Томский государственный университет

Факультет информатики

А.В. Скворцов, Т.Н. Поддубная

Создание базы данных

(методические указания к

лабораторной работе № 8)

Томск – 1999

Указания РАССМОТРЕНЫ и УТВЕРЖДЕНЫ методической комиссией факультета информатики.

Протокол №____ от «____»_____1999 г.

Председатель методической комиссии факультета информатики,

профессор _____ Поддубный В.В.

Утверждаю. Зав. кафедрой прикладной информатики, доцент

_____ Сущенко С.П. «____» _____ 1999 г.

Методические указания посвящены программной среде Borland Delphi компании Inprise Corporation – одному из мощнейших современных средств разработки приложений для Windows.

Указания разработаны для студентов межфакультетской специализации факультета информатики Томского государственного университета. Содержат подробное описание лабораторной работы: цель, задачи, описание работы и используемых компонентов Delphi. В конце работы приведены тексты программ.

Резенцент – канд. техн. наук, доцент **Ю.Л. Костюк**.

Данная лабораторная работа предназначена для знакомства с основами работы с реляционными базами данных. Условно можно считать, что реляционная база данных состоит из набора таблиц, которые в свою очередь состоят из однотипных записей с несколькими полями. Обычно записи в таблицах представляют в виде горизонтальных рядов, а в колонках – значения соответствующих полей.

Базы данных бывают серверного типа, когда они хранятся в специальном формате, и обращение к ним производится только через специальные программы-серверы баз данных, а также бывают локальные базы. В локальных базах обычно отдельные таблицы хранятся в виде одного или нескольких файлов, при этом базой данных считается каталог на диске, содержащий все эти файлы. В нашей работе мы будем работать с локальной базой данных в формате Paradox 7.

В данной работе будет необходимо создать простое приложение для ведения учета книг в домашней библиотеке. Для каждой книги в библиотеке должна быть создана запись в базе данных со следующими полями: фамилия автора, название книги, ее тип и фамилия того, кто взял книгу. Тип книг и список читателей необходимо оформить в виде таблиц-справочников, которые можно будет наполнять по мере необходимости. Просмотр списка книг должен производиться в одном из трех режимов: просмотреть все книги, книги заданного типа, либо книги, находящиеся у выбранного читателя.

В рамках работы необходимо будет освоить программу Database Desktop, предназначенную для создания и редактирования таблиц баз данных. Используя эту программу, нужно будет создать 3 таблицы, которые затем будут подключены к нашей программе в среде Delphi.

Цель работы

Изучение средств Delphi для работы с базами данных.

Задачи работы

1. Ознакомление с программой Database Desktop и создание в ней 3 таблиц базы данных домашней библиотеки.
2. Краткое знакомство с компонентами для работы с базами данных, находящихся на закладках «Database Access» и «Database Controls».
3. Создание приложения «Домашняя библиотека» для ведения учета книг в домашней библиотеке.

Описание работы

Разработка любой базы данных начинается с проектирования её логической и физической структур. Физическая схема нашей базы изображена на рис. 24. На ней слева представлена главная таблица, имеющая 4 поля: имя автора, название книги, код типа книги и код читателя. Справа представлены две таблицы-справочника, в которых хранится соответствие кода типа книги с текстовым описанием типа и кода читателя с его текстовым именем. Для быстрого поиска и сортировки содержимого таблиц они должны быть проиндексированы по полям, отмеченным звездочками. В первой таблице по полям Author и Name должен быть создан первичный индекс, а по полям Type и User два вторичных. В таблицах-справочниках индекс должен быть создан по полям ID. Он будет использоваться для быстрого поиска текстового описания по коду.

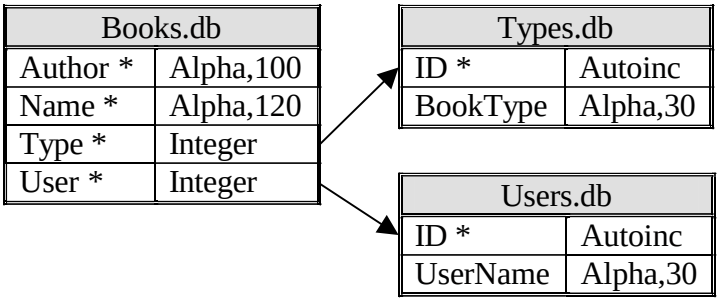

Рис. 24. Физическая схема БД домашней библиотеки

В таблицах-справочниках для полей ID необходимо указать тип данных Autoincrement, что означает, что при добавлении новой записи в таблицу для значения этого поля будет автоматически браться уникальное целочисленное значение. Тип данных Alpha означает текстовую строку. Этот тип имеет целочисленный параметр, означающий максимальную длину строки, которую можно поместить в это поле.

Для физического создания таблиц нашей базы данных можно воспользоваться программой Database Desktop, входящей в состав Delphi. Её можно запустить из Delphi с помощью команды меню Tools|Database Desktop. После ее запуска необходимо выбрать команду меню File|Working Directory… и затем в диалоге указать каталог, в котором мы будем сохранять таблицы. Пусть это будет тот же каталог, в котором мы сохраняем файлы проекта Delphi.

