

NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

**DATA-BANK PROGRAMMING FOR LIBRARY
SYSTEM**

**Graduation Project
COM- 400**

Student: Halime Yilmaz

Supervisor: Mr.Ümit Soyer

Nicosia-2008

ACKNOWLEDGMENT

"First of all I would like to thank my supervisor Mr. Umit Soyer for his way of treatment with me, he never rejected me when I had any question and also we contacted otherside of the university.

I don't forget other instructors, Mr. Elbrus Imanov was also helped me whenever I had any problem or he gave me any ideas. I thank them very much with my all for their advices.

Secondly I would like to thank my family for their molar tooth of graduated me from here. I know that because of their selfless I was here and I never want to make them disappointed. So we trust each other and I will be graduated.

Thirdly I want to thanks my friend Cüneyt Şeker, he helped me in my project and gave me ideas and parted her time with me.

Finally I would like to thank All my friends who aid me during my university life, to resist and study."

ABSTRACT

Computer science has developed tremendously over the last decades. It is possible to state this in terms of both hardware and software. Programming is always providing the scientists a continuous systematic development in their studies and research. In this project it's been constructed a special program related to Library Automation .The library management should not be regarded as an isolated and unrelated field from the other industries but it is within this framework that the history of library development should be examined .New concepts in library design have been developed more recently in an effort to meet the changing preferences and new characteristics.

The library automation program consists of many departments like give-return book, search book, update book-user, search book-user, search transactions and report transactions. The program that been given in this thesis, resumes that the briefly in a quick time in order to have quick and economic services. On the other hand, the library development is suitable for researches and students in computer science ;the development of library automation program is designed to help compute professionals who want to learn about exciting field and to serve as a basic reference.

The aim of this project is to create and to develop a project in a scientific method to introduce the gab between scientific theoretical life and work normal life.

In this project, it's been constructed a library automation program for the availability of information is incrementally important an all over the word, how to make a cays process in order to have a quick research, data process, analysis process.

Finally, all file enclosed full details about the project.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	vii
CHAPTER ONE	
1. BASIC CONCEPTS OF DELPHI	1
1.1 Introduction to Delphi	1
1.2 What is Delphi	1
1.2.1 Delphi Compilers	1
1.2.2 What Kind of Programming Can You Do with Delphi	2
1.2.3 History of Delphi	3
1.2.4 Advantages & Disadvantages of Delphi	5
1.3 Delphi 6 Editions	6
1.3.1 Delphi 6 Archite	7
1.3.2 Installation Delphi 6	7
1.4. A Tour of Environment	9
1.4.1 Running Delphi for The First Time	9
1.4.2 The Delphi IDE	10
1.4.3 The Menus & Toolbar	11
1.4.4 The component Palette	12

1.4.5	The Code Editor	13
1.4.6	The Object Inspector	13
1.4.7	The Object Tree View	14
1.4.8	Class Completion	15
1.4.9	Debugging Applications	16
1.4.10	Exploring Databases	17
1.4.11	Templates and Object Repository	18
1.5	Programming with Delphi	19
1.5.1	Starting A New Application	19
1.5.1.1	Setting Property Values	21
1.5.2	Adding Objects to The Form	21
1.5.3	Add a Table and Statusbar to the Form	21
1.5.4	Connecting to a Database	23

CHAPTER TWO

2.	WHAT is DATABASE?	27
2.1.	History	27
2.2	Database Models	29
2.2.1	Hierarchical model	29
2.2.2	Network model	29
2.2.3	Relational model	30
2.2.4	Relational operations	31
2.2.5	Normal forms	31
2.3	Database Management Systems	31
2.3.1	Relational database management systems	31

2.3.2	Post-relational database models	32
2.3.3	Object database models	32
2.4	DBMS internals	33
2.4.1	Indexing	33
2.4.2	Transactions and concurrency	34
2.4.3	Replication	35
2.4.4	Security	35
2.5	Applications of databases	35

CHAPTER THREE

3.	PARADOX (DATABASE)	37
3.1	Paradox For DOS	37
3.2	Paradox For Windows	38
3.2	Corel Paradox	40
3.3	Paradox Users	40

CHAPTER FOUR

4.	PROGRAM DESIGN PROCESS	41
4.1	Login Screen Form	41
4.2	Main Menu Form	43
4.3	Return Book	44
4.4	Give Book	47
4.5	Update	51
4.6	Add, Delete, Update User and Book	55

4.6.1 Add, Delete, Update User	56
4.6.2 Add, Delete, Update Book	58
4.7 Search	62
4.7.1 Search Book	63
4.7.2 Search User	66
4.7.3 Search Borrower	68
4.8 Report	70
4.8.1 Report Book	71
4.8.2 Report User	74
4.8.2 Report Borrower	75
4.9 About in Main Menu	77
4.9 Exit	79
4.10 Other Users	80
4.10.1 Student Login	80
4.10.2 Staff Login	81
4.10.3 Guest Login	82
4.10.4 Search Book	83
4.10.5 Statue and Location	83
CONCLUSION	85
REFERENCES	86
APPENDIX	87

LIST OF FIGURES

Figure 1.1 : Select Page For Start Installation	7
Figure 1.2 : Serial Number and Authorization Screen	8
Figure 1.3 : License Agreement Screen	8
Figure 1.4 : Setup Type And Destination Folder Screen	8
Figure 1.5 : Start Menu	9
Figure 1.6 : Borland Delphi 6 Folder	9
Figure 1.7 : IDE	10
Figure 1.8 : Menu ,Title , Speed Bar & Component Palette	11
Figure 1.9 : Component Palette	12
Figure 1.10 : Code Editor Window	13
Figure 1.11 : Object Inspector	14
Figure 1.12 : Object Tree View	15
Figure 1.13 : Class	16
Figure 1.14 : Run	17
Figure 1.15: SQL Explorer	18
Figure 1.16 : New Item	18
Figure 1.17 : Form Screen	21
Figure 1.18 : Standard Button	21
Figure 1.19 : BDE Component palette	22
Figure 1.20 : Table In The Form	22
Figure 1.21 : Select Database Name	23

Figure 1.22 : DBGrid In The Form	24
Figure 1.23 : Show Table	25
Figure 4.1 Login To program	41
Figure 4.2 Wrong Password or Name	42
Figure 4.3 Admin Entrance	42
Figure 4.4 Main Menu	43
Figure 4.5 Return Book Option	44
Figure 4.6 Return Book 1	45
Figure 4.7 Return Book	46
Figure 4.8 Give Book Option	47
Figure 4.9 Give Book By Searching with ISBN	48
Figure 4.10 Give Book by Searching with Book Name	49
Figure 4.11 Give Book by Searching with Name	50
Figure 4.12 Update Option	51
Figure 4.13 Update Form	52
Figure 4.14 Update Searching By ISBN	53
Figure 4.15 Update Searching By Book Name	54
Figure 4.16 Add, Delete, Update Option	55
Figure 4.17 Add, Delete, Update Searching By Id	56
Figure 4.18 Add, Delete, Update Searching By Name	57
Figure 4.19 Add, Delete, Update Searching By Surname	58
Figure 4.20 Add, Delete, Update Book Searching By ISBN	59
Figure 4.21 Add, Delete, Update Book Searching By Book Name	60
Figure 4.22 Add, Delete, Update Book Searching By Writer	61

Figure 4.23 Search Option	62
Figure 4.24 Search Book By ISBN	63
Figure 4.25 Search Book By ISBN	64
Figure 4.26 Search Book By Name	65
Figure 4.27 Search User	66
Figure 4.28 Search User By Id	67
Figure 4.29 Search Borrower	68
Figure 4.30 Search Borrower By Id	69
Figure 4.31 Report	70
Figure 4.32 Report Book	71
Figure 4.33 Report Book By Name	72
Figure 4.34 Lists of Books	73
Figure 4.35 Print Book	73
Figure 4.36 Report User	74
Figure 4.37 List of Users	75
Figure 4.38 Report Borrower	75
Figure 4.39 Report Borrower By Id	76
Figure 4.40 List Borrowers	76
Figure 4.41 About Option	77
Figure 4.41 About Me	78
Figure 4.42 Exit	79
Figure 4.43 Student Login	80
Figure 4.44 Staff Login	81
Figure 4.46 Guest Login	82

Figure 4.47 Search Book

83

Figure 4.48 Statue and Location Form

84

CHAPTER 1

1.BASIC CONCEPT OF DELPHI

1.1.Introduction to Delphi

Although I am not the most experienced or knowledgeable person on the forums I thought it was time to write a good introductory article for Delphi

1.2.What is Delphi?

Delphi is a Rapid Application Development (RAD) environment. It allows you to drag and drop components on to a blank canvas to create a program. Delphi will also allow you to use write console based DOS like programs.

Delphi is based around the Pascal language but is more developed object orientated derivative. Unlike Visual Basic, Delphi uses punctuation in its basic syntax to make the program easily readable and to help the compiler sort the code. Although Delphi code is not case sensitive there is a generally accepted way of writing Delphi code. The main reason for this is so that any programmer can read your code and easily understand what you are doing, because they write their code like you write yours.

For the purposes of this series I will be using Delphi 6. Delphi 6 provides all the tools you need to develop, test and deploy Windows applications, including a large number of so-called reusable components.

Borland Delphi, provides a cross platform solution when used with Borland Kylix - Borland's RAD tool for the Linux platform.

1.2.1.Delphi Compilers

There are two types compiler for Delphi

- **Turbo Delphi** : Free industrial strength Delphi RAD (Rapid Application Development) environment and compiler for Windows. It comes with 200+ components and its own Visual Component Framework.
- **Turbo Delphi for .NET**: Free industrial strength Delphi application development environment and compiler for the Microsoft .NET platform.

1.2.2. What kind of programming can you do with Delphi?

The simple answer is "more or less anything". Because the code is compiled, it runs quickly, and is therefore suitable for writing more or less any program that you would consider a candidate for the Windows operating system.

You probably won't be using it to write embedded systems for washing machines, toasters or fuel injection systems, but for more or less anything else, it can be used (and the chances are that probably someone somewhere has!)

Some projects to which Delphi is suited:

- Simple, single user database applications
- Intermediate multi-user database applications
- Large scale multi-tier, multi-user database applications
- Internet applications
- Graphics Applications
- Multimedia Applications
- Image processing/Image recognition
- Data analysis
- System tools
- Communications tools using the Internet, Telephone or LAN
- Web based applications

This is not intended to be an exhaustive list, more an indication of the depth and breadth of Delphi's applicability. Because it is possible to access any and all of the Windows

API, and because if all else fails, Delphi will allow you to drop a few lines of assembler code directly into your ordinary Pascal instructions, it is possible to do more or less anything. Delphi can also be used to write Dynamically Linked Libraries (DLLs) and can call out to DLLs written in other programming languages without difficulty. Because Delphi is based on the concept of self contained Components (elements of code that can be dropped directly on to a form in your application, and exist in object form, performing their function until they are no longer required), it is possible to build applications very rapidly. Because Delphi has been available for quite some time, the number of pre-written components has been increasing to the point that now there is a component to do more or less anything you can imagine. The job of the programmer has become one of gluing together appropriate components with code that operates them as required.

1.2.3.History Of Delphi

Delphi was one of the first of what came to be known as "RAD" tools, for Rapid Application Development, when released in 1995 for the 16-bit Windows 3.1 . Delphi 2, released a year later, supported 32-bit Windows environments, and a C++ variant, C++ Builder , followed a few years after.

The chief architect behind Delphi, and its predecessor Turbo Pascal , was Anders Hejlsberg until he was headhunted in 1996 by Microsoft , where he worked on Visual J++ and subsequently became the chief designer of C Sharp programming language|C# and a key participant in the creation of the Microsoft .NET Framework.

In 2001 a Linux version known as Kylix programming tool|Kylix became available. However, due to low quality and subsequent lack of interest, Kylix was abandoned after version 3.

Support for Linux and Windows cross platform development (through Kylix and the CLX component library) was added in 2002 with the release of Delphi 6.

Delphi 8, released December 2003, was a .NET -only release that allowed developers to compile Delphi Object Pascal code into .NET Microsoft Intermediate Language|MSIL . It was also significant in that it changed its IDE for the first time,

from the multiple-floating-window-on-desktop style IDE to a look and feel similar to Microsoft's Visual Studio.NET.

Although Borland fulfilled one of the biggest requests from developers (.NET support), it was criticized both for making it available too late, when a lot of former Delphi developers had already moved to C#, and for focusing so much on backward compatibility that it was not very easy to write new code in Delphi. Delphi 8 also lacked significant high-level features of the c sharp|C# language, as well as many of the more appealing features of Microsoft's Visual Studio IDE. (There were also concerns about the future of Delphi Win32 development. Because Delphi 8 did not support Win32, Delphi 7.1 was included in the Delphi 8 package.)

The next version, Delphi 2005 (Delphi 9), included the Win32 and .NET development in a single IDE, reiterating Borland's commitment to Win32 developers. Delphi 2005 includes design-time manipulation of live data from a database. It also includes an improved IDE and added a "for ... in" statement (like C#'s foreach) to the language. However, it was criticized by some for its bugs; both Delphi 8 and Delphi 2005 had stability problems when shipped, which were only partially resolved in service packs.

In late 2005 , Delphi 2006 was released and federated development of C# and Delphi.NET, Delphi Win32 and C++ into a single IDE. It was much more stable than Delphi 8 or Delphi 2005 when shipped, and improved even more after the service packs and several hotfixes.

On February 8 , 2006 , Borland announced that it was looking for a buyer for its IDE and database line of products, which include Delphi, to concentrate on its Application Lifecycle Management|ALM line. The news met with voluble optimism from the remaining Delphi users.

On September 6 , 2006, The Developer Tools Group (the working name of the not yet spun off company) of Borland Software Corporation released single language versions of Borland Developer Studio, bringing back the popular "Turbo" moniker. The Turbo product set includes Turbo Delphi for Win32, Turbo Delphi for .NET, Turbo C++, and Turbo C#. Each version is available in two editions: "Explorer"—a free downloadable version—and "Professional"—a relatively cheap (US\$399)

version which opens access to thousands of third-party components. Unlike earlier "Personal" editions of Delphi, new "Explorer" editions can be used for commercial development.

On November 14, 2006, Borland announced the cancellation of the sale of its Development tools; instead of that it would spin them off into an independent company named "CodeGear"

1.2.4. Advantages & Disadvantages Delphi

==Advantages==

Delphi exhibits the following advantages:

- Rapid Application Development (RAD)
- Based on a well-designed language - high-level and strongly typed, with low-level escapes for experts
- A large community on Usenet and the World Wide Web (e.g. news://newsgroups.borland.com and Borland's web access to Delphi)
- Can compile to a single executable, simplifying distribution and reducing DLL versioning issues
- Many VCL and third-party components (usually available with full source code) and tools (documentation, debug tools, etc.)
- Quick optimizing compiler and ability to use assembler code
- Multiple platform native code from the same source code
- High level of source compatibility between versions
- Cross Kylix - a third-party toolkit which allows you to compile native Kylix/Linux applications from inside the Windows Delphi IDE, hence easily enabling dual-platform development and deployment
- Cross FBC - a sister project to CrossKylix, which enables you to cross-compile your Windows Delphi applications to multi-platform targets - supported by the Free Pascal compiler - without ever leaving the Delphi IDE
- Class helpers to bridge functionality available natively in the Delphi RTL, but not available in a new platform supported by Delphi

- The language's object orientation features only class- and interface-based Polymorphism in object-oriented programming polymorphism

Disadvantages

- Limited cross-platform capability for Delphi itself. Compatibles provide more architecture/OS combinations
- Access to platform and third party libraries require header files to be translated to Pascal. This creates delays and introduces the possibilities of errors in translation.
- There are fewer published books on Delphi than on other popular programming languages such as C++ and C#
- A reluctance to break any code has lead to some convoluted language design choices, and orthogonality and predictability have suffered

1.3. Delphi 6 Editions

There are 3 editions in Delphi 6 :

- **Delphi Personal** - makes learning to develop non-commercial Windows applications fast and fun. Delphi 6 Personal makes learning Windows development easy with drag-and-drop visual programming.
- **Delphi Professional** - adds the tools necessary to create applications with the latest Windows® ME/2000 look-and-feel. Dramatically enhance functionality with minimal code using the power and flexibility of SOAP and XML to easily integrate Web Services into client-side applications.
- **Delphi Enterprise** - includes additional tools, extensive options for Internet. Delphi 6 makes next-generation e-business development with Web Services a snap. This Program will concentrate on the Enterprise edition.

1.3.1. Delphi 6 Archite

Delphi 6 Architect is designed for professional enterprise developers who need to adapt quickly to changing business rules and manage sophisticated applications that synchronize with multiple database schemas. Delphi 2006 Architect includes an advanced ECO III framework that allows developers to rapidly deploy scalable external facing Web applications with executable state diagrams, object-relational mapping, and transparent persistence.

Delphi 6 Architect includes all of the capabilities of the Enterprise edition, and includes the complete ECO III framework, including new support for ECO State Machines powered by State Chart visual diagrams, and simultaneous persistence to multiple and mixed database servers.

- State Chart Diagrams
- Executable ECO State Machines
- Multi- and Mixed- ECO database support

1.3.2.Installation Delphi 6

To install Delphi 6 Enterprise, run INSTALL.EXE (default location C:\Program Files\Borland Delphi) and follow the installation instructions.

We are prompted to select a product to install, you only have one choice "Delphi 6":



Figure 1.1 The Select Page For Start Installation

While the setup runs, you'll need to enter your serial number and the authorization key (the two you got from inside a Cd rom driver).

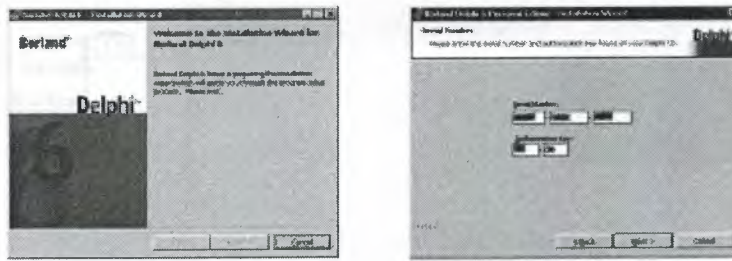


Figure 1.2 Serial Number And Authorization Screen

Later, the License Agreement screen will popup:

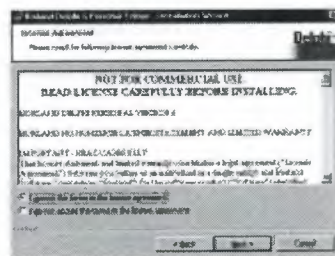


Figure 1.3 Lisanse

Agreement Screen

After that, you have to pick the Setup Type, choose Typical. This way Delphi 6 Enterprise will be installed with the most common options. The next screen prompts you to choose the Destination folder.

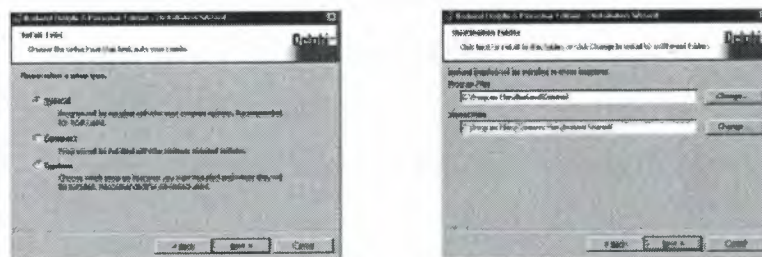


Figure 1.4.SetUp Type and Destination Folder Screen

At the end of the installation process, the set-up program will create a sub menu in the Programs section of the Start menu, leading to the main Delphi 6 Enterprise program plus some additional tools.



Figure 1.5.Start Menu

For a faster access to Delphi, create a shortcut on the Windows Desktop.

1.4. A Tour Of The Environment

This chapter explains how to start Delphi and gives you a quick tour of the main parts and tools of the Integrated Development Environment(IDE)

1.4.1. Running Delphi For The First Time

You can start Delphi in a similar way to most other Windows applications:

- Choose Programs | Borland Delphi 6 | Delphi 6 from the Windows Start menu
- Choose Run from the Windows Start menu and type Delphi32
- Double-click Delphi32.exe in the \$(DELPHI)\Bin folder. Where \$(DELPHI) is a folder where Delphi was installed. The default is C:\Program Files\Borland\Delphi6.
- Double-click the Delphi icon on the Desktop (if you've created a shortcut)

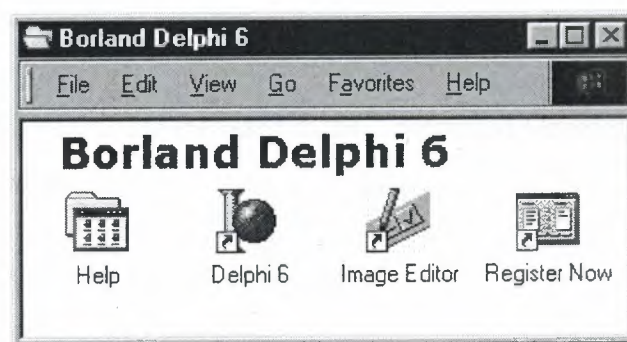


Figure 1.6.Borland Delphi 6 Folder

1.4.2. The Delphi IDE

As explained before, one of the ways to start Delphi is to choose Programs | Borland Delphi 6 | Delphi 6 from the Windows Start menu.

When Delphi starts (it could even take one full minute to start - depending on your hardware performance) you are presented with the IDE: the user interface where you can design, compile and debug your Delphi projects.

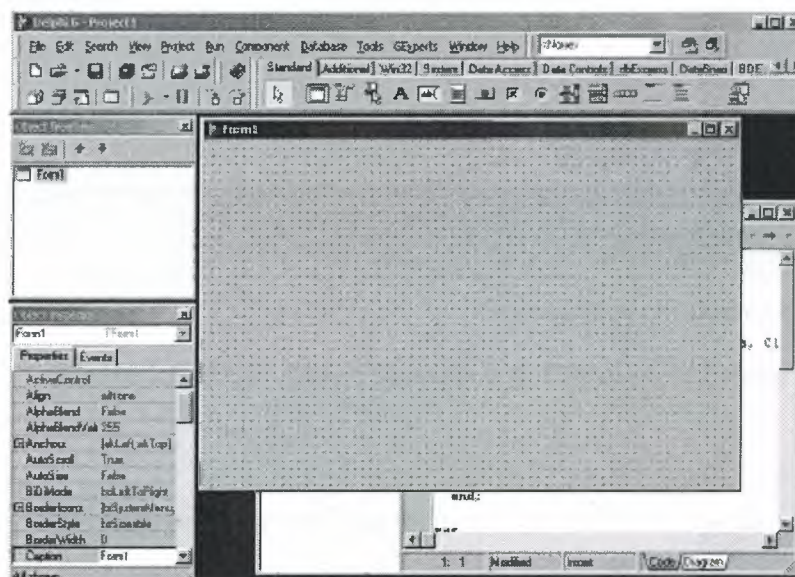


Figure 1.7.IDE

Like most other development tools (and unlike other Windows applications), Delphi IDE comprises a number of separate windows.

Some of the facilities that are included in the "Integrated Development Environment" (IDE) are listed below:

- A syntax sensitive program file editor
- A rapid optimising compiler
- Built in debugging /tracing facilities

- A visual interface developer
- Syntax sensitive help files
- Database creation and editing tools
- Image/Icon/Cursor creation / editing tools
- Version Control CASE tools

1.4.3. The Menus & Toolbar

The main window, positioned on the top of the screen, contains the main menu, toolbar and Component palette.

