для предварительного просмотра изображений, хранящихся в поле Graphic (Графические изображения) базы данных clients.xml.

Чтобы эти компоненты могли использовать данные компонента TClientDataSet, вы должны назначить компонент TDataSource свойству DataSource (Источник данных) каждого из этих трех компонентов. Чтобы компонент TDBImage отображал изображения, хранящиеся в поле ФОТО, вы должны назначить соответствующее поле его свойству DataField (Поле данных), как показано на следующем рисунке.

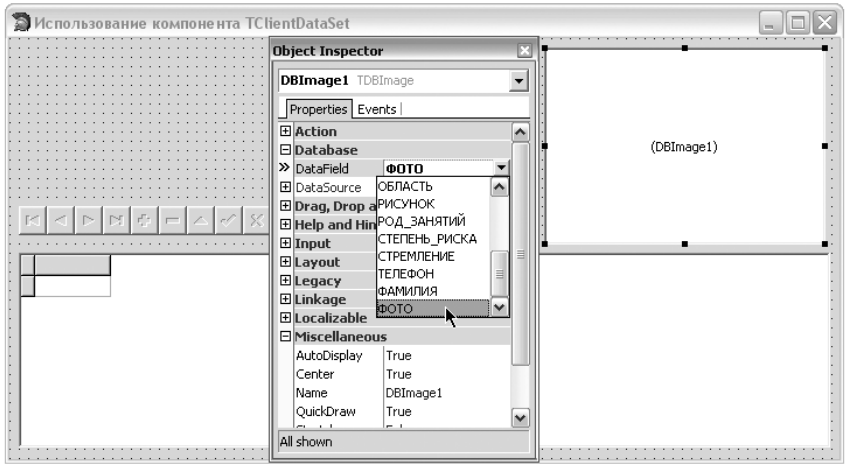
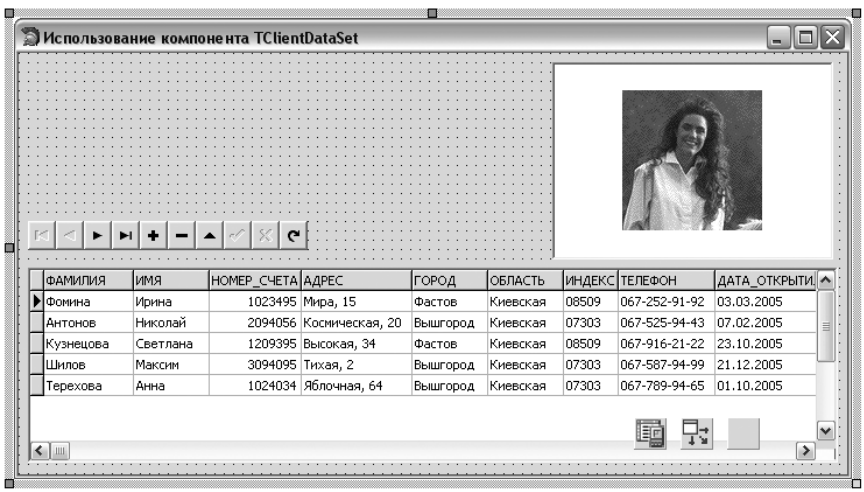


Рис. 10.2 Выбор одного из доступных полей

Теперь, когда все компоненты связаны между собой, можно присвоить свойству `Active` (Активный) компонента `TClientDataSet` значение `True`, чтобы посмотреть, все ли вы сделали правильно. Если все сделано правильно, компоненты `TDBGrid` и `TDBImage` должны отобразить данные из базы данных `clients.xml` (рис. 10.3).



ФАМИЛИЯ	ИМЯ	НОМЕР_СЧЕТА	АДРЕС	ГОРОД	ОБЛАСТЬ	ИНДЕКС	ТЕЛЕФОН	ДАТА_ОТКРЫТИ
Фомина	Ирина	1023495	Мира, 15	Фастов	Киевская	08509	067-252-91-92	03.03.2005
Антонов	Николай	2094056	Космическая, 20	Вышгород	Киевская	07303	067-525-94-43	07.02.2005
Кузнецова	Светлана	1209395	Высокая, 34	Фастов	Киевская	08509	067-916-21-22	23.10.2005
Шилов	Максим	3094095	Тихая, 2	Вышгород	Киевская	07303	067-587-94-99	21.12.2005
Терехова	Анна	1024034	Яблочная, 64	Вышгород	Киевская	07303	067-789-94-65	01.10.2005