Затем с помощью команды меню File|New|Table… необходимо создать 3 таблицы, указав список полей таблиц, их типы, а также создать индексы. При выборе этой команды, в первую очередь, в диалоговом окне (рис. 25) необходимо выбрать тип таблицы. Для нашей задачи лучше всего выбрать тип «Paradox 7», т.к. он обладает макси-

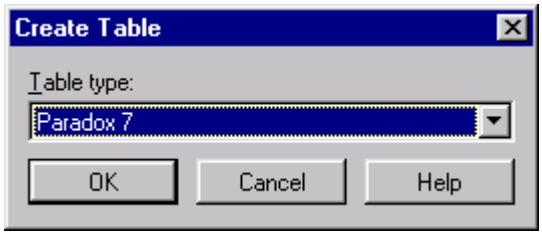

Рис. 25. Выбор формата новой таблицы в Database Desktop

мальными возможностями среди других доступных форматов не серверного типа.

После выбора типа таблицы откроется окно создания таблицы формата «Paradox 7», в котором в первую очередь необходимо сформировать список всех полей таблицы. На рис. 26 приведен внешний вид программы Database Desktop в процессе создания новой таблицы, которую мы потом должны будет сохранить под именем Books.db. В первой колонке списка полей указывается имя поля, во второй – его тип. Для выбора нужного типа можно либо нажать клавишу с первой буквой нужного типа, либо нажать пробел и на экране появится полный список всех доступных типов. В нашей работе нам понадобятся только типы Alpha для представления строковых значений, Long Integer для представления целочисленных кодов типа книги и кода читателя, а также тип + (Autoincrement).

Третья колонка списка полей предназначена для задания размера поля. В нашем случае она имеет смысл только для строковых полей. Размер поля определяет максимальную длину строки, которую можно поместить в

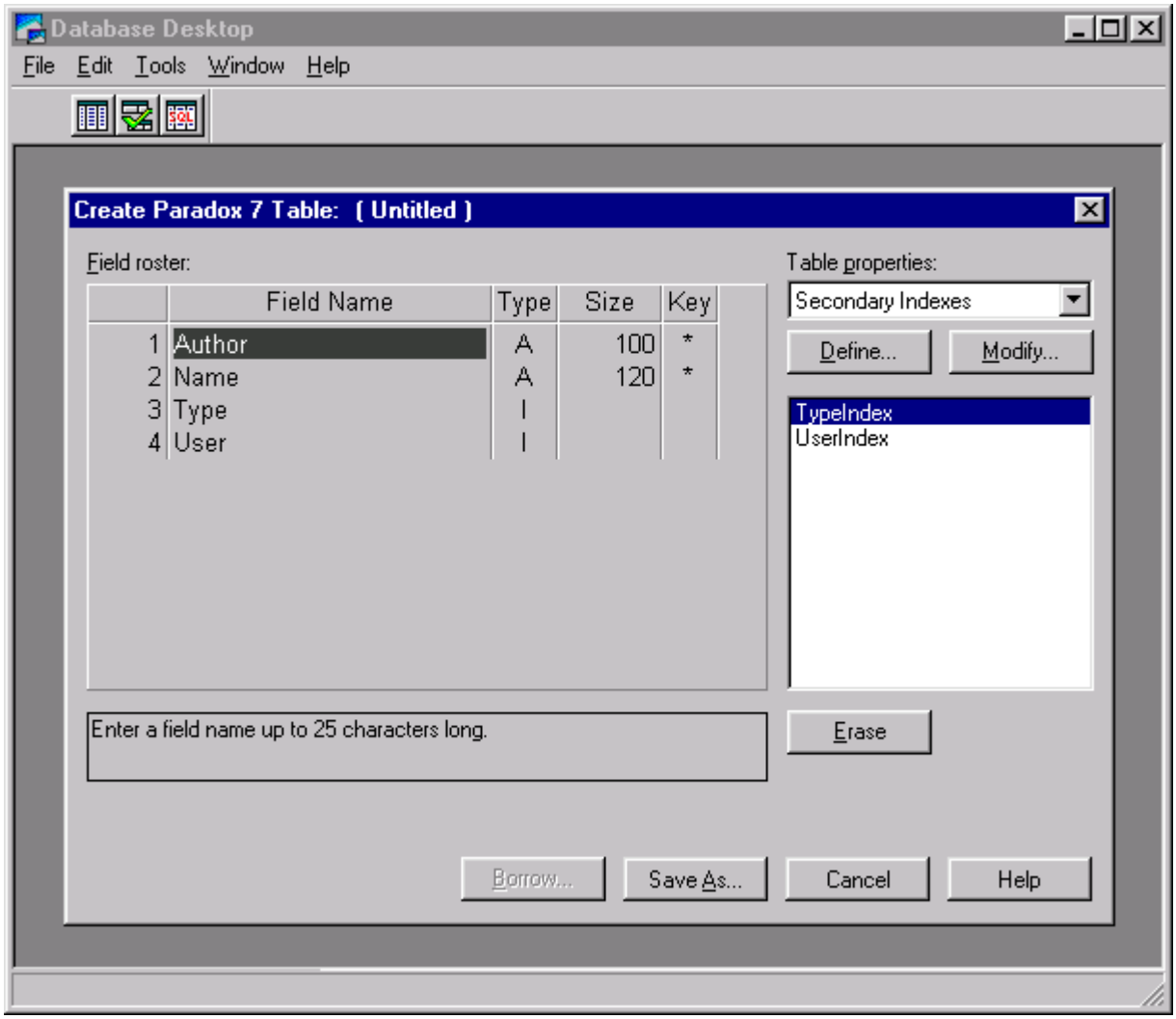

Рис. 26. Создание новой таблицы в Database Desktop

данное поле.