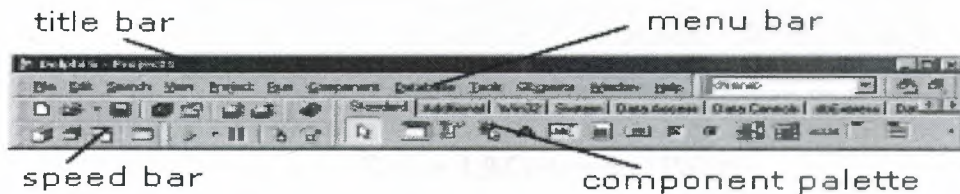


Figure 1.8.Menu ,Title , Speed Bar & Component Palette

The title bar of the main window contains the name of the current project (you'll see in some of the future chapters what exactly is a Delphi project).

The menu bar includes a dozen drop-down menus - we'll explain many of the options in these menus later through this course. The toolbar provides a number of shortcuts to most frequently used operations and commands - such as running a project, or adding a new form to a project. To find out what particular button does, point your mouse "over" the button and wait for the tooltip. As you can see from the tooltip (for example, point to [Toggle Form/Unit]), many toolbuttons have keyboard shortcuts ([F12]).

The menus and toolbars are freely customizable. I suggest you to leave the default arrangement while working through the chapters of this course.

1.4.4. The Component Palette

You are probably familiar with the fact that any window in a standard Windows application contains a number of different (visible or not to the end user) objects, like: buttons, text boxes, radio buttons, check boxes etc. In Delphi programming terminology such objects are called controls (or components). Components are the building blocks of every Delphi application. To place a component on a window you drag it from the component palette. Each component has specific attributes that enable you to control your application at design and run time.

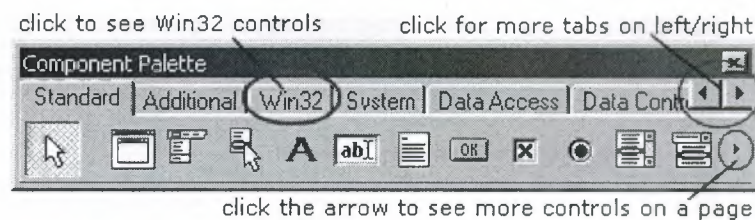


Figure 1.9.Component Palatte

Depending on the version of Delphi (assumed Delphi 6 Personal through this course), you start with more than 85 components at your disposal - you can even add more components later (those that you create or from a third party component vendor).

The components on the Component Palette are grouped according to the function they perform. Each page tab in the Component palette displays a group of icons representing the components you can use to design your application interface. For example, the Standard and Additional pages include controls such as an edit box, a button or a scroll box.

To see all components on a particular page (for example on the Win32 page) you simply click the tab name on the top of the palette. If a component palette lists more components that can be displayed on a page an arrow will appear on a far right side of the page allowing you to click it to scroll right. If a component palette has more tabs (pages) that can be displayed, more tabs can be displayed by clicking on the arrow buttons on the right-hand side.

1.4.5. The Code Editor

Each time you start Delphi, a new project is created that consists of one *empty* window. A typical Delphi application, in most cases, will contain more than one window - those windows are referred to as forms.

In our case this form has a name, it is called Form1. This form can be renamed, resized and moved, it has a caption and the three standard minimize, maximize and close buttons. As you can see a Delphi form is a regular Windows window

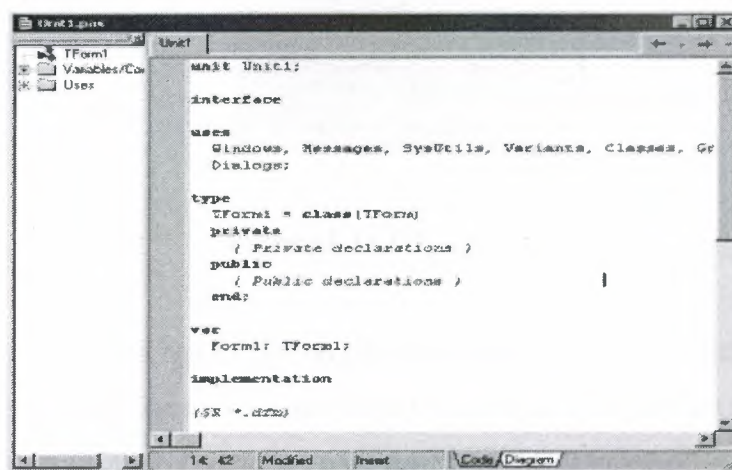


Fig.1.10.Code Editor Window

If the Form1 is the active window and you press [F12], the Code Editor window will be placed on top. As you design user interface of your application, Delphi automatically generates the underlying Object Pascal code. More lines will be added to this window as you add your own code that drives your application. This window displays code for the current form (Form1); the text is stored in a (so-called) unit - Unit1. You can open multiple files in the Code Editor. Each file opens on a new page of the Code editor, and each page is represented by a tab at the top of the window.

1.4.6. The Object Inspector

Each component and each form, has a set of properties – such as color, size, position, caption – that can be modified in the Delphi IDE or in your code, and a collection of events – such as a mouse click, keypress, or component activation – for which you can

specify some additional behavior. The Object Inspector displays the properties and events (note the two tabs) for the selected component and allows you to change the property value or select the response to some event.

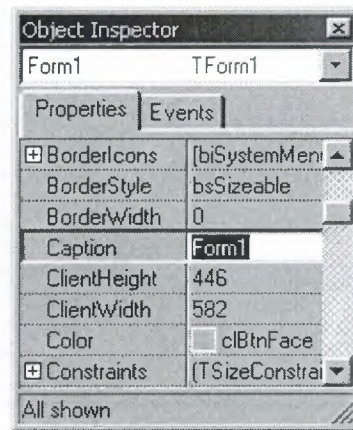


Figure 1.11.Object Inspector

For example, each form has a Caption (the text that appears on it's title bar). To change the caption of Form1 first activate the form by clicking on it. In the Object Inspector find the property Caption (in the left column), note that it has the 'Form1' value (in the right column). To change the caption of the form simply type the new text value, like 'My Form' (without the single quotes). When you press [Enter] the caption of the form will change to My Form.

Note that some properties can be changed more simply, the position of the form on the screen can be set by entering the value for the Left and Top properties - or the form can be simply dragged to the desired location.

1.4.7. The Object TreeView

Above the Object Inspector you should see the Object TreeView window. For the moment it's display is pretty simple. As you add components to the form, you'll see that it displays a component's parent-child relationships in a tree diagram. One of the great features of the Object TreeView is the ability to drag and drop components in order to change a component container without losing connections with other components.

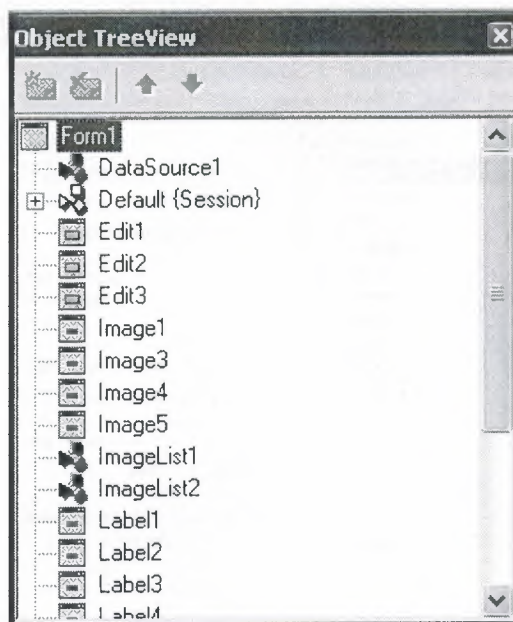


Figure 1.12.Object Tree View

The Object TreeView, Object Inspector and the Form Designer (the Form1 window) work cooperatively. If you have an object on a form (we have not placed any yet) and click it, its properties and events are displayed in the Object Inspector and the component becomes focussed in the Object TreeView.

1.4.8.Class Completion

Class Completion generates skeleton code for classes. Place the cursor anywhere within a class declaration; then press `Ctrl+Shift+C`, or right-click and select **Complete Class** at Cursor. Delphi automatically adds private **read** and **write** specifiers to the declarations for any properties that require them, then creates skeleton code for all the class's methods. You can also use Class Completion to fill in class declarations for methods you've already implemented.

To configure Class Completion, choose **Tools|Environment Options** and click the **Explorer** tab.

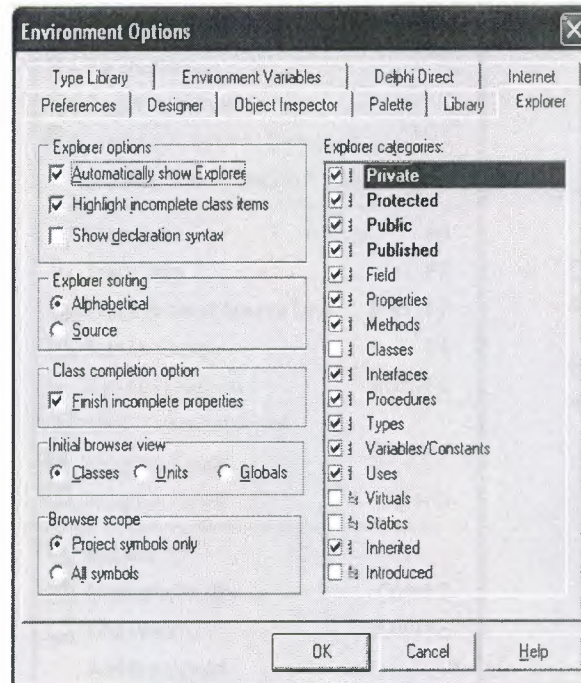
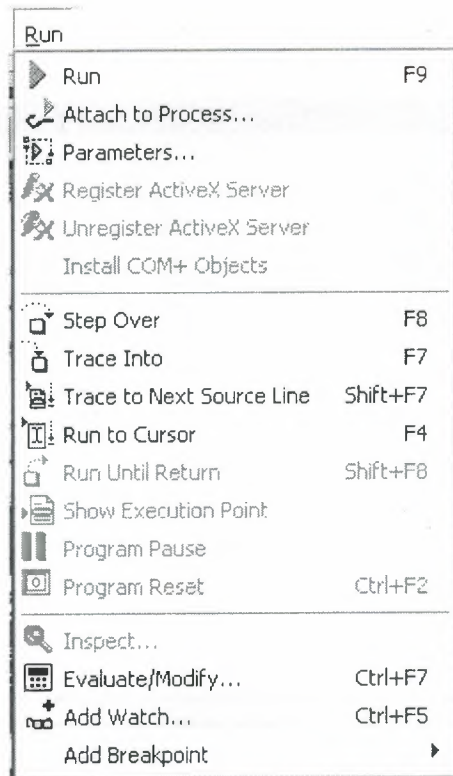


Fig.1.13.Class

1.4.9.Debugging applications

The IDE includes an integrated debugger that helps you locate and fix errors in your code. The debugger lets you control program execution, watch variables, and modify data values while your application is running. You can step through your code line by line, examining the state of the program at each breakpoint.



Choose any of the debugging commands from the Run menu.

Some commands are also available on the toolbar.

Figure 1.14. Run

To use the debugger, you must compile your program with debug information. Choose Project|Options, select the Compiler page, and check Debug Information. Then you can begin a debugging session by running the program from the IDE. To set debugger options, choose Tools|Debugger Options.

Many debugging windows are available, including Breakpoints, Call Stack, Watches, Local Variables, Threads, Modules, CPU, and Event Log. Display them by choosing View|Debug Windows. To learn how to combine debugging windows for more convenient use, see "Docking tool windows".

1.4.10. Exploring databases

The SQL Explorer (or Database Explorer in some editions of Delphi) lets you work directly with a remote database server during application development. For example, you can create, delete, or restructure tables, and you can import constraints while you are developing a database application.

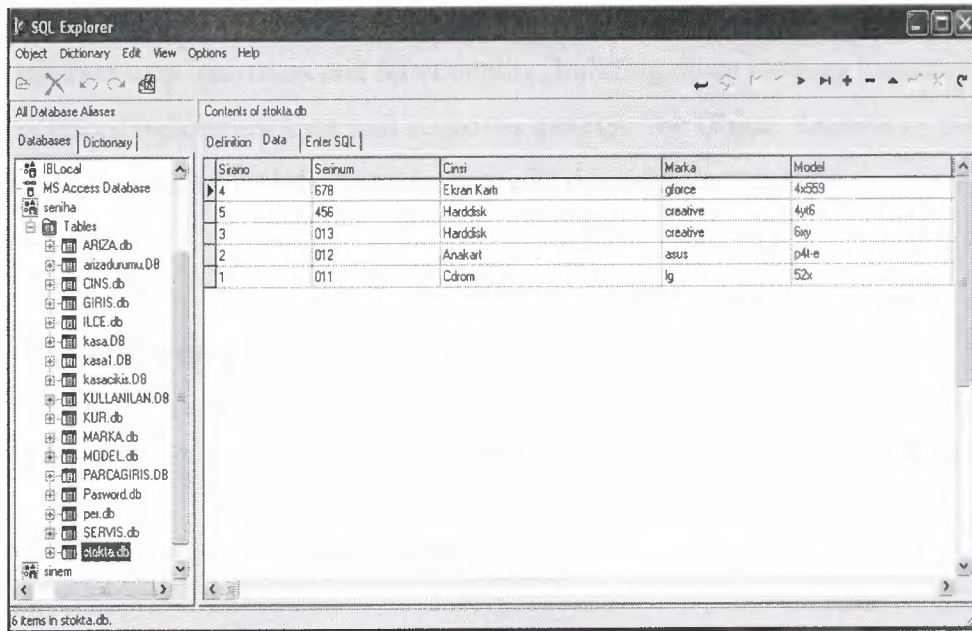


Figure 1.15. SQL Explorer

1.4.11. Templates and the Object Repository

The Object Repository contains forms, dialog boxes, data modules, wizards, DLLs, sample applications, and other items that can simplify development. Choose File|New to display the New Items dialog when you begin a project. Check the Repository to see if it contains an object that resembles one you want to create.

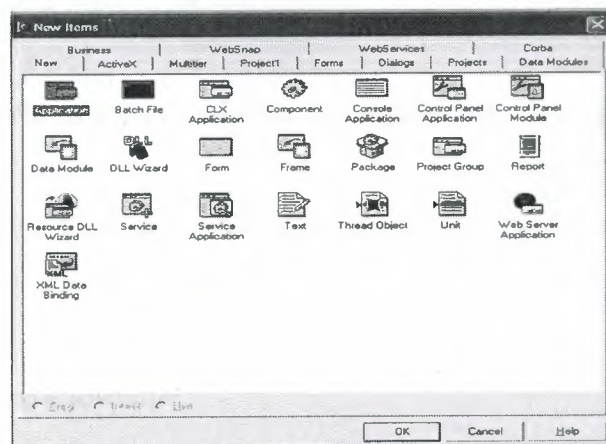


Figure 1.16. New Item

You can add your own objects to the Repository to facilitate reusing them and sharing them with other developers. Reusing objects lets you build families of applications with common user interfaces and functionality; building on an existing foundation also reduces development time and improves quality. The Object Repository provides a central location for tools that members of a development team can access over a network.

1.5.Programming With Delphi

The following section provide an overview of software development with Delphi.

1.5.1.Starting a New Application

Before beginning a new application, create a folder to hold the source files.

1. Create a folder called Seniha in the Projects directory off the main Delphi directory.
2. Open a new project.

Each application is represented by a project . When you start Delphi, it opens a blank project by default. If another project is already open, choose File|New Application to create a new project.

When you open a new project, Delphi automatically creates the following files.

- Project1.DPR : a source-code file associated with the project. This is called a project file.
- Unit1.PAS : a source-code file associated with the main project form. This is called a unit file.
- Unit1.DFM : a resource file that stores information about the main project form. This is called a form file.

Each form has its own unit and form files.

3. Choose File|Save All to save your files to disk. When the Save dialog appears, navigate to your Seniha folder and save each file using its default name.

Later on, you can save your work at any time by choosing File|Save All.

When you save your project, Delphi creates additional files in your project directory. You don't need to worry about them but don't delete them.

When you open a new project, Delphi displays the project's main form, named Form1 by default. You'll create the user interface and other parts of your application by placing components on this form.

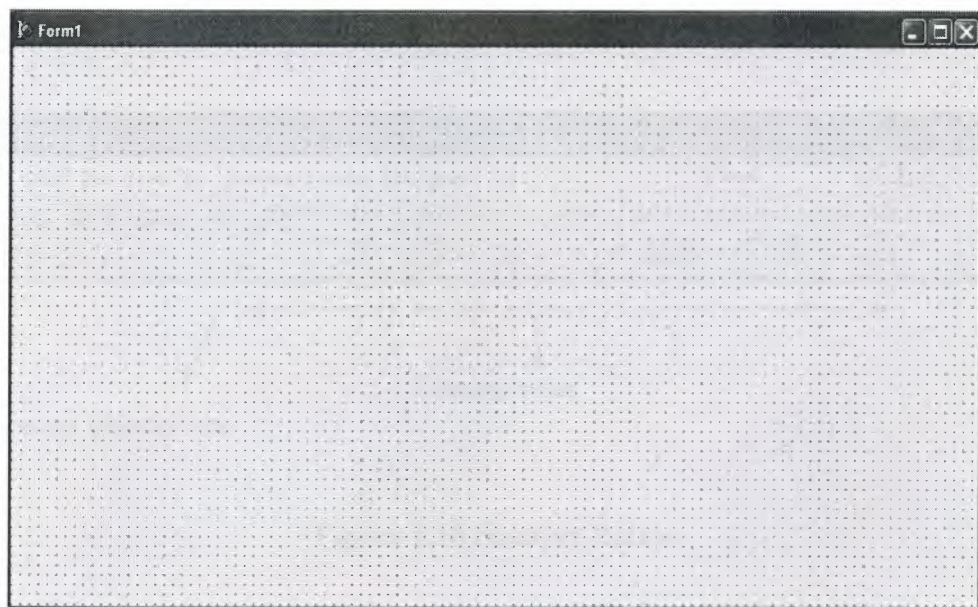


Figure 1.17.Form Screen

The default form has maximize , minimize buttons and a close button , and a control menu Next to the form, you'll see the Object Inspector, which you can use to set property values for the form and components you place on it. The drop-down list at the top of the Object Inspector shows the current selected object.when an object is sellected the Object Inspector show its properties.

1.5.1.1. Setting Property Values

When you use the Object Inspector to set properties, Delphi maintains your source code for you. The values you set in the Object Inspector are called *design-time* settings.

For Example ; Set the background color of Form1 to Aqua.

Find the form's Color property in the Object Inspector and click the drop-down list displayed to the right of the property. Choose clAqua from the list.

1.5.2. Adding objects to the form

The Component palette represents components by icons grouped onto tabbed pages. Add a component to a form by selecting the component on the palette, then clicking on the form where you want to place it. You can also double-click a component to place it in the middle of the form.

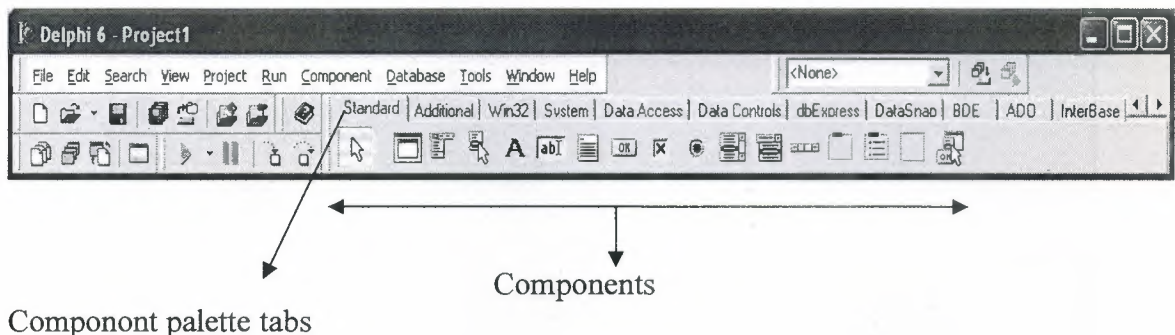


Figure 1.18.Standart Button

1.5.3.Add a Table and a StatusBar to the form:

Drop a Table component onto the form.

Click the BDE tab on the Component palette. To find the *Table* component, point at an icon on the palette for a moment; Delphi displays a Help hint showing the name of the component.

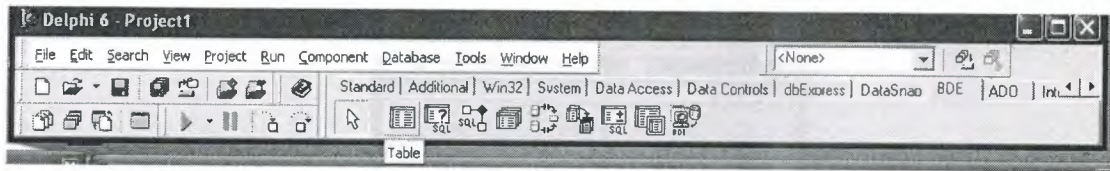


Fig.1.19.BDE Component palette

When you find the Table component, click it once to select it, then click on the form to place the component. The Table component is nonvisual, so it doesn't matter where you put it. Delphi names the object Table1 by default. (When you point to the component on the form, Delphi displays its name--Table1--and the type of object it is--TTable.)

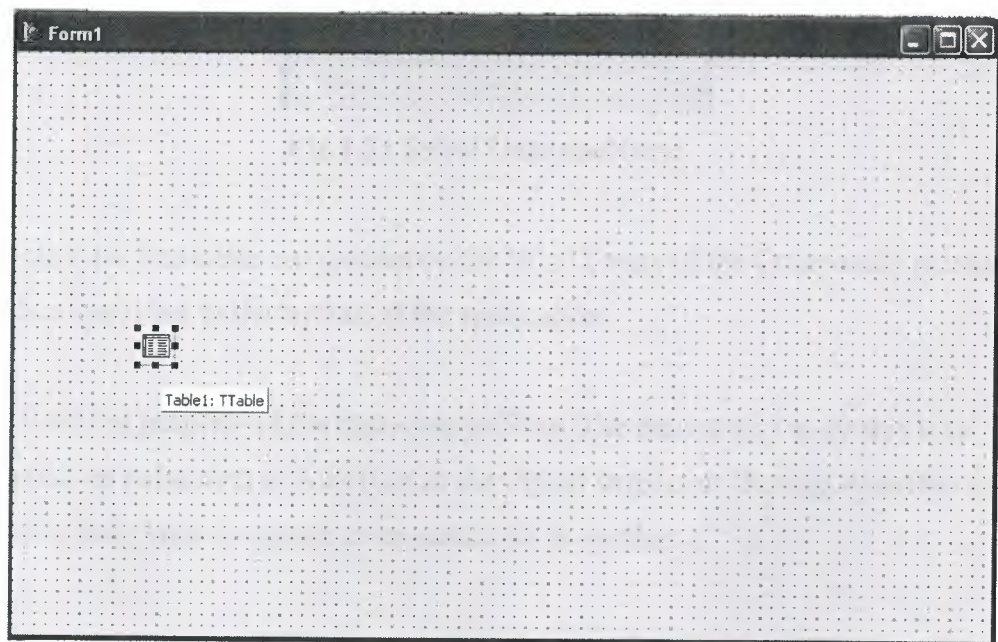


Figure 1.20.Table In The Form

Each Delphi component is a class; placing a component on a form creates an instance of that class. Once the component is on the form, Delphi generates the code necessary to construct an instance object when your application is running.

Set the DatabaseName property of Table1 to DBDEMOS. (DBDEMOS is an alias to the sample database that you're going to use.)

Select Table1 on the form, then choose the DatabaseName property in the Object Inspector. Select DBDEMOS from the drop-down list.