Рис. 10.3. Просмотр информации из БД на этапе проектирования

Когда вы используете компонент `TClientDataSet` для доступа к БД, то на этапе проектирования не следует присваивать свойству `Active` значение `True`, поскольку вся БД будет скомпонована в исполняемый файл, в результате чего размер исполняемого файла может возрасти до 1 Мбайт. Чтобы отобразить данные во время выполнения, вы должны открыть БД либо путем присвоения свойству `Active` компонента `TClientDataSet` значения `True`, либо посредством вызова его метода `Open`:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ClientDataSet1.Active := True;
  { или }
  { ClientDataSet1.Open; }
end;
```

Приложение "Каталог DVD-дисков"

Приложение "Каталог DVD-дисков" является простым приложением баз данных, на примере которого мы решим следующие задачи.

- ✓ Определение структуры компонента `TClientDataSet`.
- ✓ Загрузка данных и сохранение данных компонента `TClientDataSet` в XML-файлах.
- ✓ Фильтрация элементов в БД.
- ✓ Поиск определенного элемента в БД.
- ✓ Отслеживание изменений, произведенных в БД.
- ✓ Итерация по всем записям.
- ✓ Оптимизация размера БД.

Определение структуры компонента `TClientDataSet`

Для начала вы должны перенести на форму компоненты `TDataSource` и `TClientDataSet` и связать их между собой, назначив свойству `DataSet` компонента `TDataSource` компонент `TClientDataSet`. Чтобы сократить количество вводимых символов, компонент `TClientDataSet` можно переименовать на `CDS`.

Выберите на форме компонент `TClientDataSet`, в окне Object Inspector — свойство `FieldDefs` (Определения полей) и щелкните на кнопке с многоточием (...), чтобы отобразить редактор коллекций FieldDefs Collection Editor (рис. 10.4).

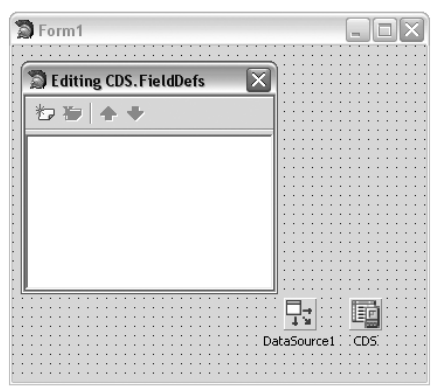


Рис. 10.4. Редактор FieldDefs Collection Editor

Чтобы добавить в таблицу новые поля, щелкните на кнопке Add New (Добавить новое) или нажмите клавишу <Insert> на клавиатуре. Поскольку нужно сохранить только идентификатор (ID) и название фильма, щелкните на кнопке Add New два раза и создайте два неопределенных поля (рис. 10.5).

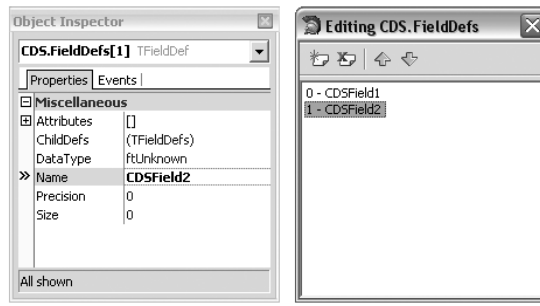


Рис. 10.5. Новые поля

В завершение процесса добавления новых полей в компонент `TClientDataSet` необходимо изменить некоторые свойства этих полей. Сначала вы должны переименовать первое поле, выбрав для него имя `MovieID`, и присвоить его свойству `DataType` (Тип данных) значение `ftInteger`. Укажите для второго поля имя `MovieName`, присвойте его свойству `DataType` значение `ftString` и, поскольку названия фильмов порой бывают слишком длинными, присвойте его свойству `Size` (Размер) значение `100`.

Наконец, закройте окно редактора FieldDefs Collection Editor, щелкните правой кнопкой мыши на компоненте `TClientDataSet` на форме и выберите в контекстном меню команду Create DataSet (Создать набор данных), чтобы создать новый набор данных. Все это необходимо сделать для того, чтобы на этапе проектирования назначить поля компонента `TClientDataSet` информационным элементам управления, что, собственно говоря, очень скоро и будет сделано.

Создание интерфейса пользователя приложения

Чтобы создать интерфейс пользователя приложения “Каталог DVD-дисков”, показанный на рис. 10.16, потребуется перенести на форму несколько компонентов.

Для начала перенесите компоненты `TActionManager`, `TActionMainMenuBar` и `TActionToolBar`, которые находятся на странице Additional (Дополнительные). Компоненты `TActionMainMenuBar` и `TActionToolBar` будут автоматически выстроены в верхней части формы.

Затем перенесите два компонента `TGroupBox` из страницы Standard (Стандартные), добавьте в левый компонент `TGroupBox` компоненты `TEdit` и `TListBox`, а в правый компонент `TGroupBox` — два информационных компонента `TDBEdit`. Наконец, добавьте столько компонентов `TLabel`, сколько потребуется для описания назначения этих текстовых окон (рис. 10.6).

Перенесите на форму компонент `TStatusBar` из страницы Win32 и присвойте его свойству `SimplePanel` (Простая панель) значение `True`. Кроме этого свяжите между собой информационные компоненты правого компонента `TGroupBox` и компонент `TDataSource`, назначив свойству `DataField` первого компонента `TDBEdit` поле `MovieID`, а свойству `DataField` второго компонента `TDBEdit` — поле `MovieName`.

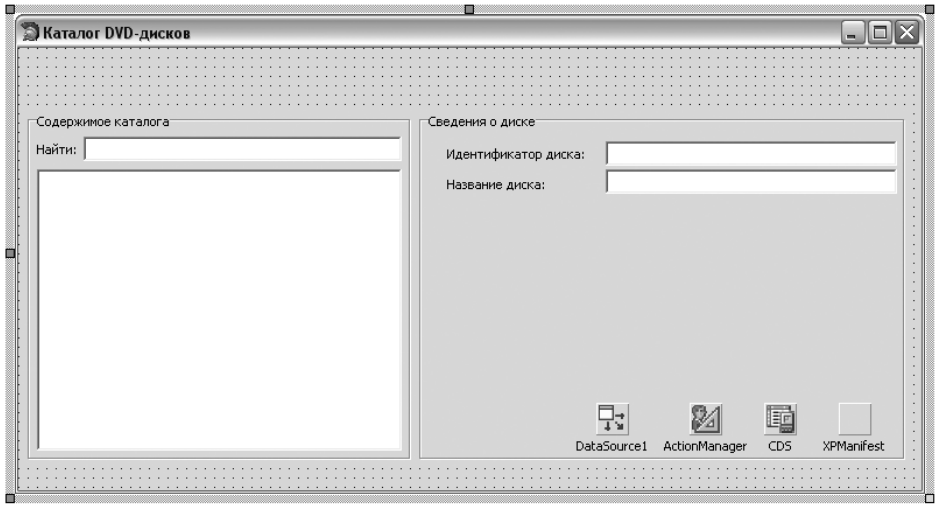


Рис. 10.6. Интерфейс пользователя приложения

Обслуживающие методы

Прежде чем добавлять действия в компонент `TActionManager`, мы должны создать три обслуживающих метода для загрузки и сохранения данных, а также для отображения названия каждого фильма в окне списка `TListBox`. Первым мы рассмотрим метод `DisplayDataset`, который отображает название каждого фильма в окне списка `TListBox`:

```
procedure TForm1.DisplayDataset;
begin
  MovieList.Clear;
  MovieList.Items.BeginUpdate;

  CDS.First;
  while not CDS.Eof do
  begin
    MovieList.Items.Add(CDS.FieldByName('MovieName').AsString);
    CDS.Next;
  end;

  MovieList.Items.EndUpdate;

  if MovieList.Items.Count = 0 then
    StatusBar.SimpleText := 'Каталог дисков пуст.'
  else
    StatusBar.SimpleText := 'Количество дисков в каталоге: ' +
IntToStr(MovieList.Items.Count);
end;
```

Как видите, метод `DisplayDataset` сам по себе не очень сложный, поскольку он будет перебирать все записи компонента `TClientDataSet` и копировать содержимое поля `MovieName` в компонент `TListBox`.

Чтобы скопировать данные из поля MovieName, метод DisplayDataset вызывает метод First компонента TClientDataSet для перехода к первой записи. Затем он входит в цикл while not Eof, в котором перебираются все записи компонента TClientDataSet. Метод FieldByName используется для нахождения поля MovieName в активной записи. Для добавления значения поля MovieName в окно списка TListBox применяется свойство AsString (В виде строки), которое позволяет обрабатывать значение поля как строку. Наконец, в цикле while производится вызов метода Next компонента TClientDataSet, который переходит к следующей записи в таблице, активизируя ее.

В методах OpenCatalog и SaveCatalog, используемых для загрузки и сохранения данных, тоже нет ничего сложного:

```
procedure TForm1.OpenCatalog(const AFileName: string);
begin
  CDS.Close;
  CDS.FileName := AFileName;

  { Вызываем процедуру CreateDataSet для обработки пустого набора
данных. }
  if not FileExists(AFileName) then CDS.CreateDataSet;
  CDS.Open;

  { Вызываем метод DisplayDataSet для автоматического отображения
    данных при открытии файла или при создании нового файла. }
  DisplayDataset;
end;

procedure TForm1.SaveCatalog(const AFileName: string);
begin
  CDS.FileName := AFileName;
  CDS.SaveToFile(CDS.FileName, dfXML);
end;
```

Метод SaveToFile компонента TClientDataSet принимает два параметра: имя искомого файла и формат, в котором вы хотите сохранить данные компонента TClientDataSet. Вы можете сохранить данные в любом из трех форматов, предлагаемых типом TDataPacketFormat:

```
TDataPacketFormat = (dfBinary, dfXML, dfXMLUTF8);
```

Значение по умолчанию dfBinary используется для того, чтобы сохранить данные в виде CDS-файлов, которые имеют меньший размер, чем XML-файлы, но, с другой стороны, их трудно просматривать и редактировать в текстовом редакторе. Значения dfXML и dfXMLUTF8 позволяют сохранять данные в формате XML.

Создание действий и меню

Теперь можно добавить в компонент TActionManager несколько действий. Дважды щелкните на компоненте TActionManager на форме, а затем щелкните на кнопке New Action (Новое действие) пять раз подряд, чтобы добавить пять новых действий, которые будут соответствовать командам Новый, Открыть, Сохранить, Сохранить как и Выход меню Файл, как показано на рис. 10.7.

Рис. 10.7. Добавление действий в компонент `TActionManager`

После создания действий нужно описать их категории. Поскольку мы собираемся использовать эти действия для реализации команд меню Файл, мы должны выбрать все пять действий и в поле свойства `Category` (Категория) каждого из них ввести `File` (рис. 10.8).

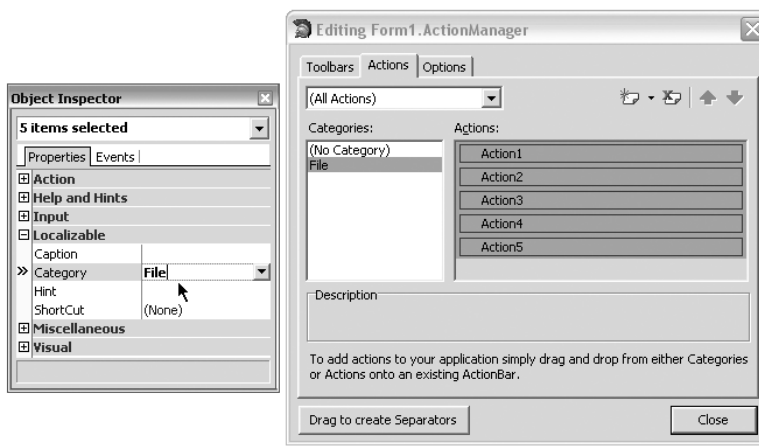


Рис. 10.8. Помещение действий в категорию

Чтобы выбрать эти пять действий, сначала выберите действие `Action1`, затем нажмите клавишу <Shift> и, удерживая ее, выберите действие `Action5`. Выбранные действия будут помещены в категорию `File` после того, как в свойстве `Category` вы введете `File` и нажмете клавишу <Enter>.

Определив категории действий в `TActionManager`, вы сможете без труда создать главное меню. Например, чтобы создать целое меню Файл, нужно будет выбрать категорию `File` в компоненте `TActionManager` и перетащить ее обычным образом в компонент `TActionMainMenuBar` на форме. Обратите внимание, что на рис. 10.9 показан компонент `TMainMenuBar`, в котором категория `File` уже была перенесена из компонента `TActionManager`.

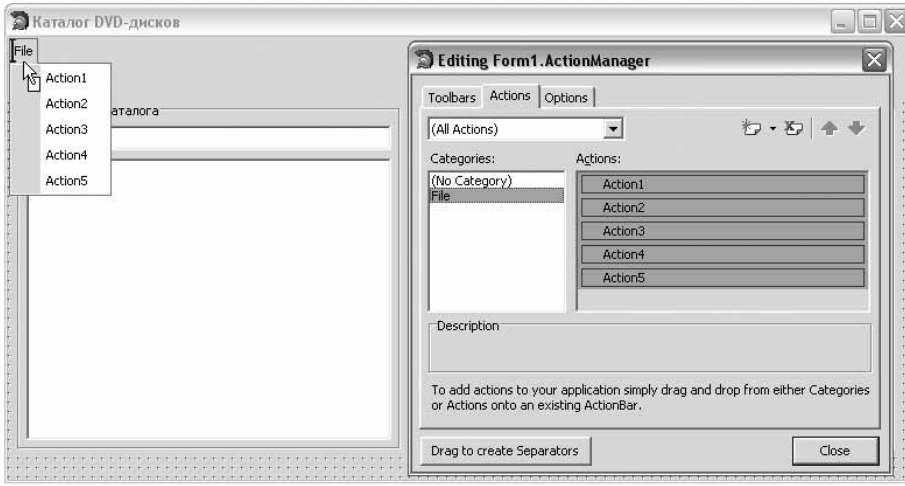


Рис. 10.9. Создание меню путем переноса категории из компонента `TActionManager` в компонент `TActionMainMenuBar`

Чтобы полностью реализовать эти действия, потребуется изменить их свойства `Caption` (Надпись), перенести на форму компоненты `TOpenDialog` и `TSaveDialog`, а также создать для каждого действия обработчик события `OnExecute` (рис. 10.10). Эти обработчики события `OnExecute` приведены в листинге 10.1.

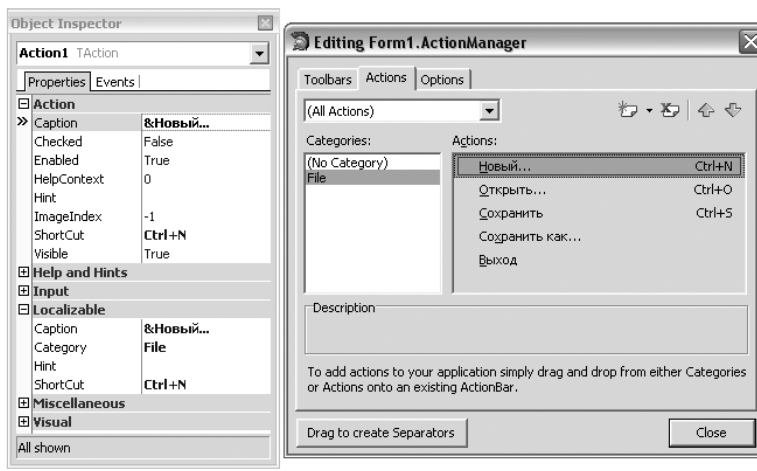


Рис. 10.10. Действия меню Файл

Поскольку мы собираемся работать только с XML-файлами, присвойте свойству `DefaultExt` (Расширение по умолчанию) обоих общих диалоговых окон значение `xml`, а в поле свойства `Filter` (Фильтр) введите `Базовая таблица XML (*.xml)|*.xml`.

Листинг 10.1. Обработчики события `OnExecute`