Четвертая колонка позволяет указать поля, которые будут входить в первичный индекс. Для его задания необходимо просто нажать пробел. Индекс (ключ) используется при отображении данных в таблице в качестве критерия сортировки записей.

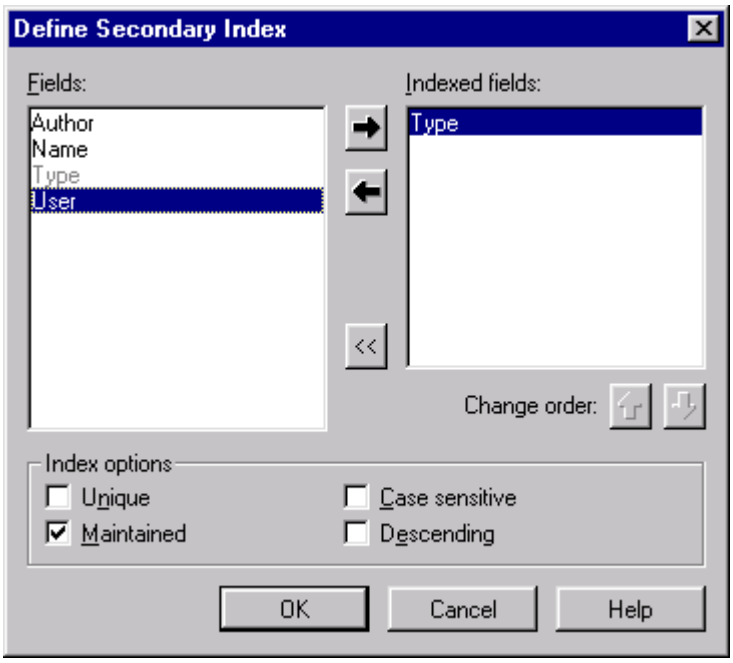

Рис. 27. Создание вторичного индекса в Database Desktop

В нашей задаче для таблицы Books.db понадобятся также еще 2 вторичных индекса, в которые будет входить только одно поле, – Type и User соответственно. Для этого в правой верхней части окна создания таблицы необходимо выбрать в выпадающем списке пункт Secondary Indexes. Затем необходимо нажать на появившуюся кнопку Define, после чего на экране появится диалог создания вторичного индекса (рис. 27).

В диалоге создания вторичного индекса необходимо выбрать в левом списке имя поля, по которому необходимо создать индекс, и нажать кнопку ⇒. При этом в правом списке отображается список полей, которые будут участвовать во вторичном индексе. В нашем случае необходимо выбрать для индекса поле Type и нажать клавишу OK. При этом появится окно, в котором необходимо указать имя созданного индекса. Обычно для имени индекса указывают имя поля, составляющего индекс, с добавлением слова Index. В нашем случае укажем имя TypeIndex.

После этого необходимо создать ещё один вторичный индекс, но уже по полю User. Аналогично сохраним этот индекс под именем UserIndex.

В дальнейшем для изменения структуры уже созданной таблицы в программе Database Desktop необходимо будет открыть её с помощью команды меню File|Open|Table, а затем выбрать команду меню Table|Restructure…

После физического создания таблиц базы данных можно переходить к созданию пользовательского интерфейса к базе данных в среде Delphi.

В Delphi для подключения к таблице базы данных используется невизуальный компонент типа `TTable`, находящийся на закладке «Database

Access» палитры компонентов. В простейшем случае для обращения к реальной таблице необходимо указать только 3 свойства, приведенные в табл. 22.

Таблица 22. Основные свойства компонентов типа `TTable`

Свойство	Тип	Комментарий
`DatabaseName`	`String`	Имя базы данных, указанной в компоненте `TDatabase`, зарегистрированном на компьютере в программе BDE Administrator или каталог, содержащий таблицы форматов DBase, FoxPro или Paradox
`TableName`	`String`	Имя таблицы указанной базы данных
`Active`	`Boolean`	Открыта ли таблица. Для открытия можно также воспользоваться методом `Open`, а для закрытия – `Close`

Для удобства разработки компоненты баз данных обычно размещают в отдельных модулях (чтобы компоненты для доступа к базам не путались с остальными): это может быть отдельная пустая невидимая форма или специальный модуль данных (Data Module), который можно создать с помощью пункта меню File| New… В этом модуле необходимо разместить компоненты типа `TTable`, соответствующие таблицам нашей базы данных.