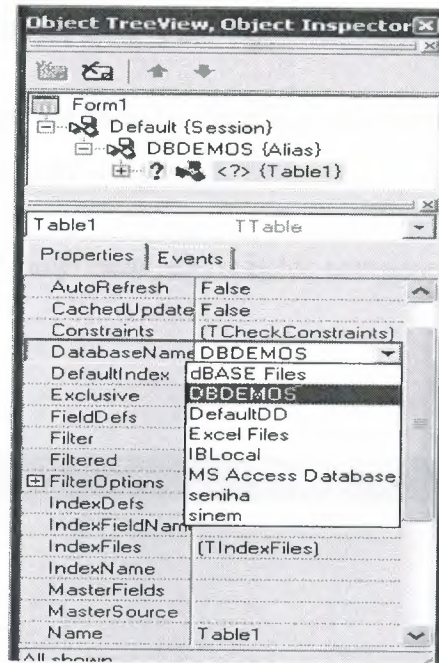


Fig.1.21.Select DatabaseName

Double-click the StatusBar component on the Win32 page of the Component palette. This adds a status bar to the bottom of the application.

Set the AutoHint property of the status bar to True. The easiest way to do this is to double-click on False next to AutoHint in the Object Inspector. (Setting AutoHint to True allows Help hints to appear in the status bar at runtime.)

1.5.4. Connecting to a Database

The next step is to add database controls and a DataSource to your form.

1. From the Data Access page of the Component palette, drop a DataSource component onto the form. The DataSource component is nonvisual, so it doesn't matter where you put it on the form. Set its DataSet property to Table1.

2. From the Data Controls page, choose the DBGrid component and drop it onto your form. Position it in the lower left corner of the form above the status bar, then expand it by dragging its upper right corner.

If necessary, you can enlarge the form by dragging its lower right corner. Your form should now resemble the following figure :

The Data Control page on Component palette holds components that let you view database tables.

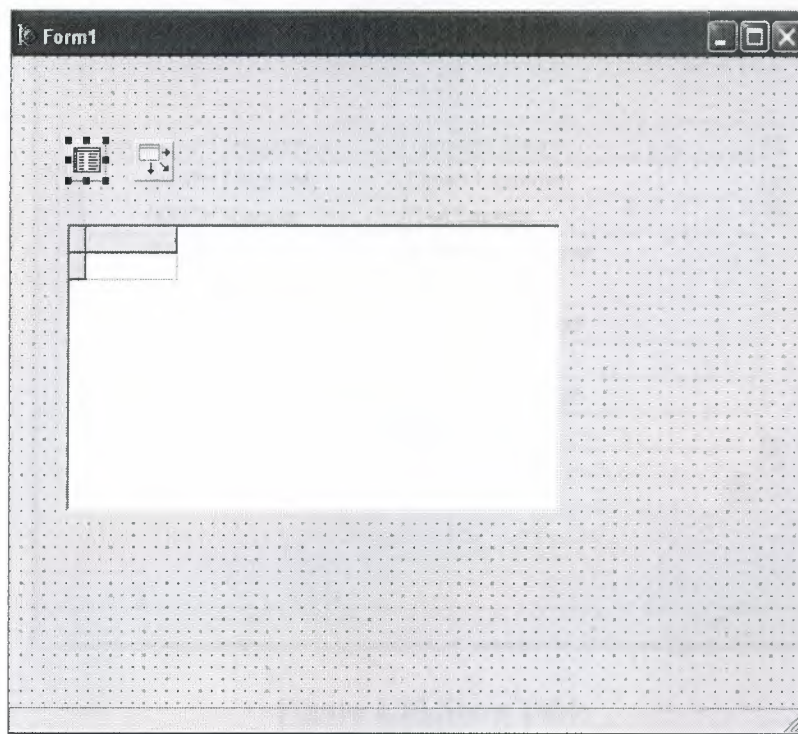


Figure 1.22.DBGrid In The Form

3. Set DBGrid properties to align the grid with the form. Double-click Anchors in the Object Inspector to display `akLeft`, `akTop`, `akRight`, and `akBottom`; set them all to `True`.
4. Set the `DataSource` property of DBGrid to `DataSource1` (the default name of the `DataSource` component you just added to the form).

Now you can finish setting up the *Table1* object you placed on the form earlier.

5. Select the Table1 object on the form, then set its TableName property to BIOLIFE.DB. (Name is still Table1.) Next, set the Active property to True.

When you set Active to True, the grid fills with data from the BIOLIFE.DB database table. If the grid doesn't display data, make sure you've correctly set the properties of all the objects on the form, as explained in the instructions above. (Also verify that you copied the sample database files into your ...\Borland Shared\Data directory when you installed Delphi.)

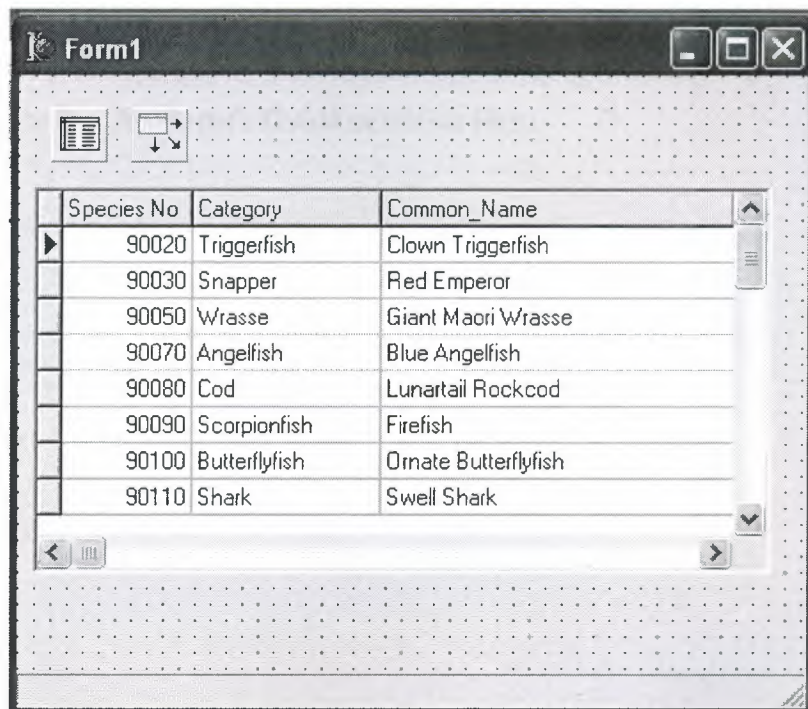


Figure 1.23.Show Table

The DBGrid control displays data at design time, while you are working in the IDE. This allows you to verify that you've connected to the database correctly. You cannot, however, edit the data at design time; to edit the data in the table, you'll have to run the application.

6. Press F9 to compile and run the project. (You can also run the project by clicking the Run button on the Debug toolbar, or by choosing Run from the Run menu.)
7. In connecting our application to a database, we've used three components and several levels of indirection. A data-aware control (in this case, a DBGrid)

points to a DataSource object, which in turn points to a dataset object (in this case, a Table). Finally, the dataset (Table1) points to an actual database table (BIOLIFE), which is accessed through the BDE alias DBDEMOS. (BDE aliases are configured through the BDE Administrator.)



This architecture may seem complicated at first, but in the long run it simplifies development and maintenance. For more information, see "Developing database applications" in the Developer's Guide or online Help.

CHAPTER 2

2. What is Database?

A database is a structured collection of records or data. A computer database relies upon software to organize the storage of data. The software models the database structure in what are known as database models. The model in most common use today is the relational model. Other models such as the hierarchical model and the network model use a more explicit representation of relationships (see below for explanation of the various database models).

Database management systems (DBMS) are the software used to organize and maintain the database. These are categorized according to the database model that they support. The model tends to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

2.1. History

The first database management systems were developed in the 1960s. A pioneer in the field was Charles Bachman. Bachman's early papers show that his aim was to make more effective use of the new direct access storage devices becoming available: until then, data processing had been based on punched cards and magnetic tape, so that serial processing was the dominant activity. Two key data models arose at this time: CODASYL developed the network model based on Bachman's ideas, and (apparently independently) the hierarchical model was used in a system developed by North American Rockwell later adopted by IBM as the cornerstone of their IMS product.

While IMS along with the CODASYL IDMS were the big, high visibility databases developed in the 1960s, several others were also born in that decade, some of which have a significant installed base today. Two worthy of mention are the PICK and

MUMPS databases, with the former developed originally as an operating system with an embedded database and the latter as a programming language and database for the development of healthcare systems.

The relational model was proposed by E. F. Codd in 1970. He criticized existing models for confusing the abstract description of information structure with descriptions of physical access mechanisms. For a long while, however, the relational model remained of academic interest only. While CODASYL products (IDMS) and network model products (IMS) were conceived as practical engineering solutions taking account of the technology as it existed at the time, the relational model took a much more theoretical perspective, arguing (correctly) that hardware and software technology would catch up in time. Among the first implementations were Michael Stonebraker's Ingres at Berkeley, and the System R project at IBM. Both of these were research prototypes, announced during 1976. The first commercial products, Oracle and DB2, did not appear until around 1980. The first successful database product for microcomputers was dBASE for the CP/M and PC-DOS/MS-DOS operating systems.

During the 1980s, research activity focused on distributed database systems and database machines. Another important theoretical idea was the Functional Data Model, but apart from some specialized applications in genetics, molecular biology, and fraud investigation, the world took little notice.

In the 1990s, attention shifted to object-oriented databases. These had some success in fields where it was necessary to handle more complex data than relational systems could easily cope with, such as spatial databases, engineering data (including software repositories), and multimedia data. Some of these ideas were adopted by the relational vendors, who integrated new features into their products as a result. The 1990s also saw the spread of Open Source databases, such as PostgreSQL and MySQL.

In the 2000s, the fashionable area for innovation is the XML database. As with object databases, this has spawned a new collection of start-up companies, but at the same time the key ideas are being integrated into the established relational products. XML databases aim to remove the traditional divide between documents and data, allowing all of an organization's information resources to be held in one place, whether they are highly structured or not.

2.2. Database Models

Various techniques are used to model data structure. Most database systems are built around one particular data model, although it is increasingly common for products to offer support for more than one model. For any one logical model various physical implementations may be possible, and most products will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance. Here are three examples:

2.2.1 Hierarchical model

In a hierarchical model, data is organized into an inverted tree-like structure, implying a multiple downward link in each node to describe the nesting, and a sort field to keep the records in a particular order in each same-level list. This structure arranges the various data elements in a hierarchy and helps to establish logical relationships among data elements of multiple files. Each unit in the model is a record which is also known as a node. In such a model, each record on one level can be related to multiple records on the next lower level. A record that has subsidiary records is called a parent and the subsidiary records are called children. Data elements in this model are well suited for one-to-many relationships with other data elements in the database.

This model is advantageous when the data elements are inherently hierarchical. The disadvantage is that in order to prepare the database it becomes necessary to identify the requisite groups of files that are to be logically integrated. Hence, a hierarchical data model may not always be flexible enough to accommodate the dynamic needs of an organisation.

2.2.2 Network model

The network model tends to store records with links to other records. Each record in the database can have multiple parents, i.e., the relationships among data elements can have a many to many relationship. Associations are tracked via "pointers". These pointers can be node numbers or disk addresses. Most network databases tend to also include some form of hierarchical model. Databases can be translated from hierarchical model to network and vice versa. The main difference between the network model and hierarchical model is that in a network model, a child can have a number of parents whereas in a hierarchical model, a child can have only one parent.

The network model provides greater advantage than the hierarchical model in that it promotes greater flexibility and data accessibility, since records at a lower level can be accessed without accessing the records above them. This model is more efficient than hierarchical model, easier to understand and can be applied to many real world problems that require routine transactions. The disadvantages are that: It is a complex process to design and develop a network database; It has to be refined frequently; It requires that the relationships among all the records be defined before development starts, and changes often demand major programming efforts; Operation and maintenance of the network model is expensive and time consuming. Examples of database engines that have network model capabilities are RDM Embedded and RDM Server.

2.2.3 Relational model

The basic data structure of the relational model is a table where information about a particular entity (say, an employee) is represented in columns and rows. The columns enumerate the various attributes of an entity (e.g. employee_name, address, phone_number). Rows (also called records) represent instances of an entity (e.g. specific employees).

The "relation" in "relational database" comes from the mathematical notion of relations from the field of set theory. A relation is a set of tuples, so rows are sometimes called tuples. All tables in a relational database adhere to three basic rule.

- The ordering of columns is immaterial
- Identical rows are not allowed in a table
- Each row has a single (separate) value for each of its columns (each tuple has an atomic value).

If the same value occurs in two different records (from the same table or different tables) it can imply a relationship between those records. Relationships between records are often categorized by their cardinality (1:1, (0), 1:M, M:M).

Tables can have a designated column or set of columns that act as a "key" to select rows from that table with the same or similar key values. A "primary key" is a key that has a unique value for each row in the table. Keys are commonly used to join or combine data from two or more tables. For example, an *employee* table may contain a column named

address which contains a value that matches the key of a *address* table. Keys are also critical in the creation of indexes, which facilitate fast retrieval of data from large tables. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

2.2.4 Relational operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many web applications, such as Wikipedia, perform SQL queries when generating pages.

In response to a query, the database returns a result set, which is the list of rows constituting the answer. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted. Often, data from multiple tables are combined into one, by doing a join. There are a number of relational operations in addition to join.

2.2.5 Normal forms

Relations are classified based upon the types of anomalies to which they're vulnerable. A database that's in the first normal form is vulnerable to all types of anomalies, while a database that's in the domain/key normal form has no modification anomalies. Normal forms are hierarchical in nature. That is, the lowest level is the first normal form, and the database cannot meet the requirements for higher level normal forms without first having met all the requirements of the lesser normal form.

2.3 Database Management Systems

2.3.1 Relational database management systems

An RDBMS implements the features of the relational model outlined above. In this context, Date's **Information Principle** states:

The entire information content of the database is represented in one and only one way. Namely as explicit values in column positions (attributes) and rows in relations (tuples). Ergo, there are no explicit pointers between related tables.

2.3.2 Post-relational database models

Several products have been identified as post-relational because the data model incorporates relations but is not constrained by the Information Principle, requiring that all information is represented by data values in relations. Products using a post-relational data model typically employ a model that actually pre-dates the relational model. These might be identified as a directed graph with trees on the nodes. Examples of models that could be classified as post-relational are PICK aka MultiValue, and MUMPS.

2.3.3 Object database models

In recent years, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the *impedance mismatch*) of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time, object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases. A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

2.4 DBMS internals

Database tables/indexes are typically stored in memory or on hard disk in one of many forms, ordered/unordered flat files, ISAM, heaps, hash buckets or B+ trees. These have various advantages and disadvantages discussed further in the main article on this topic. The most commonly used are B+ trees and ISAM.

Other important design choices relate to the clustering of data by category (such as grouping data by month, or location), creating pre-computed views known as materialized views, partitioning data by range or hash. As well memory management and storage topology can be important design choices for database designers. Just as normalization is used to reduce storage requirements and improve the extensibility of the database, conversely denormalization is often used to reduce join complexity and reduce execution time for queries.

2.4.1 Indexing

All of these databases can take advantage of indexing to increase their speed, and this technology has advanced tremendously since its early uses in the 1960s and 1970s. The most common kind of index is a sorted list of the contents of some particular table column, with pointers to the row associated with the value. An index allows a set of table rows matching some criterion to be located quickly. Typically, indexes are also stored in the various forms of data-structure mentioned above (such as B-trees, hashes, and linked lists). Usually, a specific technique is chosen by the database designer to increase efficiency in the particular case of the type of index required.

Relational DBMSs have the advantage that indexes can be created or dropped without changing existing applications making use of it. The database chooses between many different strategies based on which one it estimates will run the fastest.

In other words, indexes are transparent to the application or end-user querying the database; while they affect performance, any SQL command will run with or without index to compute the result of an SQL statement. The RDBMS will produce a plan of how to execute the query, which is generated by analyzing the run times of the different algorithms and selecting the quickest. Some of the key algorithms that deal with joins

are nested loop join, sort-merge join and hash join. Which of these is chosen depends on whether an index exists, what type it is, and its cardinality.

An index speeds up access to data, but it has disadvantages as well. First, every index increases the amount of storage on the hard drive necessary for the database file, and second, the index must be updated each time the data are altered, and this costs time. (Thus an index saves time in the reading of data, but it costs time in entering and altering data. It thus depends on the use to which the data are to be put whether an index is on the whole a net plus or minus in the quest for efficiency.)

A special case of an index is a primary index, or primary key, which is distinguished in that the primary index must ensure a unique reference to a record. Often, for this purpose one simply uses a running index number (ID number). Primary indexes play a significant role in relational databases, and they can speed up access to data considerably.

2.4.2 Transactions and concurrency

In addition to their data model, most practical databases ("transactional databases") attempt to enforce a database transaction. Ideally, the database software should enforce the ACID rules, summarized here:

- Atomicity: Either all the tasks in a transaction must be done, or none of them. The transaction must be completed, or else it must be undone (rolled back).
- Consistency: Every transaction must preserve the integrity constraints — the declared consistency rules — of the database. It cannot place the data in a contradictory state.
- Isolation: Two simultaneous transactions cannot interfere with one another. Intermediate results within a transaction are not visible to other transactions.
- Durability: Completed transactions cannot be aborted later or their results discarded. They must persist through (for instance) restarts of the DBMS after crashes.

In practice, many DBMS's allow most of these rules to be selectively relaxed for better performance.

Concurrency control is a method used to ensure that transactions are executed in a safe manner and follow the ACID rules. The DBMS must be able to ensure that only serializable, recoverable schedules are allowed, and that no actions of committed transactions are lost while undoing aborted transactions .

2.4.3 Replication

Replication of databases is closely related to transactions. If a database can log its individual actions, it is possible to create a duplicate of the data in real time. The duplicate can be used to improve performance or availability of the whole database system. Common replication concepts include:

Master/Slave Replication: All write requests are performed on the master and then replicated to the slaves

Quorum: The result of Read and Write requests are calculated by querying a "majority" of replicas.

Multimaster: Two or more replicas sync each other via a transaction identifier.

Parallel synchronous replication of databases enables transactions to be replicated on multiple servers simultaneously, which provides a method for backup and security as well as data availability.

2.4.4 Security

Database security denotes the system, processes, and procedures that protect a database from unintended activity. In the United Kingdom legislation protecting the public from unauthorized disclosure of personal information held on databases falls under the Office of the Information Commissioner. United Kingdom based organizations holding personal data in electronic format (databases for example) are required to register with the Data Commissioner.

2.5 Applications of databases

Databases are used in many applications, spanning virtually the entire range of computer software. Databases are the preferred method of storage for large multiuser applications, where coordination between many users is needed. Even individual users find them convenient, and many electronic mail programs and personal organizers are

based on standard database technology. Software database drivers are available for most database platforms so that application software can use a common Application Programming Interface to retrieve the information stored in a database. Two commonly used database APIs are JDBC and ODBC.

For example suppliers database contains the data relating to suppliers such as;

- supplier name
- supplier code
- supplier address

It is often used by schools to teach students and grade them.

CHAPTER 3

3. PARADOX (DATABASE)

Paradox is a relational database management system currently published by Corel Corporation. It was originally released for DOS by Ansa Software, but a Windows version was released by Borland in 1992.

3.1 Paradox For DOS

Paradox for DOS was a relational database management system originally written by Richard Schwartz and Robert Shostak, and released by their company Ansa Software in 1985. In September 1987, Borland purchased Ansa Software, including their Paradox/DOS 2.0 software. Notable classic versions were 3.5 and 4.5. Versions up to 3.5 were evolutions from 1.0. Version 4.0 and 4.5 were retooled in the Borland C++ windowing toolkit and used a different extended memory access scheme.

Paradox/DOS was a successful DOS-based database of the late eighties and early nineties. At that time, dBase and its xBase clones (Foxpro, Clipper programming language) dominated the market. Other notable competitors were Clarion, DataEase, R:Base, and Dataflex.

The features that distinguished Paradox/DOS were:

- a visual Query By Example implementation that was supported by an AI engine.

- effective use of memory (conventional as well as extended / expanded) - caching data tables and particularly, indexes which caused Paradox to execute tasks very quickly in contrast to the explicit skills required for xBase performance optimisation.[1]

- an innovative programming language the Paradox Application Language (PAL) that was readable, powerful, and could be recorded from keyboard actions (rather like Lotus 1-2-3 macro recording).

Lotus-like text menus and windows which was the native interface (in contrast to dBase which had a command line interface on top of which cumbersome menus were layered). Particularly in Paradox 1.0 and 2.0, the user and programming manuals won readability awards[citation needed] - they were copiously illustrated, well laid out and explanations

were written in common English. In contrast, xBase and other manuals were text heavy, sometimes even typed in plain Courier with no attempt at professional page layout.

3.2 Paradox For Windows

Paradox for Windows was a distinctly different product produced by a different team of programmers. Although key features of the DOS product, the QBE and the database engine, were ports keeping the DOS code, there was a major break in compatibility from PAL to ObjectPAL and in the shift to a GUI design metaphor for Forms and Reports. The ObjectPAL changes were controversial but forced since PAL was based on keystroke recording actions that had no equivalent in Windows. An object-based language based on ideas from Hypercard was used in place of keystroke recording. The Forms and Reports designers used device independent scaling including ability to work in zoomed mode for detailed layout. The mouse right-click was used for access to Forms and Reports properties, inspired by the Xerox Alto and Smalltalk, in a way now almost universal to Windows programs. The ObjectPAL was (like Hypercard) associated with the visual objects - also revealed by right click. Property inspection and layout tools could be "pinned up" to stay on screen, an idea borrowed from the NeXT and now fairly widely adopted in Windows.

For approximately the first year of development the object-oriented code was written in C aided by macros, until Turbo C++ was available at which point the remaining parts of the code were written in C++. The product manager up until shipping version 1.0 was Joe Duncan. The development and QA team totaled about 30 people.

Both Paradox for Windows and Quattro Pro for Windows, a closely related project, started development using beta versions of Windows 3.0, in the spring of 1990. Paradox/Windows ended up delayed about a year beyond its original plan, shipping in early 1993. The reasons were many, but not entirely surprising for a major rewrite, in OO language with new tools, shifting to a GUI paradigm, on what was essentially a first version operating system. Still it was a big problem for the company and Microsoft managed to ship Access a couple of months ahead of Paradox for Windows, a major marketing win to Microsoft.

In 1990 Borland also started work on an internal dBASE clone for both DOS and Windows, written in assembler, which was planned to ship in 1992. By early 1992 it became clear that Ashton-Tate was in difficulties on developing Windows versions of their products and so Borland switched plans, instead acquiring the company and anointing their internal project as the official successor. Part of the Ashton-Tate acquisition was the Interbase database and it was decided that Paradox/W should be able to work with Interbase as well as the Paradox engine and this led to the creation of an IDAPI engine based around Interbase.

The acquisition also shifted focus. Paradox had historically competed against dBASE in some markets, and Paradox/W originally was designed to improve the competitive position in the developer-oriented market. After dBASE was acquired this was no longer desirable and emphasis shifted towards an ease-of-use market. However the product could not be changed to match the emphasis (this occurred in later releases) at that late stage, making the product somewhat over complex for the entry level market. Access did a good job of addressing that same market and got there first, by Christmas 1992. Still, Paradox/W sold well for a while. Meanwhile, Borland was going through some serious problems caused by the Ashton-Tate acquisition. Many product lines were discontinued, corporate reorganization and consolidation was painful, and even worse the internal dBASE project at the center of the acquisition rationale was eventually cancelled for technical reasons leaving Borland with a collapse in revenues and a serious need to develop the missing dBASE for Windows in a hurry. Borland had lost the strength to fight the multiple marketing battles it needed for its range of products. Paradox was minimally marketed to the developers since the company decided it would hold out for a replacement of dBASE, which eventually came out in 1994, too late for the company.