```
procedure TForm1.NewActionExecute(Sender: TObject);
begin
  if OpenDialog1.Execute then
    OpenCatalog(OpenDialog1.FileName);
end;

procedure TForm1.OpenActionExecute(Sender: TObject);
begin
  if OpenDialog1.Execute then
    OpenCatalog(OpenDialog1.FileName);
end;

procedure TForm1.SaveActionExecute(Sender: TObject);
begin
  if CDS.FileName <> '' then
    SaveCatalog(CDS.FileName)
  else
    SaveAsAction.Execute;
end;

procedure TForm1.SaveAsActionExecute(Sender: TObject);
begin
  if SaveDialog1.Execute then
    SaveCatalog(SaveDialog1.FileName);
end;

procedure TForm1.ExitActionExecute(Sender: TObject);
begin
  Close;
end;
```

Добавление записей

Чтобы добавить новые записи в компонент `TClientDataSet`, потребуется создать простое диалоговое окно, в котором пользователь сможет ввести идентификатор (ID) и название нового фильма (рис. 10.11).

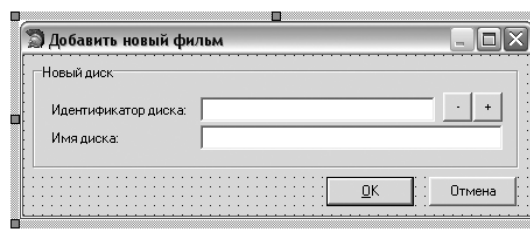


Рис. 10.11. Диалоговое окно Добавить новый фильм

Свойству `ModalResult` (Способ закрытия родительской формы) кнопки OK присваивается значение `mrNone`, поскольку сначала следует проверить, были ли введены название и идентификатор фильма. Только после этой проверки свойству `ModalResult`

возвращается значение mrOK, чтобы известить о том, что все идет как надо. Свойству ModalResult кнопки Отмена может быть автоматически присвоено значение mrCancel, которое уведомит о том, что пользователь решил не добавлять новые записи в компонент TClientDataSet.

В листинге 10.2 показан обработчик события OnClick кнопки OK, который проверяет, были ли введены необходимые данные, и обработчики события OnClick кнопок + и –.

Листинг 10.2. Код приложения диалогового окна Добавить новый фильм

```
procedure TAddMovieForm.MinusButtonClick(Sender: TObject);
var
  Number: Integer;
begin
  Number := StrToInt(MovieID.Text) - 1;
  if Number < 1 then
  begin
    Number := 1;
    MessageDlg('Идентификатор диска не может быть меньше 1!',
mtInformation, [mbOK], 0);
  end;
  MovieID.Text := IntToStr(Number);
end;

procedure TAddMovieForm.PlusButtonClick(Sender: TObject);
var
  Number: Integer;
begin
  Number := StrToInt(MovieID.Text) + 1;
  MovieID.Text := IntToStr(Number);
end;

procedure TAddMovieForm.OKButtonClick(Sender: TObject);
const
  ALLOW: array[Boolean] of TModalResult = (mrNone, mrOK);
begin
  { Разрешаем добавлять элемент в БД только в случае,
    если идентификатор и название фильма были введены правильно. }
  ModalResult := ALLOW[(MovieID.Text <> '') and (MovieName.Text <> '')];
  if ModalResult = mrNone then
    MessageDlg('Прежде чем щелкнуть на кнопке OK, введите идентификатор и
название диска.', mtInformation, [mbOK], 0);
end;
```

Теперь, когда имеется диалоговое окно, вы можете создать действие AddAction, которое отобразит диалоговое окно, где пользователь сможет ввести необходимые данные, проверит, щелкнул ли пользователь на кнопке OK в этом диалоговом окне, и вызовет метод AppendRecord компонента TClientDataSet, чтобы добавить данные пользователя в компонент TClientDataSet. После того как вы добавите действие AddAction, можете перенести его в компонент TActionToolBar, чтобы создать кнопку Добавить.

Ниже показан обработчик события OnExecute действия AddAction:

```
procedure TForm1.AddActionExecute(Sender: TObject);
begin
  with TAddMovieForm.Create(Self) do
  begin
    MovieID.Text := IntToStr(Succ(MovieList.Items.Count)); ShowModal;

    if ModalResult = mrOK then
    begin
      CDS.AppendRecord([StrToInt(MovieID.Text), MovieName.Text]);
FindEdit.Clear; { Удаляем фильтр, чтобы видеть все элементы. }

