NEAR EAST UNIVERSITY

Faculty of Engineering

Department Of Computer Engineering

BOOKSTORE PROGRAMMING WITH USING DELPHI

Graduating Project
COM - 400

Student: Serbülent KARAGÖZ

Supervisor : Assoc.Prof.Dr. Rahib ABIYEV

Nicosia 2006

INTRODUCTION

I would like to inform about this software.This program is designed Borland Delphi 7.0 with using Microsoft Sql Server,so that the program is based on client/server model That is very important for a software because the program is not depended the particular computer.

Actually,our project is occured two different section.One of them BookStore web page, and BookStore Admin Section.Bookstore Web Page is prepared with Active Server Pages that is designed my best friend is Arif Koçak.In this project, we were worked together.

Now I want to inform my project which is BookStore Admin Section.Firstly,the tables is prepared with using Microsoft Sql Server, so used relational database systems,that is very important for sequrity and straightness the datas.For instance,when it is entered the author,publisher informations,it can be seen book information section automatically, so that ,it is very useful for the user.And also it is prepared the report system for every stage the program.It can be seen author,publisher etc... or according to some criterions, so the user can be printed out any reports in program.

The program is also supported multi-user system.Easily a user can be added the system or removed the system.

The aim of this software product is covered the BookStore in Near East University

Chapter One describes Basic concept of Delphi 7
Chapter Two presents Microsoft Sql Server
Chapter Three how the ADO connection does with MSSQL Server
Chapter Four presents Flowcharts of Program
Chapter Five informs about using forms.

ABSTRACT

Nowadays the computers is developed a lot and started to use by almost anyone in the world no matter who he/she is. Because of the computers is entered to every platform of our life human needed to combine both software and hardware. Without software the computers are nothing. They need software to operate, a software system also need datas which is using daily life ...

Data, gathered around us as a collection of facts, is of no use unless it is organized and represented in some meaningful form. Data represented in some meaningful form like, tables, charts, or graphs become information, which can be easily processed. The collection of data, usually refereed to as the database, contains information about one particular enterprise. These days' databases are used by a variety of users and organizations, which are important tools in processing DBMS, which are designed to manage large amount of data.

This project has as its goal to develop software is working on based client\server structure, and processing information about activities of a bookstore company. Software is developed in this project contains many forms which informations associated with sales and purchase of books. The project can be developed by improving the software for processing all activities of the bookstrore.

TABLE OF CONTENTS

CHAPTER 1: DELPHI

1. BASIC CONCEPT OF DELPHI 7

1. Introduction

Let us start an overview of the Delphi development environment to get you started using the product right away. It also tells you where to look for details about the tools and features available in Delphi.

You will be shown a tour of the environment describes the main tools on the Delphi desktop, or integrated desktop environment (IDE). It will be explained Programming with Delphi, explains how you use some of these tools to create an application. And also Customizing the desktop describes how you can customize the Delphi IDE for your development needs.

1.1. What is Delphi?

Delphi is an object-oriented, visual programming environment for rapid application development (RAD). Using Delphi, you can create highly efficient applications for Microsoft Windows XP, Microsoft Windows 2000 and Microsoft Windows 98 with a minimum of manual coding. Delphi also provides a simple cross-platform solution when used in conjunction with Kylix, Borland's RAD tool for Linux. Delphi provides all the tools you need to develop, test, and deploy applications, including a large library of reusable components, a suite of design tools, application and form templates, and programming wizards.

1.2.1. Registering Delphi

Delphi can be registered in several ways. The first time you launch Delphi after installation, you will be prompted to enter your serial number and authorization key. Once this has been entered, a registration dialog offers four choices:

• Register using your internet connection.

Use this option to register online using your existing internet connection.

• Register by phone or Web browser.

Use this option to register by phone or through your web browser. If you received an activation key via email, use this option to select the file.

• Import software activation information from a file or email.

• Register later.

Online registration is the easiest way to register Delphi, but it requires that you have an active connection to the internet. If you are already a member of the Borland Community, or have an existing software registration account, simply enter the relevant account information. This will automatically register Delphi. If not, the registration process provides a way to create an account.

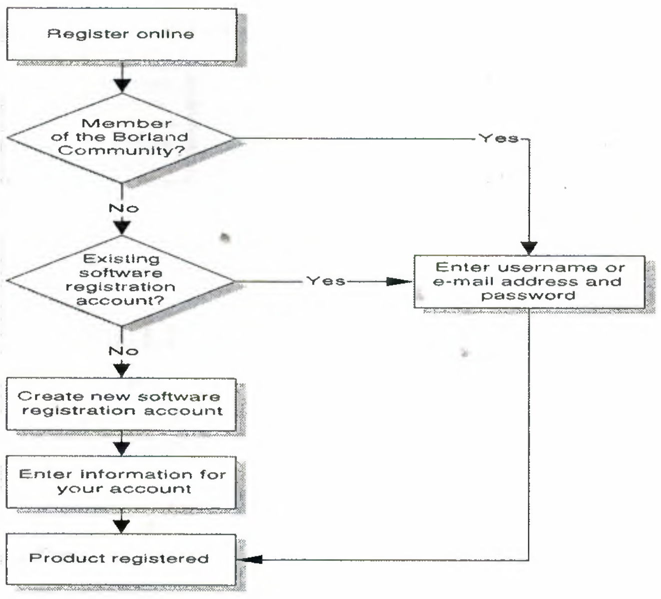

Register online
Member of the Borland Community?
Yes
No
Existing software registration account?
Yes
No
Enter username or e-mail address and password
Create new software registration account
Enter information for your account
Product registered

Fig. 1.1 Online Registration

The second option (register by phone or Web page) is useful if the machine you are installing on is not connected to the internet, or if you are behind a firewall that is blocking online registration.

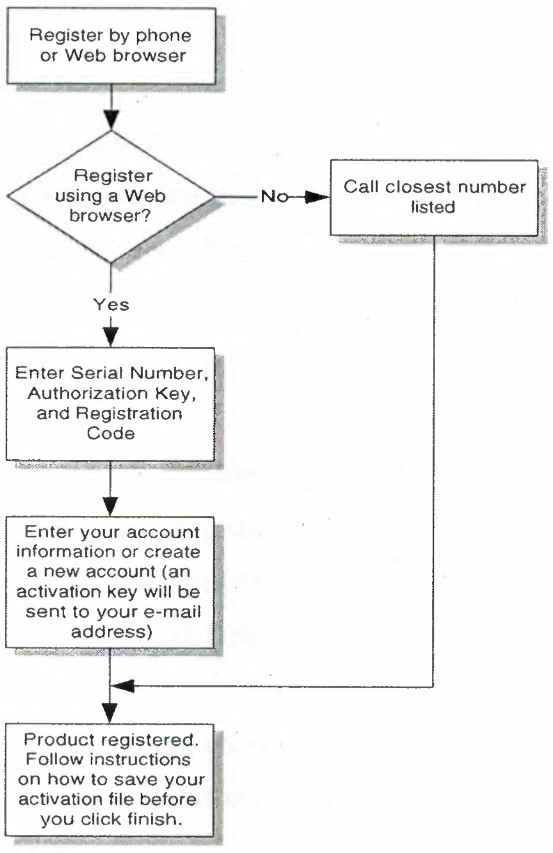

Register by phone or Web browser
Register using a Web browser?
No
Call closest number listed
Yes
Enter Serial Number, Authorization Key, and Registration Code
Enter your account information or create a new account (an activation key will be sent to your e-mail address)
Product registered. Follow instructions on how to save your activation file before you click finish.

Fig. 1.2 Register by Phone or Web Page

If you have previously received software activation information, you can select theImport software activation information from a file or email option and select the activation.slip file on your system.

If you have previously received software activation information, you can select the Import software activation information from a file or email option and select the activation.slip file on your system.

1.2.2. Finding Information

You can find information on Delphi in the following ways:

• Online Help

• Printed documentation

• Borland developer support services and Web site

For information about new features in this release, refer to What's New in the online Help Contents and to the www.borland.com Web site.

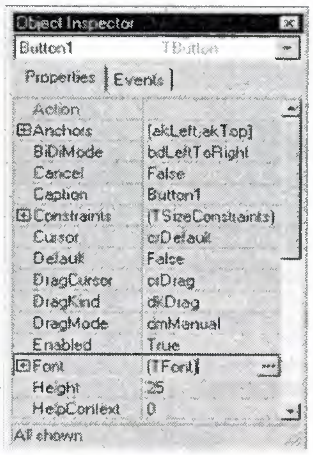
1.2.3. Online Help

The online Help system provides detailed information about user interface features, language implementation, programming tasks, and the components. It includes all the material in the Delphi Developer's Guide, Delphi Language Guide, and a host of Help files for other features bundled with Delphi.

To view the table of contents, choose Help|Delphi Help and Help|Delphi Tools, and click the Contents tab. To look up the components or any other topic, click the Index or Find tab and type your request.

1.2.4. F1 Help

You can get context-sensitive Help on any part of the development environment, including menu items, dialog boxes, toolbars, and components by selecting the item and pressing F1.



Object Inspector

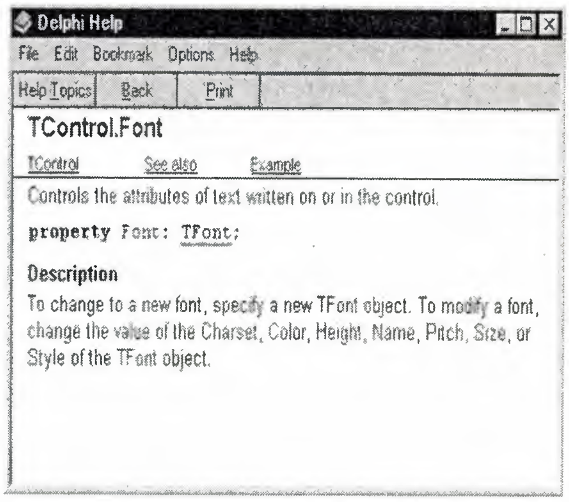
Button1 TButton

Properties | Events

Action	
⊞Anchors	[akLeft,akTop]
BiDiMode	bdLeftToRight
Cancel	False
Caption	Button1
⊞Constraints	(TSizeConstraints)
Cursor	crDefault
Default	False
DragCursor	crDrag
DragKind	dkDrag
DragMode	dmManual
Enabled	True
⊞Font	(TFont)
Height	25
HelpContext	0

All shown

Press F1 on a property or event name in the Object Inspector to display the VCL Help.



TControl.Font

TControl See also Example

Controls the attributes of text written on or in the control.

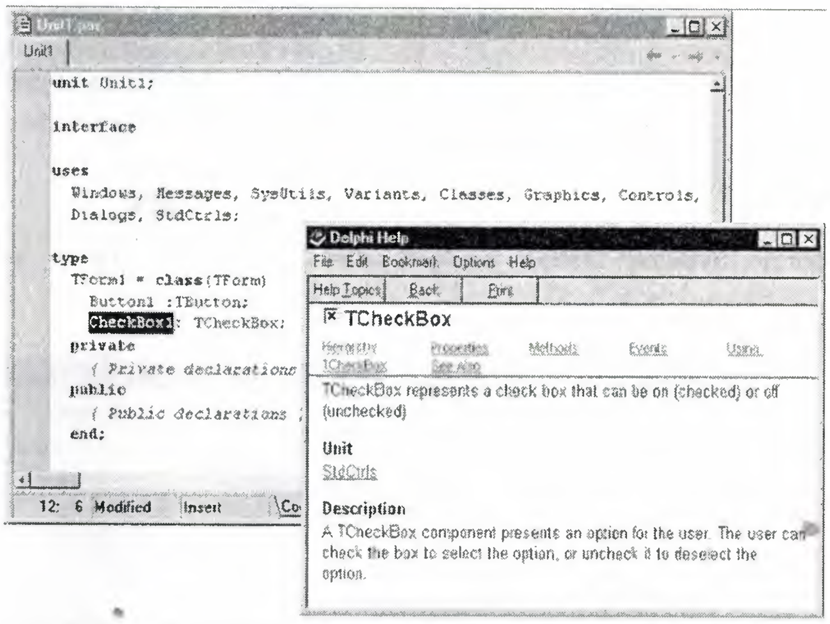
```
property Font: TFont;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

Fig. 1.3 TControl Font

In the Code editor, press F1 on a language, VCL, or CLX element.



```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1 :TButton;
    CheckBox1: TCheckBox;
  private
    { Private declarations
  public
    { Public declarations ;
  end;
```

TCheckBox

Hierarchy Properties Methods Events Using TCheckBox See Also

TCheckBox represents a check box that can be on (checked) or off (unchecked)

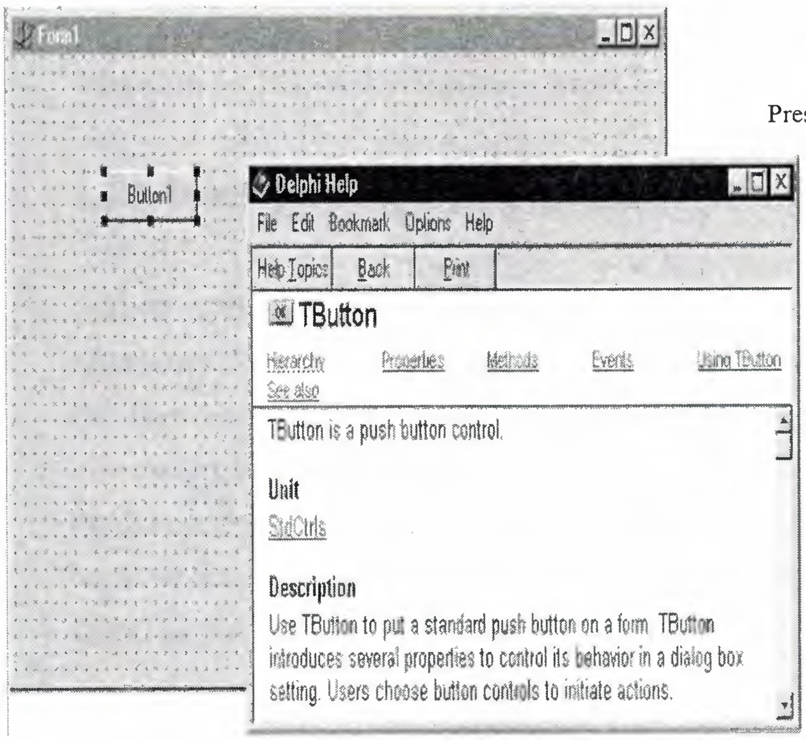
Unit

StdCtrls

Description

A TCheckBox component presents an option for the user. The user can check the box to select the option, or uncheck it to deselect the option.

Fig. 1.4 TCheckBox


Form1
Button1
Delphi Help
File Edit Bookmark Options Help
Help Topics Back Print
Hierarchy Properties Methods Events Using TButton
See also

TButton

TButton is a push button control.

Unit

StdCtrls

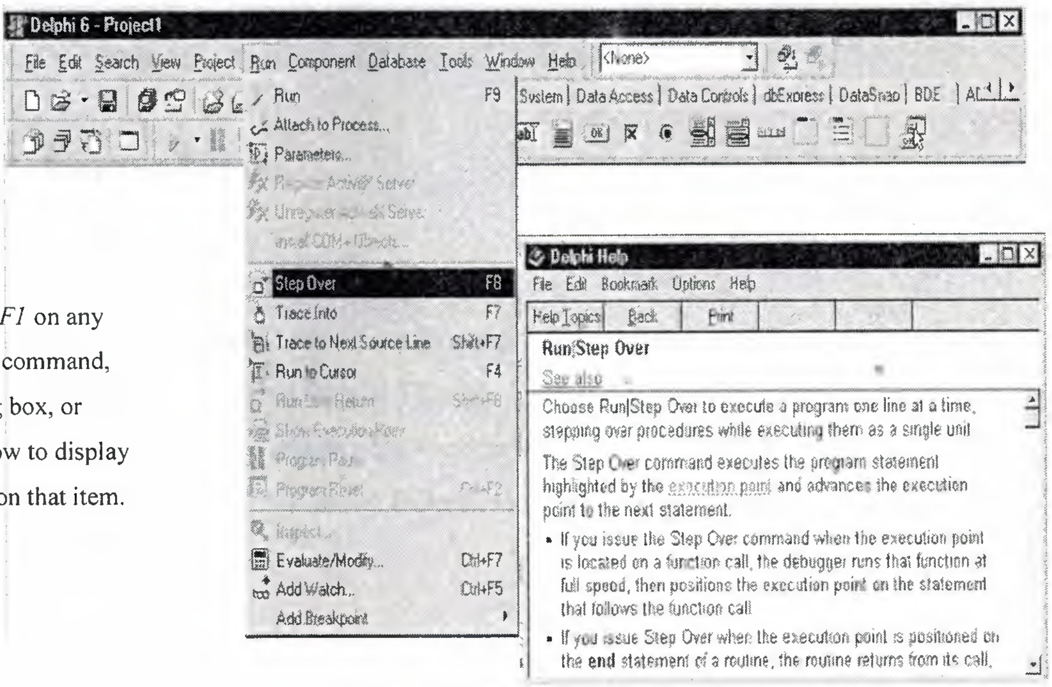
Description

Use TButton to put a standard push button on a form. TButton introduces several properties to control its behavior in a dialog box setting. Users choose button controls to initiate actions.

Press F1 on a component on a form.

Fig. 1.5 Tbutton

Pressing the Help button in any dialog box also displays context-sensitive online documentation.



Run|Step Over

See also

Choose Run|Step Over to execute a program one line at a time, stepping over procedures while executing them as a single unit.

The Step Over command executes the program statement highlighted by the execution point and advances the execution point to the next statement.

- If you issue the Step Over command when the execution point is located on a function call, the debugger runs that function at full speed, then positions the execution point on the statement that follows the function call.
- If you issue Step Over when the execution point is positioned on the **end** statement of a routine, the routine returns from its call.

Press *F1* on any menu command, dialog box, or window to display Help on that item.

Fig. 1.6 Run/Step Over

Error messages from the compiler and linker appear in a special window below the Code editor. To get Help with compilation errors, select a message from the list and press F1.

1.2.5. Developer Support Services and Web Site

Borland offers a variety of support options to meet the needs of its diverse developer community. To find out about support, refer to http://www.borland.com/devsupport/. From the Web site, you can access many newsgroups where Delphi developers exchange information, tips, and techniques. From the Web site, you can access many newsgroups where Delphi developers exchange information, tips, and techniques. The site also includes a list of books about Delphi, additional Delphi technical documents, and Frequently Asked Questions (FAQs).

1.3. 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).

1.3.1. Starting Delphi

You can start Delphi in the following ways:

• Double-click the Delphi icon (if you've created a shortcut).
• Choose Programs|Borland Delphi 7|Delphi 7 from the Windows Start menu.
• Choose Run from the Windows Start menu, then enter Delphi32.
• Double-click Delphi32.exe in the Delphi\Bin directory.

1.3.2. The IDE

When you first start Delphi, you'll see some of the major tools in the IDE. In Delphi, the IDE includes the menus, toolbars, Component palette, Object Inspector, Object TreeView, Code editor, Code Explorer, Project Manager, and many other tools. The particular features and components available to you will depend on which edition of Delphi you've purchased.

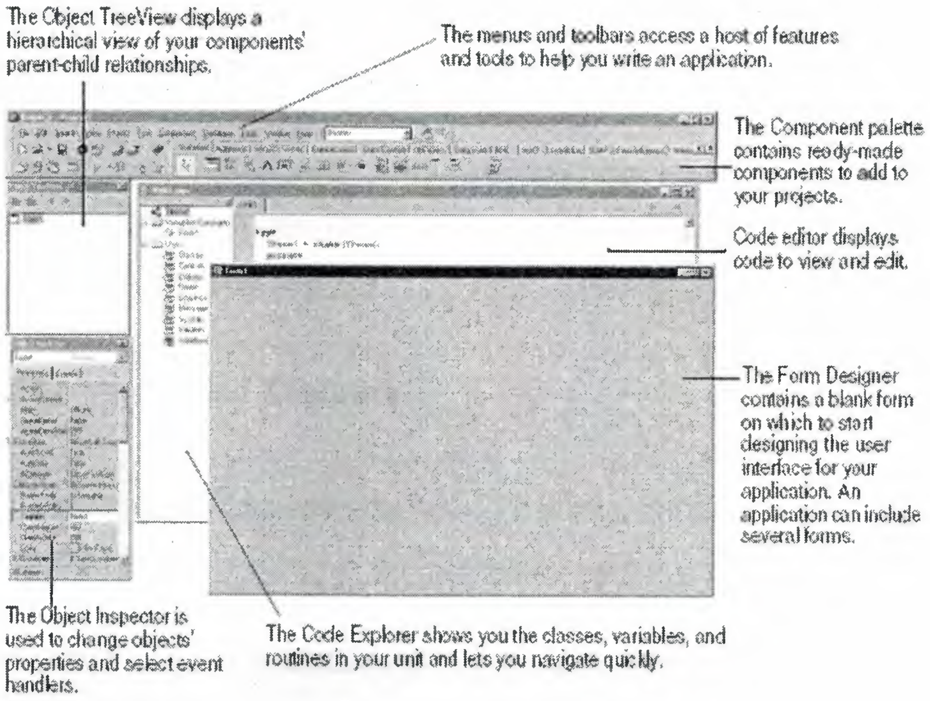

The Object TreeView displays a hierarchical view of your components' parent-child relationships.
The menus and toolbars access a host of features and tools to help you write an application.
The Component palette contains ready-made components to add to your projects.
Code editor displays code to view and edit.
The Form Designer contains a blank form on which to start designing the user interface for your application. An application can include several forms.
The Object Inspector is used to change objects' properties and select event handlers.
The Code Explorer shows you the classes, variables, and routines in your unit and lets you navigate quickly.

Fig. 1.7 IDE

Delphi's development model is based on two-way tools. This means that you can move back and forth between visual design tools and text-based code editing. For example, after using the Form Designer to arrange buttons and other elements in a graphical interface, you can immediately view the form file that contains the textual description of your form. You can also manually edit any code generated by Delphi without losing access to the visual programming environment.

From the IDE, all your programming tools are within easy reach. You can design graphical interfaces, browse through class libraries, write code, and compile, test, debug, and manage projects without leaving the IDE.

Delphi's development model is based on two-way tools. This means that you can move back and forth between visual design tools and text-based code editing. For example, after using the Form Designer to arrange buttons and other elements in a graphical interface, you can immediately view the form file that contains the textual description of your form. You can also manually edit any code generated by Delphi without losing access to the visual programming environment.

From the IDE, all your programming tools are within easy reach. You can design graphical interfaces, browse through class libraries, write code, and compile, test, debug, and manage projects without leaving the IDE.

1.3.3. The Menus and Toolbars

The main window, which occupies the top of the screen, contains the main menu, toolbars, and Component palette.

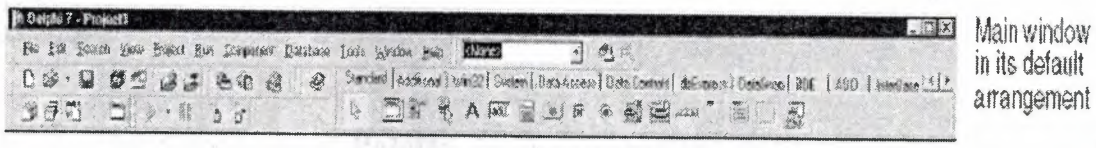

Delphi 7 - Project1
File Edit Search View Project Run Component Database Tools Window Help
Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase
Main window
in its default
arrangement

Fig. 1.8 Menus and Toolbars

Delphi's toolbars provide quick access to frequently used operations and commands. Most toolbar operations are duplicated in the drop-down menus.

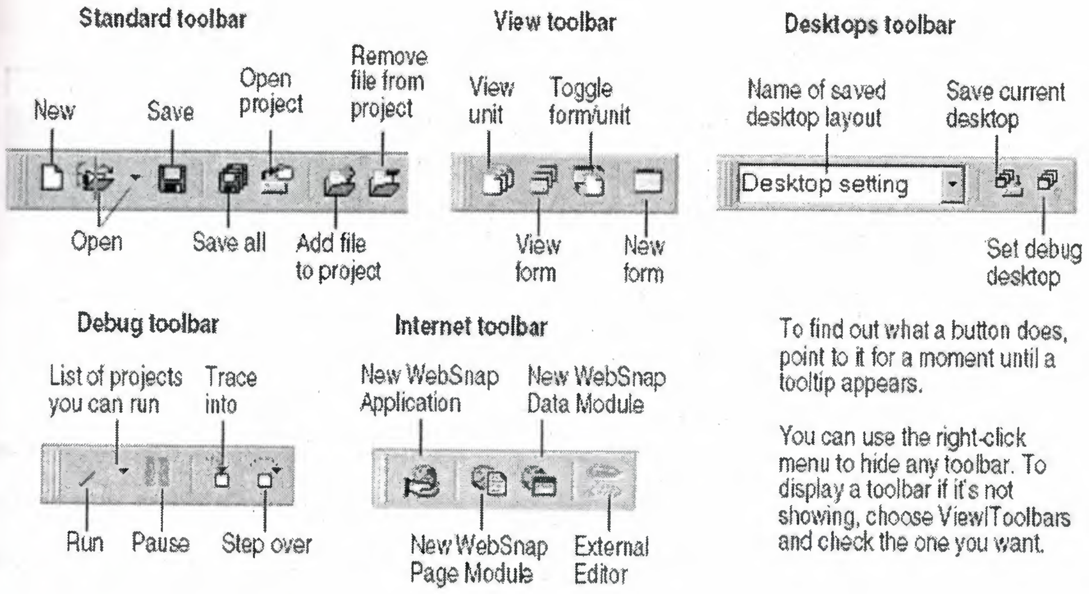


Fig. 1.9 Toolbars

Many operations have keyboard shortcuts as well as toolbar buttons. When a keyboard shortcut is available, it is always shown next to the command on the dropdown menu. You can right-click on many tools and icons to display a menu of commands appropriate to the object you are working with. These are called context menus.The toolbars are also customizable. You can add commands you want to them or move them to different locations.

1.3.4. The Component Palette, Form Designer, and Object Inspector

The Component palette, Form Designer, Object Inspector, and Object TreeView work together to help you build a user interface for your application. The Component palette includes tabbed pages with groups of icons representing visual or nonvisual components. The pages divide the components into various functional groups. For example, the Standard, Additional, and Win32 pages include windows controls such as an edit box and up/down button; the Dialogs page includes common dialog boxes to use for file operations such as opening and saving files.

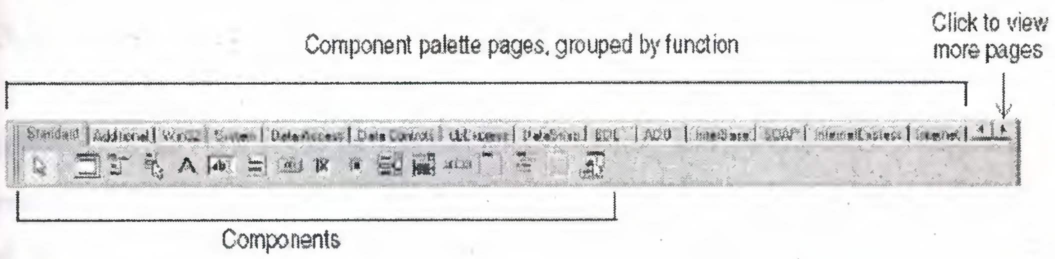


Fig. 1.10 Component Palatte

Each component has specific attributes properties, events, and methods that enable you to control your application. After you place components on the form, or Form Designer, you can arrange components the way they should look on your user interface. For the components you place on the form, use the Object Inspector to set design time properties, create event handlers, and filter visible properties and events, making the connection between your application's visual appearance and the code that makes your application run.

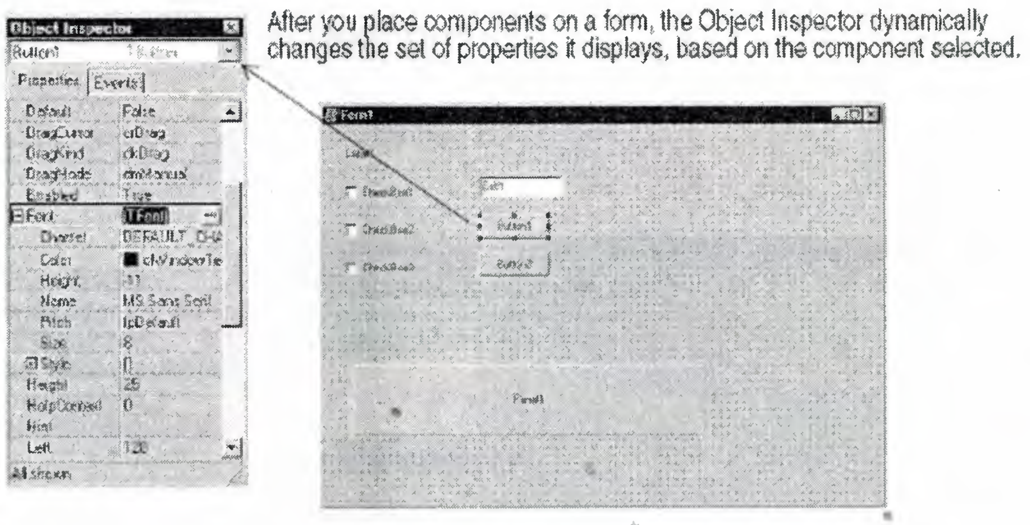

After you place components on a form, the Object Inspector dynamically changes the set of properties it displays, based on the component selected.
Object Inspector
Properties
Events
Form1

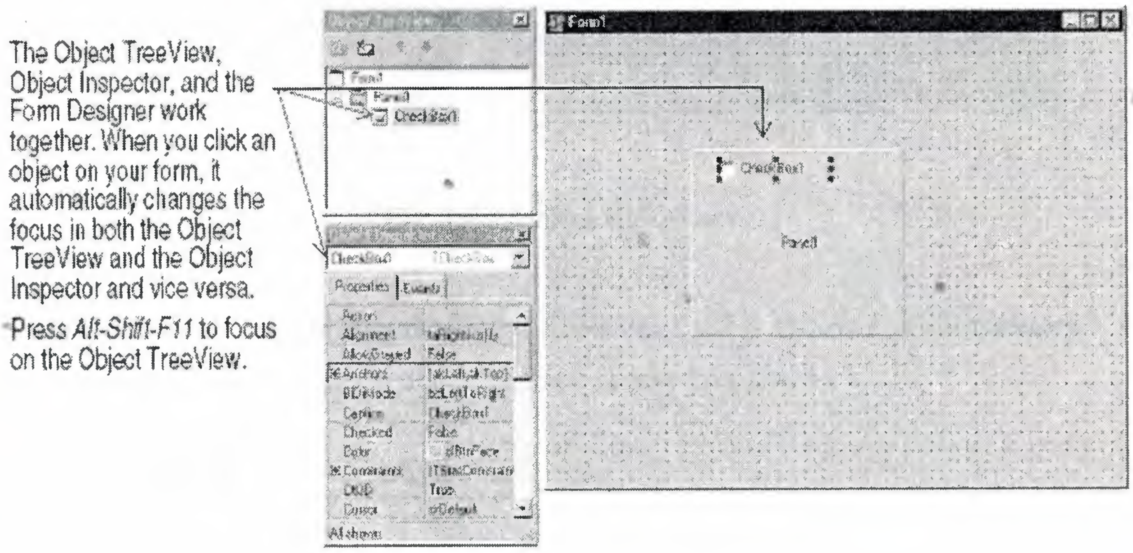
Fig. 1.11 Changing Set of Properties in Object Inspector

1.3.5. The Object TreeView

The Object TreeView displays a component's sibling and parent-child relationships in a hierarchical, or tree diagram. The tree diagram is synchronized with the Object Inspector and the Form Designer so that when you change focus in the Object TreeView, both the Object Inspector and the form change focus.

You can use the Object TreeView to change related components' relationships to each other. For example, if you add a panel and check box component to your form, the two components are siblings. But in the Object TreeView, if you drag the check box on top of the panel icon, the check box becomes the child of the panel.

If an object's properties have not been completed, the Object TreeView displays a red question mark next to it. You can also double-click any object in the tree diagram to open the Code editor to a place where you can write an event handler. If the Object TreeView isn't displayed, choose View|Object TreeView.



The Object TreeView, Object Inspector, and the Form Designer work together. When you click an object on your form, it automatically changes the focus in both the Object TreeView and the Object Inspector and vice versa.

Press *Alt-Shift-F11* to focus on the Object TreeView.

CheckBox1 TCheckBox

Properties	Events
Action	
Alignment	taRightJustify
AllowGrayed	False
⊞Anchors	[akLeft,akTop]
BiDiMode	bdLeftToRight
Caption	CheckBox1
Checked	False
Color	clBtnFace
⊞Constraints	(TSizeConstraints)
Ctl3D	True
Cursor	crDefault

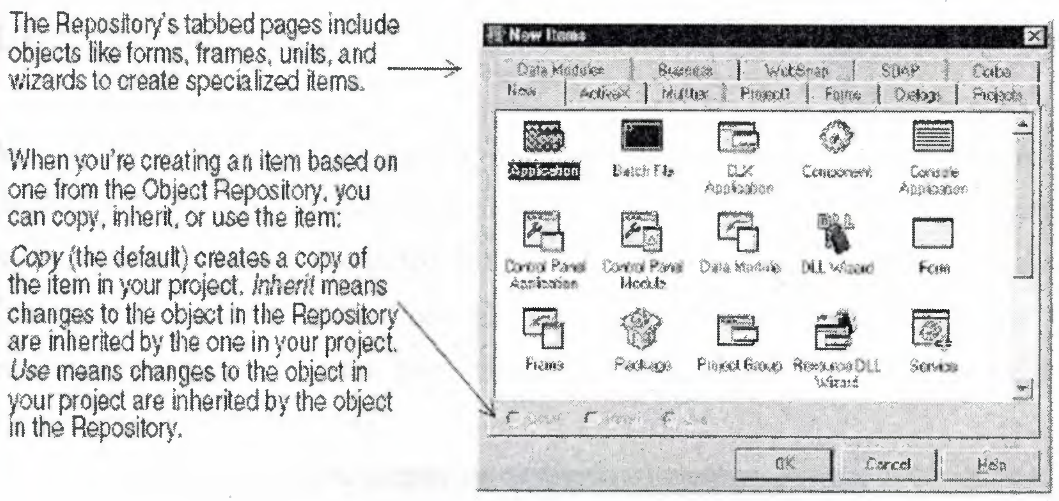
All shown

Fig. 1.12 Panel

The Object TreeView is especially useful for displaying the relationships between database objects.

1.3.6. 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|Other to display the New Items dialog box when you begin a project. The New Items dialog box is the same as the Object Repository. Check the Repository to see if it contains an object that resembles one you want to create.



The Repository's tabbed pages include objects like forms, frames, units, and wizards to create specialized items.

When you're creating an item based on one from the Object Repository, you can copy, inherit, or use the item:

Copy (the default) creates a copy of the item in your project. *Inherit* means changes to the object in the Repository are inherited by the one in your project. *Use* means changes to the object in your project are inherited by the object in the Repository.

Fig. 1.13 Object Repository

To edit or remove objects from the Object Repository, either choose Tools|Repository or right-click in the New Items dialog box and choose Properties.

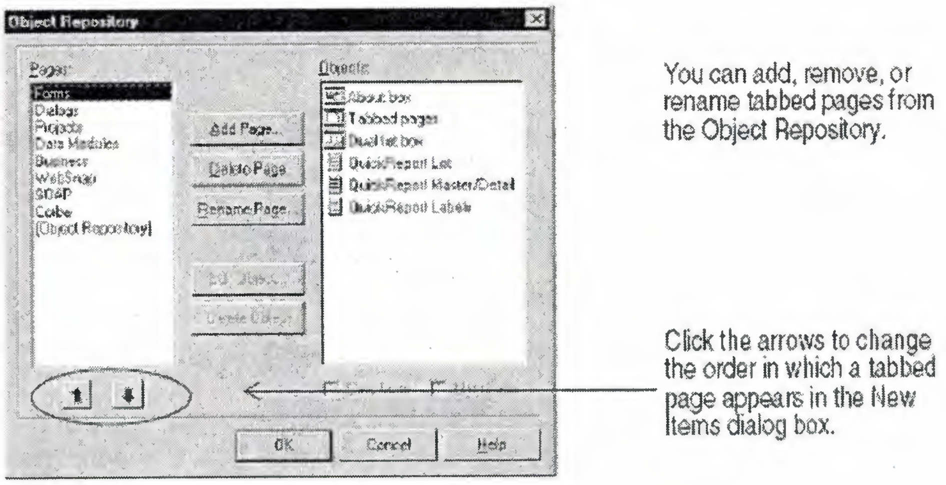
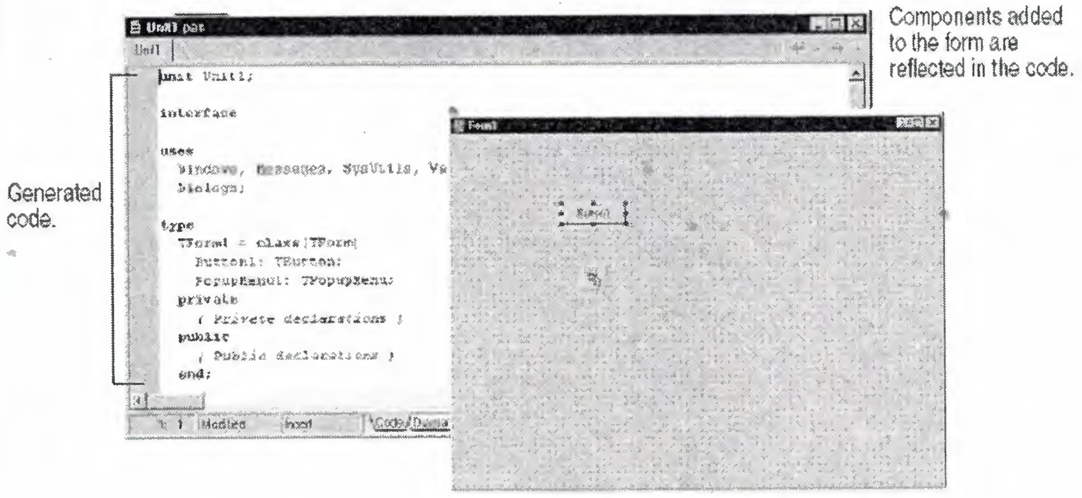


Fig. 1.14 Adding project and form templates to the Object Repository

1.3.7. The Code Editor

As you design the user interface for your application, Delphi generates the underlying Delphi code. When you select and modify the properties of forms and objects, your changes are automatically reflected in the source files. You can add code to your source files directly using the built-in Code editor, which is a full-featured ASCII editor. Delphi provides various aids to help you write code, including the Code Insight tools, class completion, and code browsing.



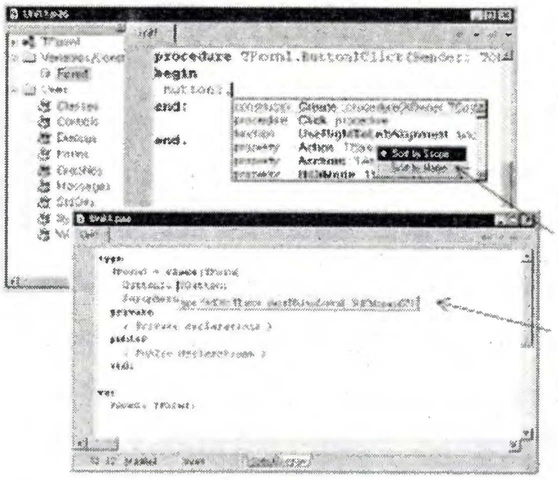
```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Va
  Dialogs;