Microsoft Access was sold for a fraction of the price of Paradox/Windows and bundled with Word, Excel and PowerPoint in Microsoft Office Professional. Furthermore, Access performance was good thanks to team contributions from FoxPro programmers. Despite solid follow-on versions with improvements to usability for entry-level users, Paradox faded from the market. It was included in the sale of Borland products to Word Perfect, which were in turn resold as Word Perfect got into financial products, and at the current time of writing Paradox for Windows, Word Perfect and Quattro Pro for

Windows are all owned by Corel and sold as part of their office suite. dBASE for Windows came out too late to be a significant player in the Windows market, most dBASE programmers by then had migrated to Microsoft FoxBase, a very similar database tool. Borland itself retained the Interbase/IDAPI server and focussed efforts on its Delphi tools which over the years gave it an influential but small part of the data-oriented developer market.

3.2 Corel Paradox

Corel acquired certain rights to develop and market Paradox in the mid-90's and released Corel Paradox 8 in 1997. It also bundled Paradox in the professional version of its WordPerfect Office suite. It has released versions 9, 10, 11, 12, X3 and the latest version Office X4 Professional Edition since then.

3.3 Paradox Users

There is a strong Paradox user base, mainly centered around the Paradox Community and its associated newsgroups. Many feel let down by Borland and Corel because they believe that Paradox is superior to all the other desktop DBMSes around.

Although there are many fans of ObjectPAL, the programming language for Paradox/Windows, PAL/DOS scripts could not easily be migrated; the object and event models were completely different forcing developers using PAL to completely rewrite their database applications.

CHAPTER 4

4.PROGRAM DESIGN PROCESS

4.1 Login Screen Form

Here the users enter the username and password to login to the program. There are five types of users: Admin, User, Guest, Student, Staff. Each of the users has different hierarchic authorities. Admin and user can use all the options in the program, but user can not change the admin's password. Student, Staff and Guest can only search book and except guest they can borrow book. The difference between student and staff is book borrowing duration. Student can borrow a book for 10 days, guest can borrow a book for 30 days.

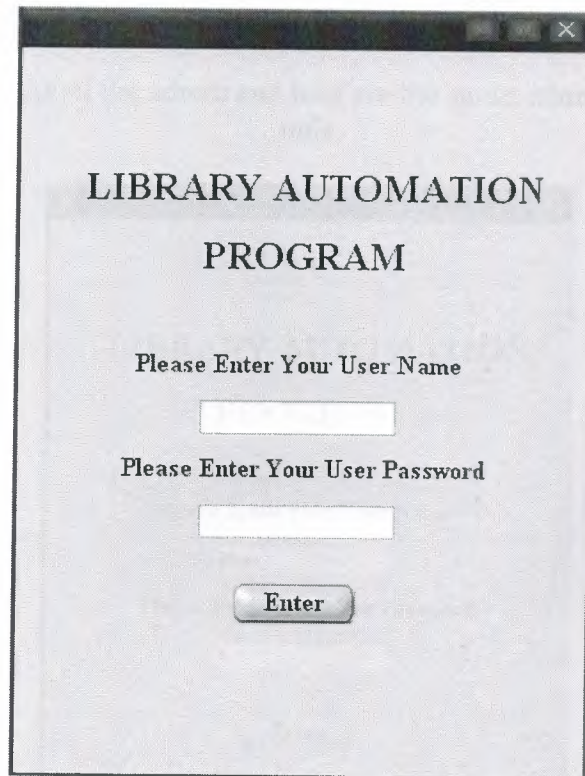
The image shows a screenshot of a login window titled "LIBRARY AUTOMATION PROGRAM". The window has a light gray background and a dark border. At the top, the title "LIBRARY AUTOMATION PROGRAM" is centered in a bold, black, sans-serif font. Below the title, there are two text prompts: "Please Enter Your User Name" and "Please Enter Your User Password", both in a bold, black, sans-serif font. Each prompt is followed by a white rectangular input field. At the bottom of the form, there is a button labeled "Enter" in a bold, black, sans-serif font. The button has a slight 3D effect with a dark border and a light gray fill.

Figure 4.1 Login To program

If the information provided by the user is correct the main form will be opened else it will not be displayed and a message box will appear to warn the user that there could be a mistake in username or password.

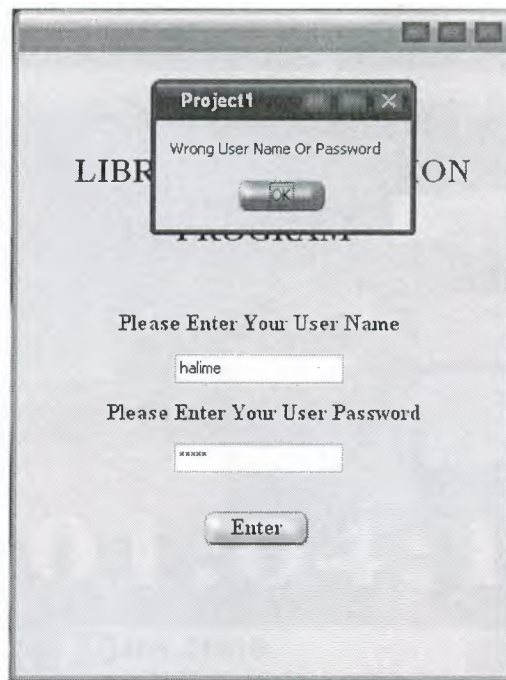


Figure 4.2 Wrong Password or Name

Because of the authority of the admin and user are the same, admin login will be shown only.



Figure 4.3 Admin Entrance

After correct informations are entered the mainmenu screenshot will be displayed.



Figure 4.4 Main Menu

This is the main page of the program.

4.2 Main Menu Form

The aim of the main menu is to use the program easily, faster and use all the process screens or necessary program at the same time. There will be options like Give Book, Return Book, Add, Delete, Update User and Book, Update Transaction, Search Book, User and Borrower, Report Book, User and Borrower. Lastly there will be an About Page in which users could see some information about the program and programmer.

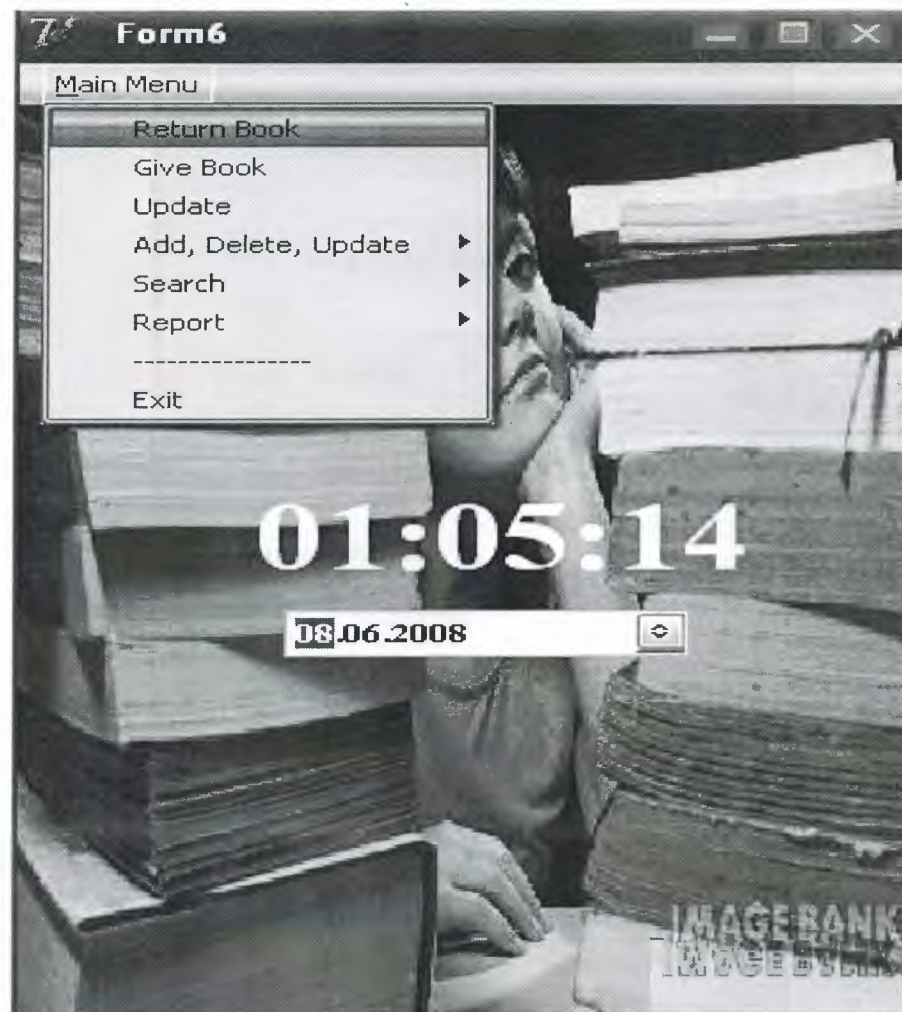


Figure 4.5 Return Book Option

4.3 Return Book

In this form, when the user selects the return book to take the book that was borrowed before, the Return book window will appear, and using search options either by isbn or by name, the user will find the important book information like rent date and return date. If the book will be given late, the program will automatically calculate lateness dept. After all data is completed, the user will change the book status from not available to available to be able to lend the book again later and press the button to send the data to the database. Available means the book is in the library, not available means the book is borrowed and not in the library. Away means the book location is unknown.

Library Automation

Main Menu

Search By ISBN Number

Search By Book Name

ISBN Number

Book Name

Borrower ID

Borrower Name

Borrower Surname

User ID

User Name

User Surname

Return Date

Late Date

Total Debt

Book Status

Return Book

Figure 4.6 Return Book

Library Automation

Main Menu

Search By ISBN Number

1

Search By Book Name

ISBN Number 111

Book Name Dudaktan Kalbe

Borrower ID 342

Borrower Name burcin

Borrower Surname baysal

User ID 546

User Name burak

User Surname aytan

Return Date 01.04.2008

Late Date 3

Total Debt 3 ytl

Book Status

Not Available

Available

Not Available

Away

Return

Figure 4.7 Return Book



Figure 4.8 Give Book Option

4.4 Give Book

In this form user can lend books to the people who wants. To give the book, user can search the book information due to isbn, book and writer name. After data is entered, book status must be changed available to not available.

7' Library Automation

Main Menu

Label1

Search By ISBN Number Search By Book Name

123

Search By Name Search By Name

ISBN Number 123 Rent Date 20.06.2008

Book Name Oracle Return Date 25.06.2008

Borrower ID 234 User ID 234

Borrower Name Murat User Name malik

Borrower Surname Karaogul User Surname düzgün

Days 10 Book Status Not Available

Subject novel

Give Book

Figure 4.9 Give Book By Searching with Isbn

7' Library Automation

Main Menu

Label1

Search By ISBN Number

Search By Book Name

Search By Name

Search By Name

ISBN Number	<input type="text" value="111"/>	Rent Date	<input type="text" value="12.03.2008"/>
Book Name	<input type="text" value="Dudaktan Kalbe"/>	Return Date	<input type="text" value="01.04.2008"/>
Borrower ID	<input type="text" value="342"/>	User ID	<input type="text" value="546"/>
Borrower Name	<input type="text" value="burcin"/>	User Name	<input type="text" value="burak"/>
Borrower Surname	<input type="text" value="baysal"/>	User Surname	<input type="text" value="aytan"/>
Days	<input type="text" value="20"/>	Book Status	<input type="text" value="Not Available"/>
Subject	<input type="text" value="novel"/>		

Give Book

Figure 4.10 Give Book by Searching with Book Name

Library Automation

Main Menu

Label1

Search By ISBN Number Search By Book Name

Search By Name Search By Name

ISBN Number Rent Date

Book Name Return Date

Borrower ID User ID

Borrower Name User Name

Borrower Surname User Surname

Days Book Status

Subject

Figure 4.11 Give Book by Searching with Name

4.5 Update

Update option will be used to make longer borrowing days. When it is requested, return date could be changed by user.

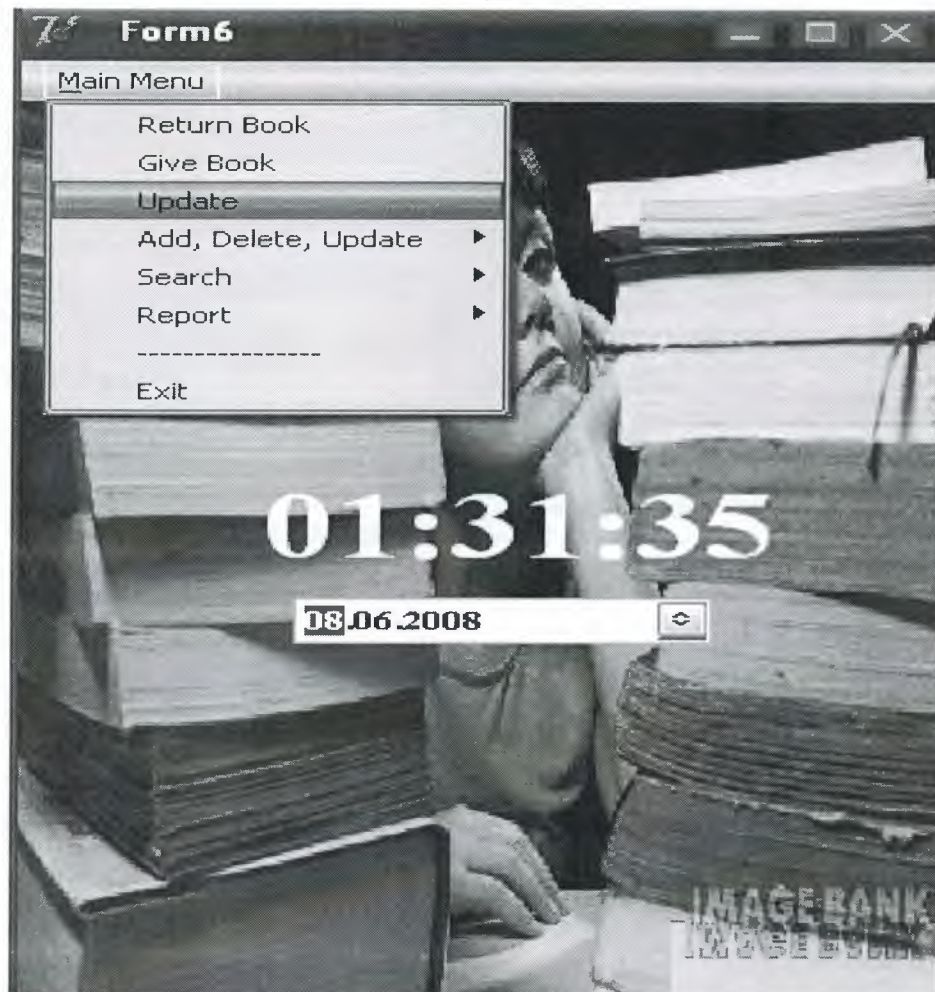


Figure 4.12 Update Option

7 Library Automation — □ ×

Main Menu

Admin

Search By ISBN Number <input style="width: 100%;" type="text"/>	Search By Book Name <input style="width: 100%;" type="text"/>
Search By Borrower Name <input style="width: 100%;" type="text"/>	Search By Borrower Surname <input style="width: 100%;" type="text"/>

ISBN Number	Rent Date
Book Name	Return Date
Borrower ID	User ID
Borrower Name	User Name
Borrower Surname	User Surname
Days	Book Status

Subject

⏮ ⏪ ⏩ ⏭ + - ↶ ✓ ✕ ↻

Figure 4.13 Update Form

User can find the last return date record for that book by searching isbn,

7' Library Automation Main Menu

Admin

Search By ISBN Number

Search By Book Name

Search By Borrower Name

Search By Borrower Surname

ISBN Number	<input type="text" value="123"/>	Rent Date	<input type="text" value="20.06.2008"/>
Book Name	<input type="text" value="Oracle"/>	Return Date	<input type="text" value="25.06.2008"/>
Borrower ID	<input type="text" value="234"/>	User ID	<input type="text" value="234"/>
Borrower Name	<input type="text" value="Murat"/>	User Name	<input type="text" value="malik"/>
Borrower Surname	<input type="text" value="Karaogul"/>	User Surname	<input type="text" value="düzgün"/>
Days	<input type="text" value="10"/>	Book Status	<input type="text" value="Not Available"/>
Subject	<input type="text" value="novel"/>		

Figure 4.14 Update Searching By Isbn

User can find the last return date record for that book by searching book name,

Library Automation

Main Menu

Admin

Search By ISBN Number

Search By Book Name

Search By Borrower Name

Search By Borrower Surname

ISBN Number **Rent Date**

Book Name **Return Date**

Borrower ID **User ID**

Borrower Name **User Name**

Borrower Surname **User Surname**

Days **Book Status**

Subject

Figure 4.15 Update Searching By Book Name

4.6 Add, Delete, Update User and Book

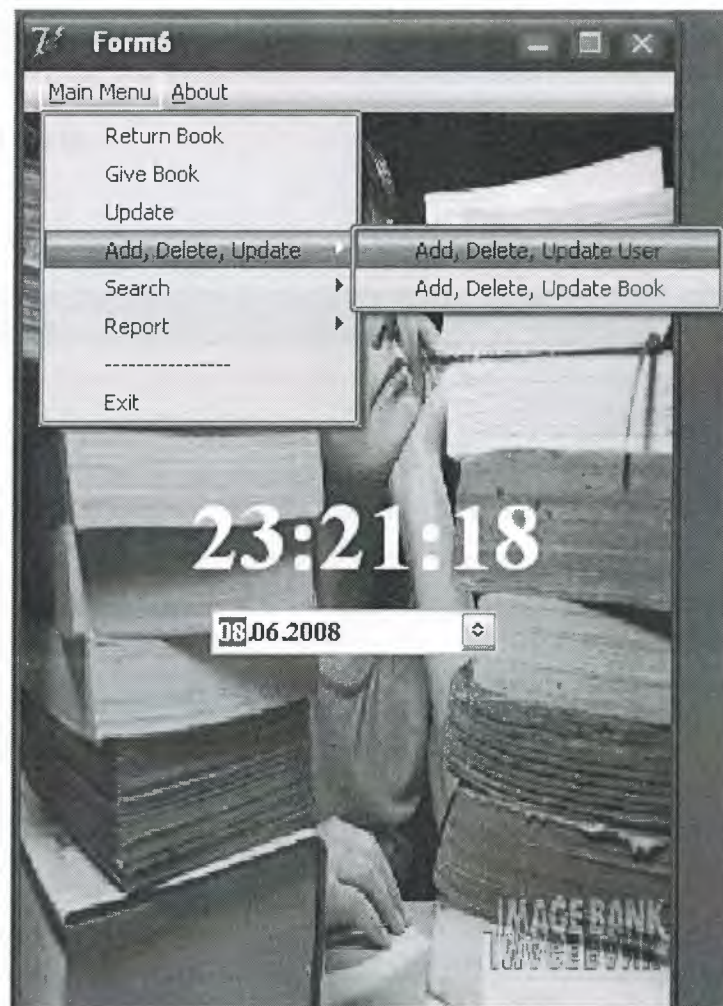


Figure 4.16 Add, Delete, Update Option

4.6.1 Add, Delete, Update User

In this window users can add a new user due to their statu's,delete a user or update a user's information.If user would update an existing record,using search options (By Id,By Name,By Surname) he/she could find and see in table below easily.

Searching By UserId,

The screenshot shows a software window titled "Library Automation" with a "Main Menu" bar. Below this is an "Admin" section. It features three search options: "Search By UserID", "Search By Name", and "Search By Surname". The "Search By UserID" option is selected, with the value "345" entered in the search field. Below the search fields are input fields for user details: "UserID" (345), "User Name" (Mehmet), "User Surname" (Tahta), "Phone" (212121212), "Password" (2003), "Mobile Phone" (2121221121), "Address" (Gonyeli / Lefkosa), "E - mail" (tahtamehmet@hotmail.c), and "Statu" (Studesnt). Below these fields is a table with columns: "userid", "username", "usersurname", and "Ph". The table contains one row with the values: "345", "Mehmet", "Tahta", and "212". At the bottom of the window is a toolbar with various icons for navigation and editing.

userid	username	usersurname	Ph
345	Mehmet	Tahta	212

Figure 4.17 Add, Delete, Update Searching By Id

Searching By Name,

Library Automation

Main Menu

Admin

Search By UserID

Search By Name

Search By Surname

UserID

Mobile Phone

User Name

Address

User Surname

Phone

E - mail

Password

Statu

userid	username	usersurname	Ph
▶ 234	Turgut	Aydin	212

<

|||

>

⏪

⏩

⏴

⏵

+

-

⏶

✓

✕

↺

Figure 4.18 Add, Delete, Update Searching By Name

Searching By Surname,

The screenshot shows a web application window titled "Library Automation" with a sub-header "Main Menu". The main section is titled "Admin". At the top, there are three search options: "Search By UserID", "Search By Name", and "Search By Surname". The "Search By Surname" field contains the text "Tahta". Below these are several form fields for user information: "UserID" (345), "User Name" (Mehmet), "User Surname" (Tahta), "Phone" (212121212), "Password" (2003), "Mobile Phone" (2121221121), "Address" (Gonyeli / Lefkosa), "E - mail" (tahtamehmet@hotmail.c), and "Statu" (Studesnt). Below the form fields is a table with the following data:

userid	username	usersurname	Ph
345	Mehmet	Tahta	212

At the bottom of the window, there is a row of navigation buttons: a left arrow, a right arrow, a double left arrow, a double right arrow, a plus sign, a minus sign, an up arrow, a down arrow, a refresh/cancel button, and a redo/undo button.

Figure 4.19 Add, Delete, Update Searching By Surname

4.6.2 Add,Delete,Update Book

In this window users can add a new book ,delete book or update a book's information.If user would update an existing record,using search options (By Isbn,By Book Name,By Writer Name) he/she could find and see in table below easily.

Searching By ISBN,

Library Automation

Main Menu

01:50:35 Admin 08.06.2008

Book Registration

Search By ISBN Search By Book Name Search By Writer Name

111

ISBN Number 111 Language turkish

Book Name Dudaktan Kalbe Publisher yağmur kitapevi

Writer Name Reşat Nuri Güntekin Date 12.04.2000

Book Status Not Available Location b14

Subject novel

Navigation buttons: Previous, Previous, Next, Next, Add, Delete, Up, Down, Refresh, Repeat

Isbn	Bookname	Borrower
111	Dudaktan Kalbe	342

Navigation: Previous, Next

Figure 4.20 Add, Delete, Update Book Searching By ISBN

Searching By Book Name,

Library Automation

Main Menu

01:54:01 **Admin** **08.06.2008**

Book Registration

Search By ISBN Search By Book Name Search By Writer Name

ISBN Number Language

Book Name Publisher

Writer Name Date

Book Status Location

Subject

Userid	Username	Usersurname

Figure 4.21 Add, Delete, Update Book Searching By Book Name

Searching By Writer Name,

The screenshot shows a web application window titled "Library Automation" with a sub-header "Main Menu". The interface is for an "Admin" user, with a timestamp of "18:37:58" and a date of "08.06.2008". The main section is titled "Book Registration".

There are three search options: "Search By ISBN", "Search By Book Name", and "Search By Writer Name". The "Search By Writer Name" option is selected, with the text "T|" entered in the search field.

Below the search options, there are several input fields for book details:

- ISBN Number: 789
- Language: turkish
- Book Name: Anna Karenina
- Publisher: gündüz kitabevi
- Writer Name: Tolstoy
- Date: 01.11.1998
- Book Status: Available (dropdown menu)
- Location: e.11.22
- Subject: novel

Below the input fields, there is a row of navigation buttons: a left arrow, a right arrow, a double left arrow, a double right arrow, a plus sign, a minus sign, an up arrow, a down arrow, a refresh button, and a search button.

At the bottom, there is a table with three columns: "Userid", "Username", and "Usersurname". The table is currently empty.

Figure 4.22 Add, Delete, Update Book Searching By Writer

4.7 Search

In this form user could search Book, User and Borrower.