Для каждого компонента `TTable` определен так называемый «курсор», определяющий текущую запись в таблице. Отметим особую роль таблиц-справочников в нашем приложении. Их необходимо будет редактировать при появлении новых читателей и типов книг, а также независимо перемещаться по ним для подмены кодов полей их строковыми значениями. Поэтому для этих таблиц необходимо будет поме-

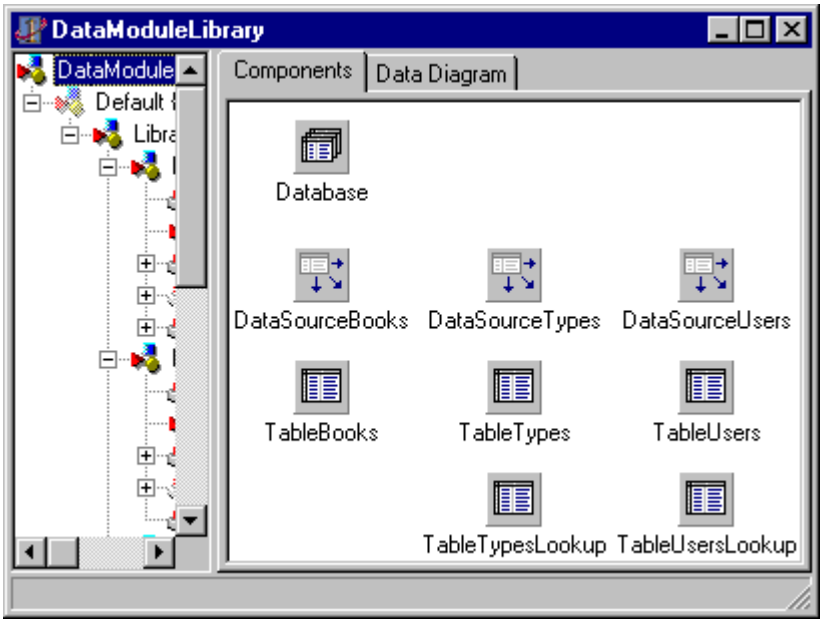

Рис. 28. Состав модуля данных домашней библиотеки

стить в модуль данных по два компонента TTable, каждый из которых имеет независимые курсоры. Состав модуля данных приведен на рис. 28.

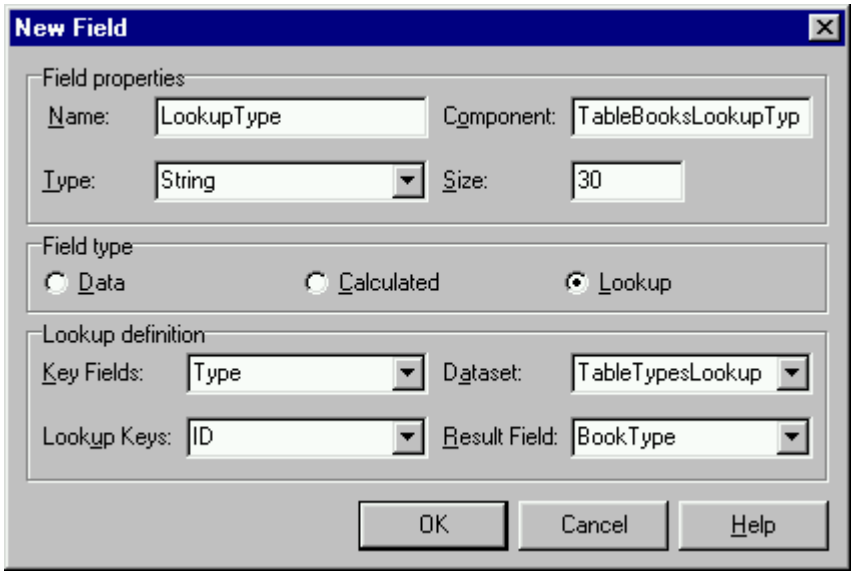

Рис. 29. Создание подстановочного поля

Список полей таблицы представляется в свойстве таблицы Fields. Для его изменения необходимо вызвать редактор списка полей таблицы, который появляется при двойном щелчке мышкой на компоненте TTable. По умолчании список полей пуст, что означает, что он будет формироваться автоматически при открытии таблицы базы данных. Если же мы хотим каким-то образом переопределить свойства полей, то необходимо в локальном меню редактора списка полей вызвать вначале команду Add all fields для получения полного списка полей. Для создания в главной таблице подстановочных полей необходимо выбрать команду локального меню New field…

В главной таблице TableBooks для подстановки текстовых значений вместо кодов типов книг и кодов читателей будем использовать компоненты с именами TableTypesLookup и TableUsersLookup. Поэтому в главной таблице кроме реальных физических полей необходимо создать два дополнительных подстановочных (lookup) поля строкового типа, поля же с кодами необходимо скрыть от пользователя.

На рис. 29 приведен пример создания подстановочного поля типа книги LookupType, подменяющего числовой код типа книги его текстовым описанием с помощью другой таблицы-справочника TableTypesLookup.

Создаваемые поля компонента TTable являются потомками класса TField. Любое поле в списке можно выбрать, и затем изменить его настройки с помощью инспектора объектов. Список основных свойств объектов типа TField приведен в табл. 23.

Таблица 23. Основные свойства компонентов типа `TField`

Свойство	Тип	Комментарий
`Alignment`	`TAlignment`	Выравнивание текста поля при просмотре: по левой стороне, по правой или по центру
`DisplayLabel`	`String`	Текст метки поля, используемый, например, в качестве заголовка столбца таблицы
`DisplayWidth`	`Integer`	Ширина колонки таблицы по умолчанию в символах
`FieldKind`	`TFieldKind`	Определяет, является ли поле отражением реальных данных в физической таблице (`fkData`), является ли поле подстановочным (`fkLookup`), вычисляемым (`fkCalculated`) и т.д.
`FieldName`	`String`	Физическое имя поля для типа `fkData`, либо логическое для других типов
`ReadOnly`	`Boolean`	Разрешено ли изменение поля пользователем
`Required`	`Boolean`	Обязательно ли заполнять значение поля при модификации записи
`Visible`	`Boolean`	Видно ли поле в таблице по умолчании
`LookupDataset`	`TDataset`	Указывает на набор данных (например на таблицу), являющийся справочником
`KeyFields`	`String`	Имя поля таблицы, где хранится код, который надо подменить по справочнику
`LookupKeyFields`	`String`	Имя поля справочника с кодом
`LookupResultField`	`String`	Имя поля справочника, возвращаемого как результат подмены кода