type
  TForm1 = class(TForm)
    Button1: TButton;
    PopupMenu1: TPopupMenu;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

Fig. 1.15 Code Editor

1.3.7.1. Code Insight

The Code Insight tools display context-sensitive pop-up windows.

Tool	How it works
Code completion	Type a class name followed by a dot (.) to display a list of properties, methods, and events appropriate to the class, select it, and press Enter. In the **interface** section of your code you can select more than one item. Type the beginning of an assignment statement and press Ctrl+space to display a list of valid values for the variable. Type a procedure, function, or method name to bring up a list of arguments.
Code parameters	Type a method name and an open parenthesis to display the syntax for the method's arguments.
Tooltip expression evaluation	While your program has paused during debugging, point to any variable to display its current value.
Tooltip symbol insight	While editing code, point to any identifier to display its declaration.
Code templates	Press Ctrl+Jto see a list of common programming statements that you can insert into your code. You can create your own templates in addition to the ones supplied with Delphi.



```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.
end;

end.
```

With code completion, when you type the dot in Button1., Delphi displays a list of properties, methods, and events for the class. As you type, the list automatically filters to the selection that pertains to that class. Select an item on the list and press *Enter* to add it to your code.

Procedures and properties are colored as teal and functions as blue.

You can sort this list alphabetically by right-clicking and clicking Sort by Name.

The tooltip symbol insight displays declaration information for any identifier when you pass the mouse over it.

Fig. 1.16 Code Completion

To turn these tools on or off, choose Tools|Editor Options and click the Code Insight tab. Check or uncheck the tools in the Automatic features section.

1.3.8. Class Completion

Class completion generates skeleton code for classes. Place the cursor anywhere within a class declaration of the **interface** section of a unit and press Ctrl+Shift+C or right click and choose Complete Class at Cursor. Delphi automatically adds private **read** and **write** specifiers to the declarations for any properties that require them, then creates keleton 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 turn on class completion, choose Tools|Environment Options, click the Explorer tab, and make sure Finish incomplete properties is checked.

1.3.9. Code Browsing

While passing the mouse over the name of any class, variable, property, method, or other identifier, the pop-up menu called Tooltip Symbol Insight displays where the identifier is declared. Press Ctrl and the cursor turns into a hand, the identifier turns blue and is underlined, and you can click to jump to the definition of the identifier. The Code editor has forward and back buttons like the ones on Web browsers. As you jump to these definitions, the Code editor keeps track of where you've been in the code. You can click the drop-down arrows next to the Forward and Back buttons to ove forward and backward through a history of these references.

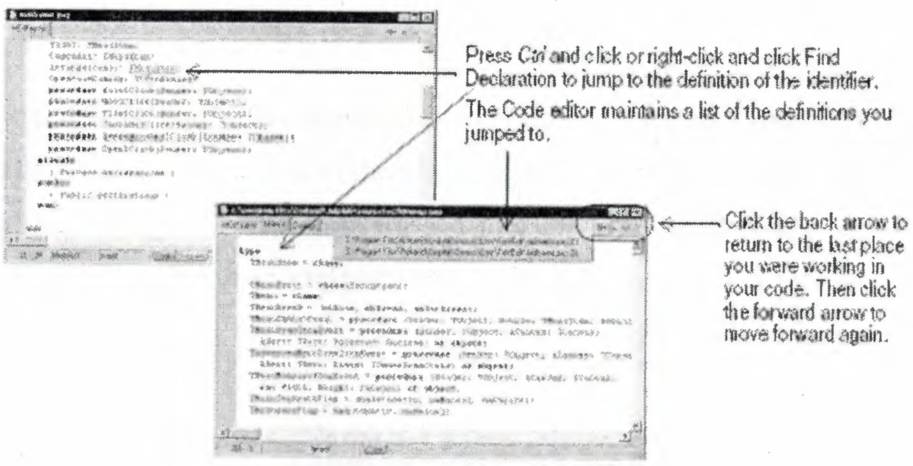

Press Ctrl and click or right-click and click Find Declaration to jump to the definition of the identifier.
The Code editor maintains a list of the definitions you jumped to.
Click the back arrow to return to the last place you were working in your code. Then click the forward arrow to move forward again.

Fig. 1.17 Code Editor

You can also move between the declaration of a procedure and its implementation by pressing Ctrl+Shift+↑ or Ctrl+Shift+↓.
To customize your code editing environment, see "Customizing the Code Editor".

1.3.10. The Diagram Page

The bottom of the Code editor may contain one or more tabs, depending on which edition of Delphi you have. The Code page, where you write all your code, appears in the foreground by default. The Diagram page displays icons and connecting lines representing the relationships between the components you place on a form or data module. These relationships include siblings, parent to children, or components to properties.

To create a diagram, click the Diagram page. From the Object TreeView, simply drag one or multiple icons to the Diagram page to arrange them vertically. To arrange them horizontally, press Shift while dragging. When you drag icons with parentchildren or component-property dependencies onto the page, the lines, or connectors, that display the dependent relationships are automatically added. For example, if you add a dataset component to a data module and drag the dataset icon plus its property icons to the Diagram page, the property connector automatically connects the property icons to the dataset icon.

For components that don't have dependent relationships but where you want to show one, use the toolbar buttons at the top of the Diagram page to add one of four connector types, including allude, property, master/detail, and lookup. You can also add comment blocks that connect to each other or to a relevant icon.

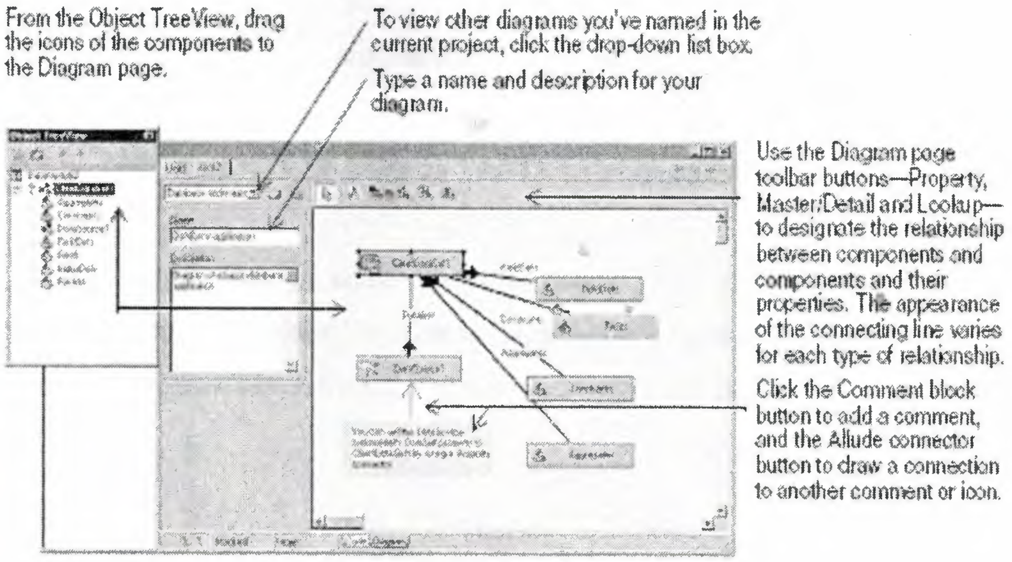
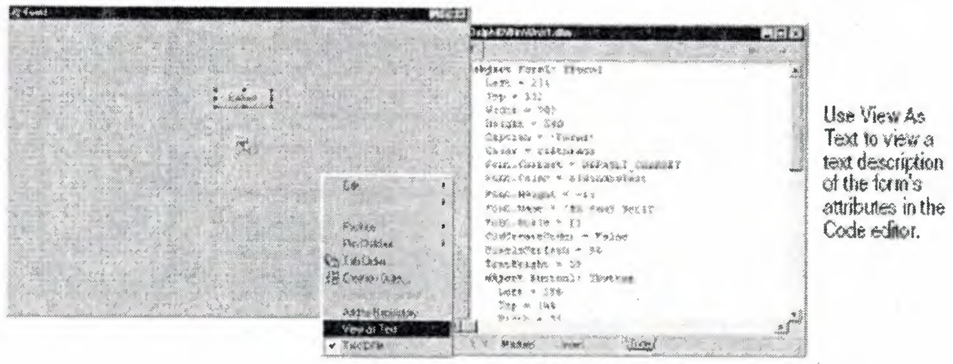

From the Object TreeView, drag the icons of the components to the Diagram page.
To view other diagrams you've named in the current project, click the drop-down list box.
Type a name and description for your diagram.
Use the Diagram page toolbar buttons—Property, Master/Detail and Lookup—to designate the relationship between components and components and their properties. The appearance of the connecting line varies for each type of relationship.
Click the Comment block button to add a comment, and the Allude connector button to draw a connection to another comment or icon.

Fig. 1.18 Diagram Page Toolbar Button

You can type a name and description for your diagram, save the diagram, and print it when you are finished.

1.3.11. *Viewing Form Code*

Forms are a very visible part of most Delphi projects they are where you design the user interface of an application. Normally, you design forms using Delphi's visual tools, and Delphi stores the forms in form files. Form files (.dfm, or .xfm for a CLX application) describe each component in your form, including the values of all persistent properties. To view and edit a form file in the Code editor, right-click the form and select View as Text. To return to the graphic view of your form, right-click and choose View as Form.



View as Text
Use View As Text to view a text description of the form's attributes in the Code editor.

```
object Form1: TForm1
  Left = [illegible]
  Top = [illegible]
  Width = [illegible]
  Height = [illegible]
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = [illegible]
    Top = [illegible]
    [illegible]
```

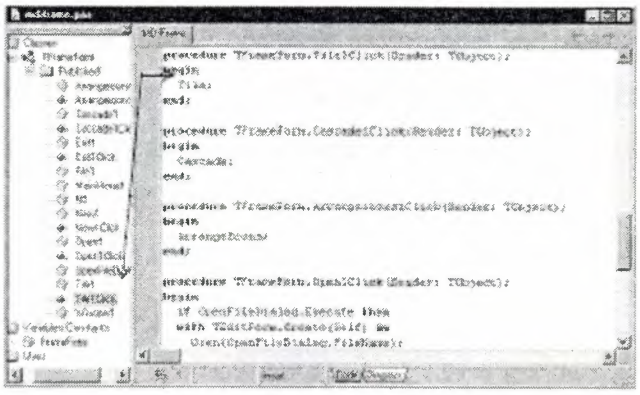
Fig. 1.19 View as Text Description of Form

You can save form files in either text (the default) or binary format. Choose Tools| Environment Options, click the Designer page, and check or uncheck the New forms as text check box to designate which format to use for newly created forms.

1.3.12. The Code Explorer

When you open Delphi, the Code Explorer is docked to the left of the Code editor window, depending on whether the Code Explorer is available in the edition of Delphi you have. The Code Explorer displays the table of contents as a tree diagram for the source code open in the Code editor, listing the types, classes, properties, methods, global variables, and routines defined in your unit. It also shows the other units listed in the **uses** clause.

You can use the Code Explorer to navigate in the Code editor. For example, if you double-click a method in the Code Explorer, a cursor jumps to the definition in the class declaration in the interface part of the unit in the Code editor.



```
procedure TFrameForm.File1Click(Sender: TObject);
begin
  Tile;
end;

procedure TFrameForm.Cascade1Click(Sender: TObject);
begin
  Cascade;
end;

procedure TFrameForm.ArrangeIcons1Click(Sender: TObject);
begin
  ArrangeIcons;
end;

procedure TFrameForm.Open1Click(Sender: TObject);
begin
  if OpenFileDialog.Execute then
  with TEditForm.Create(Self) do
    Open(OpenFileDialog.FileName);
```

Double-click an item in the Code Explorer and the cursor moves to that item's implementation in the Code editor. Press *Ctrl+Shift+E* to move the cursor back and forth between the last place you were in the Code Explorer and Code editor.

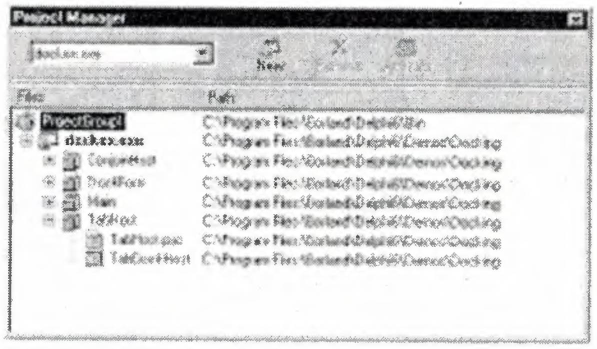
Each item in the Code Explorer has an icon that designates its type.

Fig. 1.20 Code Explorer

To configure how the Code Explorer displays its contents, choose Tools|Environment Options and click the Explorer tab.

1.3.13. The Project Manager

When you first start Delphi, it automatically opens a new project. A project includes several files that make up the application or DLL you are going to develop. You can view and organize these files such as form, unit, resource, object, and library files in a project management tool called the Project Manager. To display the Project Manager, choose View|Project Manager.



Project Manager

New

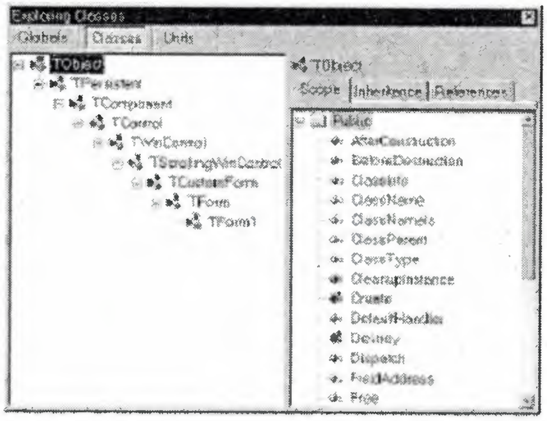
Files	Path
ProjectGroup1	C:\Program Files\Borland\Delphi6\Bin
dockex.exe	C:\Program Files\Borland\Delphi6\Demos\Docking
ConjoinHost	C:\Program Files\Borland\Delphi6\Demos\Docking
DockForm	C:\Program Files\Borland\Delphi6\Demos\Docking
Main	C:\Program Files\Borland\Delphi6\Demos\Docking
TabHost	C:\Program Files\Borland\Delphi6\Demos\Docking
TabHost.pas	C:\Program Files\Borland\Delphi6\Demos\Docking
TabDockHost	C:\Program Files\Borland\Delphi6\Demos\Docking

Fig. 1.21 Project Manager

You can use the Project Manager to combine and display information on related projects into a single project group. By organizing related projects into a group, such as multiple executables, you can compile them at the same time. To change project options, such as compiling a project, you can use Setting project options.

1.3.14. The Project Browser

The Project Browser examines a project in detail. The Browser displays classes, units, and global symbols (types, properties, methods, variables, and routines) your project declares or uses in a tree diagram. Choose View|Browser to display the Project Browser.



The Project Browser has two resizeable panes: the Inspector pane (on the left) and the Details pane. The Inspector pane has three tabs for globals, classes, and units.

Globals displays classes, types, properties, methods, variables, and routines.

Classes displays classes in a hierarchical diagram.

Units displays units, identifiers declared in each unit, and the other units that use and are used by each unit.

Fig. 1.22 Project Browser

By default, the Project Browser displays the symbols from units in the current project only. You can change the scope to display all symbols available in Delphi. Choose Tools|Environment Options, and on the Explorer page, check All symbols.

1.3.15. To-do lists

To-do lists record items that need to be completed for a project. You can add ojectwide items to a list by adding them directly to the list, or you can add specific items directly in the source code. Choose View|To-Do List to add or view information associated with a project.

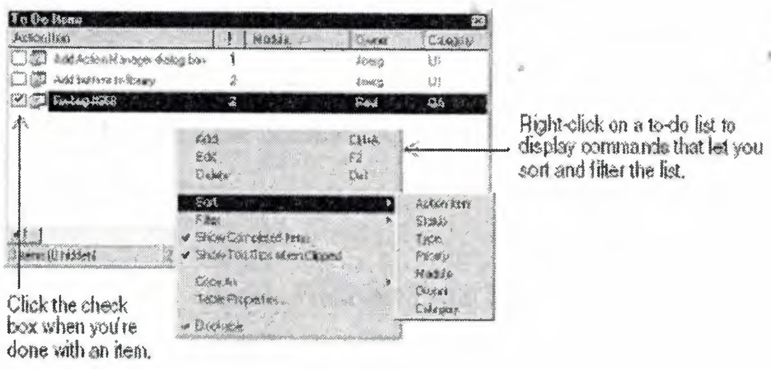


Fig. 1.23 To-Do List

1.4. Programming With Delphi

The following sections provide an overview of software development with Delphi, including creating a project, working with forms, writing code, and compiling, debugging, deploying, and internationalizing applications, and including the types of projects you can develop.

1.4.1. Creating a Project

A project is a collection of files that are either created at design time or generated when you compile the project source code. When you first start Delphi, a new project opens. It automatically generates a project file (Project1.dpr), unit file (Unit1.pas), and resource file (Unit1.dfm; Unit1.xfm for CLX applications), among others. If a project is already open but you want to open a new one, choose either File|New| Application or File|New|Other and double-click the Application icon. File|New| Other opens the Object Repository, which provides additional forms, modules, and frames as well as predesigned templates such as dialog boxes to add to your project. When you start a project, you have to know what you want to develop, such as an application or DLL.

1.4.2. Adding Data Modules

A data module is a type of form that contains nonvisual components only. Nonvisual components can be placed on ordinary forms alongside visual components. But if you plan on reusing groups of database and system objects, or if you want to isolate the parts of your application that handle database connectivity and business rules, data modules provide a convenient organizational tool.

To create a data module, choose File|New|Data Module. Delphi opens an empty data module, which displays an additional unit file for the module in the Code editor, and adds the module to the current project as a new unit. Add nonvisual components to a data module in the same way as you would to a form.

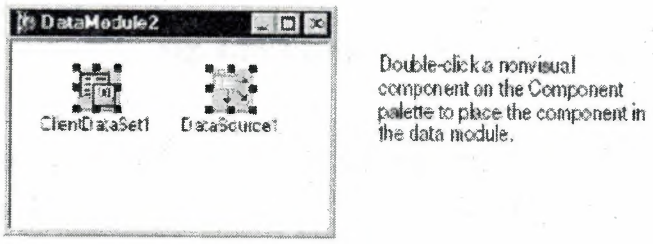

DataModule2
ClientDataSet1
DataSource1
Double-click a nonvisual component on the Component palette to place the component in the data module.

Fig. 1.24 Adding Data Modules

When you reopen an existing data module, Delphi displays its components.

1.4.3. Building the user interface

With Delphi, you first create a user interface (UI) by selecting components from the Component palette and placing them on the main form.

1.4.4. Placing components on a form

To place components on a form, either:

1 Double-click the component; or

2 Click the component once and then click the form where you want the component to appear.

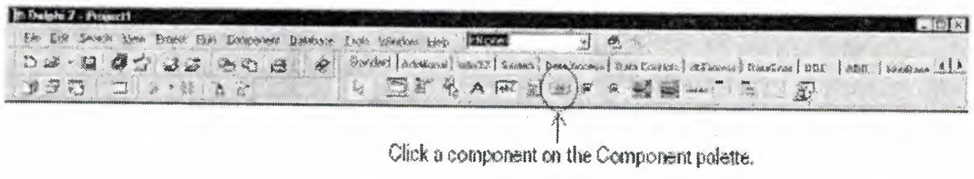


Fig. 1.25 Component Palette

Select the component and drag it to wherever you want on the form.

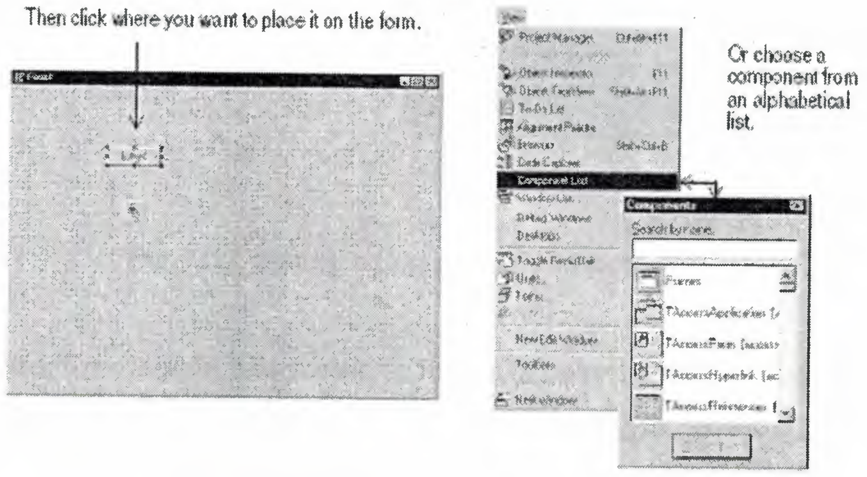

Then click where you want to place it on the form.
Or choose a component from an alphabetical list.

Fig. 1.26 Component List

1.4.5. Setting Component Properties

After you place components on a form, set their properties and code their event handlers. Setting a component's properties changes the way a component appears and behaves in your application. When a component is selected on a form, its properties and events are displayed in the Object Inspector.

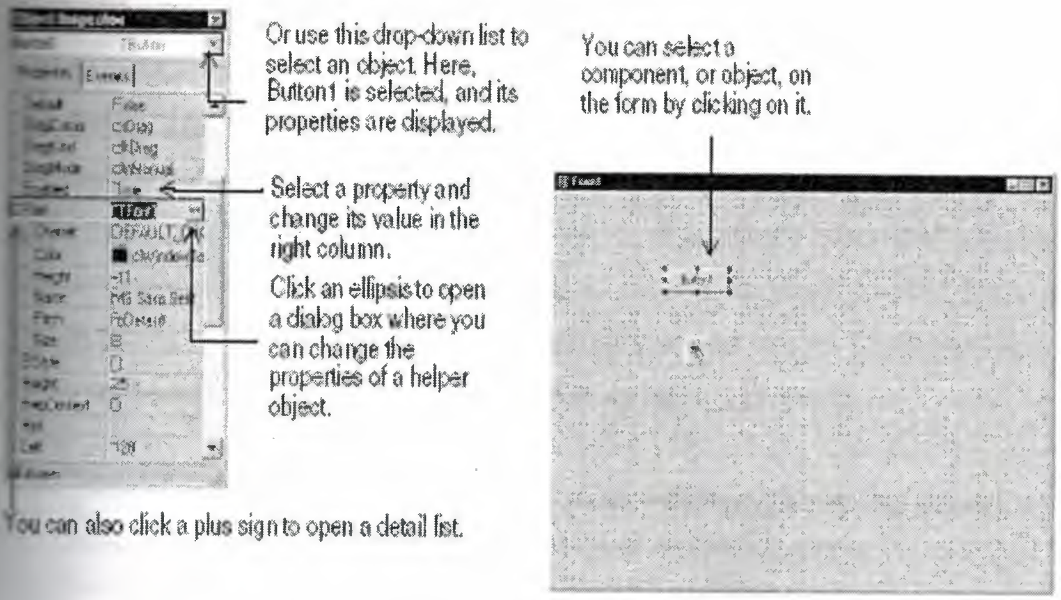

Or use this drop-down list to select an object. Here, Button1 is selected, and its properties are displayed.
You can select a component, or object, on the form by clicking on it.
Select a property and change its value in the right column.
Click an ellipsis to open a dialog box where you can change the properties of a helper object.
You can also click a plus sign to open a detail list.

Fig. 1.27 Setting a Component's Properties

Many properties have simple values such as names of colors, True or False, and integers. For Boolean properties, you can double click the word to toggle between True and False. Some properties have associated property editors to set more complex values. When you click on such a property value, you'll see an ellipsis. For some properties, such as size, enter a value.

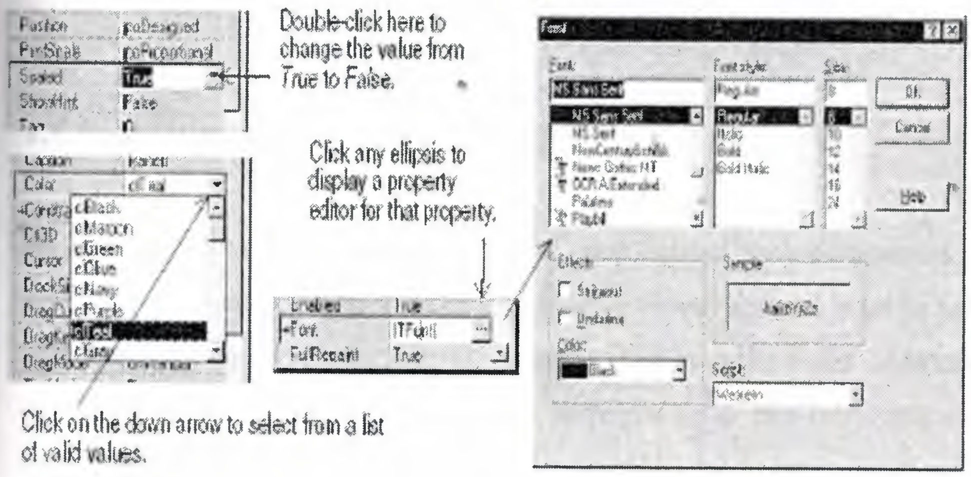
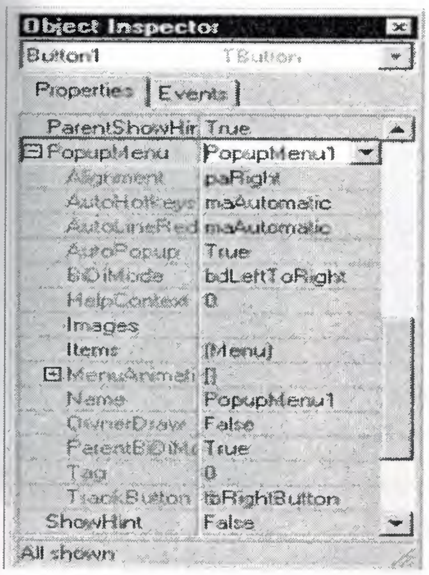

Double-click here to change the value from True to False.
Click any ellipsis to display a property editor for that property.
Click on the down arrow to select from a list of valid values.