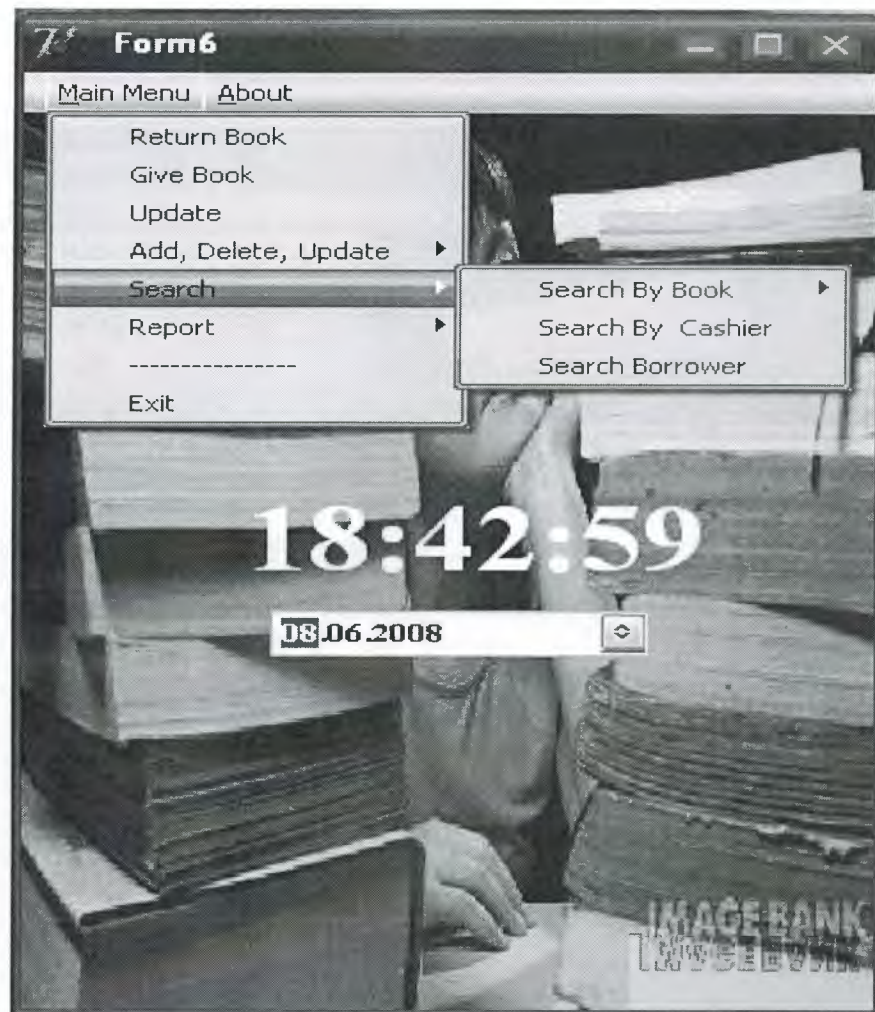


Figure 4.23 Search Option

4.7.1 Search Book

In this form, user can search borrowed books according to its isbn or name.

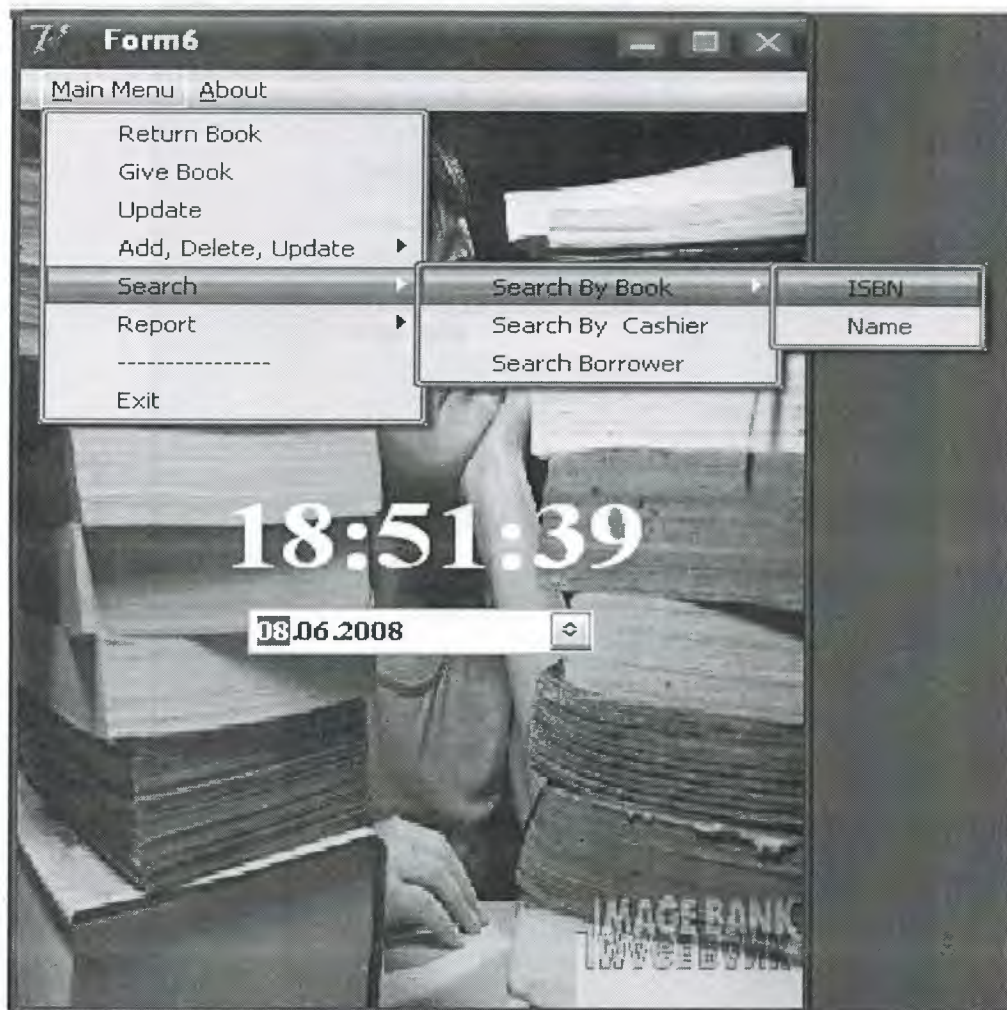


Figure 4.24 Search Book By ISBN

This form allows to user to search borrowed books according to ISBN given between two dates. Table will show the book ISBN, name, location and statue.

Library Automation

Main Menu

Search By ISBN

Search By Date Of Start
05.06.2008

Search By Date Of End
05.06.2008

ISBN Number
111

Book Name

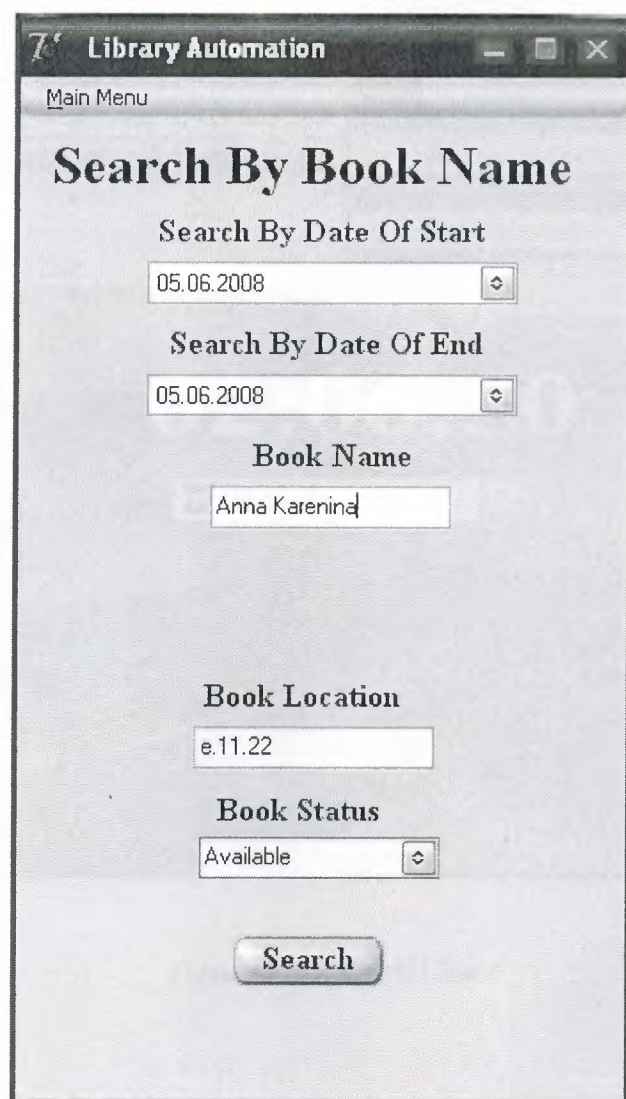
Isbn	Bookname
▶ 111	Dudaktan Kalbe

< ||| >

Search

Figure 4.25 Search Book by ISBN

This form allows to user to search borrowed books according to book name between two dates.



The screenshot shows a window titled "Library Automation" with a "Main Menu" link. The main heading is "Search By Book Name". Below this, there are two date selection fields: "Search By Date Of Start" and "Search By Date Of End", both containing the date "05.06.2008". The "Book Name" field contains the text "Anna Karenina". The "Book Location" field contains "e.11.22". The "Book Status" dropdown menu is set to "Available". A "Search" button is located at the bottom of the form.

Field	Value
Search By Date Of Start	05.06.2008
Search By Date Of End	05.06.2008
Book Name	Anna Karenina
Book Location	e.11.22
Book Status	Available

Figure 4.26 Search Book By Name

4.7.2 Search User

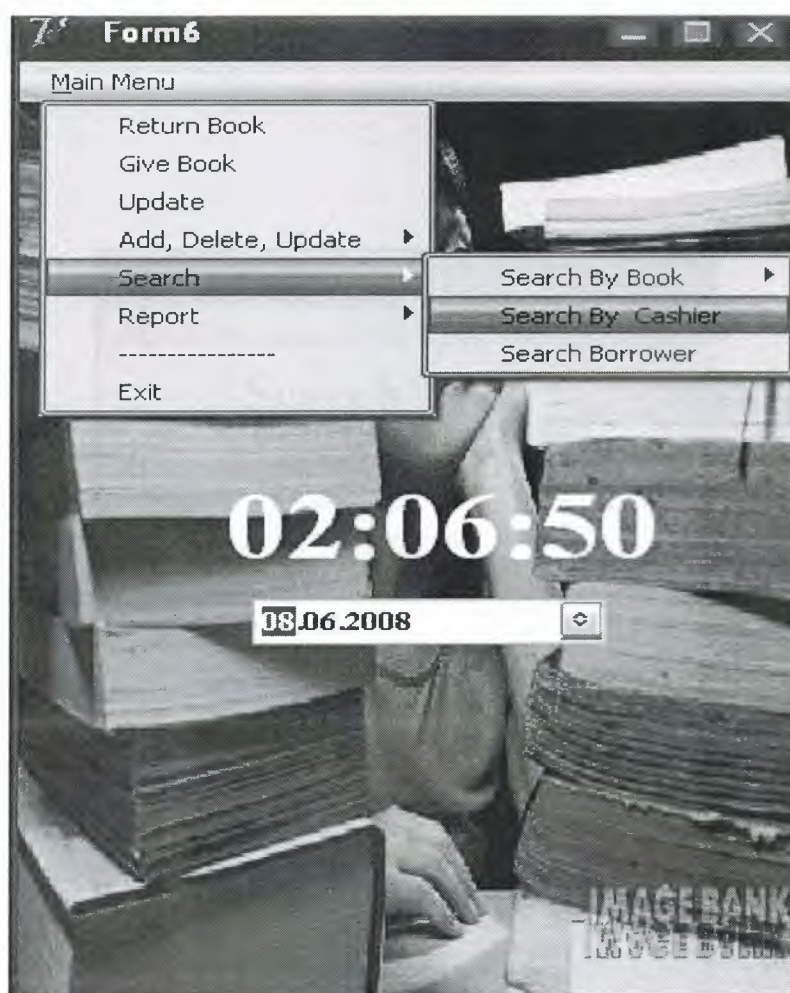


Figure 4.27 Search User

This form allows to user to search transactions have been done by that user according to his/her id in between two dates. Table will show the user ID, name, surname, borrowed book, rent date and return dates.

Library Automation

Main Menu

Search User

Search By Date Of Start

05.06.2008

Search By Date Of End

05.06.2008

User ID

234

userid	username
* 234	malik

< ||| >

Search

Figure 4.28 Search User By ID

4.7.3 Search Borrower

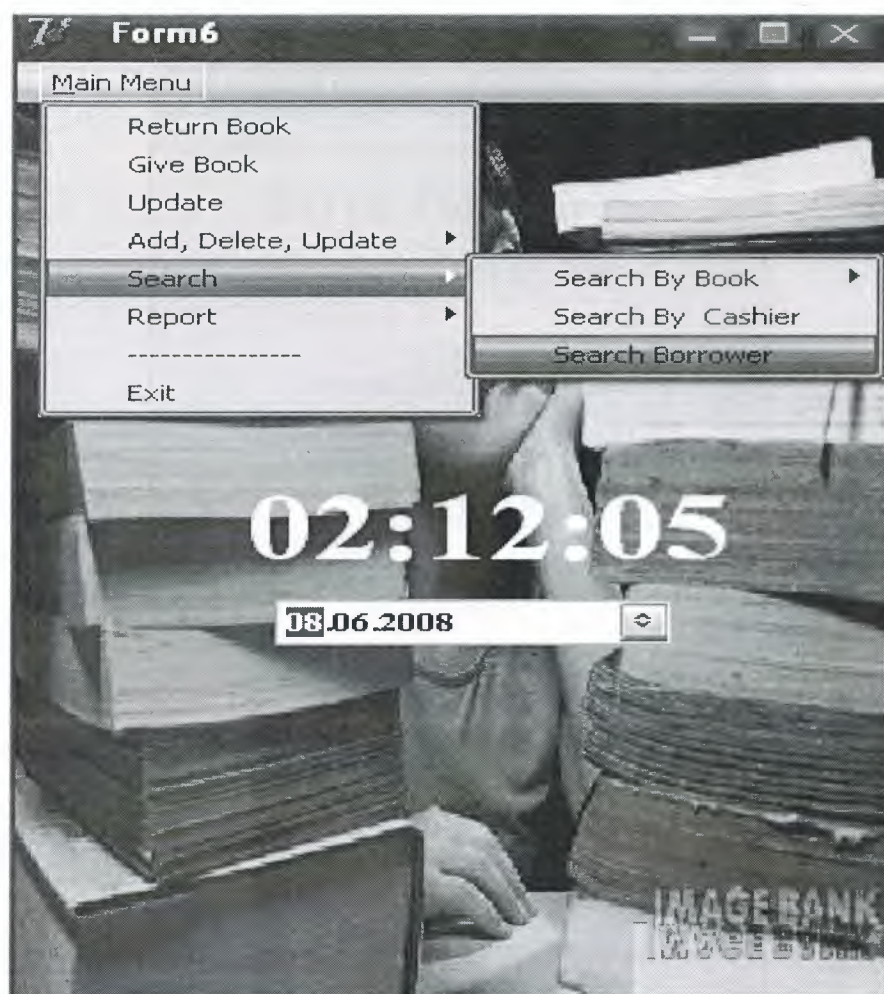


Figure 4.29 Search Borrower

This form allows to user to search books have been taken by borrower according to his/her ID between two dates.

Library Automation

Main Menu

Serch Borrower

Search By Date Of Start

05.06.2008

Search By Date Of End

05.06.2008

Borrower ID

342

borrowerid	borrowername
▶ 342	burcin

< ||| >

Search

Figure 4.30 Search Borrower By ID

4.8 Report

In this form user could report and print (if it is necessary) Book, User and Borrower informations.

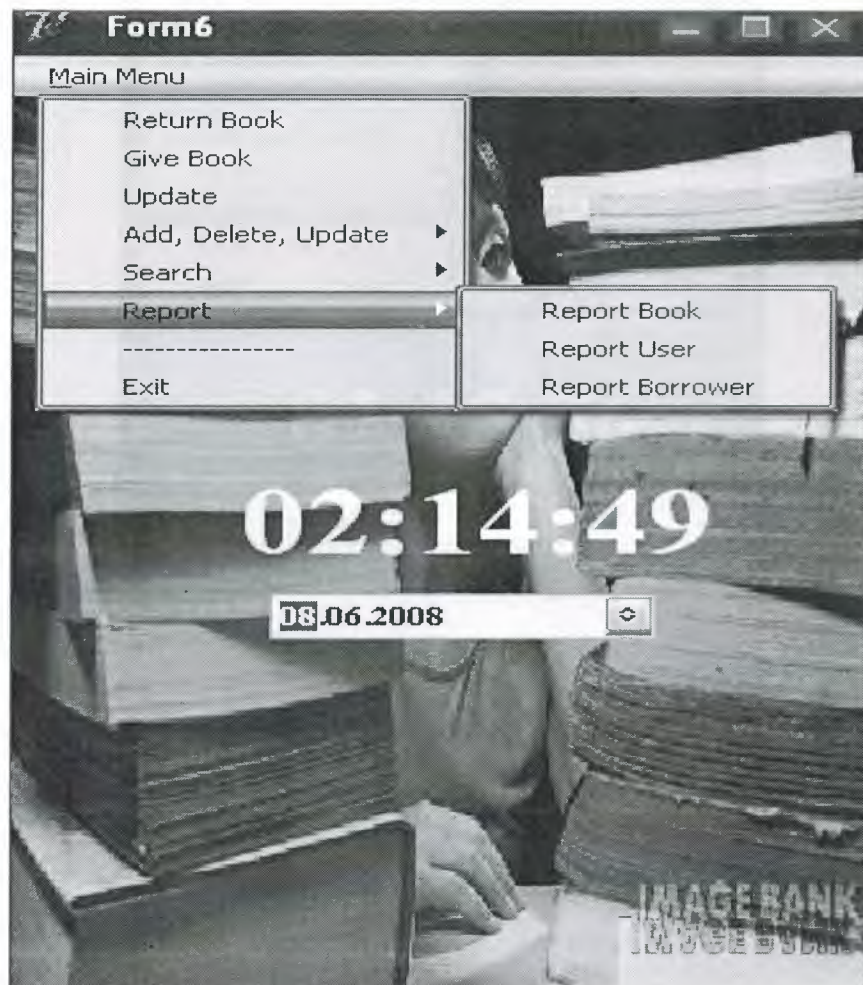


Figure 4.31 Report

4.8.1 Report Book

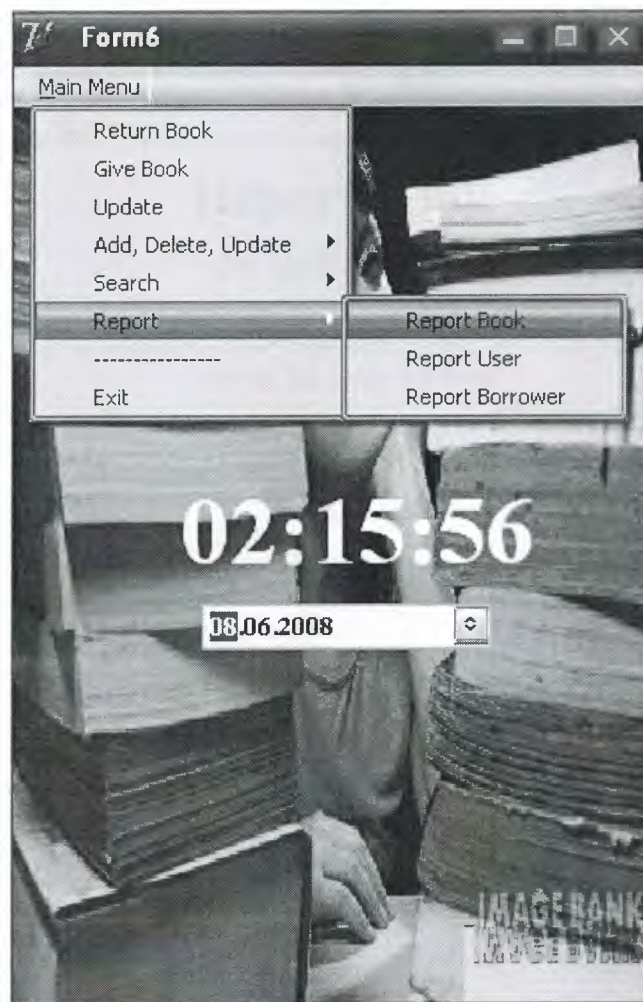


Figure 4.32 Report Book

In this section, program can report and print optionally books which have been borrowed between given two dates.

Library Automation

Main Menu

Report Book

Search By Date Of Start

25.06.2008

Search By Date Of End

25.06.2008

Book Name

Oracle

Search

Report

Print

Figure 4.33 Report Book by Name

Rent Date	Return Date	Book Name	ID	Borrower Name		ID	Cashier Name
20.06.2008	25.06.2008	Oracle	234	Murat	Karaogul	987	Halime

Figure 4.34 List of Books

Print

Printer
Name:

Where: ☒ Print to file

Pages
☒ All
☐ Current page
☐ Pages:
 Enter page numbers and/or page ranges, separated by commas. For example, 1,3,5-12

Copies
 Number of copies:
☒ Collate

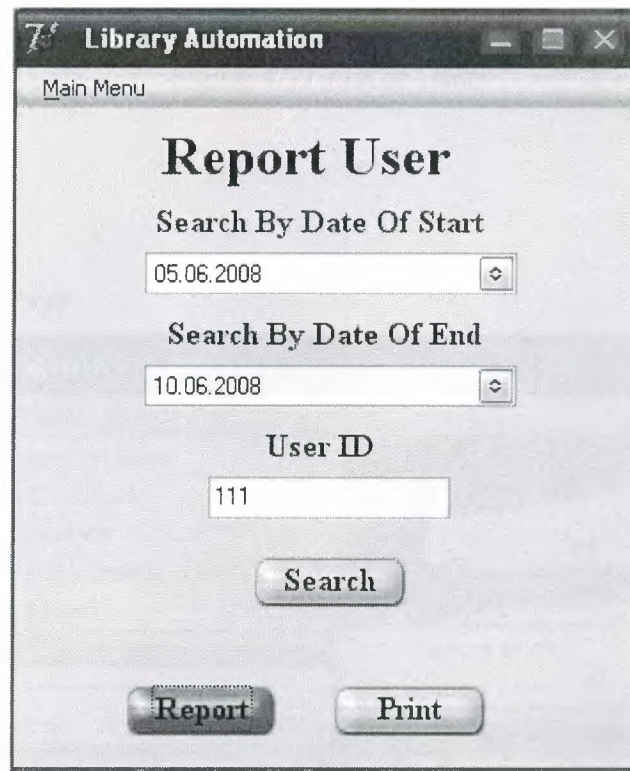
Other
 Print:
 Order:
 Duplex:

Print mode
 > Default
 Print on sheet:

Figure 4.35 Print Book

4.8.2 Report User

In this section, program can report and print optionally users which have done transactions between given two dates.