После того как таблицы открыты, к ним можно подключать визуальные компоненты для отображения содержимого и перемещения по записям. Для этого в Delphi используется двухступенчатая схема, по которой к таблицам присоединяется один компонент типа `TDataSource`, а к нему, в свою очередь, все необходимые визуальные компоненты. В Delphi, как правило, имена типов визуальных компонентов для работы с базами данных начинаются с букв `TDB`, например `TDBGrid` для отображения всей таблицы в виде сетки, `TDBNavigator` для навигации по таблице и т.д. Условно иерархия связывания визуальных и невизуальных компонентов для работы с базами данных приведена на рис. 30. В нашем приложении, таким образом, необходимо для всех отображаемых таблиц (кроме тех, которые используются в подстановочных полях) создать компоненты типа `TDataSource` и связать их с таблицами с помощью их свойства `Dataset`.

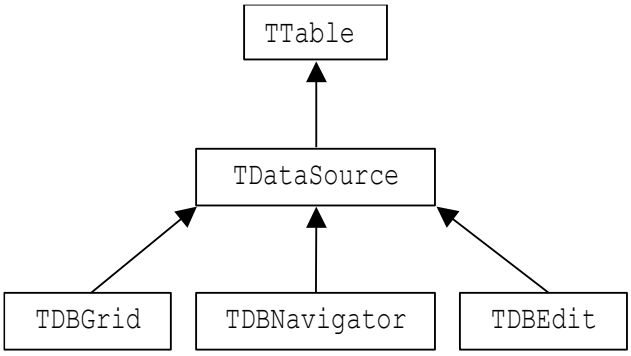

Рис. 30. Связывание компонент для работы с базами данных

Замечание

Для того чтобы в инспекторе объектов в выпадающих списках свойств были доступны компоненты, находящиеся в других модулях, необходимо его указать с помощью команды меню File|Use Unit…

В главной форме нашего приложения необходимо разместить компоненты с соответствии с рис. 31. В правой части надо разместить компонент `TDBGrid` для отображения таблицы в виде сетки. С его помощью можно редактировать содержимое таблицы, добавлять новые записи с помощью клавиши Insert, а также удалять записи с помощью комбинации клавиш Ctrl+Delete. В верхней части формы необходимо разместить компонент `TDBNavigator`, помогающий перемещаться по таблице, редактировать ее, вставлять и удалять записи. Обоим этим компонентам надо указать источник данных – свойство `DataSource` – с помощью выпадающего списка в инспекторе объектов. Но т.к. наши источники данных находятся в другом модуле, то перед этим необходимо указать, что данный модуль (главная форма) использует другой (модуль данных). Это делается в Delphi с помощью команды меню File|Use Unit…, которая по сути добавляет в секцию **`uses`** раздела **`implementation`** выбранный модуль проекта.