Fig. 1.28 Font Properties

When more than one component is selected in the form, the Object Inspector displays all properties that are shared among the selected components. The Object Inspector also supports expanded inline component references. This provides access to the properties and events of a referenced component without having to select the referenced component itself. For example, if you add a button and pop-up menu component to your form, when you select the button component, in the Object Inspector you can set the PopupMenu property to PopupMenu1, which displays all of the pop-up menu's properties.



Object Inspector

Button1 TButton

Properties | Events

ParentShowHir	True
⊟ PopupMenu	PopupMenu1
Alignment	paRight
AutoHotkeys	maAutomatic
AutoLineRed	maAutomatic
AutoPopup	True
BiDiMode	bdLeftToRight
HelpContext	0
Images	
Items	(Menu)
⊞ MenuAnimati	[]
Name	PopupMenu1
OwnerDraw	False
ParentBiDiMo	True
Tag	0
TrackButton	tbRightButton
ShowHint	False

All shown

Set the Button component's PopupMenu property to PopupMenu1, and all of the popup menu's properties appear when you click the plus sign (+). Inline component references are colored red, and their subproperties are colored green.

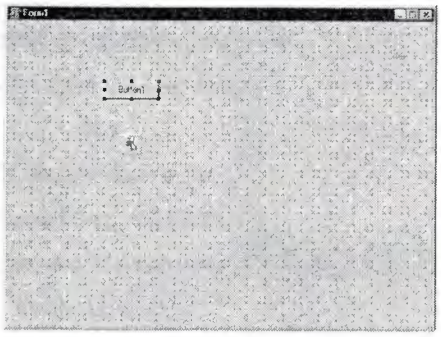


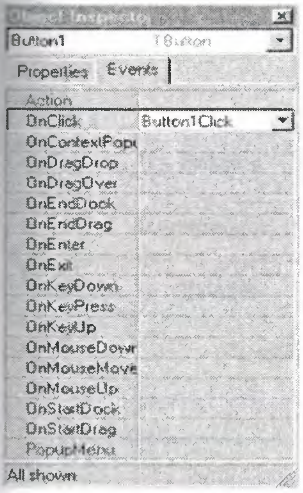
Fig. 1.29 Setting the Button Component's Popup Menu

1.4.6. Writing Code

An integral part of any application is the code behind each component. While Delphi's RAD environment provides most of the building blocks for you, such as preinstalled visual and nonvisual components, you will usually need to write event handlers, methods, and perhaps some of your own classes. To help you with this task, you can choose from thousands of objects in the class library.

1.4.6.1. Writing Event Handlers

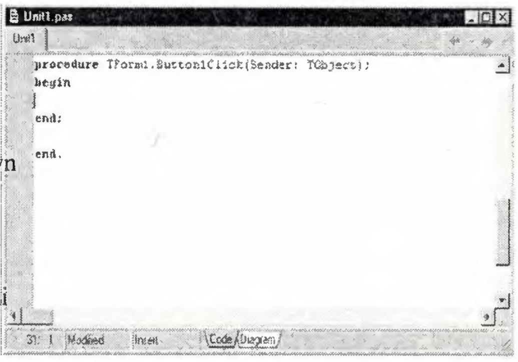
Your code may need to respond to events that might occur to a component at runtime. An event is a link between an occurrence in the system, such as clicking a button, and a piece of code that responds to that occurrence. The responding code is an event handler. This code modifies property values and calls methods. To view predefined event handlers for a component on your form, select the component and, on the Object Inspector, click the Events tab.



OnClick	Button1Click
OnContextPopu	
OnDragDrop	
OnDragOver	
OnEndDock	
OnEndDrag	
OnEnter	
OnExit	
OnKeyDown	
OnKeyPress	
OnKeyUp	
OnMouseDown	
OnMouseMove	
OnMouseUp	
OnStartDock	
OnStartDrag	
PopupMenu	

Here, Button1 is selected and its type is displayed: *TButton*. Click the Events tab in the Object Inspector to see the events that the Button component can handle.

Select an existing event handler from the dropdown list.
Or double-click in the value column, and Delphi generates skeleton code for the new event handler.



```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;

end.
```

Fig. 1.30 Event Handlers

1.4.6.2. Using The Component Library

Delphi comes with a component library made up of objects, some of which are also components or controls, that you use when writing code. You can use VCL components for Windows applications and CLX components for Windows and Linux applications. The component library includes objects that are visible at Runtime such as edit controls, buttons, and other user interface elements as well-as non visual controls like datasets and timers. The following diagram shows some of the principal classes that make up the VCL hierarchy. The CLX hierarchy is similar.

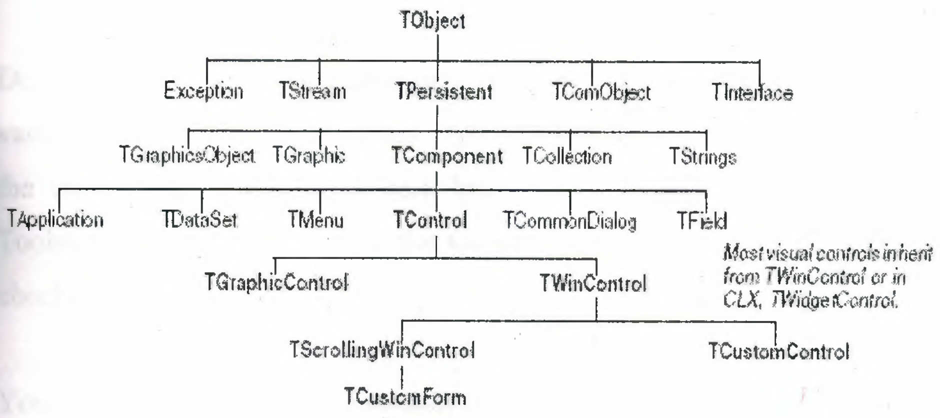


Fig. 1.31 Component Library

Objects descended from TComponent have properties and methods that allow them to be installed on the Component palette and added to Delphi forms and data modules Because the components are hooked into the IDE, you can use tools like the Form Designer to develop applications quickly.

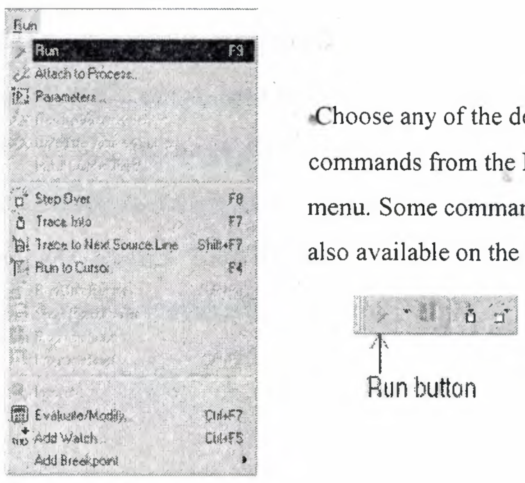
Components are highly encapsulated. For example, buttons are preprogrammed to respond to mouse clicks by firing OnClick events. If you use a button control, you don't have to write code to handle generated events when the button is clicked; you are responsible only for the application logic that executes in response to the click itself. Most editions of Delphi come with the component library source code and examples of Delphi programming techniques.

1.4.7. Compiling and Debugging Projects

After you have written your code, you will need to compile and debug your project. With Delphi, you can either compile your project first and then separately debug it, or you can compile and debug in one step using the integrated debugger. To compile your program with debug information, choose Project|Options, click the Compiler page, and make sure Debug information is checked.

Delphi uses an integrated debugger so that you can control program execution, watch variables, and modify data values. You can step through your code line by line, examining the state of the program at each breakpoint. To use the integrated debugger, choose Tools|Debugger Options, click the General page, and make sure Integrated debugging is checked.

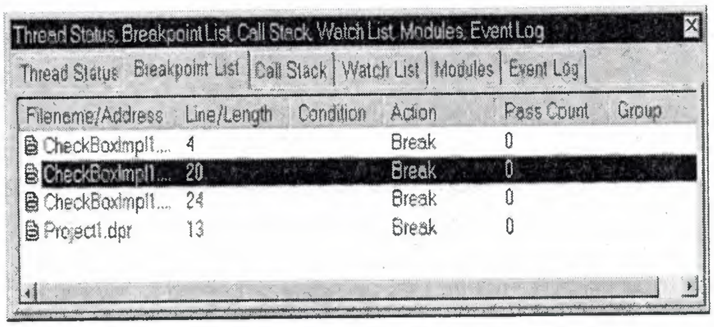
You can begin a debugging session in the IDE by clicking the Run button on the Debug toolbar, choosing Run|Run, or pressing F9.



Choose any of the debugging commands from the Run menu. Some commands are also available on the toolbar.

Fig. 1.32 Compiling and Debugging

With the integrated debugger, 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. Not all debugger views are available in all editions of Delphi.


Thread Status, Breakpoint List, Call Stack, Watch List, Modules, Event Log
Thread Status
Breakpoint List
Call Stack
Watch List
Modules
Event Log

Filename/Address	Line/Length	Condition	Action	Pass Count	Group
CheckBoxImpl1....	4		Break	0	
CheckBoxImpl1....	20		Break	0	
CheckBoxImpl1....	24		Break	0	
Project1.dpr	13		Break	0	

You can combine several debugging windows for easier use.

Fig. 1.33 Debugging

Once you set up your desktop as you like it for debugging, you can save the settings as the debugging or runtime desktop. This desktop layout will be used whenever you are debugging any application.

1.4.8. Deploying Applications

You can make your application available for others to install and run by deploying it When you deploy an application, you will need all the required and supporting files, such as the executables, DLLs, package files, and helper applications. Delphi comes bundled with a setup toolkit called InstallShield Express that helps you create an installation program with these files. To install InstallShield Express, from the Delphi setup screen, choose InstallShield Express Custom Edition for Delphi.

1.4.9. Internationalizing Applications

Delphi offers several features for internationalizing and localizing applications. The IDE and the VCL support input method editors (IMEs) and extended character sets to internationalize your project. Delphi includes a translation suite, not available in all editions of Delphi, for software localization and simultaneous development for different locales. With the translation suite, you can manage multiple localized versions of an application as part of a single project.

The translation suite includes three integrated tools:

• Resource DLL wizard, a DLL wizard that generates and manage resource DLLs.

• Translation Manager, a table for viewing and editing translated resources.

• Translation Repository, a shared database to store translations.

To open the Resource DLL wizard, choose File|New|Other and double-click the Resource DLL Wizard icon. To configure the translation tools, choose Tools| Translation Tools Options.

1.4.10. Types of Projects

All editions of Delphi support general-purpose 32-bit Windows programming, DLLs, packages, custom components, multithreading, COM (Component Object Model) and automation controllers, and multiprocess debugging. Some editions support server applications such as Web server applications, database applications, COM servers, multi-tiered applications, CORBA, and decision-support systems.

1.4.10.1. CLX Applications

You can use Delphi, to develop cross-platform 32-bit applications that run on both the Windows and Linux operating systems. To develop a CLX application, choose File| New|CLX Application. The IDE is similar to that of a regular Delphi application, except that only the components and items you can use in a CLX application appear on the

Component palette and in the Object Repository. Windows-specific features supported on Delphi will not port directly to Linux environments.

1.4.10.2. Database Applications

Delphi offers a variety of database and connectivity tools to simplify the development of database applications. To create a database application, first design your interface on a form using the Data Controls page components. Second, add a data source to a data module using the Data Access page. Third, to connect to various database servers, add a dataset and data connection component to the data module from the previous or corresponding pages of the following connectivity tools:

• dbExpress is a collection of database drivers for cross-platform applications that provide fast access to SQL database servers, including DB2, Informix, InterBase, MSSQL, MySQL, and Oracle. With a dbExpress driver, you can access databases using unidirectional datasets.

• The Borland Database Engine (BDE) is a collection of drivers that support many popular database formats, including dBASE, Paradox, FoxPro, Microsoft Access, and any ODBC data source.

• ActiveX Data Objects (ADO) is Microsoft's high-level interface to any data source, including relational and nonrelational databases, e-mail and file systems, text and graphics, and custom business objects.

• InterBase Express (IBX) components are based on the custom data access Delphi component architectures. IBX applications provide access to advanced InterBase features and offer the highest performance component interface for InterBase 5.5 and later. IBX is compatible with Delphi's library of data-aware components. Certain database connectivity tools are not available in all editions of Delphi.

1.4.11. BDE Administrator

Use the BDE Administrator (BDEAdmin.exe) to configure BDE drivers and set up the aliases used by data-aware VCL controls to connect to databases.

1.4.12. SQL Explorer (Database Explorer)

The SQL Explorer (DBExplor.exe) lets you browse and edit databases. You can use it to create database aliases, view schema information, execute SQL queries, and maintain data dictionaries and attribute sets.

1.4.13. Database Desktop

The Database Desktop (DBD32.exe) lets you create, view, and edit Paradox and dBase database tables in a variety of formats.

1.4.14. Data Dictionary

When you use the BDE, the Data Dictionary provides a customizable storage area, independent of your applications, where you can create extended field attribute sets that describe the content and appearance of data. The Data Dictionary can reside on a remote server to share additional information.

1.4.15. Custom Components

The components that come with Delphi are preinstalled on the Component palette and offer a range of functionality that should be sufficient for most of your development needs. You could program with Delphi for years without installing a new component, but you may sometimes want to solve special problems or display particular kinds of behavior that require custom components. Custom components promote code reuse and consistency across applications. You can either install custom components from third-party vendors or create your own. To create a new component, choose Component|New Component to display the New Component wizard.

1.4.16. DLLs

Dynamic-link libraries (DLLs) are compiled modules containing routines that can be called by applications and by other DLLs. A DLL contains code or resources typically used by more than one application.

1.4.17. COM and ActiveX

Delphi supports Microsoft's COM standard and provides wizards for creating ActiveX controls. Choose File|New|Other and click the ActiveX tab to access the wizards. Sample ActiveX controls are installed on the ActiveX page of the Component palette. Numerous COM server components are provided on the Servers tab of the Component palette. You can use these components as if they were VCL components. For example, you can place one of the Microsoft Word components onto a form to bring up an instance of Microsoft Word within an application interface.

1.4.18. Type Libraries

Type libraries are files that include information about data types, interfaces, member functions, and object classes exposed by an ActiveX control or server. By including a type library with your COM application or ActiveX library, you make information about these entities available to other applications and programming tools. Delphi provides a Type Library editor for creating and maintaining type libraries.

1.5. Customizing The Desktop

This chapter explains some of the ways you can customize the tools in Delphi's IDE.

1.5.1. Organizing Your Work Area

The IDE provides many tools to support development, so you'll want to reorganize your work area for maximum convenience. You can rearrange menus and toolbars, combine tool windows, and save your new desktop layout.

1.5.2. Arranging Menus and Toolbars

In the main window, you can reorganize the menu, toolbars, and Component palette by clicking the grabber on the left-hand side of each one and dragging it to another location.

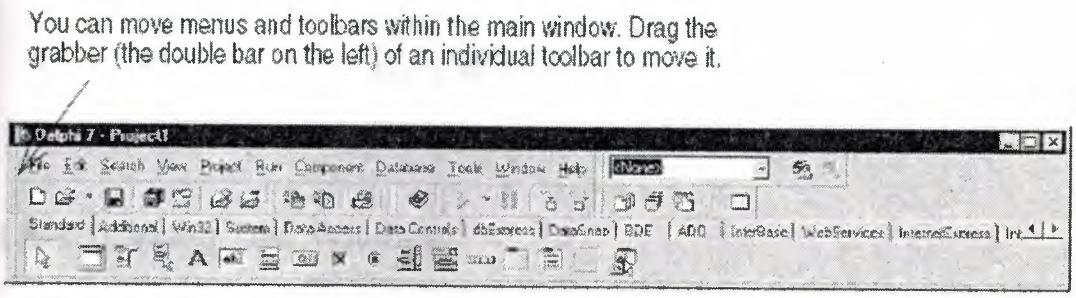

You can move menus and toolbars within the main window. Drag the grabber (the double bar on the left) of an individual toolbar to move it.
Delphi 7 - Project1
File Edit Search View Project Run Component Database Tools Window Help
Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase WebServices InternetExpress

Fig. 1.34 Arranging Menus and Toolbars

You can separate parts from the main window and place them elsewhere on the screen or remove them from the desktop altogether. This is useful if you have a dual monitor setup.

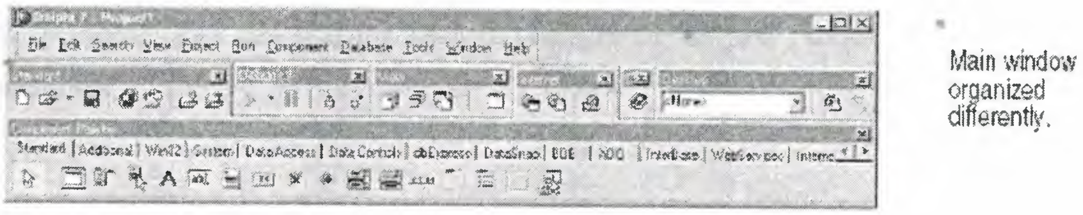
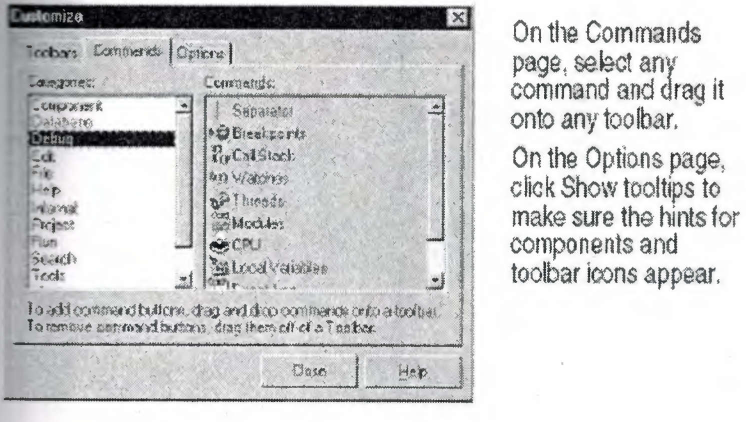

Delphi 7 - Project1
File Edit Search View Project Run Component Database Tools Window Help
Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase WebServices Internet
Main window organized differently.

Fig. 1.35 Main Window

You can add or delete tools from the toolbars by choosing View|Toolbars|Customize. Click the Commands page, select a category, select a command, and drag it to the toolbar where you want to place it.



To add command buttons, drag and drop commands onto a toolbar.
To remove command buttons, drag them off of a Toolbar.

On the Commands page, select any command and drag it onto any toolbar.

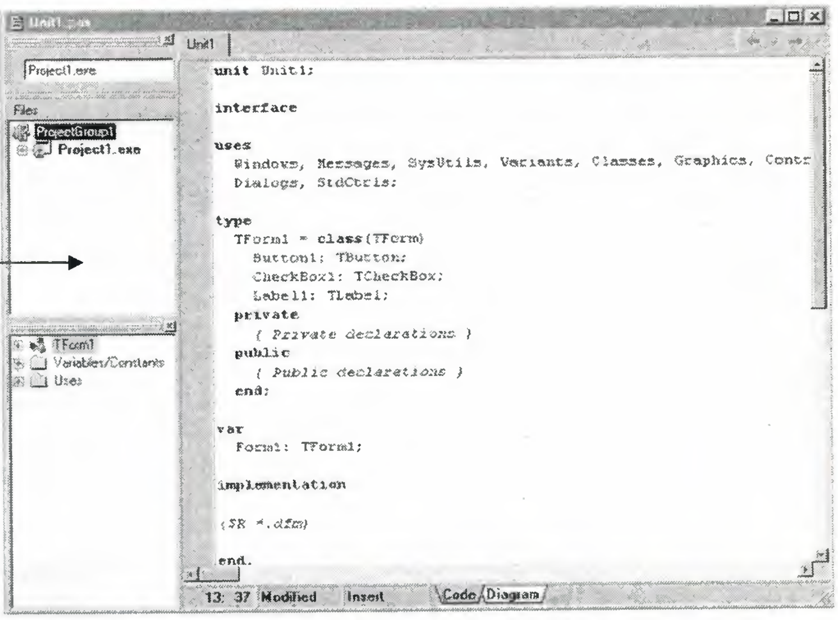
On the Options page, click Show tooltips to make sure the hints for components and toolbar icons appear.

Fig. 1.36 Customize Command

1.5.3. Docking Tool Windows

You can open and close individual tool windows and arrange them on the desktop as you wish. Many windows can also be docked to one another for easy management. Docking which means attaching windows to each other so that they move Together helps you use screen space efficiently while maintaining fast access to tools. From the View menu, you can bring up any tool window and then dock it directly to another. For example, when you first open Delphi in its default configuration, the Code Explorer is docked to the left of the Code editor. You can add the Project Manager to the first two to create three docked windows.

Here the Project Manager and Code Explorer are docked to the Code editor. You can combine, or "dock" windows with either grabbers, as on the right, or tabs, as on page 5-4.

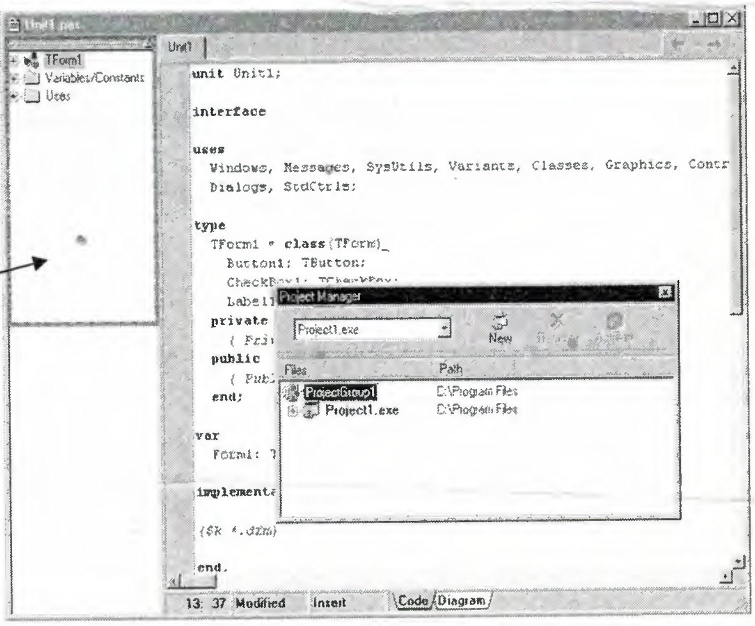


```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Contr
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    CheckBox1: TCheckBox;
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Fig. 1.37 Docking Tool Windows

To dock a window, click its title bar and drag it over the other window. When the drag outline narrows into a rectangle and it snaps into a corner, release the mouse. The two windows snap together.

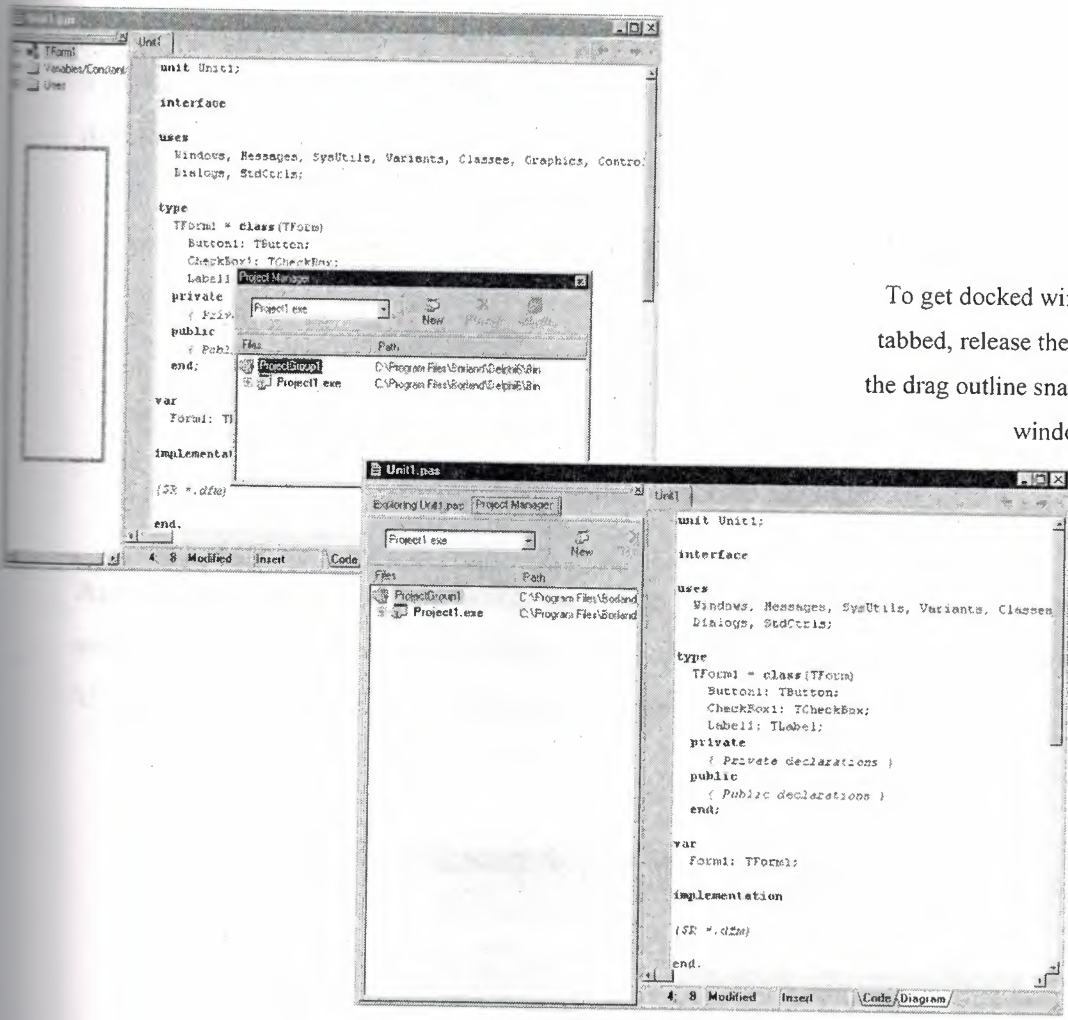
To get docked windows with grabbers, release the mouse when the drag outline snaps to the window's corner.



```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Contr
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    CheckBox1: TCheckBox;
    Label1
  private
    { Pri
  public
    { Pub
  end;

var
  Form1: T

implementa

{$R *.dfm}

end.
```

Fig. 1.38 Two Windows Snap Together

You can also dock tools to form tabbed windows.



```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    CheckBox1: TCheckBox;
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

To get docked windows that are tabbed, release the mouse *before* the drag outline snaps to the other window's corner.

Fig. 1.39 Docking Tools to Form

To undock a window, double click its grabber or tab, or click and drag the tab outside of the docking area. To turn off automatic docking, either press the Ctrl key while moving windows around the screen, or choose Tools|Environment Options, click the references page, and uncheck the Auto drag docking check box.

1.5.4. Saving Desktop Layouts

You can customize and save your desktop layout. The Desktops toolbar in the IDE includes a pick list of the available desktop layouts and two icons to make it easy to customize the desktop.

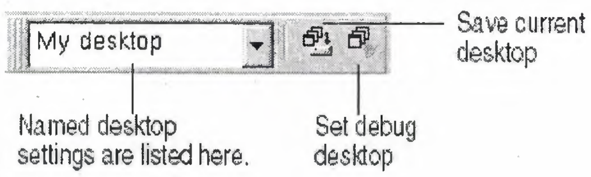


Fig. 1.40 Saving Desktop Layouts

Arrange the desktop as you want, including displaying, sizing, and docking particular windows. On the Desktops toolbar, click the Save current desktop icon or choose View|Desktops|Save Desktop, and enter a name for your new layout.

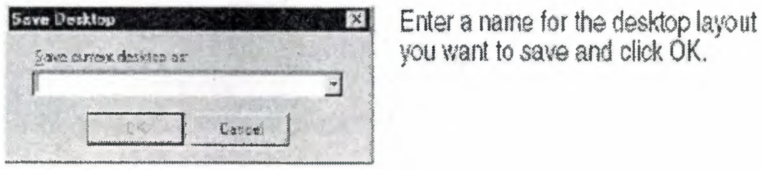

Save Desktop
Save current desktop as:
OK
Cancel
Enter a name for the desktop layout you want to save and click OK.

Fig. 1.41 Entering the Name For Desktop

1.5.5. Customizing The Component Palette

In its default configuration, the Component palette displays many useful objects organized functionally onto tabbed pages. You can customize the Component palette by:

- Hiding or rearranging components.
- Adding, removing, rearranging, or renaming pages.
- Creating component templates and adding them to the palette.

• Installing new components.

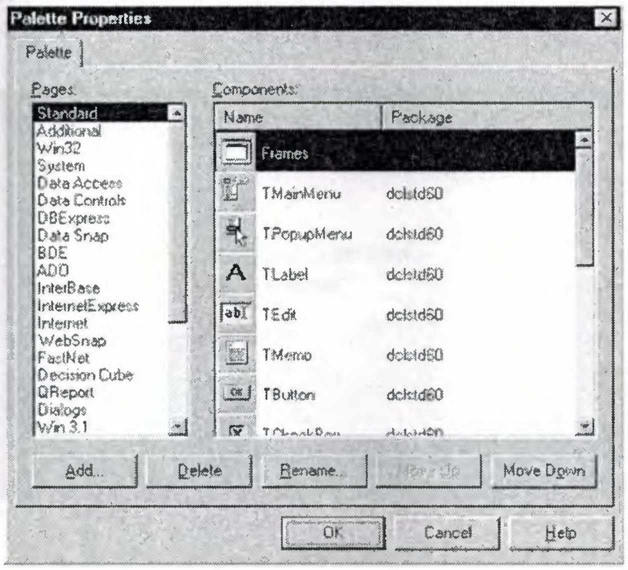
1.5.6. Arranging The Component Palette

To add, delete, rearrange, or rename pages, or to hide or rearrange components, use the Palette Properties dialog box. You can open this dialog box in several ways:

• Choose Component|Configure Palette.

• Choose Tools|Environment Options and click the Palette tab.

• Right-click the Component palette and choose Properties.



You can rearrange the palette and add new pages.

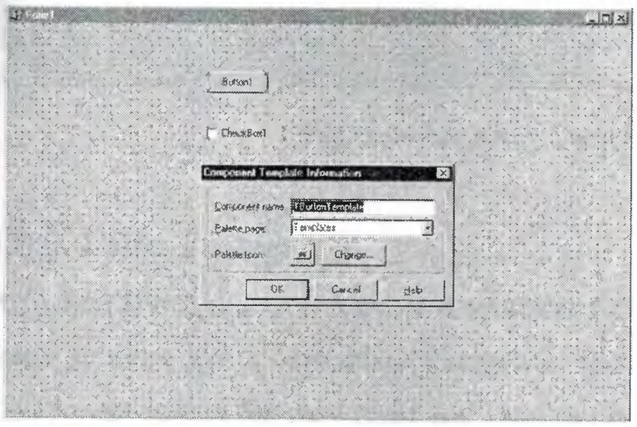
Fig. 1.42 Palette Properties Dialog Box

1.5.7. Creating Component Templates

Component templates are groups of components that you add to a form in a single operation. Templates allow you to configure components on one form, then save their arrangement, default properties, and event handlers on the Component palette to reuse on other forms.