Library Automation

Main Menu

Report User

Search By Date Of Start

05.06.2008

Search By Date Of End

10.06.2008

User ID

111

Search

Report Print

Figure 4.36 Report User

User ID	User Name	Rent Date	Book Name	Return Date
111	Ekrem	Yilmaz		

Figure 4.37 List of Users

4.8.2 Report Borrower

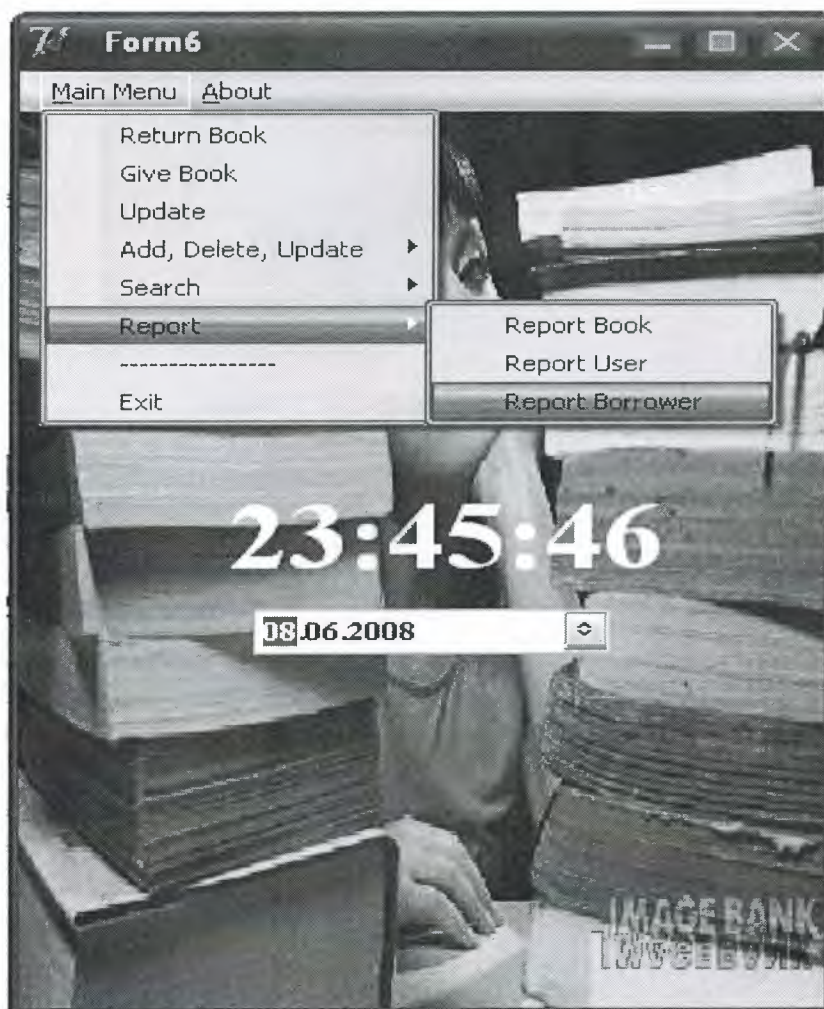


Figure 4.38 Report Borrower

In this window user can report borrowers which borrowed book in given two dates.

7 Library Automation

Main Menu

Report Borrower

Search By Date Of Start

05.06.2008

Search By Date Of End

20.06.2008

Borrower ID

234

Search

Report Print

Figure 4.39 Report Borrower by ID

Borrower ID	Borrower Name	Rent Date	Book Name	Return Date
234	Murat Karaogul	20.06.2008	Oracle	25.06.2008

Figure 4.40 List Borrowers

4.9 About in Main Menu



Figure 4.41 About Option

In this About window, user can see the some information about the programmer and program.

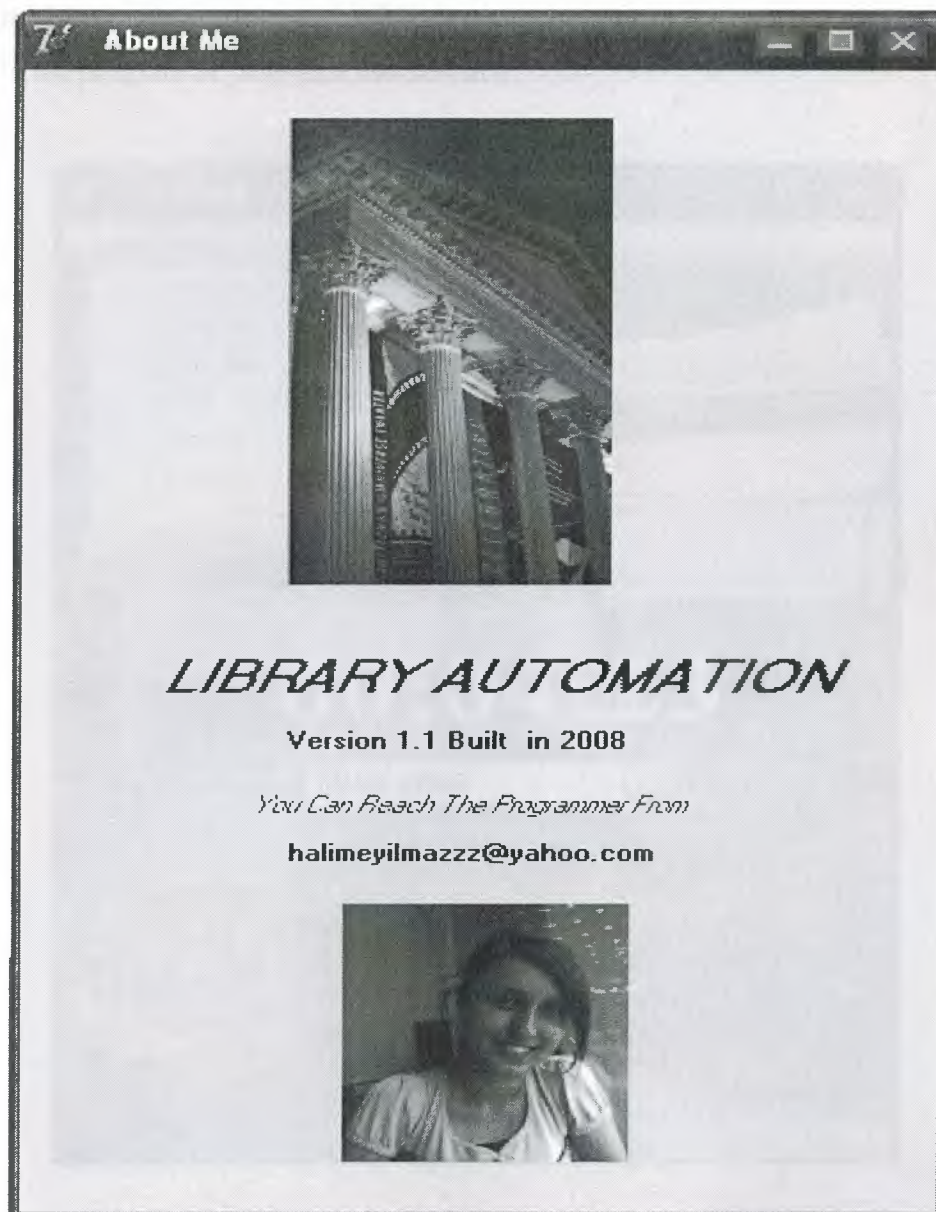


Figure 4.41 About Me

4.9 Exit

To close the program or page exit should click.

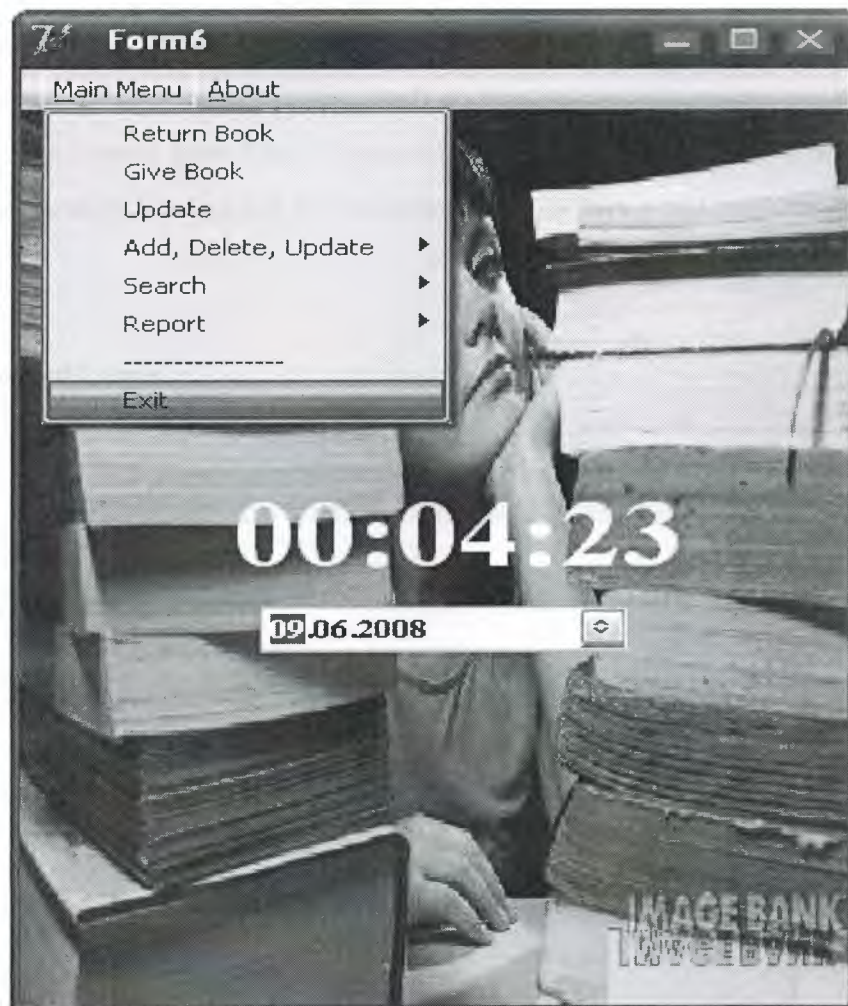


Figure 4.42 Exit

4.10 Other Users

As it is mentioned before there are five types of users:

Admin, User, Guest, Student, Staff. Each of the users has different hierarchic authorities. Admin and user can use all the options in the program, but user can not change the admin's password. Student, Staff and Guest can only search book and except guest they can borrow book. The difference between student and staff is book borrowing duration. Student can borrow a book for 10 days, guest can borrow a book for 30 days.

4.10.1 Student Login



The screenshot shows a window titled "LIBRARY AUTOMATION PROGRAM". Inside the window, the text "Please Enter Your User Name" is displayed above a text input field containing the word "Student". Below this, the text "Please Enter Your User Password" is displayed above a password input field containing three asterisks "xxx". At the bottom of the form is a button labeled "Enter".

Figure 4..43 Student Login

4.10.2 Staff Login

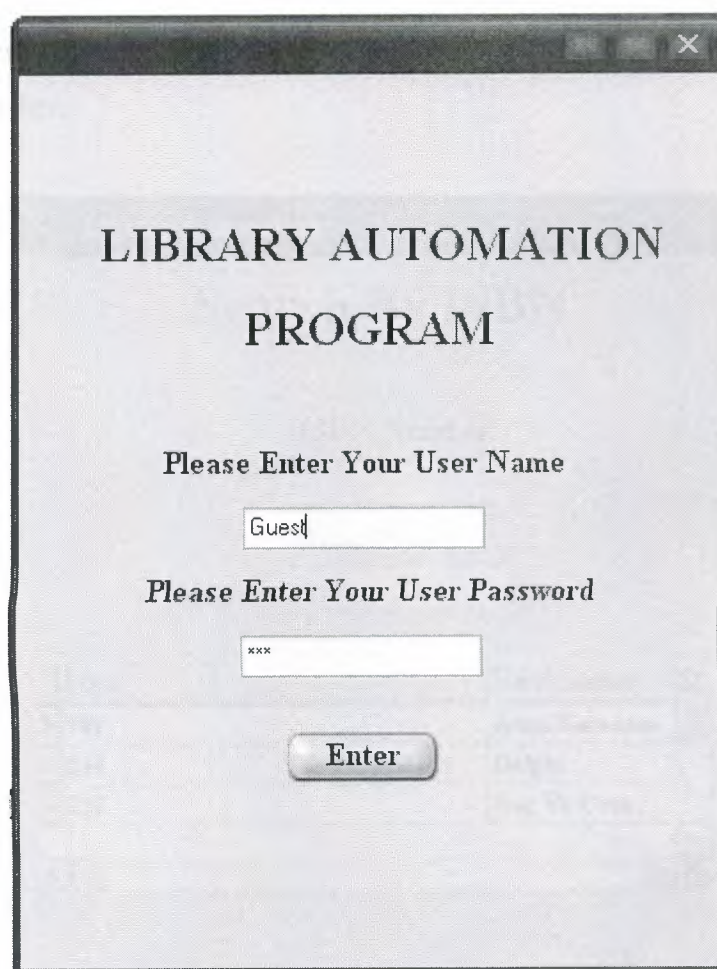


A screenshot of a software window titled "LIBRARY AUTOMATION PROGRAM". The window has a standard Windows-style title bar with a close button (X) in the top right corner. The main content area is light gray and contains the following elements:

- The title "LIBRARY AUTOMATION PROGRAM" in a large, bold, black serif font, centered at the top.
- The instruction "Please Enter Your User Name" in a smaller, bold, black serif font, centered below the title.
- A text input field containing the word "Staff" in a black serif font.
- The instruction "Please Enter Your User Password" in a smaller, bold, black serif font, centered below the username field.
- A password input field containing three asterisks "xxx" in a black serif font.
- A rounded rectangular button with the word "Enter" in a black serif font, centered below the password field.

Figure 4.44 Staff Login

4.10.3 Guest Login



A screenshot of a software window titled "LIBRARY AUTOMATION PROGRAM". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is light gray and contains the following elements:

- The title "LIBRARY AUTOMATION PROGRAM" in a large, bold, black serif font, centered at the top.
- The instruction "Please Enter Your User Name" in a smaller, bold, black serif font, centered below the title.
- A text input field containing the text "Guest".
- The instruction "Please Enter Your User Password" in a smaller, bold, black serif font, centered below the username field.
- A password input field containing three asterisks "xxx".
- A rounded rectangular button with the text "Enter" in a bold, black serif font, centered below the password field.

Figure 4.46 Guest Login

The authority of the student, staff and guest is the same and when the Student, Staff and Guest login to program can see just two forms. They can just search book.

4.10.4 Search Book

In this form users can search books by ISBN or name and can see the book ISBN, name, writer and publisher.

Isbn	Bookname
789	Anna Karenina
234	Delphi
222	Suç Ve Ceza

Figure 4.47 Search Book

After learning these information if the user wants to borrow the book, he or she should click Statue and Location button to see the book situation.

4.10.5 Statue And Location

In this form user will see the book location to take the book and the statue. If it is available user can borrow it. Also close button closes the window.

Library Automation

Search By Book Name

Book Name

Book Location

Book Status

Search By ISBN_Name

Close

Figure 4.48 Statue And Location Form

CONCLUSION

Particular needs and demands of a library made this software program possible. The features of the program that the library needs as follows:

- To give the books to the borrowers
- To return book which is taken before
- To register the books at the library database
- To see the books which library has
- To list the borrowers who take book
- To list the users who give book
- To list the books which are taken
- To keep the transaction details under control

Borland Delphi 6.0 is used for the software that will answer the needs of a big library. This program is chosen because it is highly secure and easy software to use in terms of both the user and the programmer when trying to fulfill the needs and demands of the library. Paradox is the database that is used for this software as it is the most applicable database for Delphi.

This software holds the all necessary aspects to support the heavy work load of a library and it is being tested in a library department and there have been no problems faced with the program coping with the library's load so far. The program requires further development to handle remote access perhaps through internet to facilitate remote access by users and borrowers for better communications.

REFERENCES

- [1] www.delphiturk.com
- [2] www.yazilim.com
- [3] www.programlama.com
- [4] www.wikipedia.com
- [5] www.bilisinterimleri.com
- [6] www.paradoxplaza.com
- [7] www.paradox.gen.tr
- [8] Karagülle, İhsan, Delphi 7.0, Türkmen Printing House, İstanbul, 2001

APPENDIX

Login Form

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, DBTables, StdCtrls, Mask, DBCtrls;

type
  TForm1 = class(TForm)
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    DBEdit7: TDBEdit;
    DBEdit8: TDBEdit;
    DBEdit9: TDBEdit;
    DBEdit10: TDBEdit;
    Query1: TQuery;
    Query2: TQuery;
    Query3: TQuery;
    Query4: TQuery;
    Query5: TQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    DataSource4: TDataSource;
    DataSource5: TDataSource;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
  private
    { Private declarations }
  public
```



```

    { Public declarations }
end;

var
    Form1: TForm1;

implementation

uses Unit2, Unit4, Unit5, Unit6, Unit10, Unit13, Unit20, Unit3;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    if (edit1.text=dbedit1.Text) and (edit2.text=dbedit2.Text) then
    begin
        form2.label1.caption:='Admin';
        form4.label1.caption:='Admin';
        form5.label1.caption:='Admin';
        form3.label12.Caption:='Admin';
        form6.Show;
    end
    else
    if (edit1.text=dbedit3.Text) and (edit2.text=dbedit4.Text) then
    begin
        form2.label1.caption:='User';
        form4.label1.caption:='User';
        form5.label1.caption:='User';
        form3.label12.Caption:='User';

        form6.Show;
    end
    else

    if (edit1.text=dbedit5.Text) and (edit2.text=dbedit6.Text) then
    begin
        form2.label1.caption:='guest';
        form4.label1.caption:='guest';
        form5.label1.caption:='guest';
        form20.Show;

    end
    else

    if (edit1.text=dbedit7.Text) and (edit2.text=dbedit8.Text) then
    begin
        form2.label1.caption:='student';
        form4.label1.caption:='student';
        form5.label1.caption:='student';
        form20.Show;
    end
end

```

```

end
else
if (edit1.text=dbedit9.Text) and (edit2.text=dbedit10.Text) then
begin
form2.label1.caption:='staff';
form4.label1.caption:='staff';
form5.label1.caption:='staff';
form20.Show;

end
else

showmessage('Wrong User Name Or Password');
edit1.Text:='';
edit2.Text:='';
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
edit1.Text:='';
edit2.Text:='';
dbedit1.Visible:=false;
dbedit2.Visible:=false;
dbedit3.Visible:=false;
dbedit4.Visible:=false;
dbedit5.Visible:=false;
dbedit6.Visible:=false;
dbedit7.Visible:=false;
dbedit8.Visible:=false;
dbedit9.Visible:=false;
dbedit10.Visible:=false;

end;
procedure TForm1.Edit2Change(Sender: TObject);
begin
edit2.PasswordChar:='*';
end;
end.

```

Main Menu

```
unit Unit6;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, StdCtrls, jpeg, ExtCtrls, ComCtrls;
```

```

type
  TForm6 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    DateTimePicker1: TDateTimePicker;
    Label2: TLabel;
    Timer1: TTimer;
    Timer2: TTimer;
    MainMenu1: TMainMenu;
    MainMenu2: TMenuItem;
    ReturnBook1: TMenuItem;
    GiveBook1: TMenuItem;
    Update1: TMenuItem;
    Add1: TMenuItem;
    AddUser1: TMenuItem;
    AddBook1: TMenuItem;
    Search1: TMenuItem;
    SerchBook1: TMenuItem;
    ISBN1: TMenuItem;
    Cashier1: TMenuItem;
    SerchByCashier1: TMenuItem;
    Report1: TMenuItem;
    ReportByCashier1: TMenuItem;
    reportByBorrowers1: TMenuItem;
    ReportBorrower1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    SearchBorrower1: TMenuItem;
    About1: TMenuItem;
    procedure Timer2Timer(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure ReportByCashier1Click(Sender: TObject);
    procedure reportByBorrowers1Click(Sender: TObject);
    procedure ReportBorrower1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure ISBN1Click(Sender: TObject);
    procedure Cashier1Click(Sender: TObject);
    procedure SerchByCashier1Click(Sender: TObject);
    procedure SearchBorrower1Click(Sender: TObject);
    procedure Update1Click(Sender: TObject);
    procedure AddBook1Click(Sender: TObject);
    procedure AddUser1Click(Sender: TObject);
    procedure GiveBook1Click(Sender: TObject);
    procedure ReturnBook1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure Image1Click(Sender: TObject);
  private
    { Private declarations }
  public

```



```

    { Public declarations }
end;

var
    Form6: TForm6;

implementation

uses Unit7, Unit8, Unit9, Unit1, Unit10, Unit13, Unit11, Unit12, Unit4,
    Unit2, Unit3, Unit15, Unit14, Unit22;

{$R *.dfm}

procedure TForm6.Timer2Timer(Sender: TObject);
begin
    DateTimePicker1.DateTime := Now;
end;

procedure TForm6.Timer1Timer(Sender: TObject);
begin
    label2.caption:=timetostr(time);
end;

procedure TForm6.ReportByCashier1Click(Sender: TObject);
begin

    form7.show;
end;

procedure TForm6.reportByBorrowers1Click(Sender: TObject);
begin

    form8.show;
end;

procedure TForm6.ReportBorrower1Click(Sender: TObject);
begin

    form9.show;
end;

procedure TForm6.Exit1Click(Sender: TObject);
begin
    form1.close;
end;

procedure TForm6.ISBN1Click(Sender: TObject);
begin
    form10.show;
end;

```

```

procedure TForm6.Cashier1Click(Sender: TObject);
begin
form13.show;
end;

procedure TForm6.SerchByCashier1Click(Sender: TObject);
begin
form11.show;
end;

procedure TForm6.SearchBorrower1Click(Sender: TObject);
begin
form12.show;
end;

procedure TForm6.Update1Click(Sender: TObject);
begin
form4.show;
end;

procedure TForm6.AddBook1Click(Sender: TObject);
begin
form2.show;
end;

procedure TForm6.AddUser1Click(Sender: TObject);
begin
form3.show;
end;

procedure TForm6.GiveBook1Click(Sender: TObject);
begin
form15.show;
end;

procedure TForm6.ReturnBook1Click(Sender: TObject);
begin
form14.show;
end;

procedure TForm6.About1Click(Sender: TObject);
begin
Form22.show;
end;

procedure TForm6.Image1Click(Sender: TObject);
begin

end;

```

end.

RETURN BOOK

unit Unit14;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Menus, Mask, DBCtrls, DB, DBTables;

type

TForm14 = class(TForm)
 Label11: TLabel;
 Label12: TLabel;
 Edit1: TEdit;
 Edit2: TEdit;
 Label3: TLabel;
 Label4: TLabel;
 Label5: TLabel;
 Label6: TLabel;
 Label10: TLabel;
 Label14: TLabel;
 Label15: TLabel;
 Label1: TLabel;
 Label2: TLabel;
 DBEdit1: TDBEdit;
 DBEdit2: TDBEdit;
 DBEdit3: TDBEdit;
 DBEdit4: TDBEdit;
 DBEdit5: TDBEdit;
 DBEdit6: TDBEdit;
 DBEdit7: TDBEdit;
 DBEdit8: TDBEdit;
 DBEdit9: TDBEdit;
 Label7: TLabel;
 DBEdit10: TDBEdit;
 Label8: TLabel;
 DBEdit11: TDBEdit;
 MainMenu1: TMainMenu;
 MainMenu2: TMenuItem;
 ReturnBook1: TMenuItem;
 GiveBook1: TMenuItem;
 Update1: TMenuItem;
 Add1: TMenuItem;


```

AddUser1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
SearchByBorrower1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
DBComboBox1: TDBComboBox;
Label9: TLabel;
Button1: TButton;
Query1: TQuery;
DataSource1: TDataSource;
procedure AddUser1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form14: TForm14;

implementation

uses Unit3, Unit7, Unit8, Unit9, Unit10, Unit13, Unit11, Unit12, Unit2,
    Unit4, Unit15;

```

{ \$R *.dfm }

```
procedure TForm14.AddUser1Click(Sender: TObject);
begin
form14.close;
form3.show;
end;
```

```
procedure TForm14.Exit1Click(Sender: TObject);
begin
form14.Close;
end;
```

```
procedure TForm14.ReportByCashier1Click(Sender: TObject);
begin
form14.close;
form7.show;
end;
```