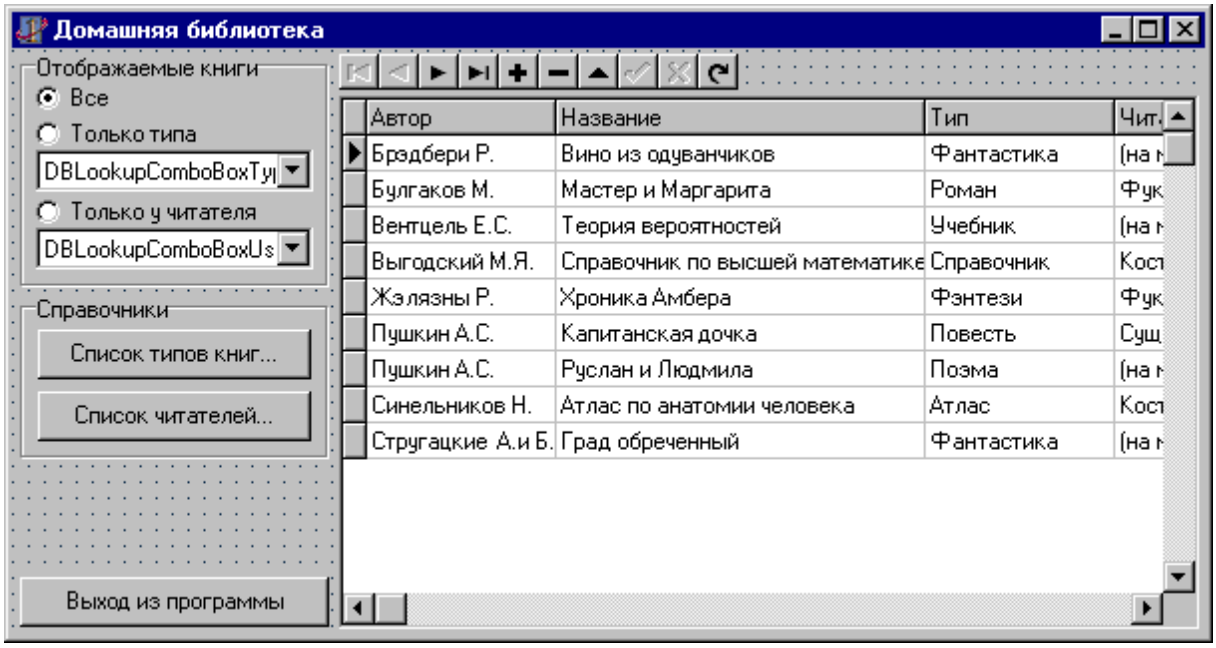

Рис. 31. Внешний вид главной формы приложения

В левой части главной формы необходимо разместить кнопку выхода из программы и две кнопки для редактирования справочников, которые просто должны выводить на экран две другие формы простого содержания, как на рис. 32-33. В этих формах также необходимо указать, что надо использовать модуль данных Data, и затем просто выставить свойства `DataSource`. Обратите внимание, что в этих формах видно только одно строковое поле, а поле ID скрыто. Просто у этого поля в редакторе списка полей свойство `Visible` установлено равным `False`.

В левой верхней части формы необходимо разместить компоненты, позволяющие фильтровать записи в таблице в соответствии с некоторым критерием. Нам необходимо сделать три варианта: все книги, книги задан-

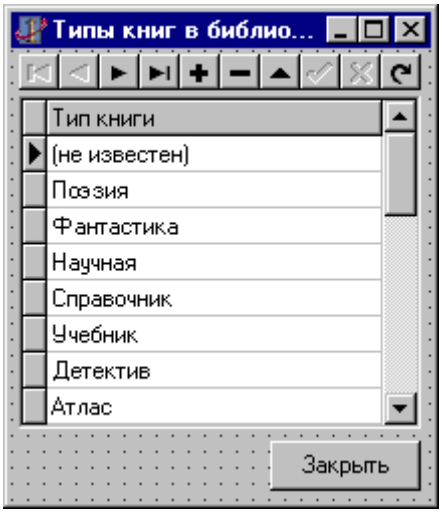

Рис. 32. Форма для редактирования справочника типов книг

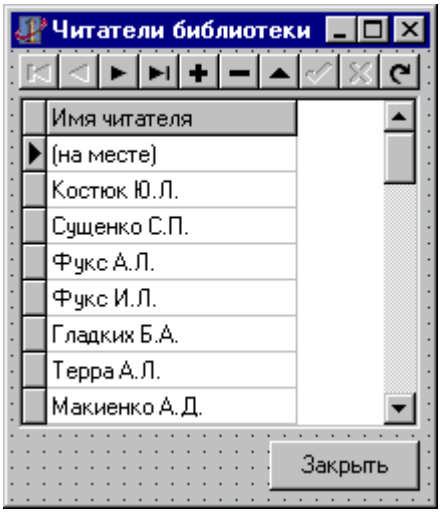

Рис. 33. Форма для редактирования списка читателей

ного типа и книги, находящиеся у заданного пользователя. Для извлечения данных из таблиц в соответствии с некоторым критерием в Delphi возможны несколько вариантов.

Один из вариантов заключается в задании выражения фильтра на записи в свойствах `Filter` и `Filtered` компонента `TTable`. Например, выражение фильтра «`Type<>1 and Author='Стругац*'`» для главной таблицы нашего приложения позволяет выбрать только те книги, код которых отличается от 1 и имя автора начинается с букв `'Стругац'`.

Другой вариант заключается в создании связки таблиц «главная-подчиненная» (master-detail) по некоторым полям. Тогда при выборе записи в главной таблице в подчиненной таблице будут отображаться только те, код связи которых совпадает с текущей записью в главной. Эта связка аналогична той, что используется в подстановочных полях: там главной таблицей являлась `TableBooks`, а подчиненными – справочники `TableTypes` и `TableUsers`.

В данном случае необходимо, чтобы, наоборот, таблицы `TableTypes` и `TableUsers` были главными, а `TableBooks` – подчиненной. При выборе типа книги или пользователя в таблице книг должны выбираться только определенные записи. Для организации такой связи в подчиненной таблице необходимо указать три свойства, краткое описание которых приведено в табл. 24.

Таблица 24. Свойства компонентов типа `TTable` для организации связи с главной таблицей в режиме подчиненной

Свойство	Тип	Комментарий
`MasterSource`	`TDataSource`	Источник данных с главной таблицей
`IndexFieldNames`	`String`	Имя индексированного поля подчиненной таблицы, по которому осуществляется связь
`MasterFields`	`String`	Имя поля главной таблицы, текущее значение которого используется для фильтрации записей в подчиненной

Итак, вернемся к нашему приложению. В левой верхней части главной формы следует разместить три радио-кнопки, при выборе которых необходимо динамически менять свойства `MasterSource`, `IndexFieldNames` и `MasterFields` компонента `TableBooks`. Для выбора типа книг или пользователя для фильтрации необходимо разместить около радио-кнопок компоненты типа `TDBLookupComboBox`, при раскрытии которых в ниспадающем списке будут отображаться записи таблиц `TableTypes` и `TableUsers`. Для