To create a component template, simply arrange one or more components on a form and set their properties in the Object Inspector, and select all of the components by dragging the mouse over them. Then choose Component|Create Component Template. When the Component Template Information dialog box opens, select a name for the template, the palette page on which you want it to appear, and an icon to represent the template on the palette.

After placing a template on a form, you can reposition the components independently, reset their properties, and create or modify event handlers for them just as if you had placed each component in a separate operation.


Component Template Information

Component name: TButtonTemplate

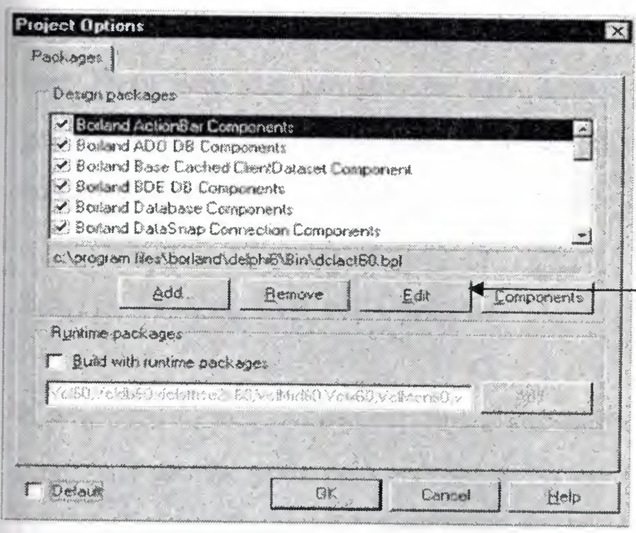
Palette page: Templates

Fig. 1.43 Creating Component Templates

1.5.8. Installing Component Packages

Whether you write custom components or obtain them from a vendor, the components must be compiled into a package before you can install them on the Component palette.A package is a special DLL containing code that can be shared among Delphi applications, the IDE, or both. Runtime packages provide functionality when a user runs an application. Design-time packages are used to install components in the IDE.

Delphi packages have a .bpl extension. If a third-party vendor's components are already compiled into a package, either follow the vendor's instructions or choose Component|Install Packages.


Project Options
Packages
Design packages
Borland ActionBar Components
Borland ADO DB Components
Borland Base Cached ClientDataset Component
Borland BDE DB Components
Borland Database Components
Borland DataSnap Connection Components
c:\program files\borland\delphi6\Bin\dclact60.bpl
Add...
Remove
Edit
Components
Runtime packages
Build with runtime packages
Default
OK
Cancel
Help

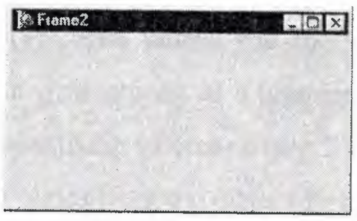
These components come preinstalled in Delphi. When you install new components from third-party vendors, their package appears in this list. Click Components to see what

Fig. 1.44 Installing Component Packages

1.5.9. Using Frames

A frame (TFrame), like a form, is a container for components that you want to reuse. A frame is more like a customized component than a form. Frames can be saved on the Component palette for easy reuse and they can be nested within forms, other frames, or other container objects. After a frame is created and saved, it continues to function as a unit and to inherit changes from the components (including other frames) it contains. When a frame is embedded in another frame or form, it continues to inherit changes made to the frame from which it derives.

To open a new frame, choose File | New | Frame.



You can add whatever visual or nonvisual components you need to the frame. A new unit is automatically added to the Code editor.

Fig. 1.45 Opening New Frame

1.5.10. Adding ActiveX Controls

You can add ActiveX controls to the Component palette and use them in your Delphi projects. Choose Component|Import ActiveX Control to open the Import ActiveX dialog box. From here you can register new ActiveX controls or select an already registered control for installation in the IDE. When you install an ActiveX control, Delphi creates and compiles a "wrapper" unit file for it.

1.5.11. Setting Project Options

If you need to manage project directories and to specify form, application, compiler, and linker options for your project, choose Project |Options. When you make changes in the Project Options dialog box, your changes affect only the current project; but you can also save your selections as the default settings for new projects.

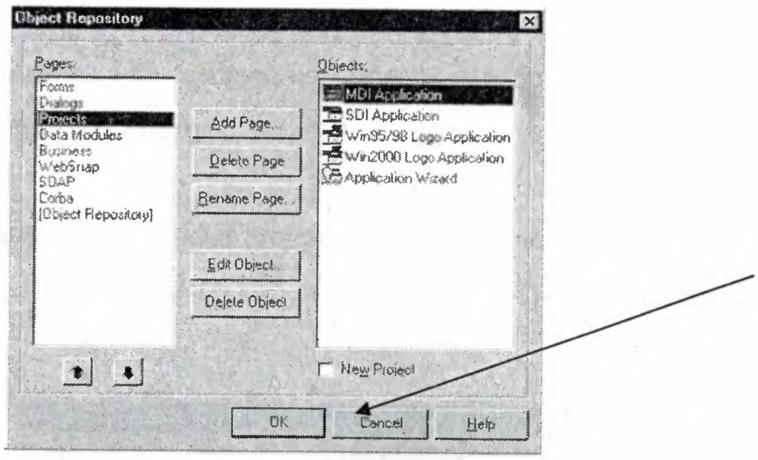
1.5.12. Setting Default Project Options

To save your selections as the default settings for all new projects, in the lower left corner of the Project Options dialog box, check Default. Checking Default writes the current settings from the dialog box to the options file defproj.dof, located in the Delphi7\Bin directory. To restore Delphi's original default settings, delete or rename the defproj.dof file.

1.5.13. Specifying Project and Form Templates As The Default

When you choose File|New|Application, Delphi creates a standard new application with an empty form, unless you specify a project template as your default project. You can save your own project as a template in the Object Repository on the Projects page by choosing Project|Add to Repository . Or you can choose from one of Delphi's existing project templates from the Object Repository.

To specify a project template as the default, choose Tools or Repository. In the Object Repository dialog box, under Pages, select Projects. If you've saved a project as a template on the Projects page, it appears in the Objects list. Select the template name, check New Project, and click OK.



The Object Repository's pages contain project templates only, form templates only, or a combination of both. To set a project template as the default, select an item in the Objects list and check New Project .To set a form template as the default, select an item in the Objects list and check New Form or Main Form.

Fig. 1.46 Specifying Project and Form Templates As The Default

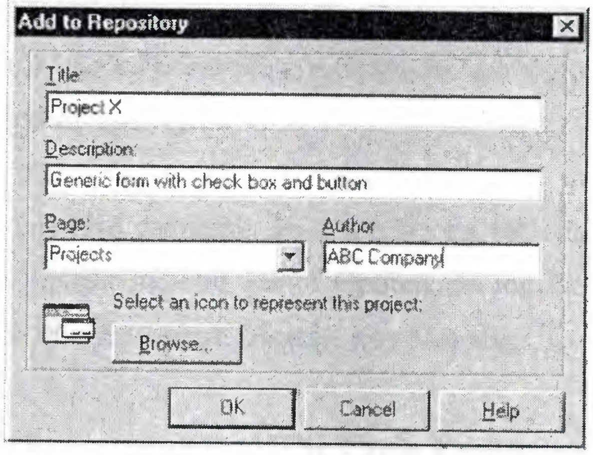
Once you've specified a project template as the default, Delphi opens it automatically whenever you choose File |New |Application.

In the same way that you specify a default project, you can specify a default new form and a default main form from a list of existing form templates in the Object Repository. The default new form is the form created when you choose File |New |Form to add an additional form to an open project. The default main form is the form created when you open a new application. If you haven't specified a default form, Delphi uses a blank form. You can override your default project or form temporarily by choosing File |New| Other and selecting a different template from the New Items dialog box.

1.5.14. Adding Templates To The Object Repository

You can add your own objects to the Object Repository as templates to reuse and share with other developers over a network. Reusing objects lets you build families of applications with common user interfaces and functionality that reduces development time and improves quality.

For example, to add a project to the Repository as a template, first save the project and choose Project|Add To Repository. Complete the Add to Repository dialog box.


Add to Repository

Title: Project X

Description: Generic form with check box and button

Page: Projects

Author: ABC Company

Enter a title, description, and author. In the Page list box, choose Projects so that your project will appear on the Repository's Projects tabbed page.

Fig. 1.47 Adding Templates To The Object Repository

The next time you open the New Items dialog box, your project template will appear on the Projects page (or the page to which you had saved it).

1.5.15. Setting Tool Preferences

You can control many aspects of the appearance and behavior of the IDE, such as the Form Designer, Object Inspector, and Code Explorer. These settings affect not just the current project, but projects that you open and compile later. To change global IDE settings for all projects, choose Tools |Environment Options.

1.5.16. Customizing The Form Designer

The Designer page of the Tools|Environment Options dialog box has settings that affect the Form Designer. For example, you can enable or disable the "snap to grid" feature, which aligns components with the nearest grid line; you can also display or hide the names, or captions, of nonvisual components you place on your form.

1.5.17. Customizing The Code Editor

One tool you may want to customize right away is the Code editor. Several pages in the Tools|Editor Options dialog box have settings for how you edit your code. For example, you can choose keystroke mappings, fonts, margin widths, colors, syntax highlighting, tabs, and indentation styles.

You can also configure the Code Insight tools that you can use within the editor on the Code Insight page of Editor Options. In the Editor Options dialog box, click the Help button on the General, Display, Key Mappings, Color, and Code Insight pages.

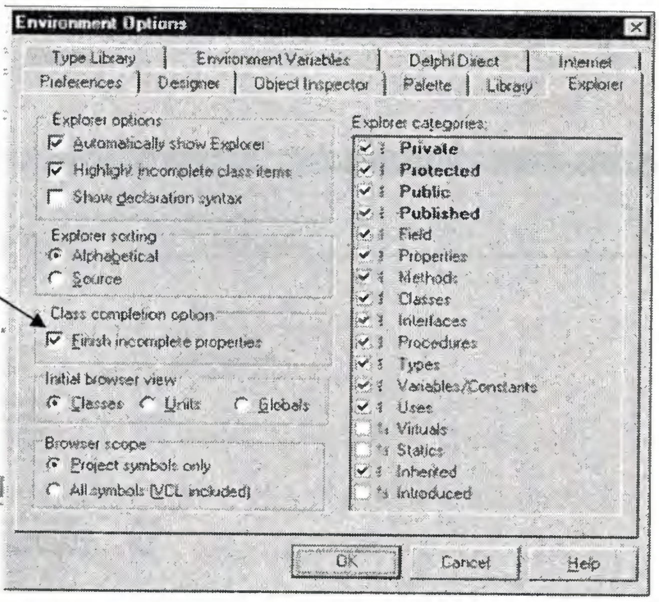
1.5.18. Customizing The Code Explorer

When you start Delphi, the Code Explorer opens automatically. If you don't want Code Explorer to open automatically, choose Tools|Environment Options, click the Explorer tab, and uncheck Automatically show Explorer.

You can change the way the Code Explorer's contents are grouped within the Code Explorer by right-clicking in the Code Explorer, choosing Properties, and, under Explorer categories, checking and unchecking the check boxes. If a category is checked, elements in that category are grouped under a single node. If a category is unchecked, each element in that category is displayed independently on the diagram's trunk. For example, if you uncheck the Published category, the Published folder disappears but not the items in it.

In the Code Explorer, you can sort all source elements alphabetically or in the order in which they are declared in the source file

To display the folder for each type of source element in the Code Explorer, check an Explorer category.



Explorer options
[✓] Automatically show Explorer
[✓] Highlight incomplete class items
[ ] Show declaration syntax

Explorer sorting
(●) Alphabetical
( ) Source

Class completion option
[✓] Finish incomplete properties

Initial browser view
(●) Classes ( ) Units ( ) Globals

Browser scope
(●) Project symbols only
( ) All symbols (VCL included)

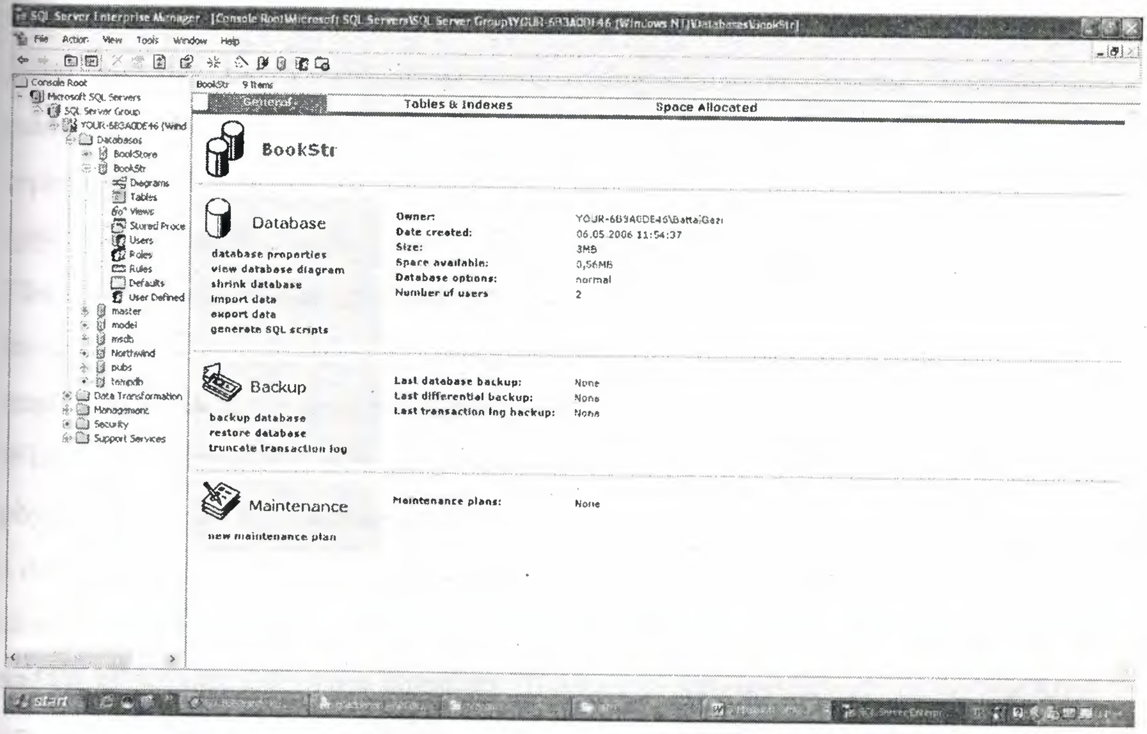
Explorer categories:
[✓] **Private**
[✓] **Protected**
[✓] **Public**
[✓] **Published**
[✓] Field
[✓] Properties
[✓] Methods
[✓] Classes
[✓] Interfaces
[✓] Procedures
[✓] Types
[✓] Variables/Constants
[✓] Uses
[ ] Virtuals
[ ] Statics
[✓] Inherited
[ ] Introduced

Fig. 1.48 Customizing The Code Explorer

CHAPTER 2: MICROSOFT SQL SERVER

2. WHAT IS SQL SERVER

SQL Server is a client/server relational database management system (RDBMS). In this section, client/server architecture is explained and you will be introduced to some of the features and characteristics of SQL Server.



Owner:	YOUR-6B3ACDE46\BattalGazi
Date created:	06.05.2006 11:54:37
Size:	3MB
Space available:	0,56MB
Database options:	normal
Number of users	2
Last database backup:	None
Last differential backup:	None
Last transaction log backup:	None
Maintenance plans:	None

Figure 1.1 : A external view

2.1 Client/Server Architecture

The terms client, server, and client/server can be used to refer to very general concepts or specific items of hardware or software. At the most general level, a client is any component of a system that requests services or resources from other system components. A server is any system component that provides services or resources to other system components.

For example, when you print a document from your workstation on a network, the workstation is the client and the print spooling machine is the server.

Any client/server data-based system consists of the following things:

The server. A collection of data items and supporting objects organized and presented to facilitate services, such as searching, sorting, recombining, retrieving, updating, and analyzing data. The database consists of the physical storage of data and the database services. All data access occurs through the server; the physical data is never directly accessed.

The client. A software program that might be used interactively by a person or that could be an automated process. This includes all software that interacts with the server, either requesting data from the database or sending data to the database. Examples are: management utilities—those that are part of the SQL Server product and those bought separately, ad-hoc query and reporting software, custom applications, off-the-shelf applications, and Web server-based applications.

The communication between the client and the server. The communication between the client and the server is largely dependent on how the client and server are implemented. Both physical and logical levels of communication can be identified. When you communicate with someone using the telephone, the telephone system is the physical layer and a spoken natural language is the logical layer of communication. For a data-based system, the physical communication can be a network, if the server and the client are on different computers. It can be via inter-process communication if the server and the client are on the same computer. The logical communication may be low-level operating system calls, a proprietary data access language, or the open Structured Query Language (SQL).

All implementations of data-based systems fall into one of three categories:

File-based systems. Commonly found on personal computers, these systems use an application that directly accesses data files on a local hard drive or on a network file server. These systems implement the database services and the logical communication as part of the client application, only the physical communication and the physical storage of data are external to the client application. In this implementation, the client application fulfills the role of client and the role of server, as shown in Figure 1.1.

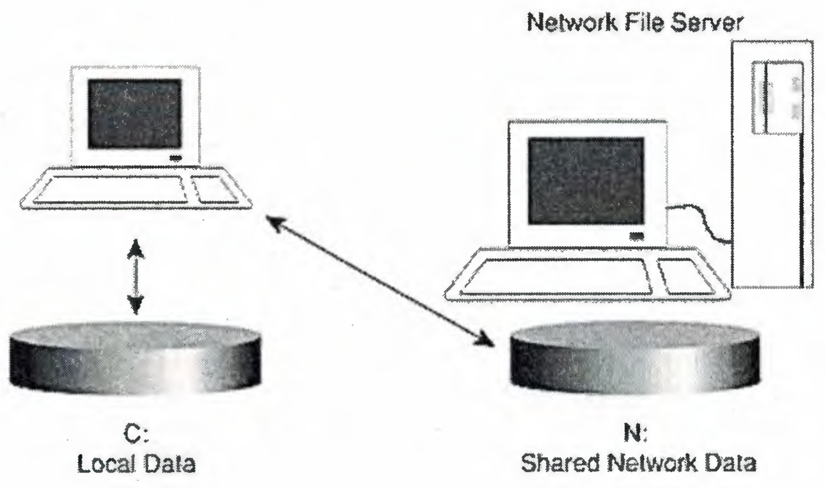


Figure 1.2 : File-based system.

Host-based systems. Typically used in legacy mainframe and mini-computer environments, these systems implement all or most of the database services and client functionality on a large central computer. The user views and interacts with the client application remotely using a terminal. The communication between the client and the database is done on the host computer. In this implementation, the host computer fulfills the role of client and the role of server, as shown in Figure 1.2.

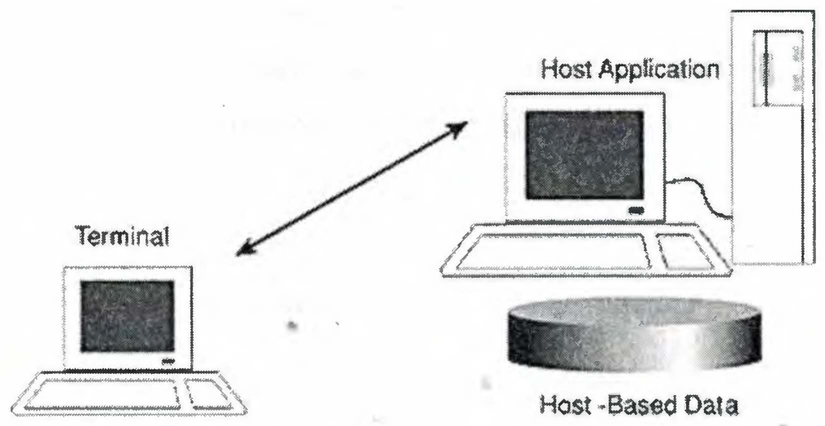


Figure 1.3 : Host-based system.

Client/server systems. These systems are designed to separate database services from the client, allowing the communication between them to be more flexible and open. Database services are implemented on a powerful computer, allowing centralized management, security, and shared resources—therefore, the server in client/server is the database and its services. Client applications are implemented on a variety of platforms, using a variety of tools. This process allows greater flexibility and high quality user applications—this is the client in client/server. Figure 1.3 shows the client application and the database server in the client/server implementation.

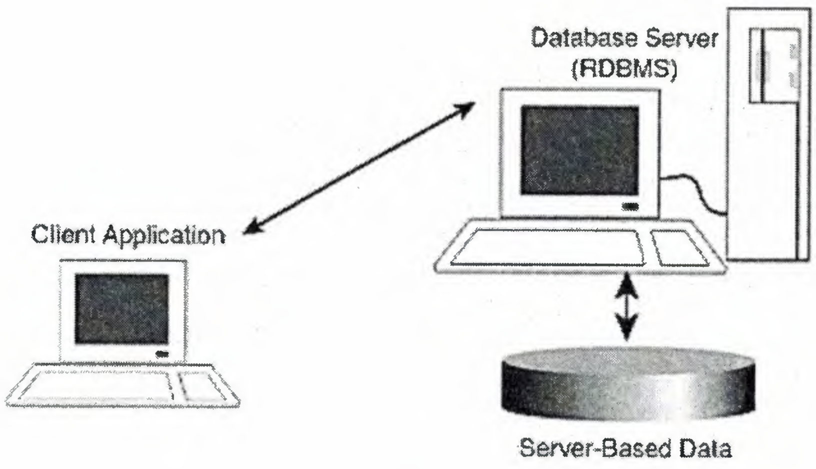


Figure 1.4 : Client/Server system.

The following table compares some of the advantages and disadvantages of file-based, host-based, and client/server systems. Many organizations now use a mix of these systems. For example, data capture may be performed on a host-based system with thousands of terminals. Data is then queried, manipulated, and analyzed by a client/server system, either directly on the host, or after transferring the data to another database.

File-based	Host-based	Client/Server
Low cost	High initial cost	Variable cost
Low security	High security	Medium to high security
Low reliability	High reliability	Medium to high reliability
Application development possible with few skills	Application development requires skilled staff	Application development requires skilled staff
Well-suited to small	Not appropriate for	Can be used for small

databases and end-user databases	small or end-user databases	databases. Not appropriate for end-userdatabases
Scalable to medium databases (~ 50 MB)	Scalable to very large databases (1000s of GB)	Scalable to very large databases (1000s of GB)
Minimal centralized management	Excellent centralized management	Excellent centralized management
Highly flexible end-user interface	Inflexible end-user interface	Flexible end-user interface
Low to medium vendor lock-in	High vendor lock-in	Medium vendor lock-in
Uses network inefficiently	Uses network efficiently	Can use network efficiently

Hundreds of commercial data-based systems are available. These systems range from those comprised of a single application running on a single personal computer to those comprised of hundreds of applications running on complex networks of mainframe, mini, and personal computers. All commercial data-based systems have these three basic components. Clearly, therefore, these basic components could all occur on a single computer or may be distributed across many computers. Try to identify these components whenever you encounter a data-based system. In a large system, each component may be further layered into many parts, but you should always be able to distinguish the database, the client, and the communication between the two.

NOTE

The key to understanding client/server (and specifically SQL Server) is that the database server (SQL Server) is a fully functional process or application that provides database services—compare this to a file on a network file server, which is a static storage structure only. Clients interact with these database services through a clearly defined communication interface, allowing for tight control and security. Clients do not have direct access to data; they always communicate with the database server, which in turn interacts with the physical data. SQL Server's own management utilities are clients that can run on the same computer or on another computer and have no more direct access to data than other clients do.

2.2 Transact-SQL

SQL Server uses Transact-SQL, a version of the Structured Query Language (SQL), as its database query and programming language. *SQL* is a set of commands that allows you to specify the information that you want to retrieve or modify. With Transact-SQL, you can access data and query, update, and manage relational database systems.
The American National Standards Institute (ANSI) and the International Standards Organization (ISO) have defined standards for SQL. Transact-SQL supports the latest ANSI SQL standard published in 1992, called ANSI SQL-92, plus many extensions to provide increased functionality.

2.3. SQL Server Platforms

Figure 1.4 summarizes the different platforms supported by SQL Server and SQL Server clients.

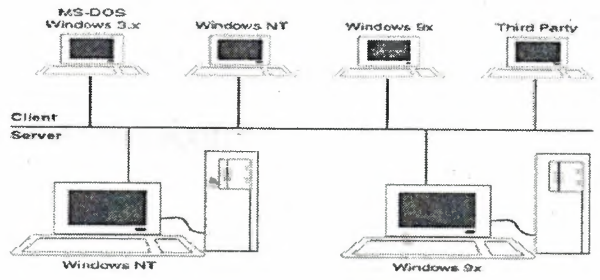


Figure 1.5 : *SQL Server platforms.*

SQL Server runs on the operating systems shown in the following table. You can use some or all of the operating system platforms to create and execute applications.

Platform	Server software	Client operating system
Microsoft Windows 95 or later	Yes—runs as an application	Yes
Microsoft Windows NT Workstation 4.0 or later	Yes—runs as a service	Yes
Microsoft Windows NT Server 4.0	Yes—runs as a service	Yes
Microsoft Windows NT Server Enterprise Edition 4.0	Yes—runs as a service	Yes
Windows 3.x	No	Yes (with some limitations)
MS-DOS	No	Yes (with some limitations)
Third party	No	Yes—such as UNIX and Apple Macintosh

2.4. SQL Server Architecture

2.4.1. Overview

SQL Server provides many structured architectures that hide underlying technical details, simplifying the development, maintenance, and management of your database applications.

2.4.2. Data Access Architecture

Users use SQL Server databases through an application that uses a data object interface or an API (Figure 1.8) to gain access to SQL Server.

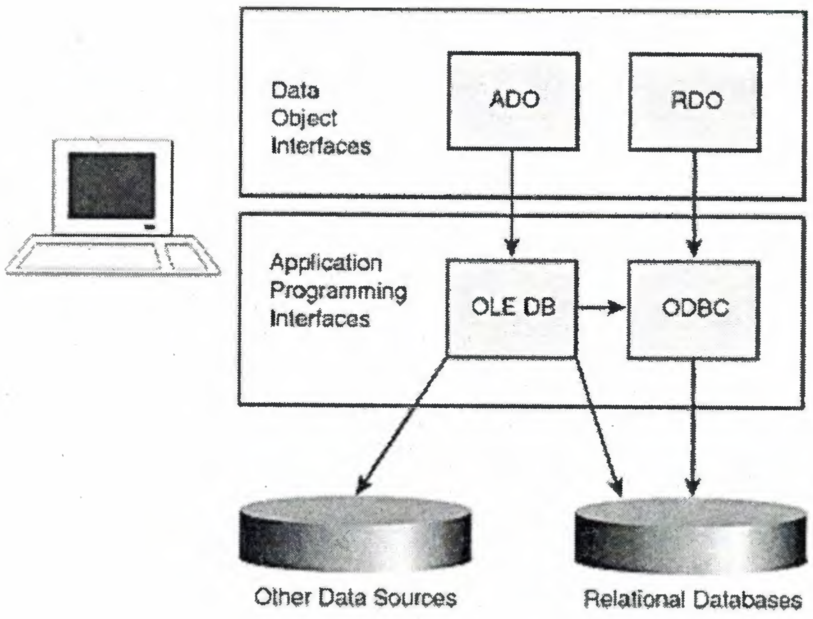


Figure 1.6 : *Application interfaces.*

SQL Server supports commonly used and emerging database interfaces. It supports low-level native APIs, as well as easy-to-use data object interfaces.

2.4.3. Application Programming Interfaces (APIs)

A database *application programming interface (API)* defines how to write an application to connect to a database and pass commands to the database. SQL Server provides native support for two main classes of database APIs, which in turn determine the data object interface that you can use. Use the database APIs to have more control over application behavior and to gain better performance.

OLE DB

OLE DB is a Component Object Model (COM)-based data access interface. It supports applications written to use OLE DB or data object interfaces that use OLE DB. OLE DB is designed to work with relational databases (such as those in SQL Server) as well as with non-relational data sources (such as a full-text index or an e-mail message store).

OLE DB uses a provider to gain access to a particular data source. Providers for SQL Server, Oracle, Jet (Microsoft Access databases), and ODBC are supplied with SQL Server. Using the OLE DB provider for ODBC, OLE DB can be used to gain access to any ODBC data source.

ODBC

ODBC is a call-level interface. It communicates directly with SQL Server and supports applications or components that are written to use ODBC or data object interfaces that use ODBC. ODBC is designed to work with relational databases (such as those in SQL Server) only, although there are limited ODBC drivers available for some nonrelational data sources.
ODBC uses a driver to gain access to a particular data source.

Data Object Interfaces

In general, *data object interfaces* are easier to use than database APIs but may not expose as much functionality as an API.

ActiveX Data Objects (ADO)

ActiveX Data Objects (ADO) encapsulates the OLE DB API in a simplified object model that reduces application development and maintenance costs. ADO can be used from many development environments, such as Microsoft Visual Basic, Microsoft Visual C++, Visual Basic for Applications, Active Server Pages (ASP), and the Microsoft Internet Explorer scripting object model. These characteristics make ADO the primary database interface for developing client/server business applications.

Remote Data Objects (RDO)

Remote Data Objects (RDO) maps over and encapsulates the ODBC API. RDO can be used from Microsoft Visual Basic and Visual Basic for Applications.

Administration Architecture

SQL Server provides a variety of management tools that minimize and automate routine administrative tasks. Figure 1.9 shows how administrative tools use different interfaces to communicate with SQL Server.

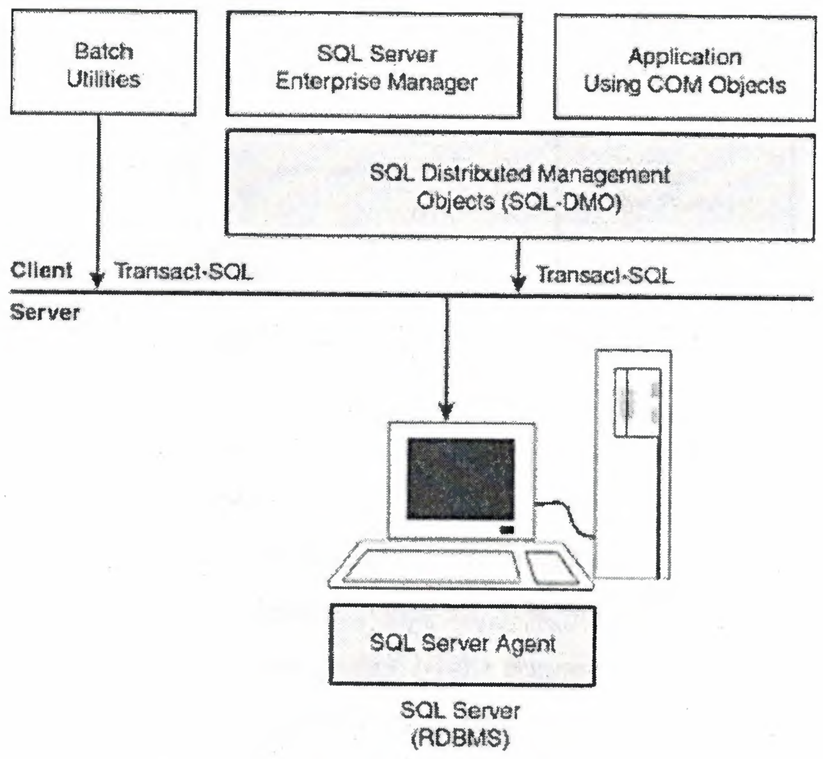

Batch Utilities
SQL Server Enterprise Manager
Application Using COM Objects
SQL Distributed Management Objects (SQL-DMO)
Client
Transact-SQL
Transact-SQL
Server
SQL Server Agent
SQL Server (RDBMS)

Figure 1.7 : *Administration architecture.*

2.5. Login Authentication

A user must have a login account to connect to SQL Server. SQL Server recognizes two login authentication mechanisms—SQL Server authentication and Windows NT authentication (Figure 1.11)—each of which has a different type of login account.