      { Обновляем окно списка. }
      DisplayDataset;
    end; // Конец условия if ModalResult.
    Free;
  end; // Конец блока with.
end;
```

Диалоговое окно Добавить новый фильм в действии показано на рис. 10.16.

Поиск записей

Чтобы выполнить поиск записей в компоненте `TClientDataSet`, можно воспользоваться методом `Locate` и передать ему имя искомого поля в первом параметре, значения, которые требуется найти, во втором параметре, а условия поиска — в третьем:

```
function TCustomClientDataSet.Locate(const KeyFields: string; const
KeyValues: Variant; Options: TLocateOptions): Boolean;
```

Тип `TLocateOptions` является множеством, которое позволяет выполнить поиск без учета регистра и поиск с указанием только части значения:

```
type
  TLocateOption = (loCaseInsensitive, loPartialKey);
  TLocateOptions = set of TLocateOption;
```

Чтобы активизировать соответствующую запись в компоненте `TClientDataSet` при выборе пользователем элемента в списке, мы должны вызвать метод `Locate` в обработчике события `OnClick` окна списка. Ниже показан обработчик события `OnClick` окна списка:

```
procedure TForm1.MovieListClick(Sender: TObject);
begin
  if MovieList.ItemIndex <> -1 then
    CDS.Locate('MovieName', MovieList.Items[MovieList.ItemIndex], []);
end;
```

Если сейчас запустить приложение и щелкнуть на элементе в окне списка, метод `Locate` отыщет и активизирует соответствующую запись, что даст возможность увидеть подробности выбранного элемента в компонентах, управляемыми данными, в правом групповом окне (рис. 10.12).

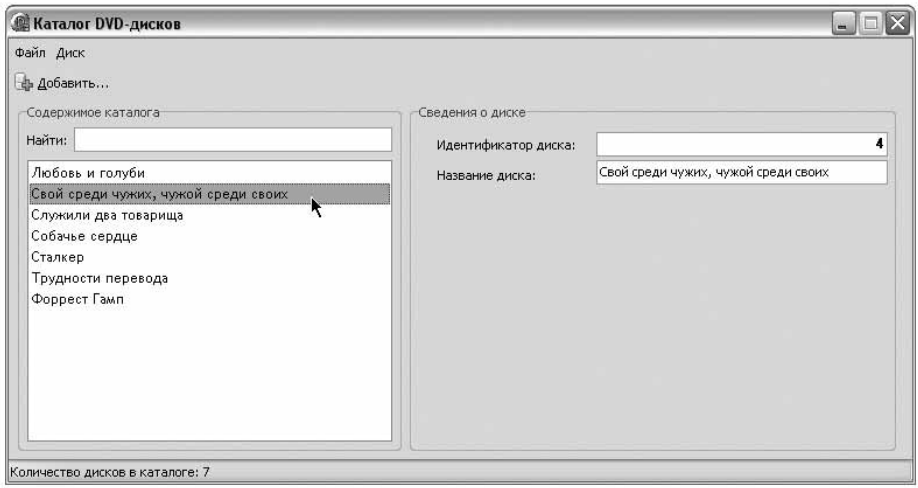


Рис. 10.12. Использование метода `Locate` для активизации выбранного элемента

Удаление записей

Для удаления активной записи потребуется создать действие `DeleteAction`. В нем будет вызываться метод `Delete` компонента `TClientDataSet` для удаления активной записи, но только при условии, что выбран допустимый элемент.

Ниже показан код обработчиков событий `OnExecute` и `OnUpdate` действия `DeleteAction`:

```
procedure TForm1.DeleteActionExecute(Sender: TObject);
const
  DELETE_MOVIE = 'Вы действительно хотите удалить диск:'#13;
begin
  if MessageDlg(DELETE_MOVIE +
    MovieList.Items[MovieList.ItemIndex] + '?', mtConfirmation,
    mbYesNo, 0) = mrYes then
  begin
    CDS.Delete;
    DisplayDataset;
  end; // Конец условия if MessageDlg.
end;