```
procedure TForm14.reportByBorrowers1Click(Sender: TObject);
begin
form14.close;
form8.show;
end;
```

```
procedure TForm14.ReportBorrower1Click(Sender: TObject);
begin
form14.close;
form9.show;
end;
```

```
procedure TForm14.ISBN1Click(Sender: TObject);
begin
form14.close;
form10.show;
end;
```

```
procedure TForm14.Cashier1Click(Sender: TObject);
begin
form14.close;
form13.show;
end;
```

```
procedure TForm14.SerchByCashier1Click(Sender: TObject);
begin
form14.close;
form11.show;
end;
```

```
procedure TForm14.SearchByBorrower1Click(Sender: TObject);
```

```

begin
form14.close;
form12.show;
end;

procedure TForm14.AddBook1Click(Sender: TObject);
begin
form14.close;
form2.show;
end;

procedure TForm14.Update1Click(Sender: TObject);
begin
form14.close;
form4.show;
end;

procedure TForm14.GiveBook1Click(Sender: TObject);
begin
form14.close;
form15.show;
end;

procedure TForm14.FormCreate(Sender: TObject);
begin

Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
edit2.Text:='';

end;

procedure TForm14.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Isbn like'+#39+(edit1.text)+'%'+#39);
query1.Open;

end;

procedure TForm14.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;

```



```

query1.sql.add('select * from halime where BookName
like'+#39+(edit2.text)+'%'+#39);
query1.Open;

end;

procedure TForm14.Button1Click(Sender: TObject);
begin
query1.Append;
end;

procedure TForm14.FormActivate(Sender: TObject);
begin

query1.Insert;
end;

end.

```

GIVE BOOK

```

unit Unit15;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, DB, DBTables, StdCtrls, DBCtrls, Grids, DBGrids,
  ExtCtrls, Mask;

type
  TForm15 = class(TForm)
    MainMenu1: TMainMenu;
    MainMenu2: TMenuItem;
    ReturnBook1: TMenuItem;
    GiveBook1: TMenuItem;
    Update1: TMenuItem;
    Add1: TMenuItem;
    AddUser1: TMenuItem;
    AddBook1: TMenuItem;
    Search1: TMenuItem;
    SerchBook1: TMenuItem;
    ISBN1: TMenuItem;
    Cashier1: TMenuItem;
    SerchByCashier1: TMenuItem;
    SearchByBorrower1: TMenuItem;
    Report1: TMenuItem;
    ReportByCashier1: TMenuItem;
    reportByBorrowers1: TMenuItem;

```

```

ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
Label3: TLabel;
Label4: TLabel;
Label9: TLabel;
Label8: TLabel;
Label2: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label10: TLabel;
Label14: TLabel;
Label15: TLabel;
Label16: TLabel;
GroupBox1: TGroupBox;
Label1: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBMemo1: TDBMemo;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
DBEdit7: TDBEdit;
DBEdit8: TDBEdit;
DBEdit9: TDBEdit;
Edit4: TEdit;
DBComboBox1: TDBComboBox;
Label17: TLabel;
Label18: TLabel;
DBEdit10: TDBEdit;
DBEdit11: TDBEdit;
Query1: TQuery;
DataSource1: TDataSource;
Button1: TButton;
procedure Update1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);

```

```

    procedure reportByBorrowers1Click(Sender: TObject);
    procedure ReportBorrower1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure ReturnBook1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form15: TForm15;

implementation

uses Unit4, Unit3, Unit2, Unit10, Unit13, Unit11, Unit12, Unit7, Unit8,
    Unit9, Unit14;

{$R *.dfm}

procedure TForm15.Update1Click(Sender: TObject);
begin
    form15.close;
    form4.show;
end;

procedure TForm15.AddUser1Click(Sender: TObject);
begin
    form15.close;
    form3.show;
end;

procedure TForm15.AddBook1Click(Sender: TObject);
begin
    form15.close;
    form2.show;
end;

procedure TForm15.ISBN1Click(Sender: TObject);
begin
    form15.close;
    form10.show;
end;

```



```

procedure TForm15.Cashier1Click(Sender: TObject);
begin
form15.close;
form13.show;
end;

procedure TForm15.SerchByCashier1Click(Sender: TObject);
begin
form15.close;
form11.show;
end;

procedure TForm15.SearchByBorrower1Click(Sender: TObject);
begin
form15.close;
form12.show;
end;

procedure TForm15.ReportByCashier1Click(Sender: TObject);
begin
form15.close;
form7.show;
end;

procedure TForm15.reportByBorrowers1Click(Sender: TObject);
begin
form15.close;
form8.show;
end;

procedure TForm15.ReportBorrower1Click(Sender: TObject);
begin
form15.close;
form9.show;
end;

procedure TForm15.Exit1Click(Sender: TObject);
begin
form15.close;
end;

procedure TForm15.ReturnBook1Click(Sender: TObject);
begin
form15.close;
form14.show;
end;

procedure TForm15.FormCreate(Sender: TObject);
begin

```

```

Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
edit2.Text:='';
edit3.Text:='';
edit4.Text:='';
end;

```

```

procedure TForm15.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Isbn like'+#39+(edit1.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm15.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where bookname like'+#39+(edit2.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm15.Edit3Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Borrowername
like'+#39+(edit3.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm15.Edit4Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Borrowersurname
like'+#39+(edit4.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm15.FormActivate(Sender: TObject);
begin
query1.Insert;
end;

```

```

procedure TForm15.Button1Click(Sender: TObject);

```

```
begin
query1.Append;
end;
end.
```

UPDATE

```
unit Unit4;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, StdCtrls, ExtCtrls, ComCtrls, DBCtrls, Mask, DB, DBTables,
Grids, DBGrids;
```

```
type
```

```
TForm4 = class(TForm)
  GroupBox1: TGroupBox;
  Label1: TLabel;
  Label11: TLabel;
  Label12: TLabel;
  Label13: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label9: TLabel;
  Label8: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  DBEdit1: TDBEdit;
  DBEdit2: TDBEdit;
  DBEdit3: TDBEdit;
  DBEdit4: TDBEdit;
  DBMemo1: TDBMemo;
  DBEdit5: TDBEdit;
  Label6: TLabel;
  Label7: TLabel;
  DBEdit6: TDBEdit;
  DBEdit7: TDBEdit;
  DBEdit8: TDBEdit;
  DBEdit9: TDBEdit;
  DBNavigator1: TDBNavigator;
  Label15: TLabel;
  Edit4: TEdit;
  MainMenu1: TMainMenu;
  MainMenu2: TMenuItem;
  ReturnBook1: TMenuItem;
  GiveBook1: TMenuItem;
  Update1: TMenuItem;
```



```

Add1: TMenuItem;
AddUser1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
SearchByBorrower1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
Label16: TLabel;
DBComboBox1: TDBComboBox;
DBEdit10: TDBEdit;
DBEdit11: TDBEdit;
Label18: TLabel;
Label10: TLabel;
Label14: TLabel;
Label17: TLabel;
Label2: TLabel;
Label5: TLabel;
Query1: TQuery;
DataSource1: TDataSource;
procedure BookRegistration2Click(Sender: TObject);
procedure ransaction21Click(Sender: TObject);
procedure EX1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
private
{ Private declarations }

```

```

public
  { Public declarations }
end;

var
  Form4: TForm4;

implementation

uses Unit2, Unit5, Unit1, Unit10, Unit13, Unit11, Unit12, Unit7, Unit8,
  Unit9, Unit15, Unit14, Unit3;

{$R *.dfm}

procedure TForm4.BookRegistration2Click(Sender: TObject);
begin
  form4.close;
  form2.show;
end;

procedure TForm4.ransaction21Click(Sender: TObject);
begin
  form4.close;
  form5.show;
end;

procedure TForm4.EX1Click(Sender: TObject);
begin
  form1.close;
end;

procedure TForm4.Exit1Click(Sender: TObject);
begin
  form4.close;
end;

procedure TForm4.ISBN1Click(Sender: TObject);
begin
  form4.Close;
  form10.show;
end;

procedure TForm4.Cashier1Click(Sender: TObject);
begin
  form4.Close;
  form13.show;
end;

procedure TForm4.SerchByCashier1Click(Sender: TObject);
begin

```

```
form4.Close;  
form11.show;  
end;
```

```
procedure TForm4.SearchByBorrower1Click(Sender: TObject);  
begin  
form4.Close;  
form12.show;  
end;
```

```
procedure TForm4.ReportByCashier1Click(Sender: TObject);  
begin  
form4.Close;  
form7.show;  
end;
```

```
procedure TForm4.reportByBorrowers1Click(Sender: TObject);  
begin  
form4.Close;  
form8.show;  
end;
```

```
procedure TForm4.ReportBorrower1Click(Sender: TObject);  
begin  
form4.Close;  
form9.show;  
end;
```

```
procedure TForm4.AddBook1Click(Sender: TObject);  
begin  
form4.Close;  
form2.Show;  
end;
```

```
procedure TForm4.GiveBook1Click(Sender: TObject);  
begin  
form4.Close;  
form15.show;  
end;
```

```
procedure TForm4.ReturnBook1Click(Sender: TObject);  
begin  
form4.close;  
form14.show;  
end;
```

```
procedure TForm4.FormCreate(Sender: TObject);  
begin  
Query1.DatabaseName:='STANDARD7';  
Query1.requestlive:=true;
```



```

query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
edit2.Text:='';
edit3.Text:='';
edit4.Text:='';

```

```

end;

```

```

procedure TForm4.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Isbn like'+#39+(edit1.text)+'%'+#39);
query1.Open;

```

```

end;

```

```

procedure TForm4.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where bookname like'+#39+(edit2.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm4.Edit3Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Borrowername
like'+#39+(edit3.text)+'%'+#39);
query1.Open;

```

```

end;

```

```

procedure TForm4.Edit4Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Borrowersurname
like'+#39+(edit4.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm4.FormActivate(Sender: TObject);
begin
query1.Insert;
end;

```

```

procedure TForm4.AddUser1Click(Sender: TObject);
begin
form3.show;
form4.Close;
end;

end.

```

ADD DELETE UPDATE USER

```

unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Mask, DBCtrls, SkinCaption, WinSkinData, DB, DBTables,
  Menus, ExtCtrls, Grids, DBGrids;

type
  TForm3 = class(TForm)
    SkinData1: TSkinData;
    SkinCaption1: TSkinCaption;
    MainMenu1: TMainMenu;
    MainMenu2: TMenuItem;
    ReturnBook1: TMenuItem;
    GiveBook1: TMenuItem;
    Update1: TMenuItem;
    Add1: TMenuItem;
    AddUser1: TMenuItem;
    AddBook1: TMenuItem;
    Search1: TMenuItem;
    SerchBook1: TMenuItem;
    ISBN1: TMenuItem;
    Cashier1: TMenuItem;
    SerchByCashier1: TMenuItem;
    SearchByBorrower1: TMenuItem;
    Report1: TMenuItem;
    ReportByCashier1: TMenuItem;
    reportByBorrowers1: TMenuItem;
    ReportBorrower1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;

```

```

Label12: TLabel;
Label3: TLabel;
DBEdit3: TDBEdit;
Label4: TLabel;
DBEdit4: TDBEdit;
Label21: TLabel;
DBEdit15: TDBEdit;
Label22: TLabel;
DBEdit16: TDBEdit;
Label23: TLabel;
DBEdit17: TDBEdit;
Label24: TLabel;
DBMemo2: TDBMemo;
Label25: TLabel;
DBEdit18: TDBEdit;
Label1: TLabel;
Label2: TLabel;
DBEdit1: TDBEdit;
DBGrid1: TDBGrid;
DBNavigator1: TDBNavigator;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
DBComboBox1: TDBComboBox;
Query1: TQuery;
DataSource1: TDataSource;
procedure AddBook1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }

```



```

end;

var
  Form3: TForm3;

implementation

uses Unit2, Unit4, Unit10, Unit13, Unit11, Unit12, Unit7, Unit8, Unit9,
  Unit15, Unit14, Unit17, Unit19, Unit18, Unit16;

{$R *.dfm}

procedure TForm3.AddBook1Click(Sender: TObject);
begin
  form3.Close;
  form2.show;
end;

procedure TForm3.Update1Click(Sender: TObject);
begin
  form3.Close;
  form4.show;
end;

procedure TForm3.Exit1Click(Sender: TObject);
begin
  form3.Close;
end;

procedure TForm3.ISBN1Click(Sender: TObject);
begin
  form3.Close;
  form10.show;
end;

procedure TForm3.Cashier1Click(Sender: TObject);
begin
  form3.Close;
  form13.show;
end;

procedure TForm3.SerchByCashier1Click(Sender: TObject);
begin
  form3.Close;
  form11.show;
end;

procedure TForm3.SearchByBorrower1Click(Sender: TObject);
begin
  form3.Close;

```

```
form12.show;  
end;
```

```
procedure TForm3.ReportByCashier1Click(Sender: TObject);  
begin  
form3.Close;  
form7.show;  
end;
```

```
procedure TForm3.reportByBorrowers1Click(Sender: TObject);  
begin  
form3.Close;  
form8.show;  
end;
```

```
procedure TForm3.ReportBorrower1Click(Sender: TObject);  
begin  
form3.Close;  
form9.show;  
end;
```

```
procedure TForm3.GiveBook1Click(Sender: TObject);  
begin  
form3.close;  
form15.show;  
end;
```

```
procedure TForm3.ReturnBook1Click(Sender: TObject);  
begin  
form3.close;  
form14.show;  
end;
```

```
procedure TForm3.FormCreate(Sender: TObject);  
begin  
Query1.DatabaseName:='STANDARD7';  
Query1.requestlive:=true;  
query1.SQL.Text:='select  
userid,username,usersurname,Phone,halime."password",mobilphone,address,email,statu  
s from halime';  
query1.Active:=true;  
edit1.Text:='';  
edit2.Text:='';  
edit3.Text:='';  
end;
```

```
procedure TForm3.Edit1Change(Sender: TObject);  
begin  
query1.close;  
query1.SQL.Clear;
```

```

query1.sql.add('select
userid,username,usersurname,Phone,halime."password",mobilphone,address,email,statu
s from halime where userid like'+#39+(edit1.text)+'%'+#39);
query1.Open;

```

```

end;

```

```

procedure TForm3.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select
userid,username,usersurname,Phone,halime."password",mobilphone,address,email,statu
s from halime where username like'+#39+(edit2.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm3.Edit3Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select
userid,username,usersurname,Phone,halime."password",mobilphone,address,email,statu
s from halime where usersurname like'+#39+(edit3.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm3.FormActivate(Sender: TObject);
begin
query1.Insert;
end;
end.

```

ADD DELETE UPDATE BOOK

```

unit Unit2;

```

```

interface

```

```

uses

```

```

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DBCtrls, Mask, Menus, ExtCtrls, ComCtrls, Grids,
  DBGrids, DB, DBTables;

```

```

type

```

```

  TForm2 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;

```

Label4: TLabel;
 Label5: TLabel;
 Label6: TLabel;
 Label7: TLabel;
 Label8: TLabel;
 DBEdit1: TDBEdit;
 DBEdit2: TDBEdit;
 DBEdit3: TDBEdit;
 DBEdit4: TDBEdit;
 DBMemo1: TDBMemo;
 DBEdit6: TDBEdit;
 Label10: TLabel;
 DateTimePicker1: TDateTimePicker;
 Timer1: TTimer;
 Timer2: TTimer;
 DBGrid1: TDBGrid;
 DBNavigator1: TDBNavigator;
 Label11: TLabel;
 Label12: TLabel;
 Label13: TLabel;
 Edit1: TEdit;
 Edit2: TEdit;
 Edit3: TEdit;
 MainMenu1: TMainMenu;
 MainMenu2: TMenuItem;
 ReturnBook1: TMenuItem;
 GiveBook1: TMenuItem;
 Update1: TMenuItem;
 Add1: TMenuItem;
 AddUser1: TMenuItem;
 AddBook1: TMenuItem;
 Search1: TMenuItem;
 SerchBook1: TMenuItem;
 ISBN1: TMenuItem;
 Cashier1: TMenuItem;
 SerchByCashier1: TMenuItem;
 SearchByBorrower1: TMenuItem;
 Report1: TMenuItem;
 ReportByCashier1: TMenuItem;
 reportByBorrowers1: TMenuItem;
 ReportBorrower1: TMenuItem;
 N1: TMenuItem;
 Exit1: TMenuItem;
 Label9: TLabel;
 DBEdit5: TDBEdit;
 Label14: TLabel;
 Label15: TLabel;
 DBEdit7: TDBEdit;
 DBComboBox1: TDBComboBox;
 Query1: TQuery;


```

DataSource1: TDataSource;
procedure Label1Click(Sender: TObject);
procedure ChangeAllPasswords1Click(Sender: TObject);
procedure EX1Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure Timer2Timer(Sender: TObject);
procedure ransaction11Click(Sender: TObject);
procedure ransaction21Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form2: TForm2;

implementation

uses Unit3, Unit1, Unit4, Unit5, Unit10, Unit13, Unit11, Unit12, Unit7,
  Unit8, Unit9, Unit15, Unit14;

{$R *.dfm}

procedure TForm2.Label1Click(Sender: TObject);
begin
  form3.show;
end;

procedure TForm2.ChangeAllPasswords1Click(Sender: TObject);
begin
  form3.show;
end;

```

```

procedure TForm2.EX1Click(Sender: TObject);
begin
form1.close;
end;

procedure TForm2.Timer1Timer(Sender: TObject);
begin
label10.caption:=timetostr(time);
end;

procedure TForm2.Timer2Timer(Sender: TObject);
begin
DateTimePicker1.DateTime := Now;
end;

procedure TForm2.ransaction11Click(Sender: TObject);
begin
form2.close;
form4.show;
end;

procedure TForm2.ransaction21Click(Sender: TObject);
begin
form2.close;
form5.show;
end;

procedure TForm2.Update1Click(Sender: TObject);
begin
form2.Close;
form4.Show;
end;

procedure TForm2.ISBN1Click(Sender: TObject);
begin
form2.Close;
form10.Show;
end;

procedure TForm2.Cashier1Click(Sender: TObject);
begin
form2.Close;
form13.Show;
end;

procedure TForm2.SerchByCashier1Click(Sender: TObject);
begin
form2.Close;
form11.Show;
end;

```

```

end;

procedure TForm2.SearchByBorrower1Click(Sender: TObject);
begin
form2.Close;
form12.Show;
end;

procedure TForm2.ReportByCashier1Click(Sender: TObject);
begin
form2.Close;
form7.Show;
end;

procedure TForm2.reportByBorrowers1Click(Sender: TObject);
begin
form2.Close;
form8.Show;
end;

procedure TForm2.ReportBorrower1Click(Sender: TObject);
begin
form2.Close;
form9.Show;
end;

procedure TForm2.Exit1Click(Sender: TObject);
begin
form2.Close;
end;

procedure TForm2.AddUser1Click(Sender: TObject);
begin
form2.Close;
form3.Show;
end;

procedure TForm2.GiveBook1Click(Sender: TObject);
begin
form2.close;
form15.show;
end;

procedure TForm2.ReturnBook1Click(Sender: TObject);
begin
form2.close;
form14.show;
end;

procedure TForm2.FormCreate(Sender: TObject);

```

```

begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
edit2.Text:='';
edit3.Text:='';

```

```

end;

```

```

procedure TForm2.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where Isbn like'+#39+(edit1.text)+'%'+#39);
query1.Open;

```

```

end;

```

```

procedure TForm2.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where bookname like'+#39+(edit2.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm2.Edit3Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where writername
like'+#39+(edit3.text)+'%'+#39);
query1.Open;
end;

```

```

procedure TForm2.FormActivate(Sender: TObject);
begin
query1.Insert;
end;
end.

```

SEARCH BOOK BY ISBN

```

unit Unit10;

```

```

interface

```

```

uses

```


Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Menus, Mask, DBCtrls, StdCtrls, ComCtrls, Grids, DBGrids, DB, DBTables;

type

```
TForm10 = class(TForm)
  Label1: TLabel;
  DateTimePicker1: TDateTimePicker;
  DateTimePicker2: TDateTimePicker;
  Label15: TLabel;
  Label16: TLabel;
  Label2: TLabel;
  Edit1: TEdit;
  Label4: TLabel;
  MainMenu1: TMainMenu;
  MainMenu2: TMenuItem;
  ReturnBook1: TMenuItem;
  GiveBook1: TMenuItem;
  Update1: TMenuItem;
  Add1: TMenuItem;
  AddUser1: TMenuItem;
  AddBook1: TMenuItem;
  Search1: TMenuItem;
  SerchBook1: TMenuItem;
  ISBN1: TMenuItem;
  Cashier1: TMenuItem;
  SerchByCashier1: TMenuItem;
  SearchByBorrower1: TMenuItem;
  Report1: TMenuItem;
  ReportByCashier1: TMenuItem;
  reportByBorrowers1: TMenuItem;
  ReportBorrower1: TMenuItem;
  N1: TMenuItem;
  Exit1: TMenuItem;
  DBGrid1: TDBGrid;
  Button1: TButton;
  Query1: TQuery;
  DataSource1: TDataSource;
  Edit2: TEdit;
  procedure Cashier1Click(Sender: TObject);
  procedure SerchByCashier1Click(Sender: TObject);
  procedure SearchByBorrower1Click(Sender: TObject);
  procedure Exit1Click(Sender: TObject);
  procedure ReportByCashier1Click(Sender: TObject);
  procedure reportByBorrowers1Click(Sender: TObject);
  procedure ReportBorrower1Click(Sender: TObject);
  procedure Update1Click(Sender: TObject);
  procedure AddBook1Click(Sender: TObject);
  procedure AddUser1Click(Sender: TObject);
  procedure GiveBook1Click(Sender: TObject);
```

```

    procedure ReturnBook1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form10: TForm10;

implementation

uses Unit13, Unit11, Unit12, Unit7, Unit8, Unit9, Unit4, Unit2, Unit3,
    Unit15, Unit14, Unit1;

{$R *.dfm}

procedure TForm10.Cashier1Click(Sender: TObject);
begin
    form10.Close;
    form13.show;
end;

procedure TForm10.SerchByCashier1Click(Sender: TObject);
begin
    form10.Close;
    form11.show;
end;

procedure TForm10.SearchByBorrower1Click(Sender: TObject);
begin
    form10.Close;
    form12.show;
end;

procedure TForm10.Exit1Click(Sender: TObject);
begin
    form1.Close;
end;

procedure TForm10.ReportByCashier1Click(Sender: TObject);
begin
    form10.Close;
    form7.show;
end;

```

```
procedure TForm10.reportByBorrowers1Click(Sender: TObject);
begin
form10.Close;
form8.show;
end;
```

```
procedure TForm10.ReportBorrower1Click(Sender: TObject);
begin
form10.Close;
form9.show;
end;
```

```
procedure TForm10.Update1Click(Sender: TObject);
begin
form10.Close;
form4.show;
end;
```

```
procedure TForm10.AddBook1Click(Sender: TObject);
begin
form10.Close;
form2.show;
end;
```

```
procedure TForm10.AddUser1Click(Sender: TObject);
begin
form10.Close;
form3.show;
end;
```

```
procedure TForm10.GiveBook1Click(Sender: TObject);
begin
form10.close;
form15.show;
end;
```

```
procedure TForm10.ReturnBook1Click(Sender: TObject);
begin
form10.close;
form14.show;
end;
```

```
procedure TForm10.FormCreate(Sender: TObject);
begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select Isbn, Bookname,Bookstatus, Location from halime';
query1.Active:=true;
edit1.Text:='';
edit2.Text:='';
```