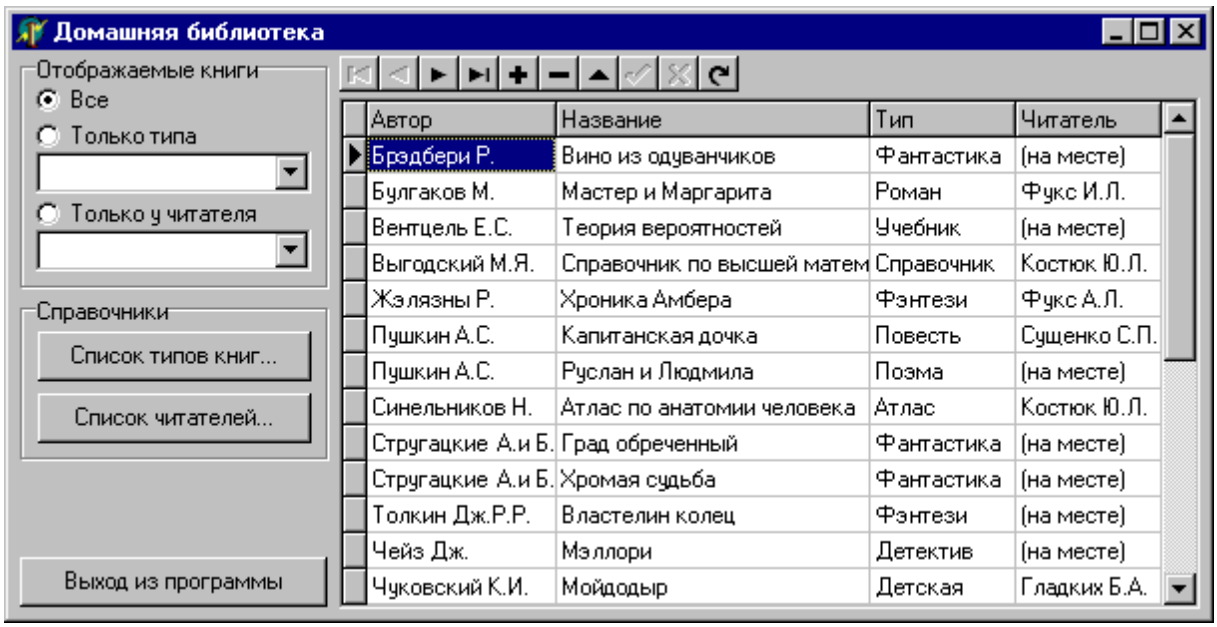

Рис. 34. Внешний вид запущенного приложения

этого нужно у компонентов `TDBLookupComboBox` установить свойства `ListSource` в значение `DataSourceType` (или `DataSourceUsers`); `ListField` равным «BookType» (или «UserName») и свойство `KeyField` равным «ID».

На рис. 34 приведен внешний вид запущенного приложения. В листингах 15-19 приведен текст всех файлов проектов.

Листинг 15. Текст файла проекта Lab8.dpr

```
program Lab8;

uses
  Forms,
  HomeLib in 'HomeLib.pas' {LibraryForm},
  Data in 'Data.pas' {DataModuleLibrary: TDataModule},
  Types in 'Types.pas' {FormBookTypes},
  Users in 'Users.pas' {FormUsers};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TLibraryForm, LibraryForm);
  Application.CreateForm(TDataModuleLibrary, DataModuleLibrary);
  Application.CreateForm(TFormBookTypes, FormBookTypes);
  Application.CreateForm(TFormUsers, FormUsers);
  Application.Run;
end.
```

Листинг 16. Текст модуля главной формы HomeLib.pas

```
unit HomeLib;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, DBCtrls, Grids, DBGrids;

type
  TLibraryForm = class(TForm)
    GroupBoxView: TGroupBox;
    RadioButtonAll: TRadioButton;
    RadioButtonType: TRadioButton;
    DBLookupComboBoxType: TDBLookupComboBox;
    RadioButtonUser: TRadioButton;
    DBLookupComboBoxUser: TDBLookupComboBox;
    GroupBoxDictionaries: TGroupBox;
    ButtonTypes: TButton;
    ButtonUsers: TButton;
    ButtonExit: TButton;
    DBGrid: TDBGrid;
    DBNavigator: TDBNavigator;
    procedure ButtonTypesClick(Sender: TObject);
    procedure ButtonUsersClick(Sender: TObject);
    procedure ButtonExitClick(Sender: TObject);
    procedure RadioButtonAllClick(Sender: TObject);
    procedure RadioButtonTypeClick(Sender: TObject);
    procedure RadioButtonUserClick(Sender: TObject);
    procedure DBLookupComboBoxTypeClick(Sender: TObject);
    procedure DBLookupComboBoxUserClick(Sender: TObject);
  end;

var
  LibraryForm: TLibraryForm;

implementation

uses
```

```
  Data, Types, Users;

{$R *.DFM}

procedure TLibraryForm.ButtonTypesClick(Sender: TObject);
begin // Вывести форму со справочником типов книг
  FormBookTypes.Show;
end;

procedure TLibraryForm.ButtonUsersClick(Sender: TObject);
begin // Вывести форму со списком читателей
  FormUsers.Show;
end;

procedure TLibraryForm.ButtonExitClick(Sender: TObject);
begin // Выход из программы
  Close;
end;

procedure TLibraryForm.RadioButtonAllClick(Sender: TObject);
begin // Просматривать все книги (не фильтровать)
  DataModuleLibrary.TableBooks.MasterSource:=nil;
end;

procedure TLibraryForm.RadioButtonTypeClick(Sender: TObject);
begin // Отфильтровать книги по типу книг
  with DataModuleLibrary do
  begin
    TableBooks.MasterSource:=DataSourceTypes;
    TableBooks.IndexFieldNames:='Type';
    TableBooks.MasterFields:='ID';
  end;
end;

procedure TLibraryForm.RadioButtonUserClick(Sender: TObject);
begin // Отфильтровать книги по читателям
  with DataModuleLibrary do
  begin
    TableBooks.MasterSource:=DataSourceUsers;
    TableBooks.IndexFieldNames:='User';
    TableBooks.MasterFields:='ID';
  end;
end;

procedure TLibraryForm.DBLookupComboBoxTypeClick(Sender: TObject);
begin // Если выбран новый тип книг, то установить режим фильтра по типам
  RadioButtonType.Checked:=True;
end;

procedure TLibraryForm.DBLookupComboBoxUserClick(Sender: TObject);
begin // Если выбран новый читатель, то установить режим фильтра по читателям
  RadioButtonUser.Checked:=True;
end;

end.
```