procedure TForm1.DeleteActionUpdate(Sender: TObject);
begin
  DeleteAction.Enabled := MovieList.ItemIndex <> -1;
end;
```

На рис. 10.13 показано действие `DeleteAction` во время выполнения.

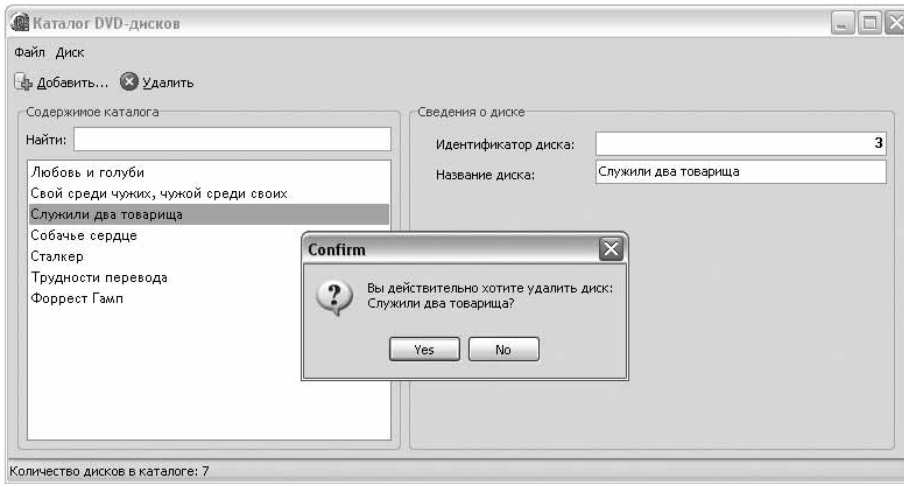


Рис. 10.13. Удаление записей

Фильтрация

Фильтрация записей компонента `TClientDataSet` возможна благодаря двум свойствам: `Filter` и `Filtered`. Если нужно отобразить только те записи, которые удовлетворяют определенному условию, то свойство `Filtered` (Фильтрация) должно иметь значение `True`, а в поле свойства `Filter` должна быть записана строка фильтра. Например, если вы хотите отобразить только те фильмы, название которых начинается с буквы `A`, то должны будете записать следующим образом:

```
MovieName = 'A'
```

Сейчас мы реализуем инкрементный фильтр, который будет фильтровать записи компонента `TClientDataSet` по мере их набора в текстовом окне Найти. Для этого мы должны воспользоваться методом `Substring` в строке `Filter`. Так, чтобы проверить, являются ли символы `ab` первыми символами в названии фильма, потребуется написать следующий фильтр:

```
Substring(MovieName, 1, 2) = 'ab'
```

В листинге 10.3 представлен код обработчика события `OnChange` текстового поля Найти.

Листинг 10.3. Инкрементный фильтр

```
procedure TForm1.FindEditChange(Sender: TObject);
const
  SUBSTRING = 'Substring(MovieName, 1, %d) = ''%s''';
var
  selectedMovie: string;
begin
  CDS.Filter := Format(SUBSTRING, [Length(FindEdit.Text), FindEdit.Text]);

  { Разрешаем фильтрацию, если в окне редактирования FindEdit имеется текст. }
  CDS.Filtered := FindEdit.Text <> '';
```

```
  { Вызываем метод DisplayDataSet, чтобы показать только
    отфильтрованные элементы. }
  DisplayDataSet;

  selectedMovie := CDS.FieldByName('MovieName').AsString;
  MovieList.ItemIndex := MovieList.Items.IndexOf(selectedMovie);
end;
```

Чтобы посмотреть, как работает фильтрация, взгляните на рис. 10.14.

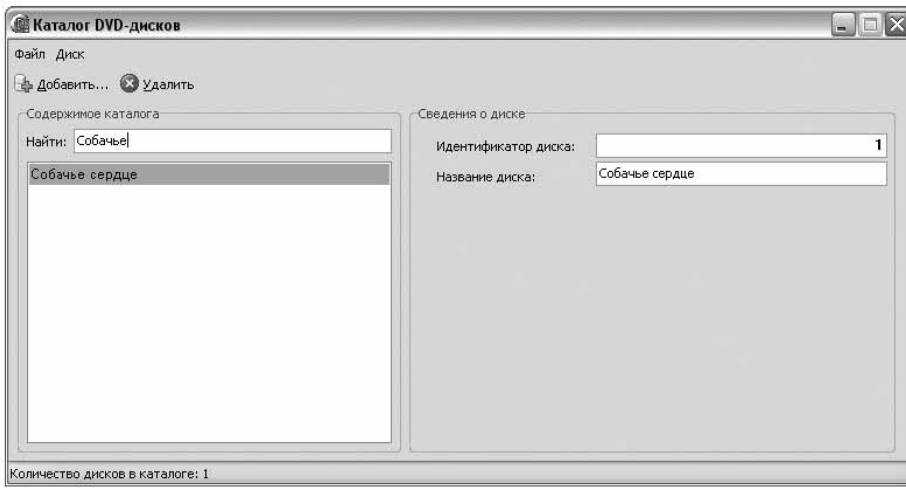


Рис. 10.14. Инкрементная фильтрация во время выполнения

Принятие и отклонение изменений

Компонент `TClientDataSet` хранит данные в двух отдельных пакетах, называемых `Data` и `Delta`. В пакете `Data` содержится текущее состояние данных, а в пакете `Delta` регистрируются изменения, произведенные в компоненте `TClientDataSet`, и хранятся вставленные, обновленные и удаленные записи. Пакет `Delta` очень полезен во время выполнения, поскольку с его помощью можно определить, сколько записей было изменено, и можно выбрать один из двух вариантов: принять или отклонить изменения, внесенные в данные. Однако пакет `Delta` вместе с обычными данными тоже записывается в файл, что приводит к нежелательному увеличению размера файла.

На рис. 10.15 показан файл `SampleDB.xml`, который имеет обычные данные и журнал изменений (XML-дескриптор `PARAMS` и атрибут `RowState`).

Чтобы приложение работало должным образом и дабы сократить размер файла на диске, мы должны вызвать метод `MergeChangeLog` до сохранения данных в методе `SaveCatalog`.

Ниже показан обновленный метод `SaveCatalog`:

```
procedure TForm1.SaveCatalog(const AFileName: string);
begin
  CDS.FileName := AFileName;
```

```
    { Вызываем метод MergeChangeLog, чтобы принять изменения
      в БД, сократить ее размер и добиться должного
      функционирования обработчика события OnCloseQuery. }
    CDS.MergeChangeLog;

    CDS.SaveToFile(CDS.FileName, dfXML);
end;
```

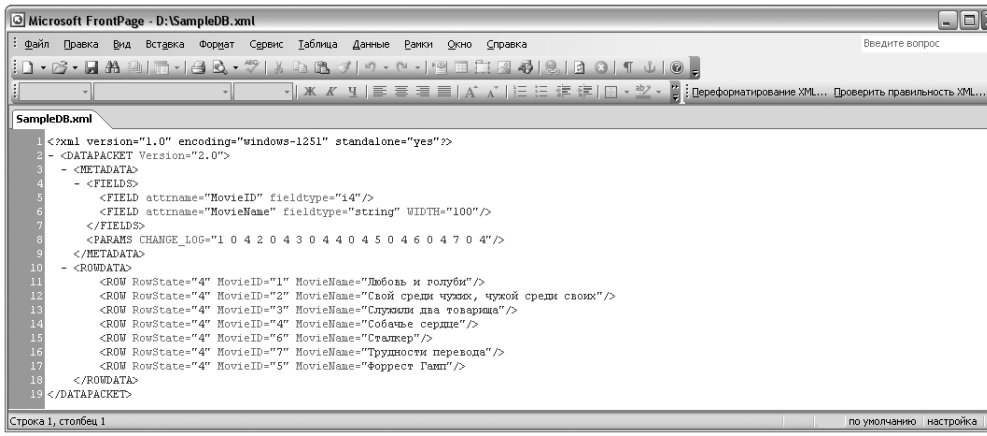


Рис. 10.15. XML-файл, содержащий обычные данные и журнал изменений

Последнее, что потребуется сделать — это написать обработчик события `OnCloseQuery`, который будет спрашивать пользователя о том, желает ли он сохранить или отклонить изменения в данных. Чтобы узнать, были ли изменены данные, проверьте свойство `ChangeCount` компонента `TClientDataSet`, которое хранит количество изменений в журнале изменений. Имейте в виду — для того чтобы это свойство безошибочно сообщало о количестве произведенных изменений, в файле, загруженном с диска, не должно быть журнала изменений (именно по этой причине следует объединять изменения перед сохранением данных на диске).

Если пользователь желает сохранить изменения в данных, вызовите обработчик события `OnExecute` действия `SaveAction`. Если пользователь желает отклонить изменения, вызовите метод `CancelUpdates` компонента `TClientDataSet`.

Ниже представлен полный код обработчика события `OnCloseQuery`:

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
begin
  if CDS.ChangeCount > 0 then
  begin
    case MessageDlg('Сохранить изменения в базе данных?',
       mtConfirmation, mbYesNoCancel, 0) of
       mrYes: SaveAction.Execute;
       mrNo: CDS.CancelUpdates;
       mrCancel: CanClose := False;
    end; // Конец блока case.
  end; // Конец условия if.
end;
```

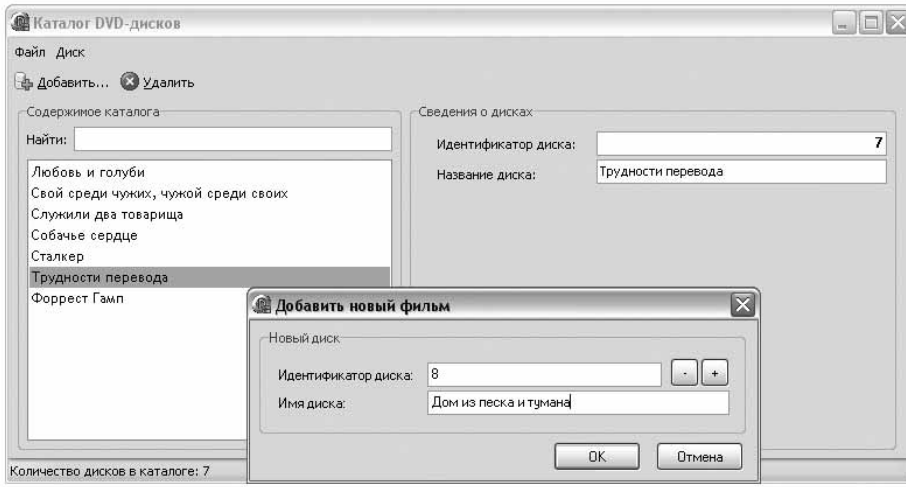


Рис. 10.16. Приложение “Каталог DVD-дисков” в работе