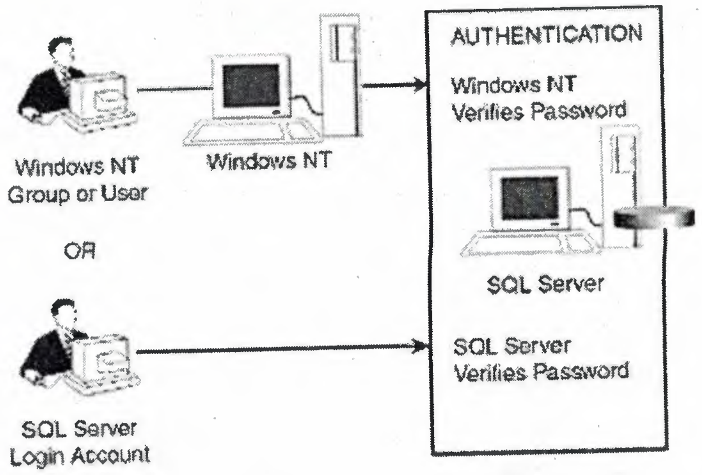


Figure 1.8 : *Login authentication.*

2.5.1.Permission Validation

SQL Server accepts Transact-SQL statements after a user's login has been successfully authenticated. Each time a user sends a statement, SQL Server checks that the user has permission to carry out the action requested by the statement. If the user has permission, the action is carried out; if not, an error is returned to the user (Figure 1.12)

.

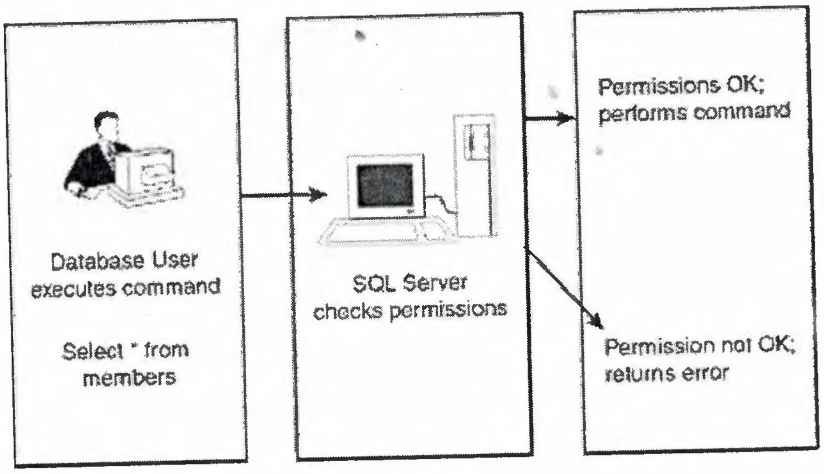

Database User executes command
Select * from members
SQL Server checks permissions
Permissions OK; performs command
Permission not OK; returns error

Figure 1.9 : *Permission validation.*

NOTE

In most cases, the user will be interacting with an application user interface, unaware of the Transact-SQL statements that their actions are generating.

Database User Accounts and Roles

User accounts and roles, which identify a successfully logged-in user within a database, are used to control ownership of objects. Permissions to execute statements and use objects in a database are granted to users and roles. Each user account is mapped to a SQL Server login, as shown in Figure 1.13.

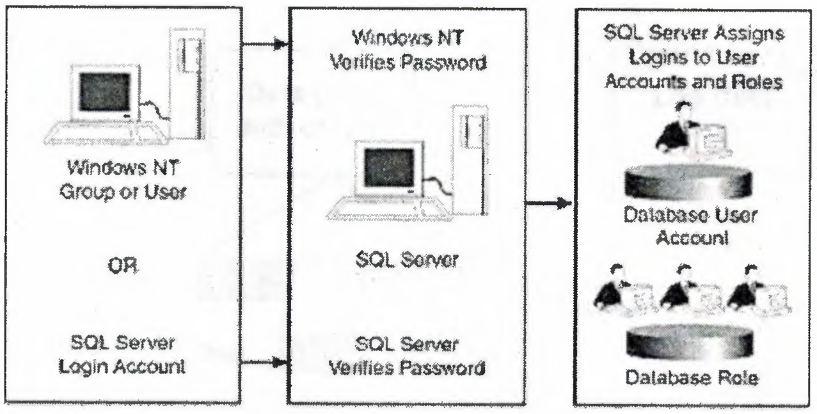


Figure 1.10 : *Users and roles.*

2.6. Introduction to SQL Server Databases

When you create a database, you set up the data storage structure. This structure includes at least one data file and one transaction log file. Before you create a database, you should understand how Microsoft SQL Server version 7.0 stores data, as well as the function of the transaction log.

2.6.1. How Data Is Stored

When you are creating a database, you will have more insight into capacity planning, data integrity, and performance if you understand how Microsoft SQL Server stores data. Figure 3.1 illustrates how data is allocated for storage.

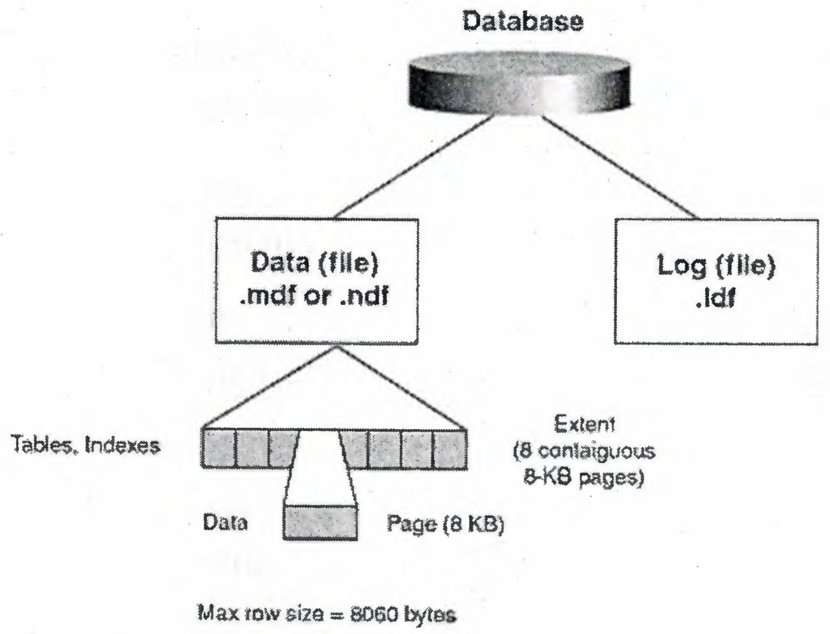


Figure 1.11 : *Database storage allocation.*

2.7. Creating Data Types

Before you can create a table, you must define the data types for the table. Data types specify the type of information (characters, numbers, or dates) that a column can hold, as well as how the data is stored. Microsoft SQL Server supplies various system data types. SQL Server also allows user-defined data types that are based on system data types.

System-Supplied Data Types

SQL Server provides several different data types. Certain types of data have several associated SQL Server system supplied data types. For example, you could use the int, decimal, or float data type to store numeric data. However, you should choose appropriate data types in order to optimize performance and conserve disk space.

Categories of System-Supplied Data Types

The following table maps common types of data to SQL Server system-supplied data types. The table includes data type synonyms for ANSI compatibility.

Type of data	System-supplied data types	ANSI synonym	Number of bytes
Binary	binary[(n)] varbinary[(n)]	- binary varying[(n)]	1-8000
Character	char[(n)] varchar[(n)]	character[(n)] char[acter] varying[(n)]	1-8000 (8000 characters)
Unicode character	nchar[(n)] nvarchar[(n)]	national char[acter][(n)] national char[acter] varying[(n)]	2-8000 (1 - 4000 characters)
Date and time	Datetime, smalldatetime	-	8 (2 4-byte integers) 4 (2 2-byte integers)
Exact numeric	decimal[(p[, s])] numeric[(p[, s])]	dec	5-17
Approximate numeric	float[(n)] real	Double precision or float[(n)] float[(n)]	4-8 4
Global identifier	uniqueidentifier	-	16

Integer	int	integer	4
	smallint, tinyint	-	2, 1
Monetary	money, smallmoney	-	8, 4
Special	bit, cursor, sysname, timestamp,	-	1, 0-8
Text and image	text, image	-	0-2 GB
Unicode text	Ntext	national text	0-2 GB

NOTE

SQL Server supports multiple languages with the nchar, nvarchar, and ntext Unicode string data types. Unicode strings use two bytes per character.

Creating Tables

After you define all the data types for your table, you can create tables, add and drop columns, and generate column values.

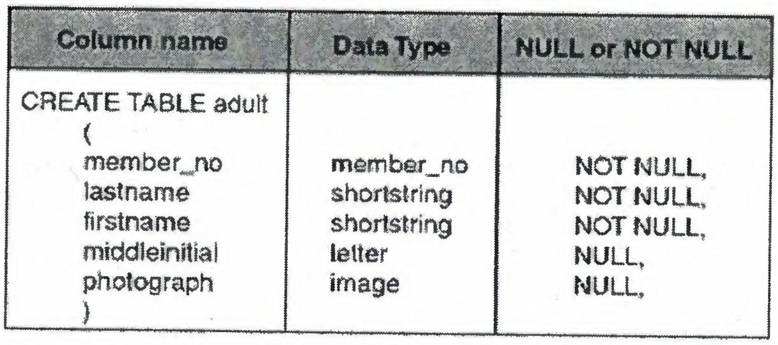
Creating a Table

Consider the following facts when you create tables in SQL Server. You can have up to:

- Two billion tables per database
- 1024 columns per table
- 8060 bytes per row (image and text data types each use 16 bytes per row)

Specifying NULL or NOT NULL

You can specify in the table definition whether to allow null values in each column, as shown in Figure 4.1. If you do not specify NULL or NOT NULL, SQL Server determines whether the column may or may not accept null values based on the session- or database-level default. However, these defaults can change, so do not rely on them. NOT NULL is the SQL Server default; NULL is the ANSI default.



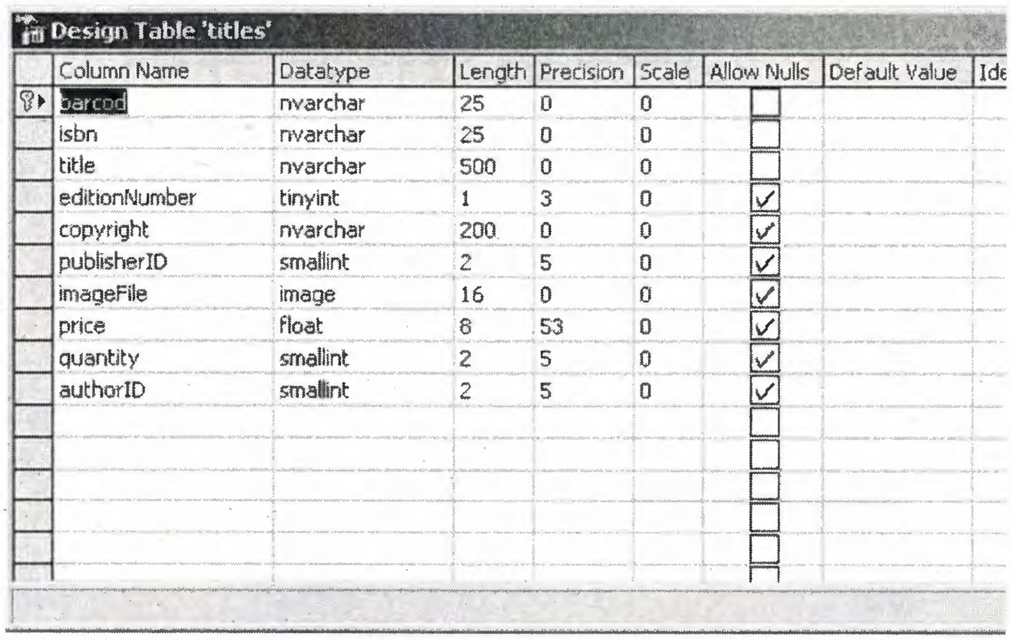
Column name	Data Type	NULL or NOT NULL
CREATE TABLE adult		
(		
member_no	member_no	NOT NULL,
lastname	shortstring	NOT NULL,
firstname	shortstring	NOT NULL,
middleinitial	letter	NULL,
photograph	image	NULL,
)		

Figure 1.12 : *The CREATE TABLE statement.*

To Create a Table Using SQL Server Enterprise Manager

In this exercise, you will use SQL Server Enterprise Manager to create a table in the library database. At this time, do not create the primary and foreign keys, indexes, or other items listed in the library schema. Open SQL Server Enterprise Manager.

1. Expand your server group; then expand your server.
2. Expand Databases; then expand the library database.
3. Right-click Tables; then click New Table.
4. In the Choose Name dialog box, type the name **titles** for the table. Click OK.
5. Fill in the columns as specified in the following table. Each row represents one column in the table.
6. Close the New Table window and save changes to the titles table.



Design Table 'titles'

Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Ide
barcod	nvarchar	25	0	0	☐		
isbn	nvarchar	25	0	0	☐		
title	nvarchar	500	0	0	☐		
editionNumber	tinyint	1	3	0	✓		
copyright	nvarchar	200	0	0	✓		
publisherID	smallint	2	5	0	✓		
imageFile	image	16	0	0	✓		
price	float	8	53	0	✓		
quantity	smallint	2	5	0	✓		
authorID	smallint	2	5	0	✓		

Figure 1.13 : titles table in BookStr Database

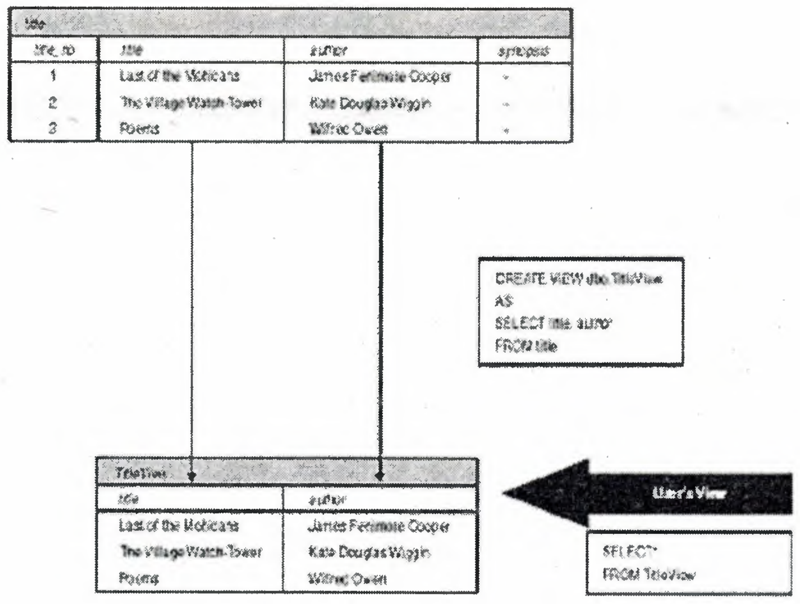
7. Close the New Table window and save changes to the adult table.

2.8. Introduction to Views

Views are a powerful way to create permanent query definition. This lesson defines views and discusses the advantages of views to administrators, programmers, and users.

2.8.1. What Is a View?

When you use a *view*, you can store a predefined query as an object in the database for later use, as shown in Figure 11.1. The tables queried in a view are called *base tables*. With a few exceptions, any SELECT statement can be named and stored as a view.



title

title_no	title	author	synopsis
1	Last of the Mohicans	James Fenimore Cooper	-
2	The Village Watch-Tower	Kate Douglas Wiggin	-
3	Poems	Wilfred Owen	-

```
CREATE VIEW dbo.TitleView
AS
SELECT title, author
FROM title
```

TitleView

title	author
Last of the Mohicans	James Fenimore Cooper
The Village Watch-Tower	Kate Douglas Wiggin
Poems	Wilfred Owen

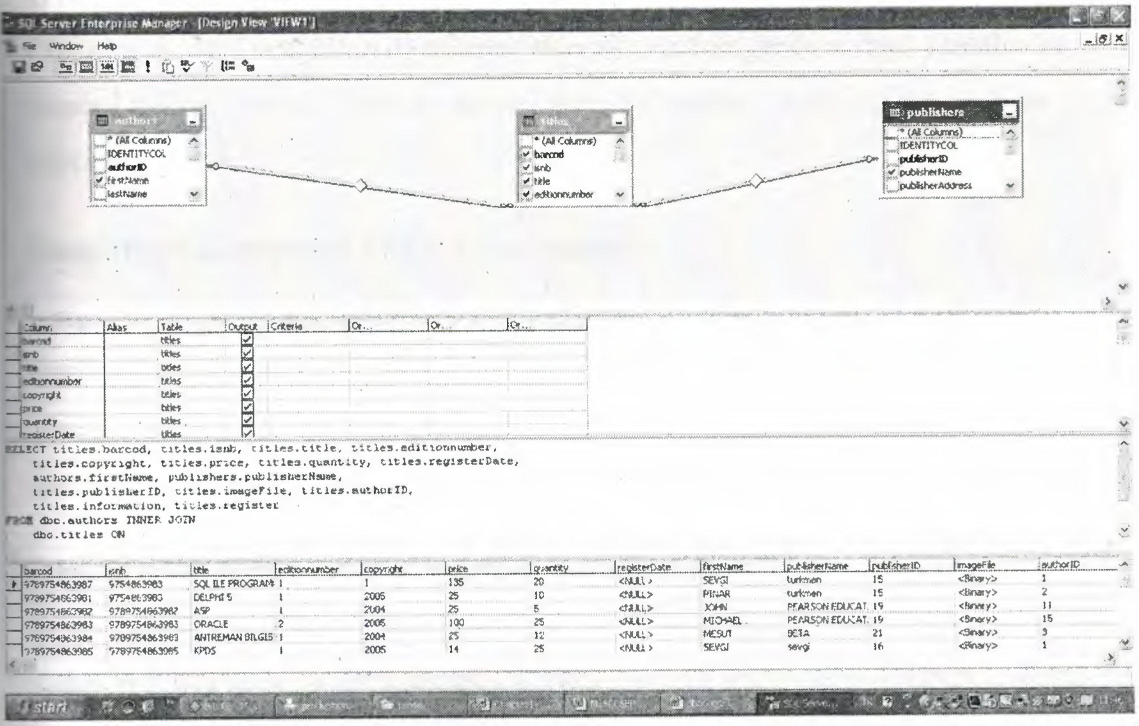
```
SELECT *
FROM TitleView
```

Figure 1.14 : *How a view works.*

Common examples of views include the following:

- A subset of rows or columns of a base table
- A union of two or more base tables
- A join of two or more base tables
- A statistical summary of a base table
- A subset of another view, or some combination of views and base tables

Example



Column	Alias	Table	Output	Criteria	Or...	Or...	Or...
barcod		titles	✓				
isnb		titles	✓				
title		titles	✓				
editionnumber		titles	✓				
copyright		titles	✓				
price		titles	✓				
quantity		titles	✓				
registerDate		titles	✓				

```
SELECT titles.barcod, titles.isnb, titles.title, titles.editionnumber,
    titles.copyright, titles.price, titles.quantity, titles.registerDate,
    authors.firstName, publishers.publisherName,
    titles.publisherID, titles.imageFile, titles.authorID,
    titles.information, titles.register
FROM dbo.authors INNER JOIN
    dbo.titles ON
```

barcod	isnb	title	editionnumber	copyright	price	quantity	registerDate	firstName	publisherName	publisherID	imageFile	authorID
9789754863987	9754863983	SQL ILE PROGRAM	1	1	135	20	<NULL>	SEVGI	turkmen	15	<Binary>	1
9789754863981	9754863983	DELPHI 5	1	2005	25	10	<NULL>	PINAR	turkmen	15	<Binary>	2
9789754863982	9789754863982	ASP	1	2004	25	5	<NULL>	JOHN	PEARSON EDUCAT	19	<Binary>	11
9789754863983	9789754863983	ORACLE	2	2005	100	25	<NULL>	MICHAEL	PEARSON EDUCAT	19	<Binary>	15
9789754863984	9789754863983	ANTREMAN BILGIS	1	2004	25	12	<NULL>	MESUT	BETA	21	<Binary>	3
9789754863985	9789754863985	KPDS	1	2005	14	25	<NULL>	SEVGI	sevgi	16	<Binary>	1

Figure 1.15 : How a view works in my project

2.8.2. Advantages of Views

Views offer several advantages, including focusing data for users, masking data complexity, simplifying permission management, and organizing data for export to other applications.

Focus the Data for Users

Views create a controlled environment that allows access to specific data and conceals other data. Data that is unnecessary, sensitive, or inappropriate can be left out of a view. Users can manipulate the display of data in a view, similar to a table. In addition, with the proper permissions and a few restrictions, users can modify the data that a view produces.

Mask Database Complexity

Views shield the complexity of the database design from the user. This means that developers can change the design without affecting user interaction with the database.

In addition, users can see a friendlier version of the data by using names that are easier to understand than the cryptic names that are often used in databases.
Complex queries, including distributed queries to heterogeneous data, can also be masked through views. The user queries the view instead of writing the query or executing a script.

Simplify Management of User Permissions

Instead of granting permission for users to query specific columns in base tables, database owners can grant permission for users to query data through views only. This type of querying also protects changes in the design of the underlying base tables. Users can continue to query the view without interruption.

Organize Data for Export to Other Applications

You can create a view based on a complex query that joins two or more tables and then export the data to another application for further analysis.

CHAPTER 3: ADO COMPONENT

3.1 Working with ADO components:

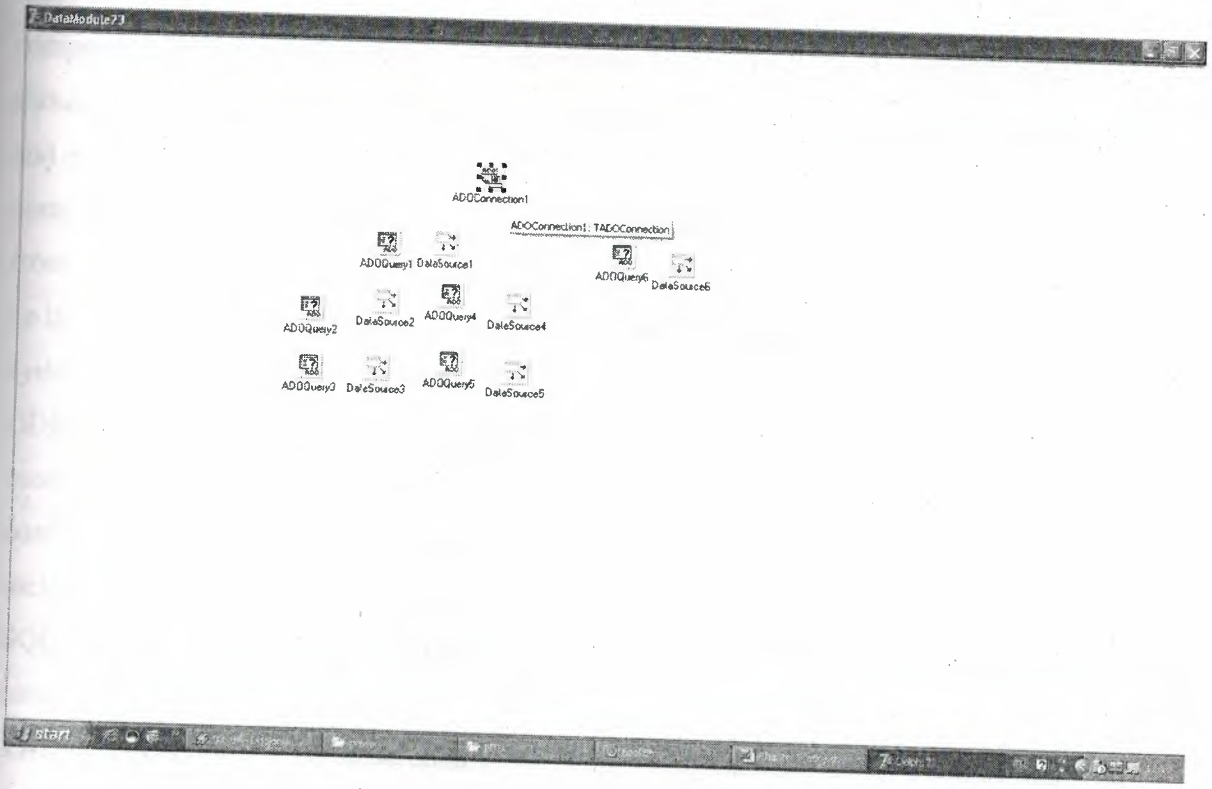


Figure 3.1 : ADO Connection,Ado Table and ADO Query components

The dbGo components provide data access through the ADO framework. ADO, (Microsoft ActiveX Data Objects) is a set of COM objects that access data through an OLE DB provider. The dbGo components encapsulate these ADO objects in the Delphi database architecture.

The ADO layer of an ADO-based application consists of Microsoft ADO 2.1, an OLE DB provider or ODBC driver for the data store access, client software for the specific database system used (in the case of SQL databases), a database back-end system accessible to the application (for SQL database systems), and a database. All of these must be accessible to the ADO-based application for it to be fully functional.The ADO objects that figure most prominently are the Connection, Command, and Recordset objects. These ADO objects are wrapped by the TADOConnection, TADOCommand, and ADO dataset components. The ADO framework includes other "helper" objects, like the Field and Properties objects, but these are typically not used directly in dbGo applications and are not wrapped by dedicated components.Before reading about the features peculiar to the dbGo components, you should familiarize yourself with the common features of database connection components and datasets .The ADO page of the Component palette hosts the dbGo components. These components let you connect to an ADO data store, execute commands, and retrieve data from tables in databases using the ADO framework. They require ADO 2.1 (or higher) to be installed on the host computer. Additionally, client software for the target database system (such as Microsoft SQL Server) must be installed, as well as an OLE DB driver or ODBC driver specific to the particular database system.Most dbGo components have direct counterparts in the components available for other data access mechanisms: a database connection component (TADOConnection) and various types of datasets. In addition, dbGo includes TADOCommand, a simple component that is not a dataset but which represents an SQL command to be executed on the ADO data store.The following table lists the ADO components.ADO components,

TADOConnection : A database connection component that establishes a connection with an ADO data store; multiple ADO dataset and command components can share this connection to execute commands, retrieve data, and operate on metadata.

TADODataSet : The primary dataset for retrieving and operating on data; TADODataSet can retrieve data from a single or multiple tables; can connect directly to a data store or use a TADOConnection component.

TADOTable : A table-type dataset for retrieving and operating on a recordset produced by a single database table; TADOTable can connect directly to a data store or use a TADOConnection component.

TADOQuery : A query-type dataset for retrieving and operating on a recordset produced by a valid SQL statement; TADOQuery can also execute data definition language (DDL) SQL statements. It can connect directly to a data store or use a TADOConnection component

TADOStoredProc: A stored procedure-type dataset for executing stored procedures;
TADOStoredProc: Executes stored procedures that may or may not retrieve data. It can connect directly to a data store or use a TADOConnection component.
TADOCommand: A simple component for executing commands (SQL statements that do not return result sets); TADOCommand can be used with a supporting dataset component, or retrieve a dataset from a table; It can connect directly to a data store or use TADOConnection component.

3.2 Connecting to ADO data stores

DbGo applications use Microsoft ActiveX Data Objects (ADO) 2.1 to interact with an OLE DB provider that connects to a data store and accesses its data. One of the items a data store can represent is a database. An ADO-based application requires that ADO 2.1 be installed on the client computer. ADO and OLE DB is supplied by Microsoft and installed with Windows.

An ADO provider represents one of a number of types of access, from native OLE DB drivers to ODBC drivers. These drivers must be installed on the client computer. OLE DB drivers for various database systems are supplied by the database vendor or by a third-party. If the application uses an SQL database, such as Microsoft SQL Server or Oracle, the client software for that database system must also be installed on the client computer. Client software is supplied by the database vendor and installed from the database systems CD (or disk).
To connect your application with the data store , use an ADO connection component (TADOConnection). Configure the ADO connection component to use one of the available ADO providers. Although TADOConnection is not strictly required, because ADO command and dataset components can establish connections directly using their ConnectionString property, you can use TADOConnection to share a single connection among several ADO components. This can reduce resource consumption, and allows you to create transactions that span multiple datasets.Like other database connection components , TADOConnection provides support for

Controlling connections

Controlling server login

Managing transactions

Working with associated datasets

Sending commands to the server

Obtaining metadata

In addition to these features that are common to all database connection components, TADOConnection provides its own support for

A wide range of options you can use to fine-tune the connection .

The ability to list the command objects that use the connection .

Additional events when performing common tasks.

Connecting to a data store using TADOConnection

One or more ADO dataset and command components can share a single connection to a data store by using TADOConnection . To do so, associated dataset and command components with the connection component through their Connection properties. At design-time, select the desired connection component from the drop-down list for the Connection property in the Object Inspector. At runtime, assign the reference to the Connection property. For example, the following line associates a TADODataSet component with a TADOConnection component.

```
ADODataSet1.Connection := ADOConnection1;
```

The connection component represents an ADO connection object . Before you can use the connection object to establish a connection, you must identify the data store to which you want to connect. Typically, you provide information using the ConnectionString property. ConnectionString is a semicolon delimited string that lists one or more named connection parameters. These parameters identify the data store by specifying either the name of a file that contains the connection information or the name of an ADO provider and a reference identifying the data store. Use the following, predefined parameter names to supply this information:

Connection parameters

Parameter	**Description**
Provider:	The name of a local ADO provider to use for the connection.
Data Source:	The name of the data store.
File name:	The name of a file containing connection information.
Remote Provider:	The name of an ADO provider that resides on a remote machine.
Remote Server:	The name of the remote server when using a remote provider.

Thus, a typical value of ConnectionString has the form

Provider=MSDASQL.1;Persist Security Info=False;User ID=sevgi;Extended Properties="DRIVER=SQL Server;SERVER=YOUR-6B3A0DE46;UID=sevgi;APP=Enterprise;WSID=YOUR-6B3A0DE46;Network=DBMSLPCN;Trusted_Connection=Yes"

Note:

The connection parameters in ConnectionString do not need to include the Provider or Remote Provider parameter if you specify an ADO provider using the Provider property. Similarly, you do not need to specify the Data Source parameter if you use the DefaultDatabase property.

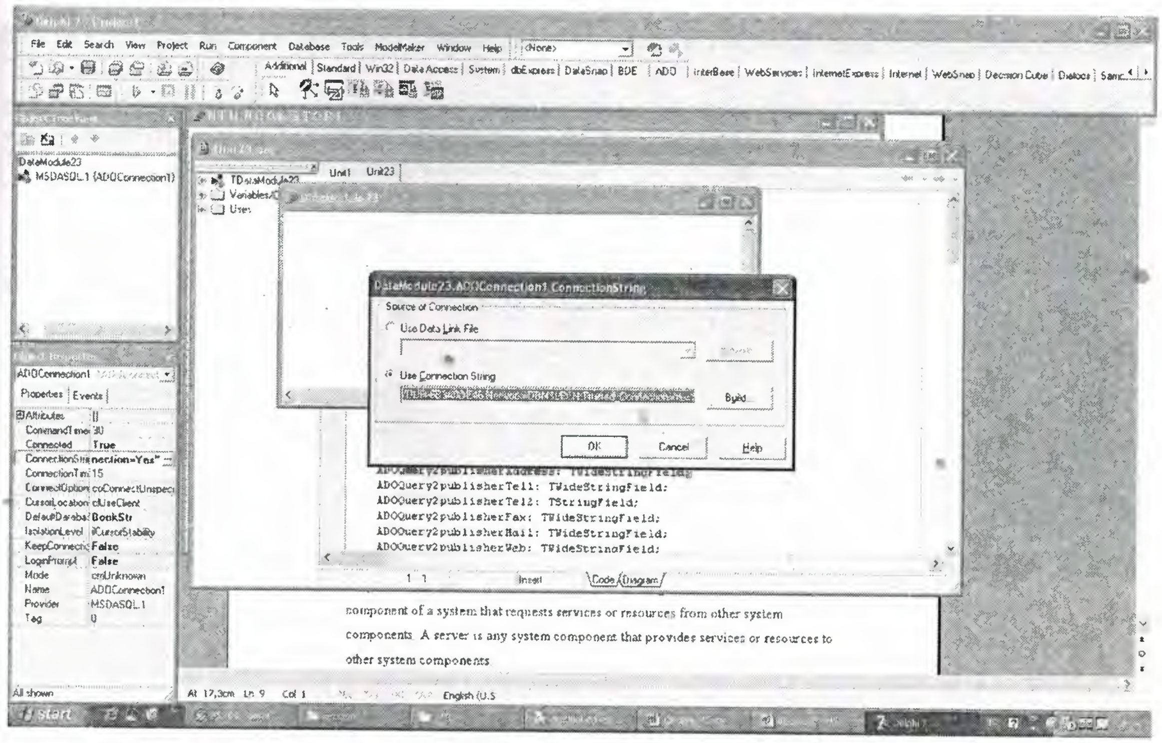
In addition, to the parameters listed above, ConnectionString can include any connection parameters peculiar to the specific ADO provider you are using. These additional connection parameters can include user ID and password if you want to hardcode the login information.