Листинг 17. Текст модуля данных Data.pas

```
unit Data;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, Db, DBTables;

type
  TDataModuleLibrary = class(TDataModule)
    Database: TDatabase;
    DataSourceBooks: TDataSource;
    TableBooks: TTable;
    TableTypes: TTable;
    DataSourceTypes: TDataSource;
    TableUsers: TTable;
    DataSourceUsers: TDataSource;
    TableBooksAuthor: TStringField;
    TableBooksName: TStringField;
    TableBooksType: TIntegerField;
    TableBooksUser: TIntegerField;
    TableBooksLookupType: TStringField;
    TableBooksLookupUser: TStringField;
    TableTypesID: TAutoIncField;
    TableTypesBookType: TStringField;
    TableUsersID: TAutoIncField;
    TableUsersUserName: TStringField;
    TableTypesLookup: TTable;
    AutoIncField1: TAutoIncField;
    StringField1: TStringField;
    TableUsersLookup: TTable;
    AutoIncField2: TAutoIncField;
    StringField2: TStringField;
    procedure DataModuleCreate(Sender: TObject);
    procedure TableBooksNewRecord(DataSet: TDataSet);
  end;

var
  DataModuleLibrary: TDataModuleLibrary;

implementation

{$R *.DFM}

procedure TDataModuleLibrary.DataModuleCreate(Sender: TObject);
begin // При запуске программы указать, что база данных находится
  Database.Connected:=False; // в том же каталоге, что и exe-файл
  Database.Params.Values['Path']:=ExtractFilePath(paramstr(0));
  TableBooks.Active:=True;
  TableTypes.Active:=True;
  TableUsers.Active:=True;
  TableTypesLookup.Active:=True;
  TableUsersLookup.Active:=True;
end;

procedure TDataModuleLibrary.TableBooksNewRecord(DataSet: TDataSet);
begin // Указать значения полей при создании новой записи
  TableBooksType.AsInteger:=1; // Тип книги = 1 (не известен)
  TableBooksUser.AsInteger:=1; // Читатель = 1 (на месте)
end;

end.
```

Листинг 18. Текст модуля со справочником типов книг Types.pas

```
unit Types;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, DBCtrls, Grids, DBGrids;

type
  TFormBookTypes = class(TForm)
    DBNavigator: TDBNavigator;
    DBGrid: TDBGrid;
    ButtonClose: TButton;
    procedure ButtonCloseClick(Sender: TObject);
  end;

var
  FormBookTypes: TFormBookTypes;

implementation

uses Data;

{$R *.DFM}

procedure TFormBookTypes.ButtonCloseClick(Sender: TObject);
begin // Закрыть форму
  Close;
end;

end.
```

Листинг 19. Текст модуля со списком пользователей Users.pas

```
unit Users;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, DBCtrls, Grids, DBGrids;

type
  TFormUsers = class(TForm)
    DBNavigator: TDBNavigator;
    DBGrid: TDBGrid;
    ButtonClose: TButton;
    procedure ButtonCloseClick(Sender: TObject);
  end;

var
  FormUsers: TFormUsers;

implementation

uses
  Data;

{$R *.DFM}
```

```
procedure TFormUsers.ButtonCloseClick(Sender: TObject);
begin // Закрыть форму
  Close;
end;

end.
```

Вопросы и задания для самостоятельной работы

1. Как скрыть поле таблицы от пользователя?
2. Какие бывают типы полей в таблицах? Что означает автоинкрементный тип? Как создаются справочные поля?
3. Как отобразить содержимое таблицы в форме? Какие компоненты нужны для этого?
4. Что такое «курсор» таблицы? Для чего используются в нашей задаче два компонента `TTable`, связанных с одной физической таблицей? Как устанавливается связь «главный-подчинённый»?
5. Для чего компоненты для работы с базами данными вынесены в отдельный модуль? Можно ли их разместить на главной форме?
6. Как назначить свойство (например `DataSource` у `TDBGrid`) указывающим на компонент, находящийся в другой форме?
7. Для чего используются индексы (ключи) в таблицах?
8. Как создать таблицу в Database Desktop?

Учебное издание

Алексей Владимирович Скворцов
Поддубная Тамара Николаевна

Создание базы данных

(методические указания к
лабораторной работе № 8)

Томский государственный университет. Томск, 1999. – 20 с.