```

end;

procedure TForm10.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select Isbn, Bookname,Bookstatus, Location from halime where Isbn
like'+#39+(edit1.text)+'%'+#39);
query1.Open;
end;

procedure TForm10.Edit2Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select Isbn, Bookname,Bookstatus, Location from halime where
BookName like'+#39+(edit2.text)+'%'+#39);
query1.Open;
end;

procedure TForm10.Button1Click(Sender: TObject);
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.add('SELECT Isbn, Bookname,Bookstatus, Location ');
query1.sql.add('FROM halime');
query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
query1.Open;
dbgrid1.DataSource:=datasource1
end;
end.

```

SEARCH BOOK BY NAME

```

unit Unit13;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, Mask, DBCtrls, StdCtrls, ComCtrls, DB, DBTables;

type
  TForm13 = class(TForm)
    Label1: TLabel;
    DateTimePicker1: TDateTimePicker;
    DateTimePicker2: TDateTimePicker;

```



```

Label15: TLabel;
Label16: TLabel;
Label2: TLabel;
Edit1: TEdit;
Label3: TLabel;
MainMenu1: TMainMenu;
MainMenu2: TMenuItem;
ReturnBook1: TMenuItem;
GiveBook1: TMenuItem;
Update1: TMenuItem;
Add1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
SearchByBorrower1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
DBComboBox1: TDBComboBox;
Label9: TLabel;
DBEdit1: TDBEdit;
Button1: TButton;
AddDeleteUpdateUser1: TMenuItem;
Query1: TQuery;
DataSource1: TDataSource;
procedure Exit1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure AddDeleteUpdateUser1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public

```

```

    { Public declarations }
end;

var
    Form13: TForm13;

implementation

uses Unit7, Unit8, Unit9, Unit10, Unit11, Unit12, Unit4, Unit2, Unit3,
    Unit15, Unit14;

{$R *.dfm}

procedure TForm13.Exit1Click(Sender: TObject);
begin
    form13.Close;
end;

procedure TForm13.ReportByCashier1Click(Sender: TObject);
begin
    form13.Close;
    form7.show;
end;

procedure TForm13.reportByBorrowers1Click(Sender: TObject);
begin
    form13.Close;
    form8.show;
end;

procedure TForm13.ReportBorrower1Click(Sender: TObject);
begin
    form13.Close;
    form9.show;
end;

procedure TForm13.ISBN1Click(Sender: TObject);
begin
    form13.Close;
    form10.show;
end;

procedure TForm13.SerchByCashier1Click(Sender: TObject);
begin
    form13.Close;
    form11.show;
end;

procedure TForm13.SearchByBorrower1Click(Sender: TObject);
begin

```

```
form13.Close;  
form12.show;  
end;
```

```
procedure TForm13.Update1Click(Sender: TObject);  
begin  
form13.Close;  
form4.show;  
end;
```

```
procedure TForm13.AddBook1Click(Sender: TObject);  
begin  
form13.Close;  
form2.show;  
end;
```

```
procedure TForm13.AddUser1Click(Sender: TObject);  
begin  
form13.Close;  
form3.show;  
end;
```

```
procedure TForm13.GiveBook1Click(Sender: TObject);  
begin  
form13.close;  
form15.show;  
end;
```

```
procedure TForm13.ReturnBook1Click(Sender: TObject);  
begin  
form13.close;  
form14.show;  
end;
```

```
procedure TForm13.AddDeleteUpdateUser1Click(Sender: TObject);  
begin  
form13.Close;  
form3.show;  
end;
```

```
procedure TForm13.FormCreate(Sender: TObject);  
begin  
Query1.DatabaseName:='STANDARD7';  
Query1.requestlive:=true;  
query1.SQL.Text:='select * from halime';  
query1.Active:=true;  
edit1.Text:="";  
  
end;
```

```

procedure TForm13.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where BookName
like'+#39+(edit1.text)+'%'+#39);
query1.Open;
end;

procedure TForm13.Button1Click(Sender: TObject);
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.add('SELECT * ');
query1.sql.add('FROM halime');
query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
query1.Open;

end;
end.

```

SEARCH USER

```
unit Unit11;
```

```
interface
```

```
uses
```

```

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, Grids, DBGrids, Mask, DBCtrls, StdCtrls, ComCtrls, DB,
DBTables;

```

```
type
```

```

TForm11 = class(TForm)
Label1: TLabel;
Label15: TLabel;
DateTimePicker1: TDateTimePicker;
Label16: TLabel;
DateTimePicker2: TDateTimePicker;
Label2: TLabel;
Edit1: TEdit;
DBGrid1: TDBGrid;
MainMenu1: TMainMenu;
MainMenu2: TMenuItem;
ReturnBook1: TMenuItem;
GiveBook1: TMenuItem;
Update1: TMenuItem;

```



```

Add1: TMenuItem;
AddUser1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
SearchByBorrower1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
Button1: TButton;
Query1: TQuery;
DataSource1: TDataSource;
procedure Exit1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form11: TForm11;

implementation

uses Unit7, Unit8, Unit9, Unit10, Unit13, Unit12, Unit4, Unit2, Unit3,
    Unit15, Unit14;

{$R *.dfm}

procedure TForm11.Exit1Click(Sender: TObject);

```

```

begin
form11.Close;
end;

procedure TForm11.ReportByCashier1Click(Sender: TObject);
begin
form11.Close;
form7.show;
end;

procedure TForm11.reportByBorrowers1Click(Sender: TObject);
begin
form11.Close;
form8.show;
end;

procedure TForm11.ReportBorrower1Click(Sender: TObject);
begin
form11.Close;
form9.show;
end;

procedure TForm11.ISBN1Click(Sender: TObject);
begin
form11.Close;
form10.show;
end;

procedure TForm11.Cashier1Click(Sender: TObject);
begin
form11.Close;
form13.show;
end;

procedure TForm11.SearchByBorrower1Click(Sender: TObject);
begin
form11.Close;
form12.show;
end;

procedure TForm11.Update1Click(Sender: TObject);
begin
form11.Close;
form4.show;
end;

procedure TForm11.AddBook1Click(Sender: TObject);
begin
form11.Close;
form2.show;

```

```

end;

procedure TForm11.AddUser1Click(Sender: TObject);
begin
form11.Close;
form3.show;
end;

procedure TForm11.GiveBook1Click(Sender: TObject);
begin
form11.close;
form15.show;
end;

procedure TForm11.ReturnBook1Click(Sender: TObject);
begin
form11.close;
form14.show;
end;

procedure TForm11.FormCreate(Sender: TObject);
begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select userid, username,usersurname,bookname,rentdate,returndate
from halime';
query1.Active:=true;
edit1.Text:='';

end;

procedure TForm11.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select userid, username,usersurname,bookname,rentdate,returndate from
halime where cashierid like'+#39+(edit1.text)+'%'+#39);
query1.Open;
end;

procedure TForm11.Button1Click(Sender: TObject);
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.add('SELECT userid,
username,usersurname,bookname,rentdate,returndate');
query1.sql.add('FROM halime');
query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;

```

```

query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
query1.Open;
dbgrid1.DataSource:=datasource1
end;
end.

```

SEARCH BORROWER

```

unit Unit12;

```

```

interface

```

```

uses

```

```

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, Grids, DBGrids, Mask, DBCtrls, StdCtrls, ComCtrls, DB,
  DBTables;

```

```

type

```

```

  TForm12 = class(TForm)
    Label1: TLabel;
    Label15: TLabel;
    DateTimePicker1: TDateTimePicker;
    Label16: TLabel;
    DateTimePicker2: TDateTimePicker;
    Label2: TLabel;
    Edit1: TEdit;
    DBGrid1: TDBGrid;
    MainMenu1: TMainMenu;
    MainMenu2: TMenuItem;
    ReturnBook1: TMenuItem;
    GiveBook1: TMenuItem;
    Update1: TMenuItem;
    Add1: TMenuItem;
    AddUser1: TMenuItem;
    AddBook1: TMenuItem;
    Search1: TMenuItem;
    SerchBook1: TMenuItem;
    ISBN1: TMenuItem;
    Cashier1: TMenuItem;
    SerchByCashier1: TMenuItem;
    SearchByBorrower1: TMenuItem;
    Report1: TMenuItem;
    ReportByCashier1: TMenuItem;
    reportByBorrowers1: TMenuItem;
    ReportBorrower1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    Button1: TButton;
    Query1: TQuery;
    DataSource1: TDataSource;

```



```

procedure Exit1Click(Sender: TObject);
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form12: TForm12;

implementation

uses Unit7, Unit8, Unit9, Unit10, Unit13, Unit11, Unit4, Unit2, Unit3,
  Unit15, Unit14;

{$R *.dfm}

procedure TForm12.Exit1Click(Sender: TObject);
begin
  form12.Close;
end;

procedure TForm12.ReportByCashier1Click(Sender: TObject);
begin
  form12.Close;
  form7.show;
end;

procedure TForm12.reportByBorrowers1Click(Sender: TObject);
begin
  form12.Close;
  form8.show;
end;

procedure TForm12.ReportBorrower1Click(Sender: TObject);
begin

```

```
form12.Close;  
form9.show;  
end;
```

```
procedure TForm12.ISBN1Click(Sender: TObject);  
begin  
form12.Close;  
form10.show;  
end;
```

```
procedure TForm12.Cashier1Click(Sender: TObject);  
begin  
form12.Close;  
form13.show;  
end;
```

```
procedure TForm12.SerchByCashier1Click(Sender: TObject);  
begin  
form12.Close;  
form11.show;  
end;
```

```
procedure TForm12.Update1Click(Sender: TObject);  
begin  
form12.Close;  
form4.show;  
end;
```

```
procedure TForm12.AddBook1Click(Sender: TObject);  
begin  
form12.Close;  
form2.show;  
end;
```

```
procedure TForm12.AddUser1Click(Sender: TObject);  
begin  
form12.Close;  
form3.show;  
end;
```

```
procedure TForm12.GiveBook1Click(Sender: TObject);  
begin  
form12.close;  
form15.show;  
end;
```

```
procedure TForm12.ReturnBook1Click(Sender: TObject);  
begin  
form12.close;  
form14.show;
```

end;

```
procedure TForm12.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='STANDARD7';
  Query1.requestlive:=true;
  query1.SQL.Text:='select
  borrowerid,borrowername,borrowersurname,bookname,rentdate,returndate from
  halime';
  query1.Active:=true;
  edit1.Text:='';
```

end;

```
procedure TForm12.Edit1Change(Sender: TObject);
begin
  query1.close;
  query1.SQL.Clear;
  query1.sql.add('select
  borrowerid,borrowername,borrowersurname,bookname,rentdate,returndate from halime
  where borrowerid like'+#39+(edit1.text)+'%'+#39);
  query1.Open;
end;
```

```
procedure TForm12.Button1Click(Sender: TObject);
begin
  query1.Close;
  query1.SQL.Clear;
  query1.SQL.add('SELECT
  borrowerid,borrowername,borrowersurname,bookname,rentdate,returndate ');
  query1.sql.add('FROM halime');
  query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
  query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
  query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
  query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
  query1.Open;
  dbgrid1.DataSource:=datasource1
end;
end.
```

REPORT BOOK

unit Unit7;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Mask, DBCtrls, StdCtrls, ComCtrls, Menus, DB, DBTables,
frxClass, frxDBSet;

type

```
TForm7 = class(TForm)
  Label1: TLabel;
  MainMenu1: TMainMenu;
  MainMenu2: TMenuItem;
  ReturnBook1: TMenuItem;
  GiveBook1: TMenuItem;
  Update1: TMenuItem;
  Add1: TMenuItem;
  AddUser1: TMenuItem;
  AddBook1: TMenuItem;
  Search1: TMenuItem;
  SerchBook1: TMenuItem;
  ISBN1: TMenuItem;
  Cashier1: TMenuItem;
  SerchByCashier1: TMenuItem;
  Report1: TMenuItem;
  ReportByCashier1: TMenuItem;
  reportByBorrowers1: TMenuItem;
  N1: TMenuItem;
  Exit1: TMenuItem;
  DateTimePicker1: TDateTimePicker;
  DateTimePicker2: TDateTimePicker;
  Label15: TLabel;
  Label16: TLabel;
  Label2: TLabel;
  Edit1: TEdit;
  ReportBorrower1: TMenuItem;
  SearchBorrower1: TMenuItem;
  Button2: TButton;
  Button1: TButton;
  Query1: TQuery;
  DataSource1: TDataSource;
  Button3: TButton;
  frxDBDataset1: TfrxDBDataset;
  frxReport1: TfrxReport;
  procedure reportByBorrowers1Click(Sender: TObject);
  procedure ReportBorrower1Click(Sender: TObject);
  procedure Exit1Click(Sender: TObject);
  procedure ISBN1Click(Sender: TObject);
  procedure Cashier1Click(Sender: TObject);
  procedure SerchByCashier1Click(Sender: TObject);
  procedure SearchBorrower1Click(Sender: TObject);
  procedure Update1Click(Sender: TObject);
  procedure AddBook1Click(Sender: TObject);
  procedure AddUser1Click(Sender: TObject);
  procedure GiveBook1Click(Sender: TObject);
  procedure ReturnBook1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
```



```

    procedure Edit1Change(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form7: TForm7;

implementation

uses Unit8, Unit9, Unit10, Unit13, Unit11, Unit12, Unit4, Unit2, Unit3,
    Unit15, Unit14;

{$R *.dfm}

procedure TForm7.reportByBorrowers1Click(Sender: TObject);
begin
    form7.Close;
    form8.show;
end;

procedure TForm7.ReportBorrower1Click(Sender: TObject);
begin
    form7.Close;
    form9.show;
end;

procedure TForm7.Exit1Click(Sender: TObject);
begin
    form7.Close;
end;

procedure TForm7.ISBN1Click(Sender: TObject);
begin
    form7.Close;
    form10.show;
end;

procedure TForm7.Cashier1Click(Sender: TObject);
begin
    form7.Close;
    form13.show;
end;

procedure TForm7.SerchByCashier1Click(Sender: TObject);

```

```

begin
form7.Close;
form11.show;
end;

procedure TForm7.SearchBorrower1Click(Sender: TObject);
begin
form7.Close;
form12.show;
end;

procedure TForm7.Update1Click(Sender: TObject);
begin
form7.Close;
form4.show;
end;

procedure TForm7.AddBook1Click(Sender: TObject);
begin
form7.Close;
form2.show;
end;

procedure TForm7.AddUser1Click(Sender: TObject);
begin
form7.Close;
form3.show;
end;

procedure TForm7.GiveBook1Click(Sender: TObject);
begin
form7.close;
form15.show;
end;

procedure TForm7.ReturnBook1Click(Sender: TObject);
begin
form7.close;
form14.show;
end;

procedure TForm7.FormCreate(Sender: TObject);
begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
end;

```

```

procedure TForm7.Edit1Change(Sender: TObject);
begin
  query1.close;
  query1.SQL.Clear;
  query1.sql.add('select * from halime where BookName
  like'+#39+(edit1.text)+'%'+#39);
  query1.Open;

```

```

end;

```

```

procedure TForm7.Button3Click(Sender: TObject);
begin
  query1.Close;
  query1.SQL.Clear;
  query1.SQL.add('SELECT * ');
  query1.sql.add('FROM book');
  query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
  query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
  query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
  query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
  query1.Open;

```

```

end;

```

```

procedure TForm7.Button2Click(Sender: TObject);
begin
  frxreport1.Print;
end;

```

```

procedure TForm7.Button1Click(Sender: TObject);
begin
  frxReport1.ShowReport(true);
end;
end.

```

REPORT USER

```

unit Unit8;

```

```

interface

```

```

uses

```

```

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, Mask, DBCtrls, StdCtrls, ComCtrls, Menus, DB,
  DBTables, frxClass, frxDBSet;

```

```

type

```

```

  TForm8 = class(TForm)
    Label1: TLabel;
    Label15: TLabel;

```

```

DateTimePicker1: TDateTimePicker;
Label16: TLabel;
DateTimePicker2: TDateTimePicker;
Label2: TLabel;
Edit1: TEdit;
MainMenu1: TMainMenu;
MainMenu2: TMenuItem;
ReturnBook1: TMenuItem;
GiveBook1: TMenuItem;
Update1: TMenuItem;
Add1: TMenuItem;
AddUser1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
SearchByBorrower1: TMenuItem;
Button2: TButton;
Button1: TButton;
Query1: TQuery;
DataSource1: TDataSource;
Button3: TButton;
frxDBDataset1: TfrxDBDataset;
frxReport1: TfrxReport;
procedure ReportByCashier1Click(Sender: TObject);
procedure ReportBorrower1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
private

```



```

    { Private declarations }
public
    { Public declarations }
end;

var
    Form8: TForm8;

implementation

uses Unit7, Unit9, Unit10, Unit13, Unit11, Unit12, Unit4, Unit2, Unit3,
    Unit15, Unit14;

{$R *.dfm}

procedure TForm8.ReportByCashier1Click(Sender: TObject);
begin
    form8.Close;
    form7.show;
end;

procedure TForm8.ReportBorrower1Click(Sender: TObject);
begin
    form8.Close;
    form9.show;
end;

procedure TForm8.Exit1Click(Sender: TObject);
begin
    Form8.Close;
end;

procedure TForm8.ISBN1Click(Sender: TObject);
begin
    form8.Close;
    form10.show;
end;

procedure TForm8.Cashier1Click(Sender: TObject);
begin
    form8.Close;
    form13.show;
end;

procedure TForm8.SerchByCashier1Click(Sender: TObject);
begin
    form8.Close;
    form11.show;
end;

```

```

procedure TForm8.SearchByBorrower1Click(Sender: TObject);
begin
form8.Close;
form12.show;
end;

```

```

procedure TForm8.Update1Click(Sender: TObject);
begin
form8.Close;
form4.show;
end;

```

```

procedure TForm8.AddBook1Click(Sender: TObject);
begin
form8.Close;
form2.show;
end;

```

```

procedure TForm8.AddUser1Click(Sender: TObject);
begin
form8.Close;
form3.show;
end;

```

```

procedure TForm8.GiveBook1Click(Sender: TObject);
begin
form8.close;
form15.show;
end;

```

```

procedure TForm8.ReturnBook1Click(Sender: TObject);
begin
form8.close;
form14.show;
end;

```

```

procedure TForm8.FormCreate(Sender: TObject);
begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
end;

```

```

procedure TForm8.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where userid like'+#39+(edit1.text)+'%'+#39);

```

```
query1.Open;
end;
```

```
procedure TForm8.Button3Click(Sender: TObject);
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.add('SELECT * ');
query1.sql.add('FROM book');
query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
query1.Open;
```

```
end;
```

```
procedure TForm8.Button2Click(Sender: TObject);
begin
frxreport1.Print;
end;
```

```
procedure TForm8.Button1Click(Sender: TObject);
begin
frxReport1.ShowReport(true);
end;
```

```
end.
```

REPORT BORROWER

```
unit Unit9;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Grids, DBGrids, Mask, DBCtrls, StdCtrls, ComCtrls, Menus, DB,
DBTables, frxClass, frxDBSet;
```

```
type
```

```
TForm9 = class(TForm)
Label1: TLabel;
Label15: TLabel;
DateTimePicker1: TDateTimePicker;
Label16: TLabel;
DateTimePicker2: TDateTimePicker;
Label2: TLabel;
Edit1: TEdit;
MainMenu1: TMainMenu;
MainMenu2: TMenuItem;
```

```

ReturnBook1: TMenuItem;
GiveBook1: TMenuItem;
Update1: TMenuItem;
Add1: TMenuItem;
AddUser1: TMenuItem;
AddBook1: TMenuItem;
Search1: TMenuItem;
SerchBook1: TMenuItem;
ISBN1: TMenuItem;
Cashier1: TMenuItem;
SerchByCashier1: TMenuItem;
Report1: TMenuItem;
ReportByCashier1: TMenuItem;
reportByBorrowers1: TMenuItem;
ReportBorrower1: TMenuItem;
N1: TMenuItem;
Exit1: TMenuItem;
SearchByBorrower1: TMenuItem;
Button1: TButton;
Button2: TButton;
Query1: TQuery;
DataSource1: TDataSource;
Button3: TButton;
frxDBDataset1: TfrxDBDataset;
frxReport1: TfrxReport;
procedure ReportByCashier1Click(Sender: TObject);
procedure reportByBorrowers1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure ISBN1Click(Sender: TObject);
procedure Cashier1Click(Sender: TObject);
procedure SerchByCashier1Click(Sender: TObject);
procedure SearchByBorrower1Click(Sender: TObject);
procedure Update1Click(Sender: TObject);
procedure AddBook1Click(Sender: TObject);
procedure AddUser1Click(Sender: TObject);
procedure GiveBook1Click(Sender: TObject);
procedure ReturnBook1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form9: TForm9;

```


implementation

uses Unit7, Unit8, Unit10, Unit13, Unit11, Unit12, Unit4, Unit2, Unit3,
Unit15, Unit14;

{ \$R *.dfm }

```
procedure TForm9.ReportByCashier1Click(Sender: TObject);
begin
form9.Close;
form7.show;
end;
```

```
procedure TForm9.reportByBorrowers1Click(Sender: TObject);
begin
form9.Close;
form8.show;
end;
```

```
procedure TForm9.Exit1Click(Sender: TObject);
begin
form9.Close;
end;
```

```
procedure TForm9.ISBN1Click(Sender: TObject);
begin
form9.Close;
form10.show;
end;
```

```
procedure TForm9.Cashier1Click(Sender: TObject);
begin
form9.Close;
form13.show;
end;
```

```
procedure TForm9.SerchByCashier1Click(Sender: TObject);
begin
form9.Close;
form11.show;
end;
```

```
procedure TForm9.SearchByBorrower1Click(Sender: TObject);
begin
form9.Close;
form12.show;
end;
```

```
procedure TForm9.Update1Click(Sender: TObject);
```

```
begin
form9.Close;
form4.show;
end;
```

```
procedure TForm9.AddBook1Click(Sender: TObject);
begin
form9.Close;
form2.show;
end;
```

```
procedure TForm9.AddUser1Click(Sender: TObject);
begin
form9.Close;
form3.show;
end;
```

```
procedure TForm9.GiveBook1Click(Sender: TObject);
begin
form9.close;
form15.show;
end;
```

```
procedure TForm9.ReturnBook1Click(Sender: TObject);
begin
form9.close;
form14.show;
end;
```

```
procedure TForm9.FormCreate(Sender: TObject);
begin
Query1.DatabaseName:='STANDARD7';
Query1.requestlive:=true;
query1.SQL.Text:='select * from halime';
query1.Active:=true;
edit1.Text:='';
end;
```

```
procedure TForm9.Edit1Change(Sender: TObject);
begin
query1.close;
query1.SQL.Clear;
query1.sql.add('select * from halime where borrowerid like'+#39+(edit1.text)+'%'+#39);
query1.Open;
end;
```

```
procedure TForm9.Button3Click(Sender: TObject);
begin
query1.Close;
query1.SQL.Clear;
```

```

query1.SQL.add('SELECT * ');
query1.sql.add('FROM book');
query1.sql.add('where (ReturnDate>=:Gelen) and (ReturnDate<=:Gelen1)');
query1.sql.add('or (RentDate>=:Gelen) and (RentDate<=:Gelen1)');
query1.ParamByName('Gelen').AsDatetime := DateTimePicker1.DateTime;
query1.ParamByName('Gelen1').AsDatetime := DateTimePicker2.DateTime;
query1.Open;

```

```

end;

```

```

procedure TForm9.Button2Click(Sender: TObject);
begin
    frxreport1.Print;
end;

```

```

procedure TForm9.Button1Click(Sender: TObject);
begin
    frxReport1.ShowReport(true);
end;
end.

```

ABOUT

```

unit Unit22;

```

```

interface

```

```

uses

```

```

    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ExtCtrls;

```

```

type

```

```

    TForm22 = class(TForm)
        Image1: TImage;
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        Label4: TLabel;
        Image2: TImage;
        procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

```

```

var

```

```

    Form22: TForm22;

```

implementation

{ \$R *.dfm }

```
procedure TForm22.FormCreate(Sender: TObject);  
begin  
end;  
end.
```