At design-time, you can use the Connection String Editor to build a connection string by selecting connection elements (like the provider and server) from lists. Click the ellipsis button for the ConnectionString property in the Object Inspector to launch the Connection String Editor, which is an ActiveX property editor supplied by ADO.

Once you have specified the ConnectionString property (and, optionally, the Provider property), you can use the ADO connection component to connect to or disconnect from the ADO data store, although you may first want to use other properties to fine-tune the connection . When connecting to or disconnecting from the data store, TADOConnection lets you respond to a few additional events beyond those common to all database connection components..

Note :

If you do not explicitly activate the connection by setting the connection component's Connected property to True, it automatically establishes the connection when the first dataset component is opened or the first time you use an ADO command component to execute a command.



⊞Attributes	[]
CommandTime	30
Connected	True
ConnectionStr	nection=Yes"
ConnectionTi	15
ConnectOption	coConnectUnspeci
CursorLocation	clUseClient
DefaultDataba	BookStr
IsolationLevel	ilCursorStability
KeepConnectio	False
LoginPrompt	False
Mode	cmUnknown
Name	ADOConnection1
Provider	MSDASQL.1
Tag	0

DataModule23.ADOConnection1.ConnectionString

Source of Connection

( ) Use Data Link File

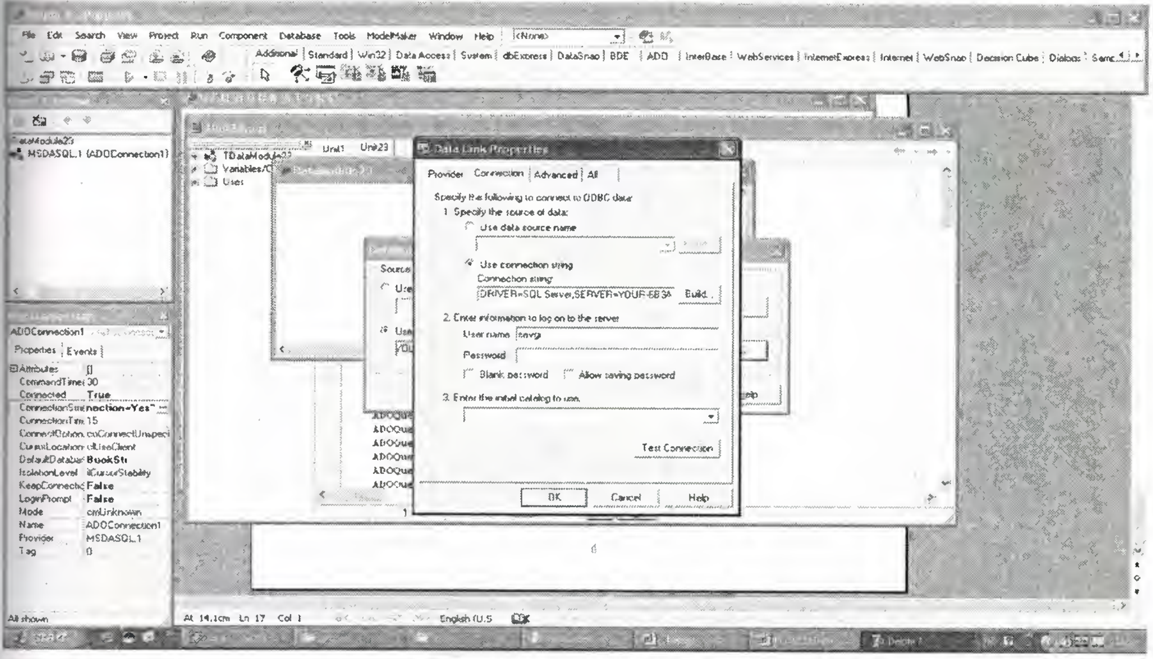
(•) Use Connection String

```
ADOQuery2publisherAddress: TWideStringField;
ADOQuery2publisherTel1: TWideStringField;
ADOQuery2publisherTel2: TStringField;
ADOQuery2publisherFax: TWideStringField;
ADOQuery2publisherMail: TWideStringField;
ADOQuery2publisherWeb: TWideStringField;
```

component of a system that requests services or resources from other system components. A server is any system component that provides services or resources to other system components

File Edit Search View Project Run Component Database Tools ModelMaker Window Help
Additional Standard Win32 Data Access System dbExpress DataSnap BDE ADO InterBase WebServices InternetExpress Internet WebSnap Decision Cube Dialogs
DataModule23
MSDASQL.1 (ADOConnection1)
Unit1 Unit23
TDataModule23
Variables/Constants
Uses
ADOConnection1
Properties Events
Build...
OK
Cancel
Help
Insert
Code Diagram
All shown
At 17,3cm Ln 9 Col 1
English (U.S
start

Figure 3.2 : How Ado Connection does in Delphi (step1)



Data Link Properties

Provider | Connection | Advanced | All

Specify the following to connect to ODBC data:

1. Specify the source of data:
   - ( ) Use data source name
   - (•) Use connection string

     Connection string: DRIVER=SQL Server;SERVER=YOUR-6B3A

2. Enter information to log on to the server
   - User name: savig
   - Password:
   - [ ] Blank password [ ] Allow saving password

3. Enter the initial catalog to use:

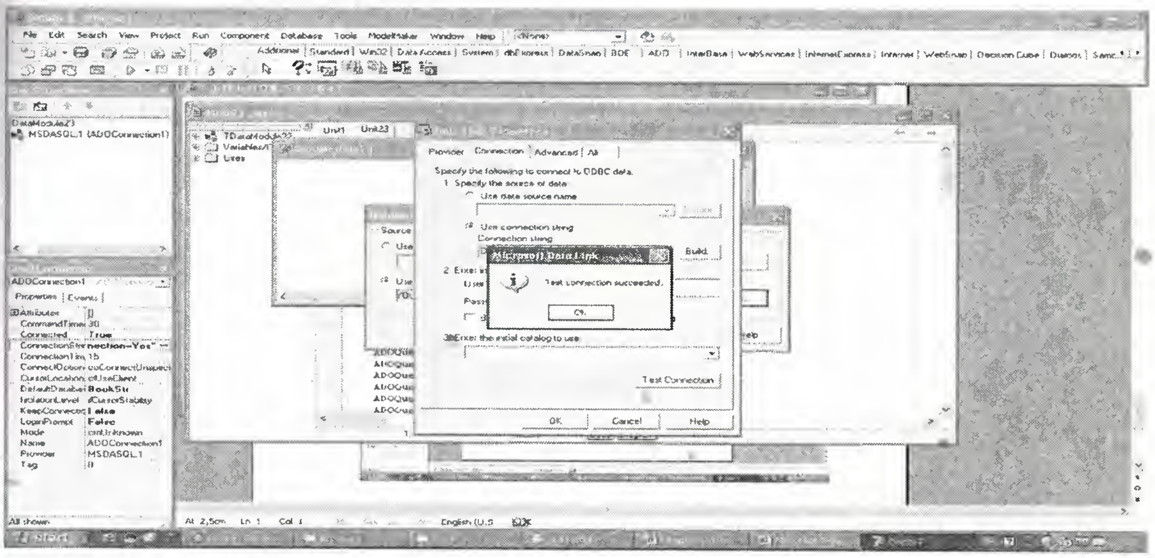
Test Connection

OK | Cancel | Help

ADOConnection1

Properties	Events
Attributes	[]
CommandTimeout	30
Connected	True
ConnectionString	nection=Yes"
ConnectionTimeout	15
ConnectOptions	coConnectUnspeci
CursorLocation	clUseClient
DefaultDatabase	BookSt
IsolationLevel	ilCursorStability
KeepConnection	False
LoginPrompt	False
Mode	cmUnknown
Name	ADOConnection1
Provider	MSDASQL.1
Tag	0

Figure 3.3 : How Ado Connection does in Delphi (step2)



Microsoft Data Link

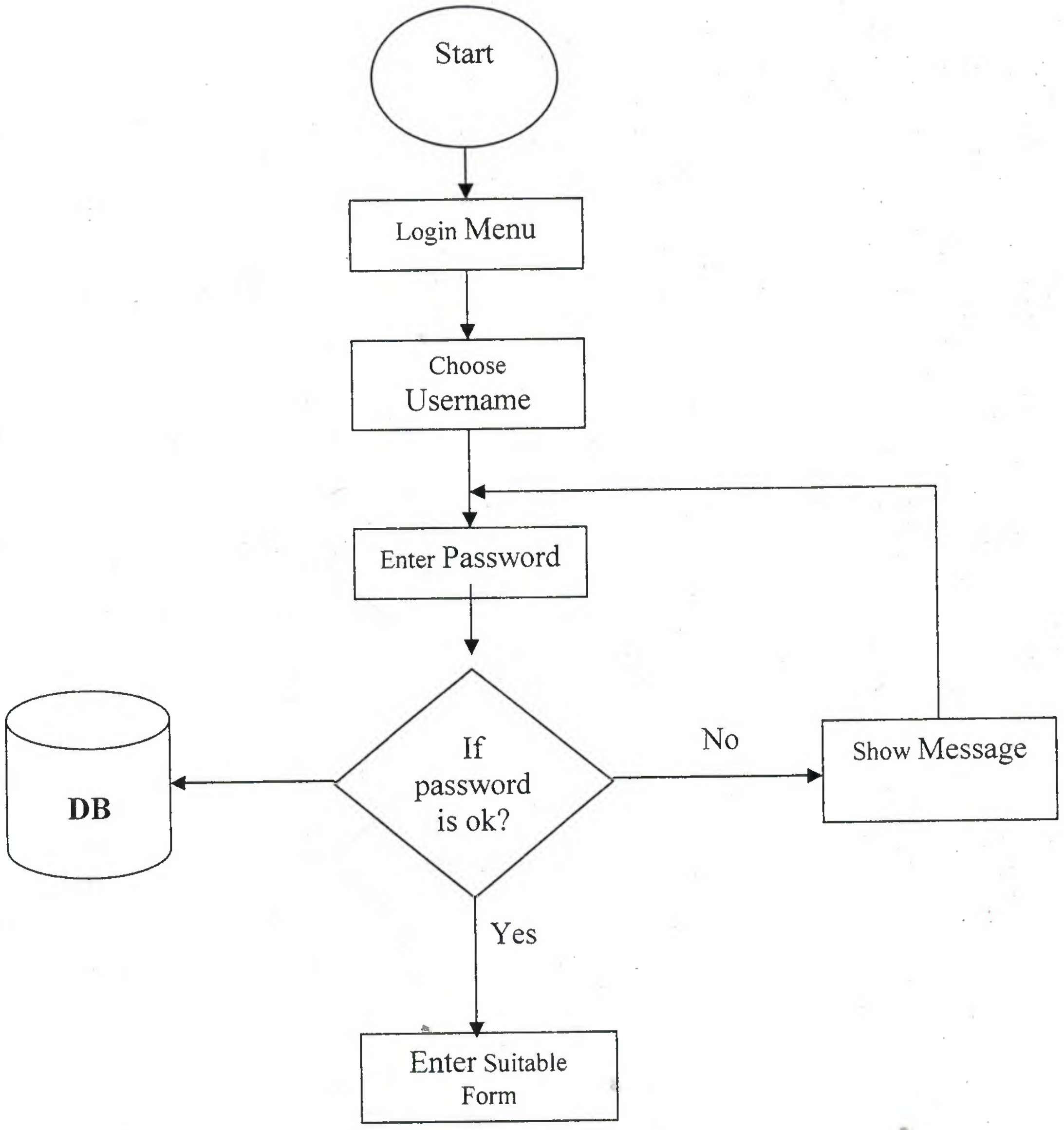
Test connection succeeded.

OK

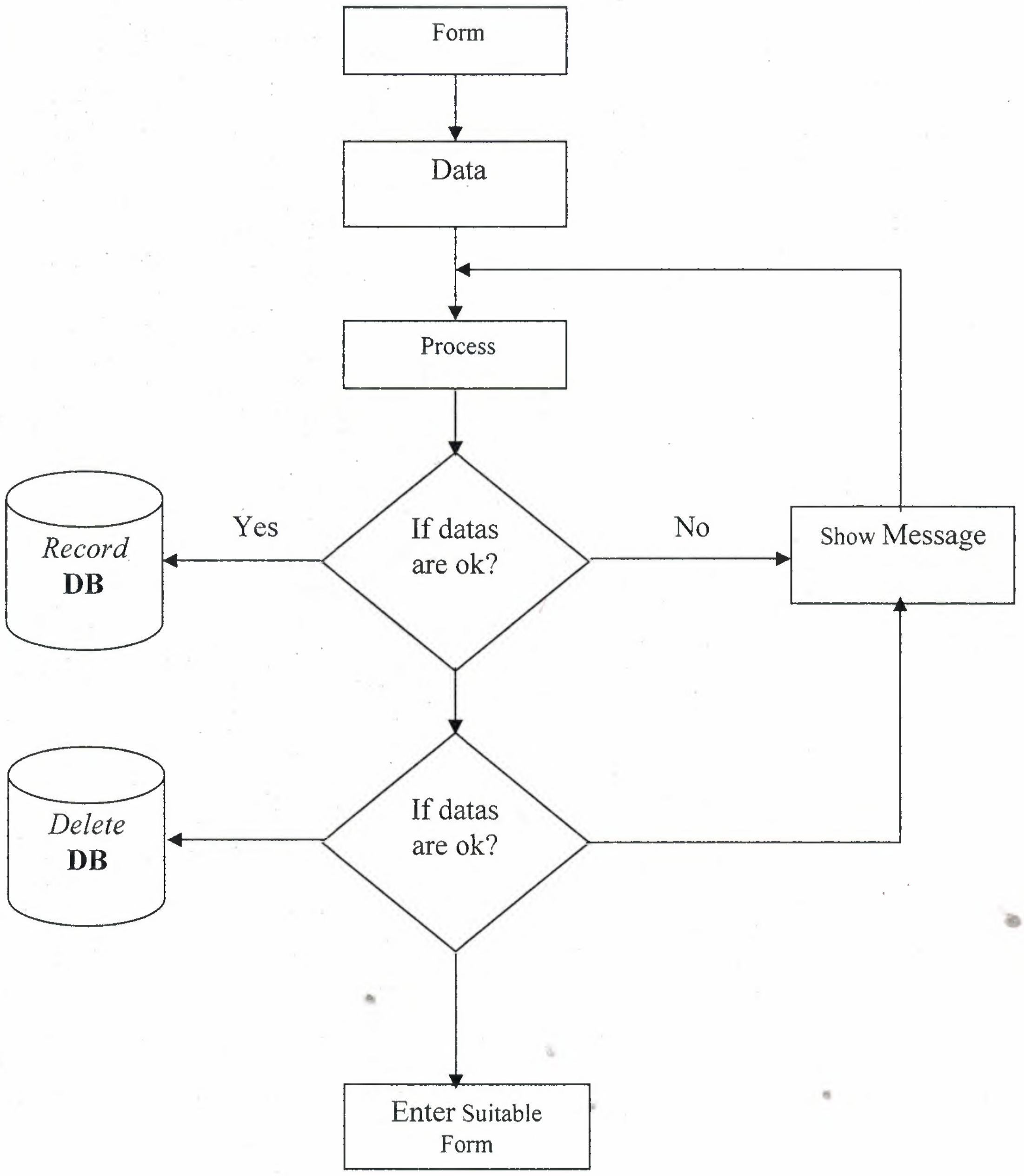
Figure 3.4 : How Ado Connection does in Delphi (step 3)

CHAPTER 4 : FLOWCHARTS OF PROGRAM

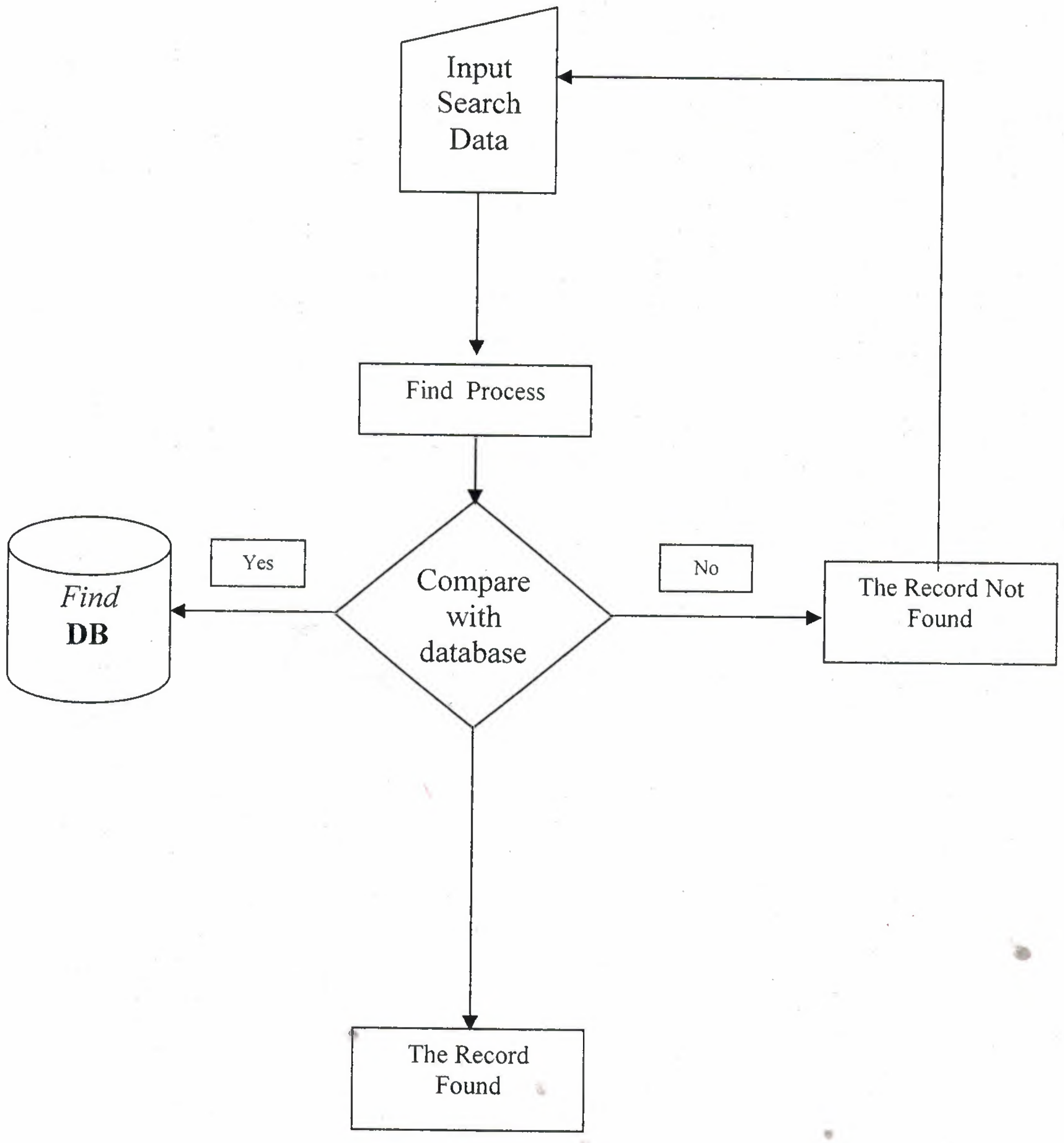
4.1 Flowchart Of Login Form



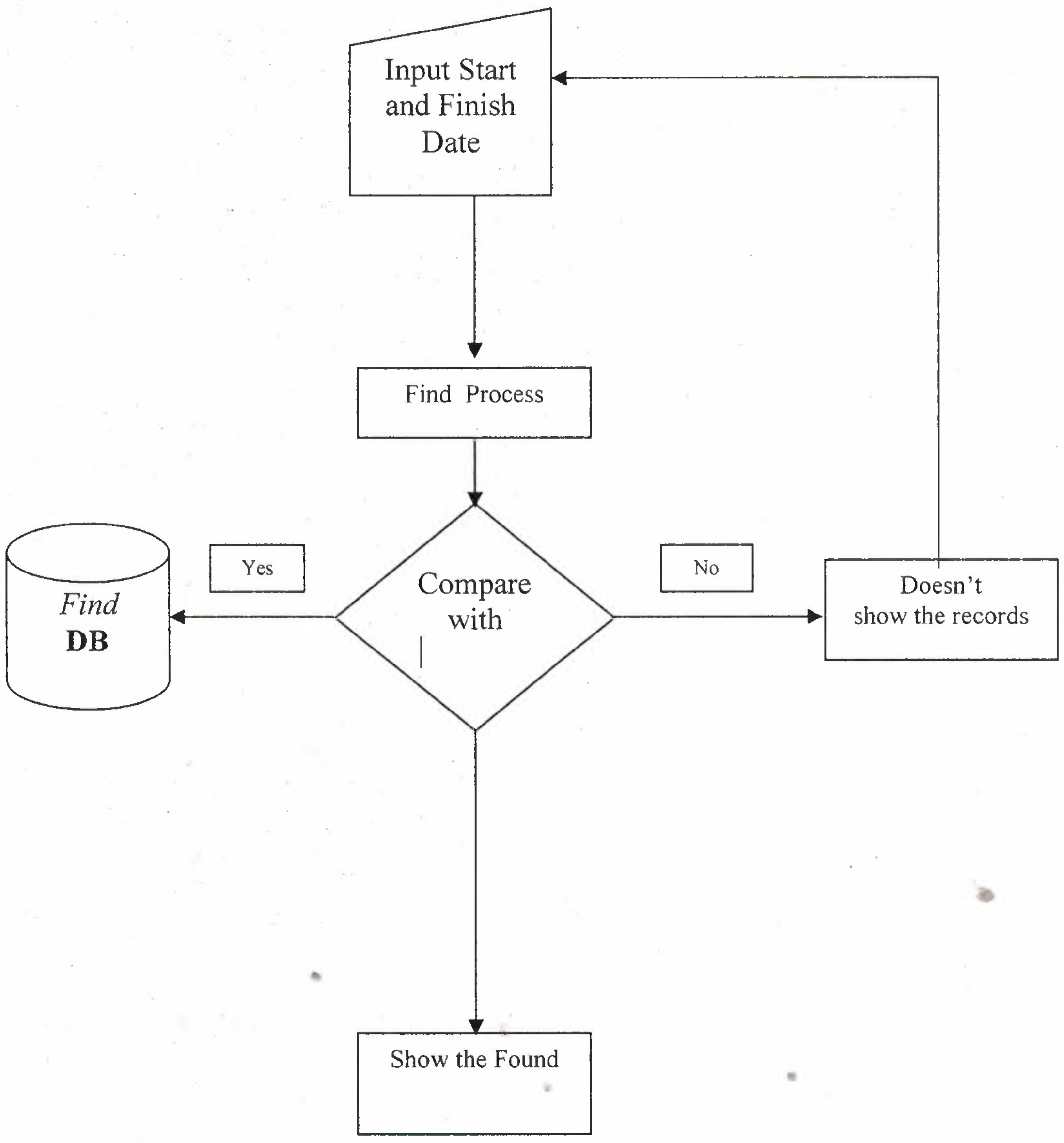
4.2 Flowchart Of Author-Publisher-Book Information Form



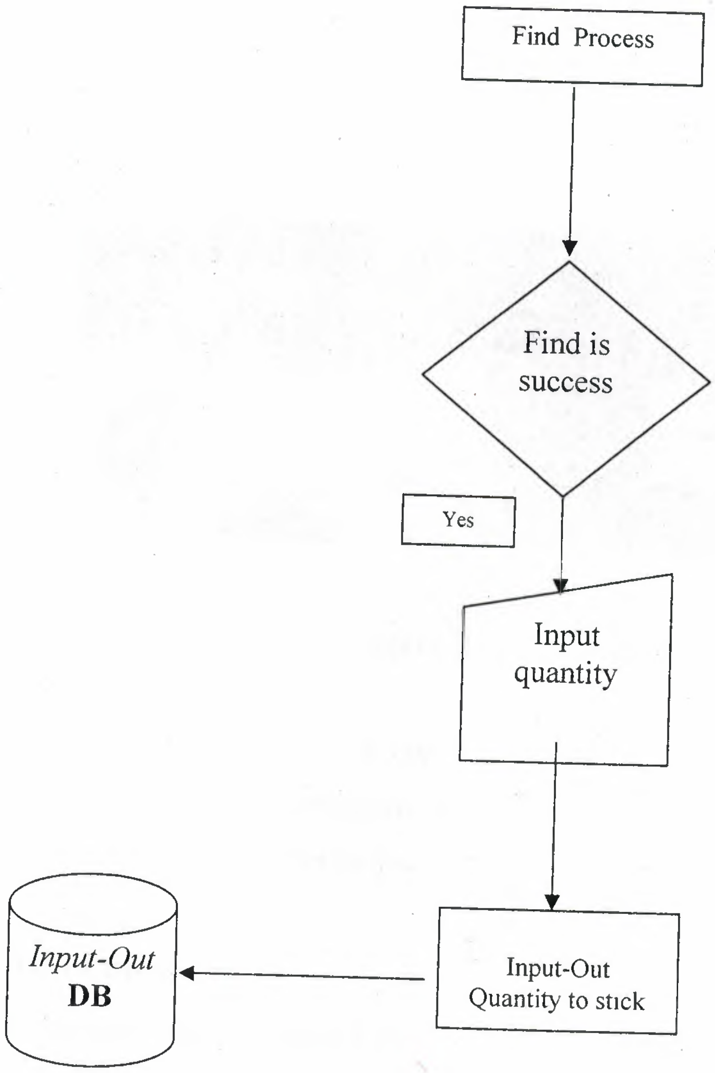
4.3 Find Process



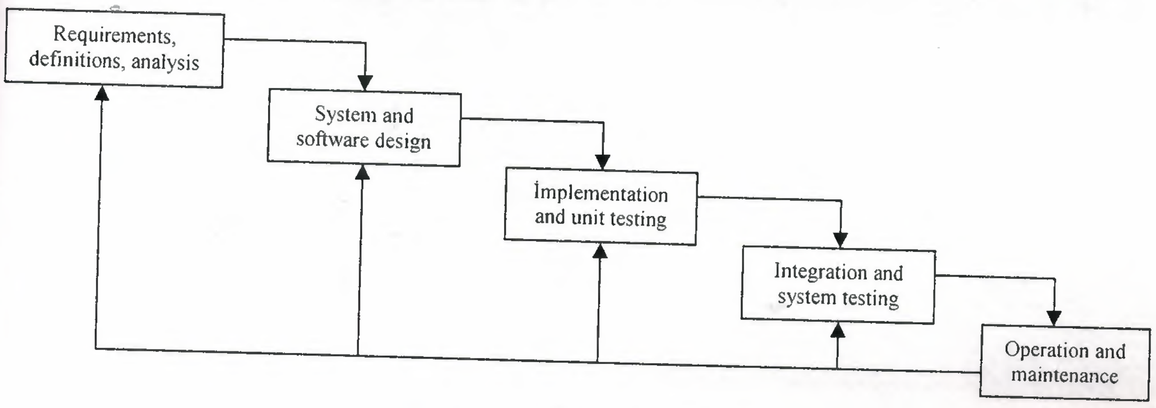
4.4 Finding between two dates datas



4.5 Stok Input-Output



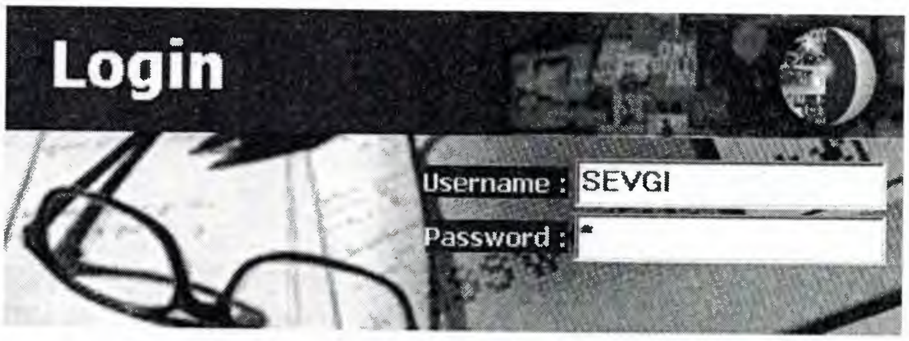
4.6 WATERFALL MODEL


Requirements, definitions, analysis
System and software design
İmplementation and unit testing
Integration and system testing
Operation and maintenance

CHAPTER 5 : BOOK STORE SYSTEM

5.1. Login Form

Login Form consists of user name and password entry. For the program security and to respect all user.



Username : SEVGI

Password : *

Figure 5.1 : Login Form

If you enter wrong password or username, program will give a message to you. Of course it is possible. But, program will guide you. What is wrong. When you enter suitable password and user name you will see Main Form Window.

5.2. Main Form

When you enter the system as a user, you will meet with main form with these types menu.

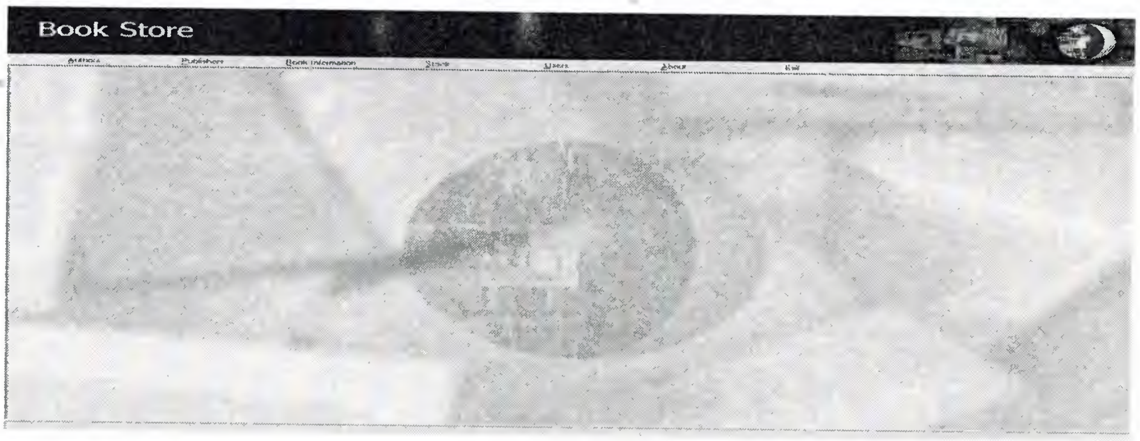
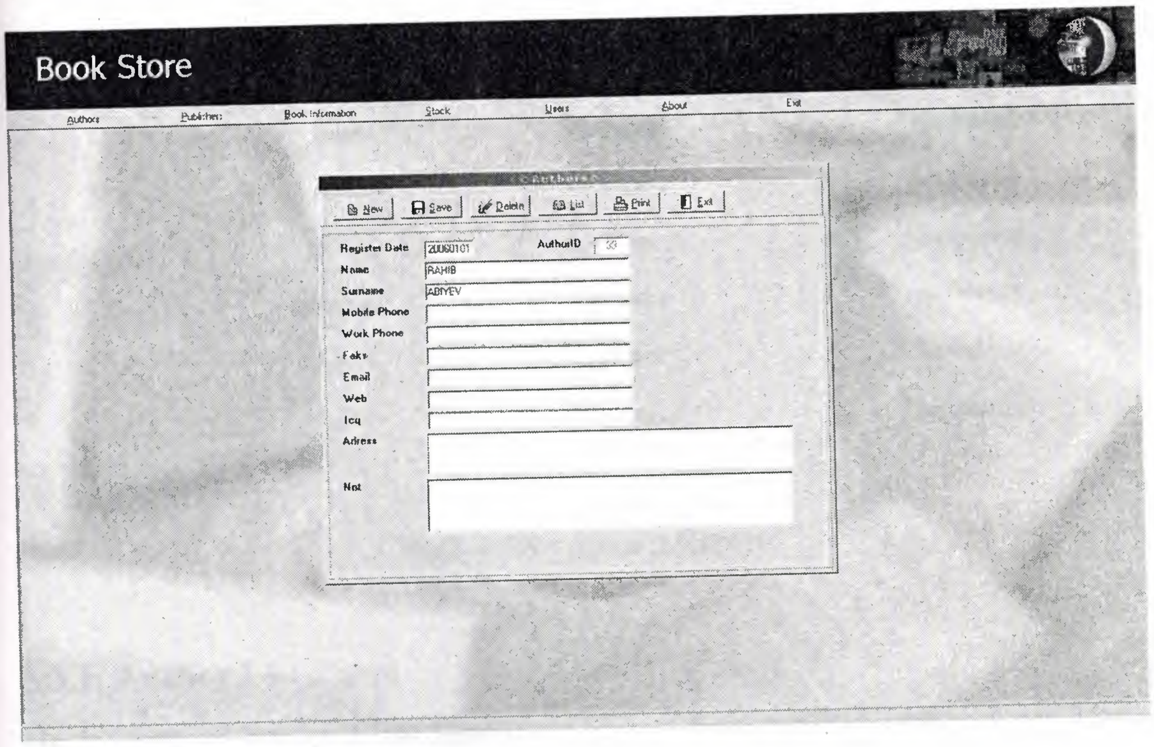


Figure 5.2 : Main Form

5.3 Authors Form



Register Date	20060101
AuthorID	33
Name	RAHIB
Surname	ABIYEV
Mobile Phone	
Work Phone	
Faks	
Email	
Web	
Icq	
Adress	
Not	

Figure 5.3 : Authors Form

In this form, you can entry a new record or delete the exist records about authors which name,surname,mobile phone etc... ,also update authors informations.You can search authors acording to name, whatever you know as you see on figure above, his/her.Also search process and record tour is optional.If you want to search and record tour in authors,you can click right clik mouse button and show→search or navigator which the commands it uses.You can print out the informations about authors in the following type.

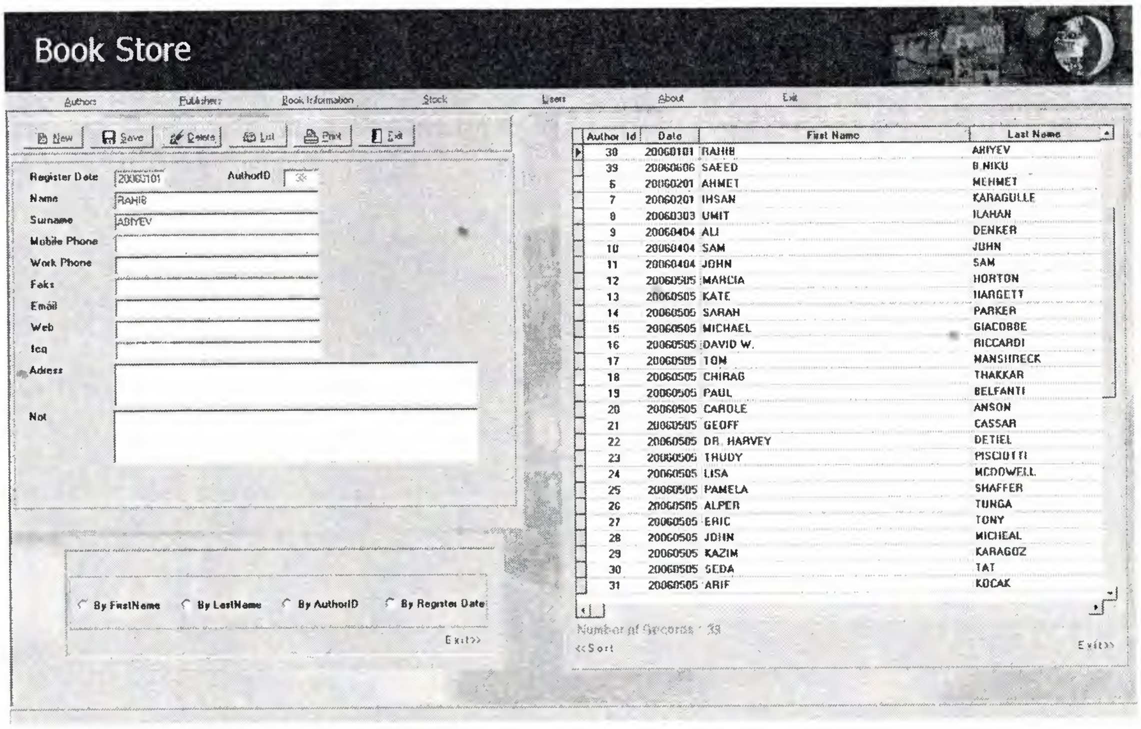
AUTHOR INFORMATION 6/3/2006 1:18:48 AM

ID	Name	Surname	Telefon
1	SEVGI	YAVUZ	05354600856
2	PINAR	MISIRLISOY	05352401223
3	MESUT	BULUT	05338645684
4	EYYUP	SOLKOL	05428846622
5	SERBULENT	KARAGOZ	05338687926
6	AHMET	MEHMET	05354600102
7	IHSAN	KARAGULLE	05358600203

Figure 5.4 : Authors Report

5.3.1. Author List Form

In this form,all authors is listed,and it can be seen the number of author record .If you click any record Author List Form you can see the details about authors..Also these records can be ordered some criterions which last name,first name,authorid and register date...



Register Date	20060101
AuthorID	3[illegible]
Name	RAHIB
Surname	ABIYEV
Mobile Phone	
Work Phone	
Faks	
Email	
Web	
Icq	
Adress	
Not	

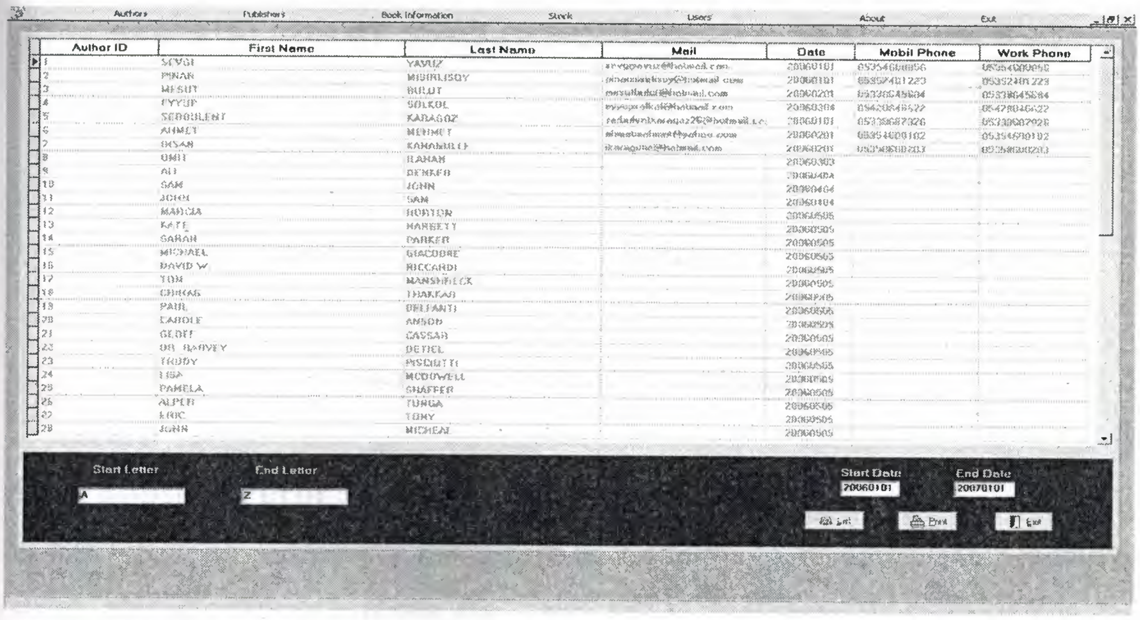
Author Id	Date	First Name	Last Name
30	20060101	RAHIB	AHIYEV
39	20060606	SAEED	B.NIKU
6	20060201	AHMET	MEHMET
7	20060201	IHSAN	KARAGULLE
8	20060303	UMIT	ILAHAN
9	20060404	ALI	DENKER
10	20060404	SAM	JOHN
11	20060404	JOHN	SAM
12	20060505	MARCIA	HORTON
13	20060505	KATE	HARGETT
14	20060505	SARAH	PARKER
15	20060505	MICHAEL	GIACOBBE
16	20060505	DAVID W.	RICCARDI
17	20060505	TOM	MANSHRECK
18	20060505	CHIRAG	THAKKAR
19	20060505	PAUL	BELFANTI
20	20060505	CAROLE	ANSON
21	20060505	GEOFF	CASSAR
22	20060505	DR. HARVEY	DETIEL
23	20060505	THUDY	PISCIOTTI
24	20060505	LISA	MCDOWELL
25	20060505	PAMELA	SHAFFER
26	20060505	ALPER	TUNGA
27	20060505	ERIC	TONY
28	20060505	JOHN	MICHEAL
29	20060505	KAZIM	KARAGOZ
30	20060505	SEDA	TAT
31	20060505	ARIF	KOCAK

Number of Records : 39

Figure 5.5 : Author List Form

You can find author just writing author name when writing author name what will be come in suitable form.

5.3.2. Authors Report



Author ID	First Name	Last Name	Mail	Date	Mobil Phone	Work Phone
1	SEVGI	YAVUZ	sevgiyavuz@hotmail.com	20060101	05354600856	05354600856
2	PINAR	MISIRLISOY	pinarmisirlisoy@hotmail.com	20060101	05352401223	05352401223
3	MESUT	BULUT	mesutbulut@hotmail.com	20060201	05338645884	05338645884
4	EYYUP	SOLKOL	eyyupsolkol@hotmail.com	20060304	05420846522	05478046522
5	SEDBULENT	KARAGOZ	sedabulentkaragoz26@hotmail.c	20060101	05338687026	05338687026
6	AHMET	MEHMET	ahmetmehmet@yahoo.com	20060201	05354600102	05354670102
7	IHSAN	KARABULUT	[illegible]	20060201	05354680203	05354680203
8	UMIT	ILAHAN		20060303		
9	ALI	DEMKER		20060404		
10	SAM	JOHN		20060404		
11	JOHN	SAM		20060104		
12	MARCIA	HORTON		20060505		
13	KATE	HARBETT		20060505		
14	SARAH	PARKER		20060505		
15	MICHAEL	GIACOBBE		20060505		
16	DAVID W.	RICCARDI		20060505		
17	TOM	MANSFIELCK		20060505		
18	CHIRAG	THAKKAR		20060505		
19	PAUL	DELFANTI		20060505		
20	CAROLE	ANSON		20060505		
21	GEOFF	CASSAR		20060505		
22	DR. HARVEY	DEITEL		20060505		
23	TRUDY	PISCIOTTI		20060505		
24	LISA	MCDOWELL		20060505		
25	PAMELA	SHAFFER		20060505		
26	ALPER	TUNGA		20060505		
27	ERIC	TONY		20060505		
28	JOHN	MICHEAL		20060505		

Start Letter: A

End Letter: Z

Start Date: 20060101

End Date: 20070101

Figure 5.6 : Author Report

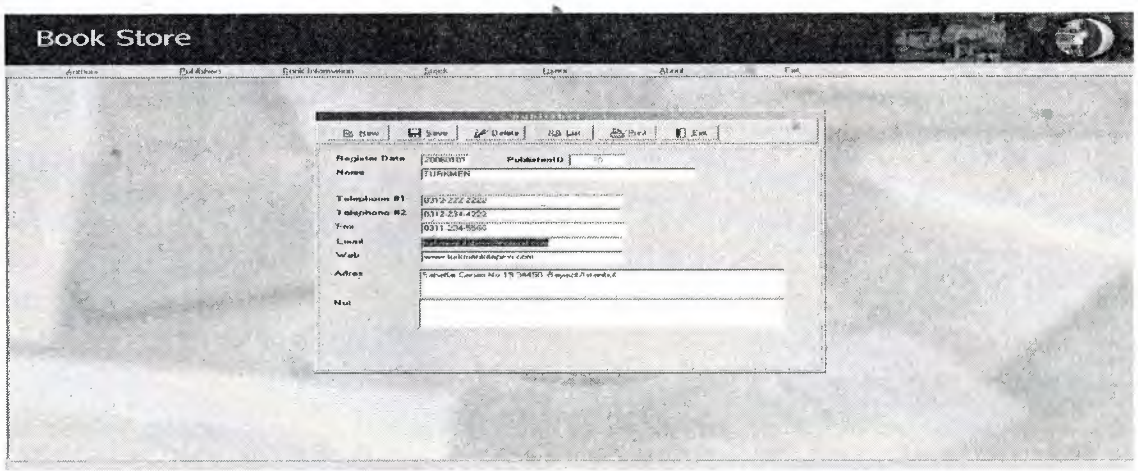
You can see this form about authors according to some criterions.It can be seen the informations about authors between two date value and authors name, and also print out these informations in the following type.

AUTHOR BALANCE			6/3/2006 8:16:17 AM
ID	Name	Last Name	Date
1	SEVGI	YAVUZ	20060101
2	PINAR	MISIRLISOY	20060101
3	MESUT	BULUT	20060201
4	EYYUP	SOLKOL	20060304
5	SERBULENT	KARAGOZ	20060101
6	AHMET	MEHMET	20060201
7	IHSAN	KARAGULLE	20060201
8	UMIT	ILAHAN	20060303
9	ALI	DENKER	20060404

Figure 5.7 : Author Balance Report

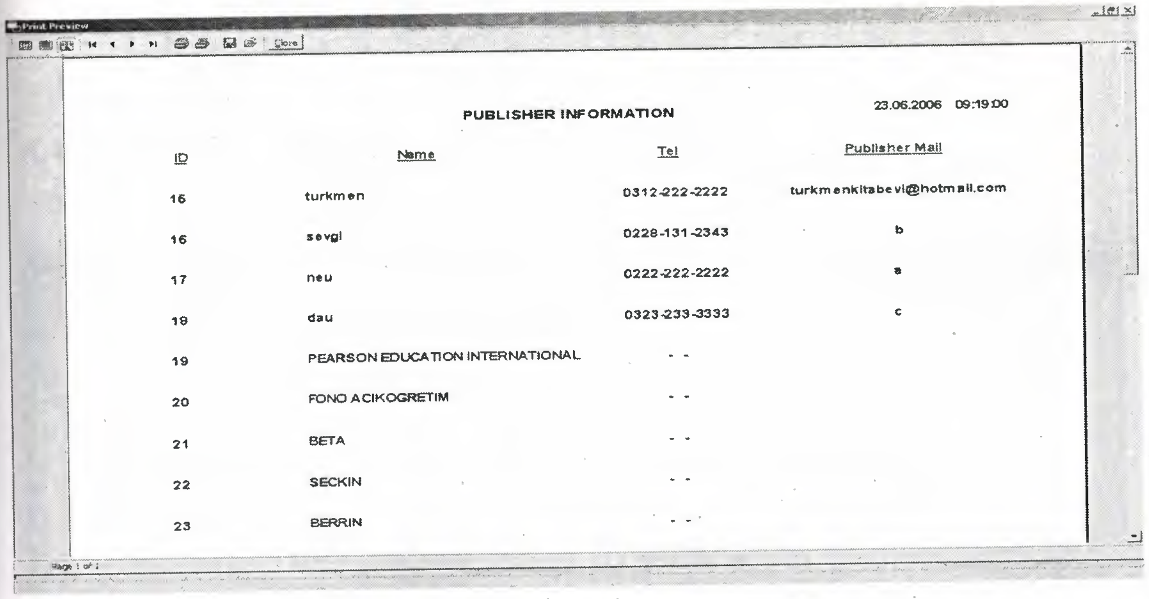
5.4. Publisher Form

In this form, you can entry a new record or delete the exist records about publisher which name,email,mobile phone etc... ,also update publishers informations.You can search publisher acording to name, whatever you know as you see on figure above, his/her.Also search process and record tour is optional.If you want to search and record tour in authors,you can click right clik mouse button and show→search or navigtor which the commands it uses.You can print out the informations about publishers in the following type.



Register Date: 20060101 PublisherID: [illegible]
Name: TURKMEN
Telephone #1: 0312 222 2222
Telephone #2: 0312 234 4222
Fax: 0311 234 5566
Email: [illegible]
Web: www.[illegible].com
Adres: [illegible] No 13 34450 [illegible]/Istanbul
Not:

Figure 5.8 : Publisher Form



PUBLISHER INFORMATION

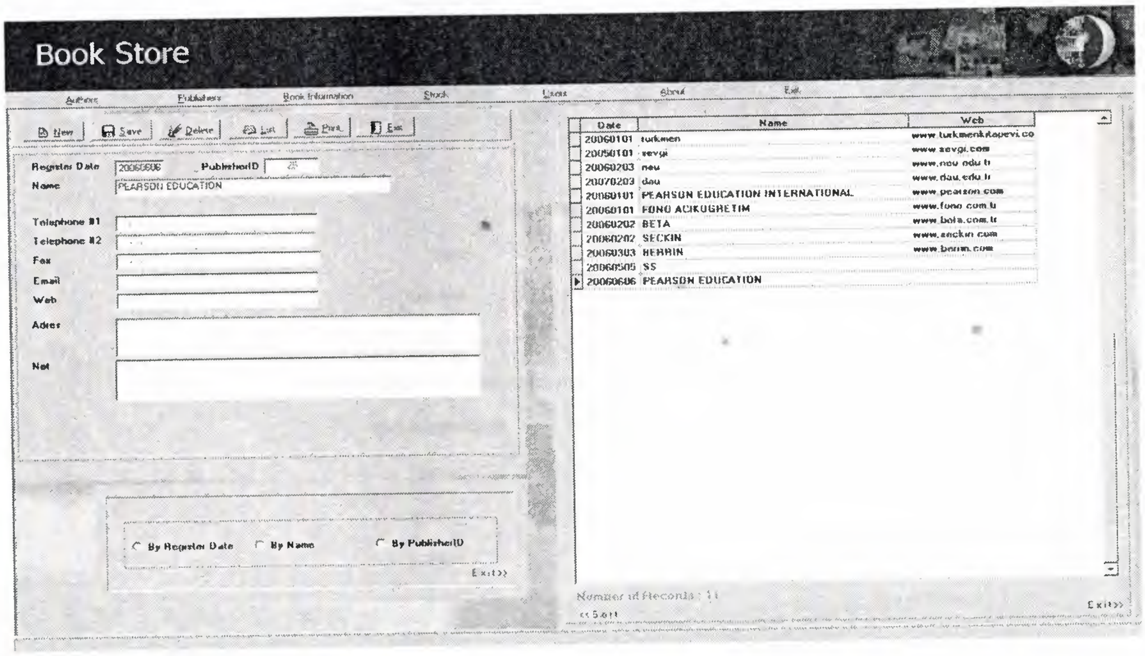
23.06.2006 09:19:00

ID	Name	Tel	Publisher Mail
16	turkmen	0312-222-2222	turkmenkitabevi@hotmail.com
16	sevgi	0228-131-2343	b
17	neu	0222-222-2222	a
18	dau	0323-233-3333	c
19	PEARSON EDUCATION INTERNATIONAL	- -	
20	FONO ACIKOGRETIM	- -	
21	BETA	- -	
22	SECKIN	- -	
23	BERRIN	- -	

Figure 5.9 : Publisher Report

5.4.1. Publisher List Form

In this form,all publishers is listed,and it can be seen the number of publisher record .If you click any record Publisher List Form you can see the details about publishers..Also these records can be ordered some criterions which last name,first name,authorid and register date...



Book Store

Register Date: 20060606 PublisherID: 25
Name: PEARSON EDUCATION
Telephone #1:
Telephone #2:
Fax:
Email:
Web:
Adres:
Not:

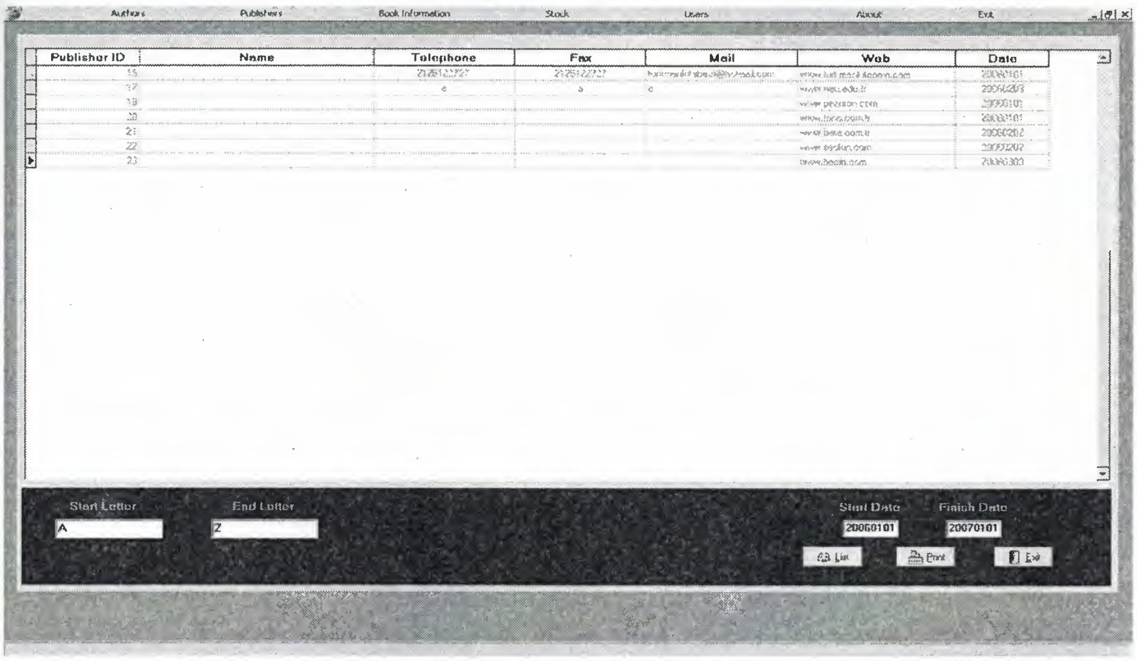
By Register Date / By Name / By PublisherID

Date	Name	Web
20060101	turkmen	www.turkmenkitapevi.co
20050101	sevgi	www.sevgi.com
20060203	neu	www.neu.edu.tr
20070203	dau	www.dau.edu.tr
20060101	PEARSON EDUCATION INTERNATIONAL	www.pearson.com
20060101	FONO ACIKOGRETIM	www.fono.com.tr
20060202	BETA	www.beta.com.tr
20060202	SECKIN	www.seckin.com
20060303	BERRIN	www.berrin.com
20060505	SS	
20060606	PEARSON EDUCATION	

Number of Records : 11

Figure 5.10 : Publisher List Form

5.4.2 Publishers Report

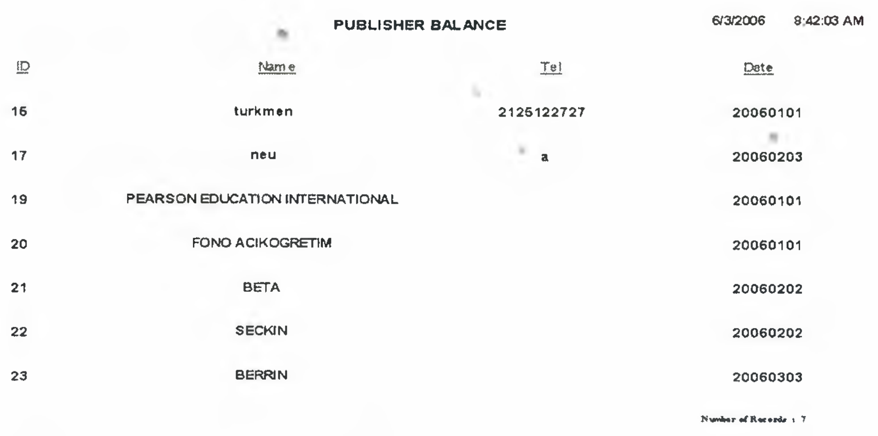


Publisher ID	Name	Telephone	Fax	Mail	Web	Date
16		2125122727	2125122727	[illegible]	[illegible]	20060101
17		e	a	e	www.neu.edu.tr	20060203
19					www.pearson.com	20060101
20					www.fono.com.tr	20060101
21					www.beta.com.tr	20060202
22					www.seckin.com	20060202
23					www.berrin.com	20060303

Start Letter: A
End Letter: Z
Start Date: 20060101
Finish Date: 20070101

Figure 5.11 : Publishers Report

You can see this form about publishers according to some criterions.It can be seen the informations about publishers between two date value and publisher name, and also print out these informations in the following type.



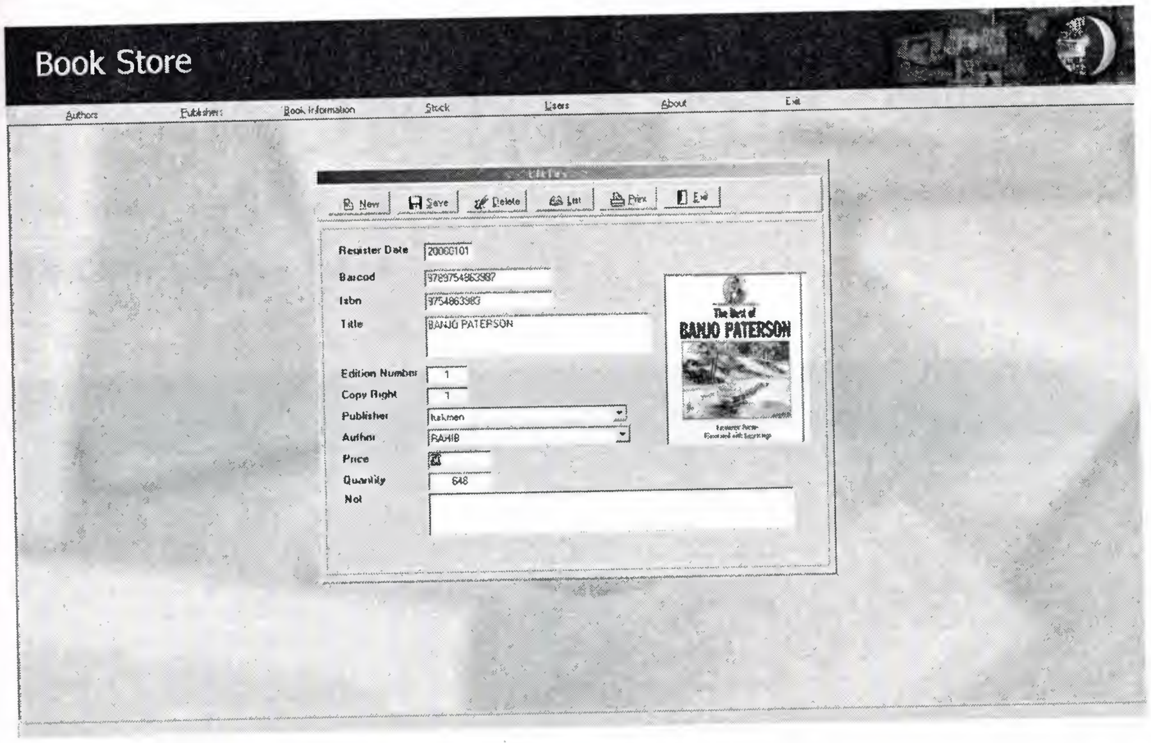
PUBLISHER BALANCE 6/3/2006 8:42:03 AM

ID	Name	Tel	Date
16	turkmen	2125122727	20060101
17	neu	a	20060203
19	PEARSON EDUCATION INTERNATIONAL		20060101
20	FONO ACIKOGRETIM		20060101
21	BETA		20060202
22	SECKIN		20060202
23	BERRIN		20060303

Number of Records : 7

Figure 5.12 : Publisher Balance Report

5.5 Book Information Form



Register Date 20060101
Barcod 9789754863987
Isbn 9754863983
Title BANJO PATERSON
Edition Number 1
Copy Right 1
Publisher hakmen
Author RAHIB
Price 23
Quantity 648
Not

Figure 5.13 : Book Information

In the above figure, you can entry a new informations or remove the informations about book and here when you entry the informations author and publisher names will come automatically,so the user doesn't entry incorect information,also if you want to add a picture,the picture can be added with double mouse click on the picture.

5.5.1 Book Information Report

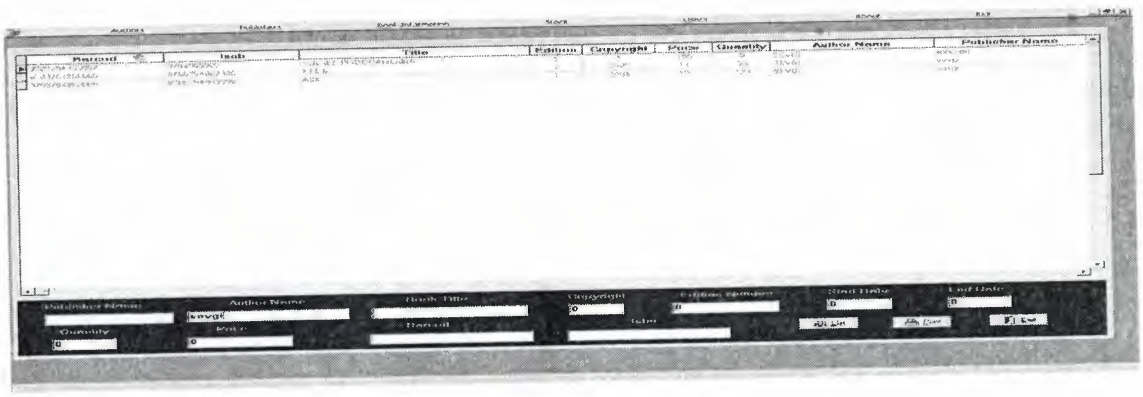
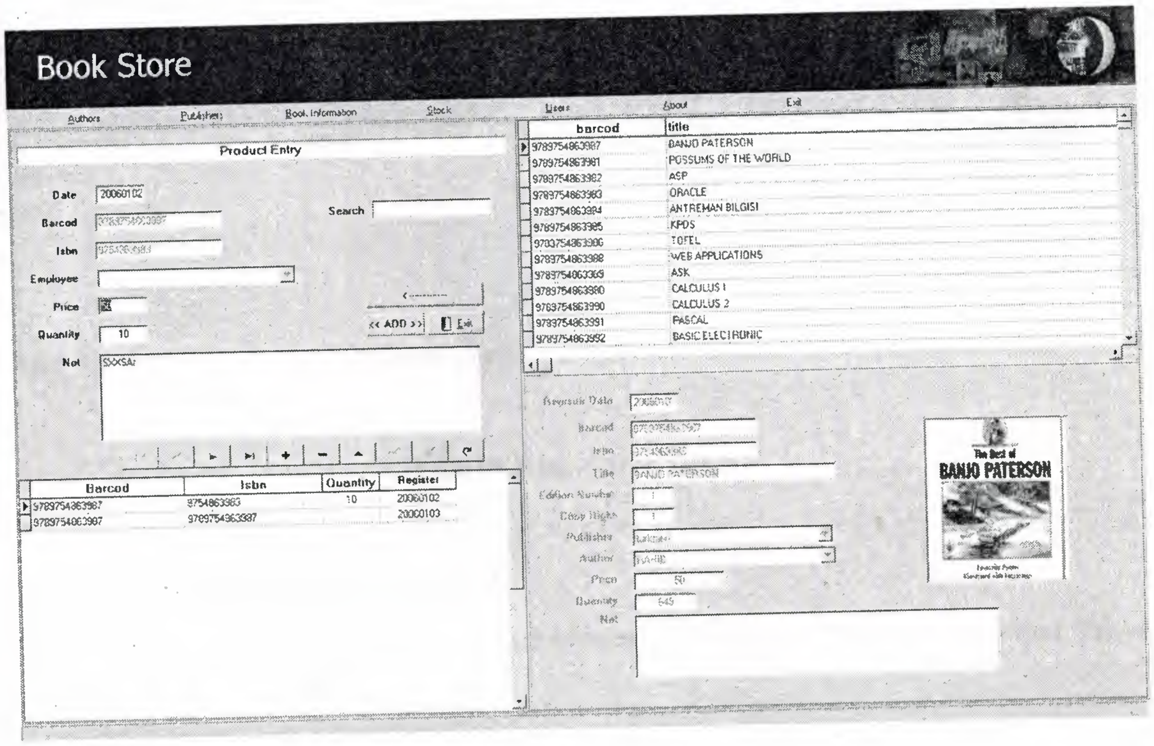


Figure 5.14 : Book Balance Report

In the above figure,the book informations can be filtered some criterions which publisher name,author name,book title,copyrigth etc....In here my aim is to show The user various informations about books.For example show me that publisher name is 'Turkmen' or Price is 10 etc....

5.6. Stock

5.6.1 Stock Entry Form



Date	20060102
Barcod	[illegible]
Isbn	[illegible]
Employee	
Price	[illegible]
Quanlity	10
Not	SXXSAr
Search	

Barcod	Isbn	Quantity	Register
9789754863987	9754863383	10	20060102
9789754863987	9789754863387		20000103

barcod	title
9789754863987	BANJO PATERSON
9789754863981	POSSUMS OF THE WORLD
9789754863982	ASP
9789754863983	ORACLE
9789754863984	ANTREMAN BILGISI
9789754863985	KPDS
9789754863986	TOFEL
9789754863988	WEB APPLICATIONS
9789754863369	ASK
9789754863980	CALCULUS I
9789754863990	CALCULUS 2
9789754863991	PASCAL
9789754863992	BASIC ELECTRONIC

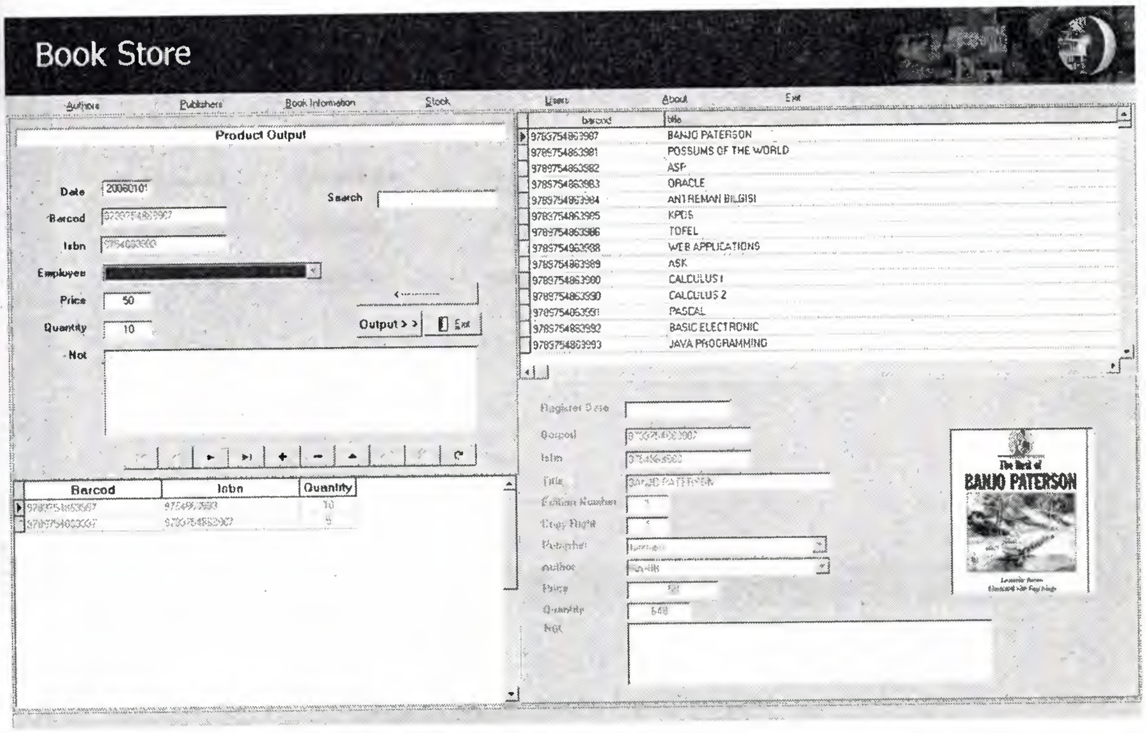
[illegible] Date	[illegible]
Barcod	[illegible]
Isbn	[illegible]
Title	BANJO PATERSON
Edition Number	1
Copy Right	1
Publisher	[illegible]
Author	[illegible]
Price	50
Quantity	645
Not	

Figure 5.15 : Stok Entry

In this form,user can entry to stock when entry the search the barcod or click the list of books automatically the book informations can be shown.After that user can entry the date,employee name,isbn,price,quantity,and memo about book.After that the user click the add button the quantity automatically is added in stock.

5.6.2 Stok Output Form

In the following form,user can decrease to stok when entry the search the barcod or click the list of books automatically the book informations can be shown.After that user can entry the date,employee name,isbn,price,quantity,and memo about book.After that the user click the output button the quantity automatically is decreased in stock.


Book Store

Authors | Publishers | Book Information | Stock | Users | About | Exit

Product Output

Date: 2006010[illegible]
Barcod: [illegible]
Isbn: [illegible]
Employee:
Price: 50
Quantity: 10
Not:

Search:

Output > > | Exit

Barcod	Isbn	Quantity
[illegible]	[illegible]	10
[illegible]	[illegible]	5

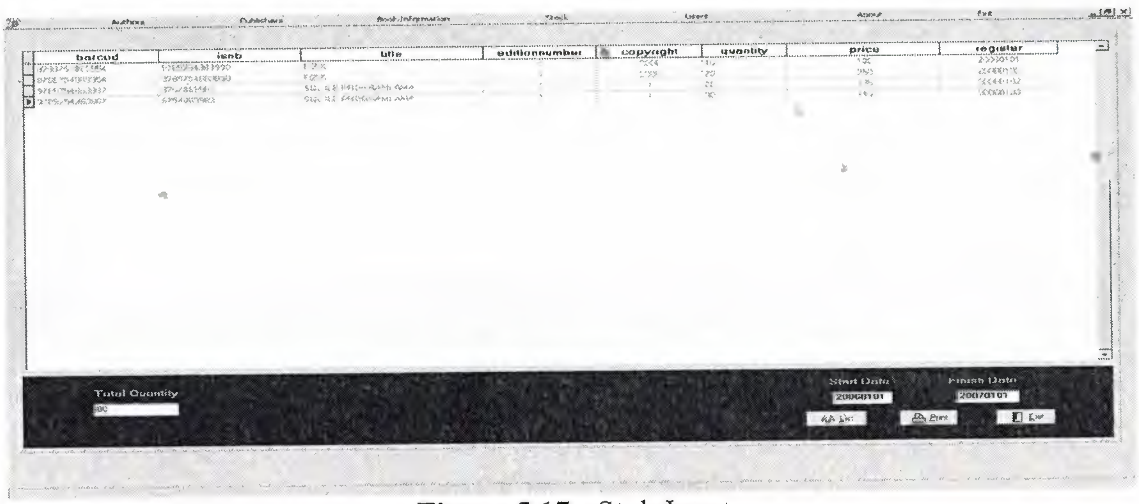
barcod	title
9783754863987	BANJO PATERSON
9789754863981	POSSUMS OF THE WORLD
9789754863982	ASP
9789754863983	ORACLE
9789754863984	ANTREMAN BILGISI
9789754863985	KPDS
9789754863986	TOFEL
9789754963988	WEB APPLICATIONS
9789754863989	ASK
9789754863980	CALCULUS I
9789754863990	CALCULUS 2
9789754063991	PASCAL
9789754863992	BASIC ELECTRONIC
9789754863993	JAVA PROGRAMMING

Register Date:
Barcod: [illegible]
Isbn: [illegible]
Title: BANJO PATERSON
Edition Number: [illegible]
Copy Right: [illegible]
Publisher: [illegible]
Author: [illegible]
Price: [illegible]
Quantity: 648
Not:

Figure 5.16 : Stok Output

5.6.3 Stok Input Balance Form

In the following form,the stock is listed between two dates values,and the input values is be showed to total quantity.


Authors | Publishers | Book Information | Stock | Users | About | Exit

barcod	isnb	title	editionnumber	copyright	quantity	price	registar
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

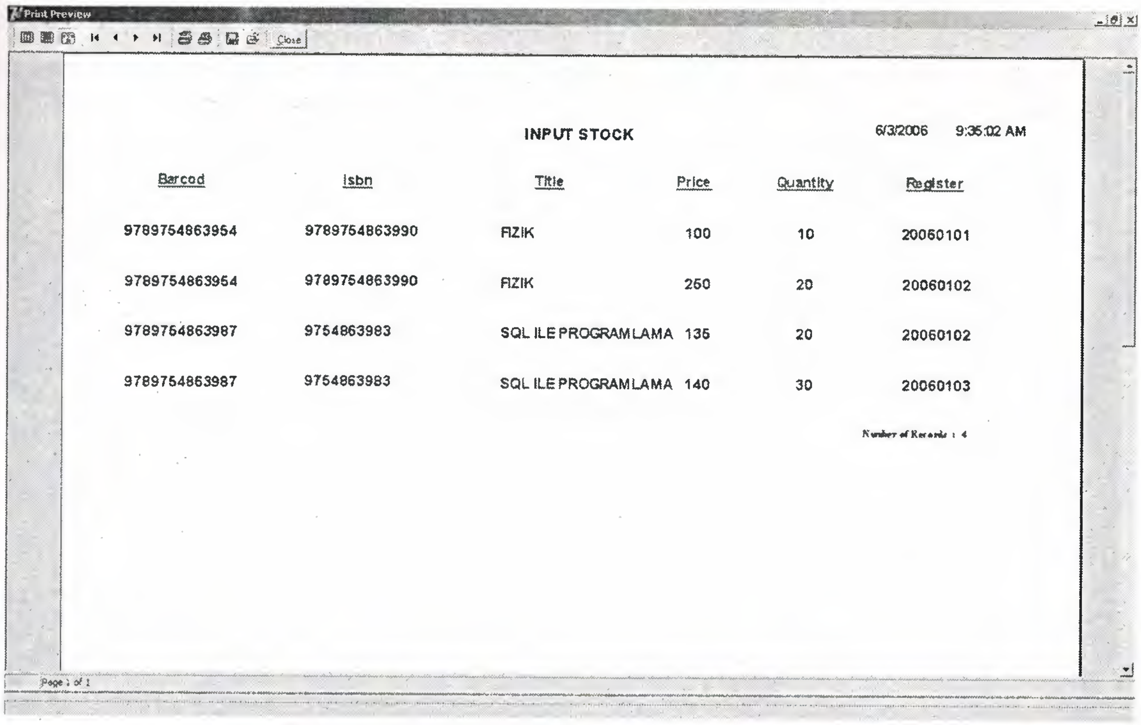
Total Quantity: [illegible]

Start Date: 20060101
Finish Date: 20070103

List | Print | Exit

Figure 5.17 : Stok Input

Also print out these informations in the following type.



INPUT STOCK 6/3/2006 9:35:02 AM

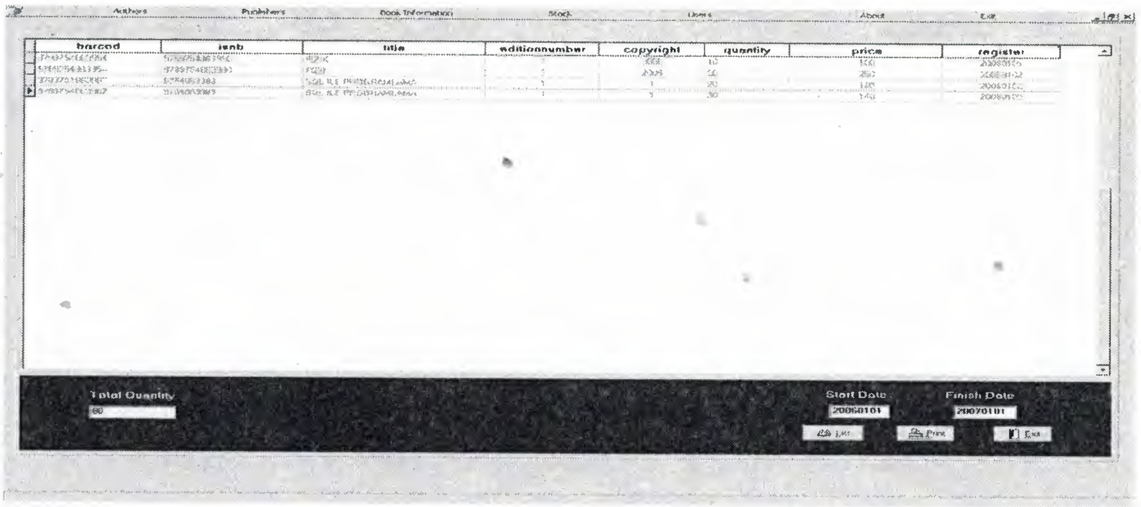
Barcod	Isbn	Title	Price	Quantity	Register
9789754863954	9789754863990	FIZIK	100	10	20060101
9789754863954	9789754863990	FIZIK	250	20	20060102
9789754863987	9754863983	SQL ILE PROGRAMLAMA	135	20	20060102
9789754863987	9754863983	SQL ILE PROGRAMLAMA	140	30	20060103

Number of Records : 4

Figure 5.18 : Stok Input Report

5.6.4 Stok Output Balance Form

In the following form,the stock is listed between two dates values,and the output values is be showed to total quantity.



barcod	isbn	title	editionnumber	copyright	quantity	price	register
[illegible]	[illegible]	FIZIK	1	[illegible]	10	100	[illegible]
[illegible]	[illegible]	FIZIK	1	2005	20	250	[illegible]
[illegible]	9754863983	SQL ILE PROGRAMLAMA	1	1	20	135	[illegible]
[illegible]	[illegible]	SQL ILE PROGRAMLAMA	1	1	30	140	[illegible]

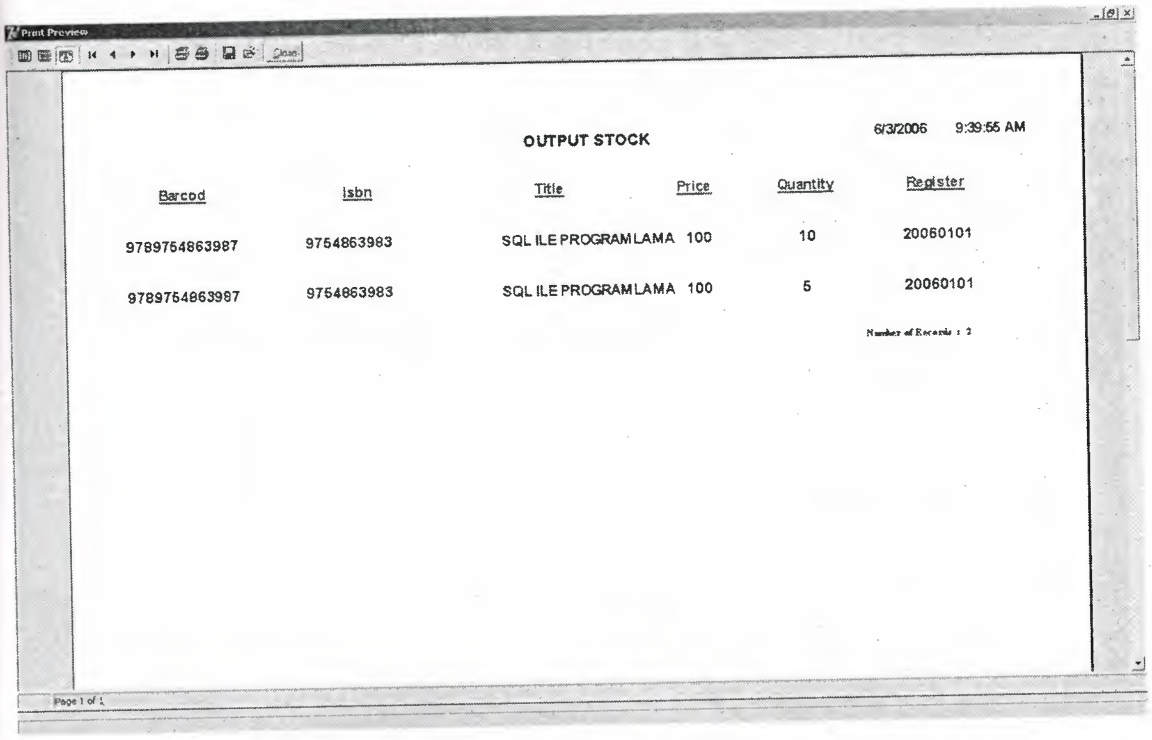
Total Quantity: 80

Start Date: 20060101

Finish Date: 20070101

Figure 5.19 : Stok Output

Also print out these informations in the following type



OUTPUT STOCK

6/3/2006 9:39:55 AM

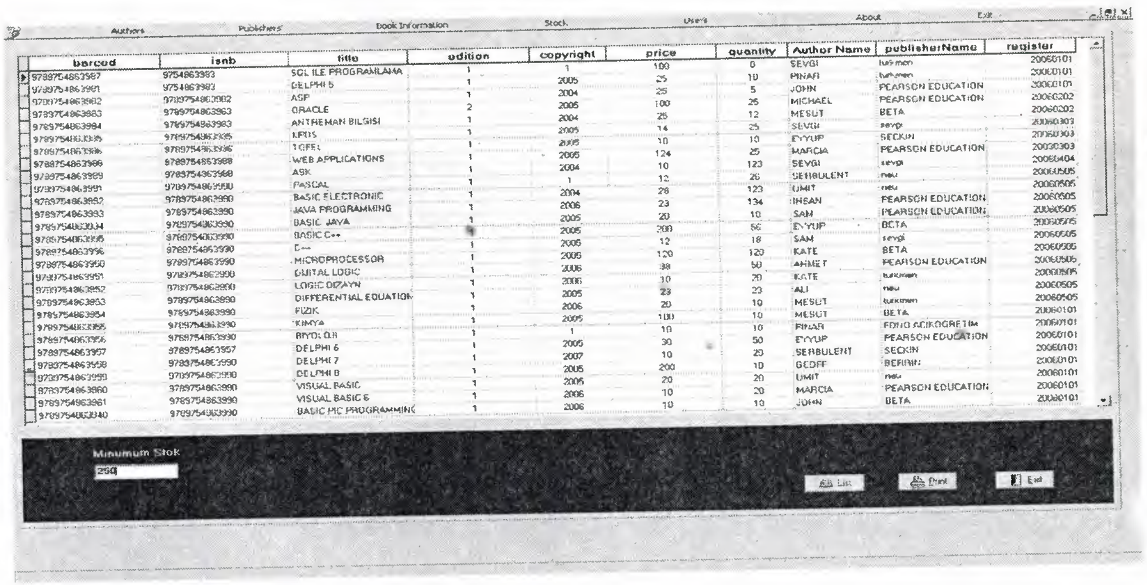
Barcod	Isbn	Title	Price	Quantity	Register
9789754863987	9754863983	SQL ILE PROGRAMLAMA	100	10	20060101
9789754863987	9754863983	SQL ILE PROGRAMLAMA	100	5	20060101

Number of Records : 2

Figure 5.20 : Output Stock

5.6.5.Minumum Stock

In the form user can be seen the stock values according to the user...

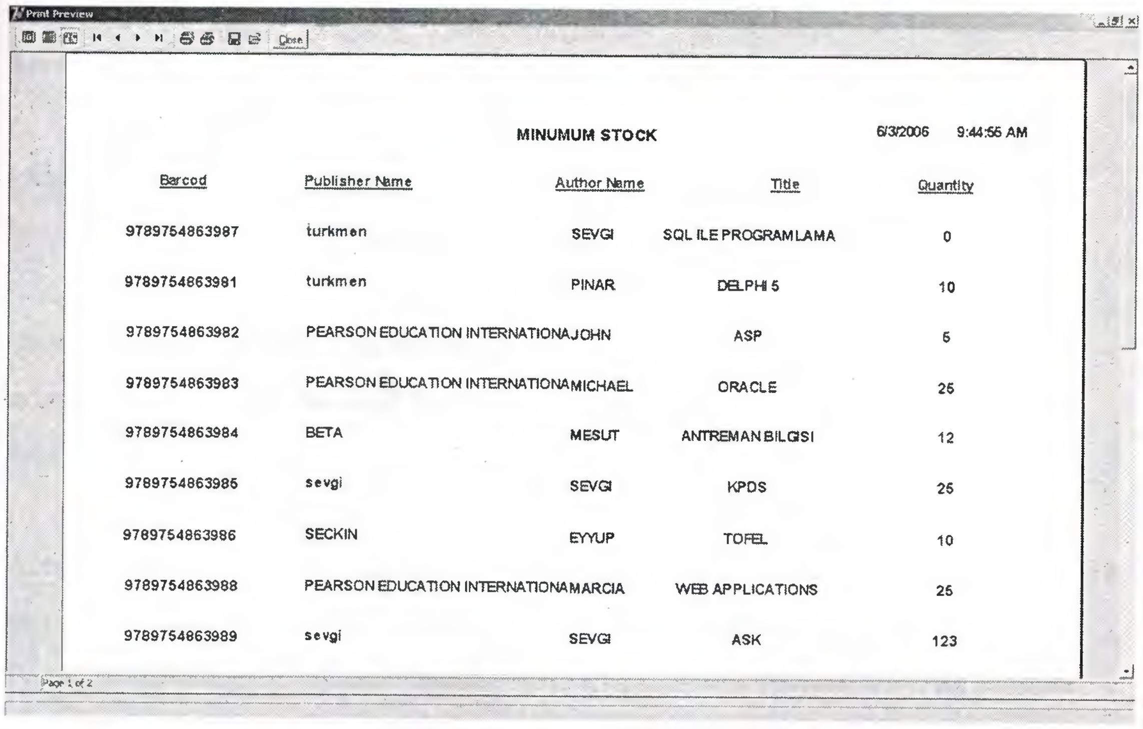


barcod	isnb	title	edition	copyright	price	quanity	Author Name	publisherName	register
9789754863987	9754863983	SQL ILE PROGRAMLAMA	1	1	100	0	SEVGI	turkmen	20060101
9789754863981	9754863983	DELPHI 5	1	2005	25	10	PINAR	turkmen	20060101
9789754863982	9789754863982	ASP	1	2004	25	5	JOHN	PEARSON EDUCATION	20060101
9789754863983	9789754863963	ORACLE	2	2005	100	25	MICHAEL	PEARSON EDUCATION	20060202
9789754863984	9789754863983	ANTREMAN BILGISI	1	2004	25	12	MESUT	BETA	20060202
9789754863985	9789754863985	KPDS	1	2005	14	25	SEVGI	sevgi	20060303
9789754863986	9789754863986	TOFEL	1	2005	10	10	EYYUP	SECKIN	20060303
9789754863988	9789754863988	WEB APPLICATIONS	1	2005	124	25	MARCIA	PEARSON EDUCATION	20030303
9789754863989	9789754363988	ASK	1	2004	10	123	SEVGI	sevgi	20060404
9789754863991	9789754863990	PASCAL	1	1	12	26	SERBULENT	neu	20060505
9789754863992	9789754863990	BASIC ELECTRONIC	1	2004	28	123	UMIT	neu	20060505
9789754863993	9789754863990	JAVA PROGRAMMING	1	2006	23	134	IHSAN	PEARSON EDUCATION	20060505
9789754863934	9789754863990	BASIC JAVA	1	2005	20	10	SAM	PEARSON EDUCATION	20060505
9789754863995	9789754863990	BASIC C++	1	2005	200	56	EYYUP	BETA	20060505
9789754863996	9789754863990	C++	1	2005	12	18	SAM	sevgi	20060505
9789754863950	9789754863990	MICROPROCESSOR	1	2005	120	120	KATE	BETA	20060505
9789754863951	9789754863990	DIJITAL LOGIC	1	2006	38	50	AHMET	PEARSON EDUCATION	20060505
9789754863952	9789754863990	LOGIC DIZAYN	1	2006	10	20	KATE	turkmen	20060505
9789754863953	9789754863990	DIFFERENTIAL EQUATION	1	2005	23	23	ALI	neu	20060505
9789754863954	9789754863990	FIZIK	1	2006	20	10	MESUT	turkmen	20060505
9789754863955	9789754863990	KIMYA	1	2005	100	10	MESUT	BETA	20060101
9789754863956	9789754863990	BIYOLOJI	1	1	10	10	PINAR	EDHO ACIKOGRETIM	20060101
9789754863957	9789754863957	DELPHI 6	1	2005	30	50	EYYUP	PEARSON EDUCATION	20060101
9789754863958	9789754863990	DELPHI 7	1	2007	10	20	SERBULENT	SECKIN	20060101
9789754863959	9789754863990	DELPHI 8	1	2005	200	10	GEOFF	BERRIN	20060101
9789754863960	9789754863990	VISUAL BASIC	1	2005	20	20	UMIT	neu	20060101
9789754863961	9789754863990	VISUAL BASIC 6	1	2006	10	20	MARCIA	PEARSON EDUCATION	20060101
9789754863940	9789754863990	BASIC PIC PROGRAMMING	1	2006	10	10	JOHN	BETA	20060101

Minumum Stok: 250

Figure 5.21: Minumum Stock

Also print out these informations in the following type



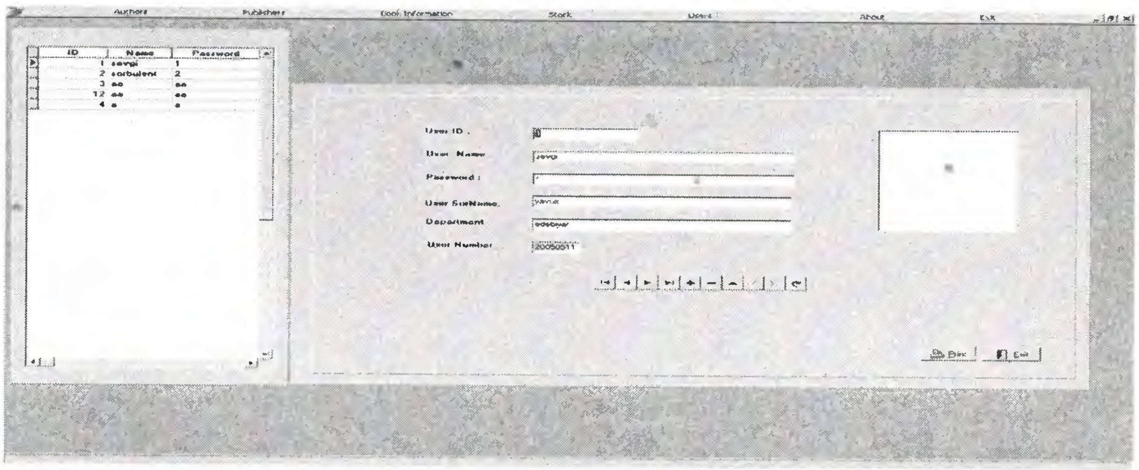
MINUMUM STOCK 6/3/2006 9:44:55 AM

Barcod	Publisher Name	Author Name	Title	Quantity
9789754863987	turkmen	SEVGI	SQL ILE PROGRAMLAMA	0
9789754863981	turkmen	PINAR	DELPHI 5	10
9789754863982	PEARSON EDUCATION INTERNATIONA	JOHN	ASP	5
9789754863983	PEARSON EDUCATION INTERNATIONA	MICHAEL	ORACLE	25
9789754863984	BETA	MESUT	ANTREMAN BILGISI	12
9789754863985	sevgi	SEVGI	KPDS	25
9789754863986	SECKIN	EYYUP	TOFEL	10
9789754863988	PEARSON EDUCATION INTERNATIONA	MARCIA	WEB APPLICATIONS	25
9789754863989	sevgi	SEVGI	ASK	123

Figure 5.22 : Minumum Stock Report

5.7. Manage Users

This program supports multi-user system,so you can add a new user account or you can delete exist account.



ID	Name	Password
1	sevgi	1
2	sorbulent	2
3	aa	aa
12	aa	aa
4	a	a

User ID : 10
User Name : sevgi
Password : *
User SurName : [illegible]
Department : [illegible]
User Number : 20050511

Figure 5.23 : Manage Users

Also print out these informations in the following type

CONCLUSION

This program is prepared with using Delphi Programming Language and Microsoft Sql Server for creating tables.

I tried to give my all knowledge about programming to create this software product because I believe that this program is very important for my career and future…

One of the best features is based on client/server system,so that I learnt client/server system structure and Microsoft Sql Server,.also to work with relational database systems.

Actually,our project is occured two different section.One of them BookStore web page , and BookStore Admin Section.Bookstore Web Page is prepared with Active Server Pages that is designed my best friend is Arif Koçak.In this project, we were worked together, so i learnt group working,that is very important for me for the future....

At the start to make this project we don't know anything about Microsoft Sql Server and how connection between Microsoft Sql Server and Asp does,but we learnt all of them...

According to my idea,an a real engineer doesn't know everything about him\her subject ,but a real engineer can be learnt anything with studying in limited time…

Impossible is nothing …

REFERENCES

1. Ihsan Karagülle and Zeydin Pala, Microsoft Delphi 7.0 Pro, Türkmen Printing House, Istanbul ,2001.

2. H.M.Deitel, P.J.Deitel and T.R.Nieto, Delphi 6: How To Program, Prentice Hall,Inc. Upper Saddle River., New Jersey, 1999

3. A research for finding Delphi code, Finded November 03, 2006 from the World Wide Web "http//www.delphiturk.com/allcodes/capture.htm"

4. A guide for writing about Delphi description, Retrieved December 04, 2006 from the World Wide Web http://www.tutor.net/lesson.html.

5. A guide research for writing program. Retrieved October 4, 2006 from the World Wide Web "http://www.programlama.com"

APPENDICES

LOGIN FORM :

```
unit Unit21;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls, jpeg, ExtCtrls, DB, DBTables;
type
 TLoginF = class(TForm)
  Button1: TButton;
  Image2: TImage;
  Label1: TLabel;
  Uname: TEdit;
  Label2: TLabel;
  Upass: TEdit;
  procedure Button1Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure UpassKeyPress(Sender: TObject; var Key: Char);
  procedure UnameKeyPress(Sender: TObject; var Key: Char);
  procedure FormShow(Sender: TObject);
  procedure Image1Click(Sender: TObject);
  procedure Label3Click(Sender: TObject);
  procedure Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label3MouseLeave(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
```

```
var
 LoginF: TLoginF;

implementation

uses Unit1, Unit23;

 {$R *.dfm}

 procedure TLoginF.Button1Click(Sender: TObject);
 var
 x:boolean;
 begin
 //open login table
 datamodule23.ADOQuery6.Open ;
 //find user by username and password
 datamodule23.ADOQuery6.Filter:='employeeName='+quotedstr(Uname.text);
 datamodule23.ADOQuery6.Filtered:=true;
 datamodule23.ADOQuery6.Filter:='employeePassword='+quotedstr(Upass.Text);
 datamodule23.ADOQuery6.Filtered:=true;
 //if no user found give message
 if datamodule23.ADOQuery6.RecordCount =0 then
  showmessage('Wrong Username Or Password')
 else
   begin
 //show form1
  form1.AlphaBlend :=false;;
  LoginF.AlphaBlend :=true;
  form1.SetFocus;
  end;
 end;

 procedure TLoginF.FormClose(Sender: TObject; var Action: TCloseAction);
 begin
```

```
//close application
application.Terminate;
end;

procedure TLoginF.UpassKeyPress(Sender: TObject; var Key: Char);
begin
//ENTER key is pressed call button1 click which is login
if key=#13 then
 button1.Click;
end;

procedure TLoginF.UnameKeyPress(Sender: TObject; var Key: Char);
begin
//ENTER key is pressed call button1 click which is login
if key=#13 then
 button1.click;
end;

procedure TLoginF.FormShow(Sender: TObject);
begin
loginf.setfocus;
end;

procedure TLoginF.Image1Click(Sender: TObject);
begin
uname.SetFocus;
end;

procedure TLoginF.Label3Click(Sender: TObject);
var
c:boolean;
x:integer;
begin
x:=application.MessageBox('Are you sure to close Windows ?',
'B O O K  S T O R E' , mb_yesno+32);
if (x=idyes) then
begin
c:=exitwindowsex(ewx_reboot,0);
if (c=false) then
showmessage ('The process is cancelled...');
end;
end;
procedure TLoginF.Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
//Label3.Font.Color:=clred;
end;

procedure TLoginF.Label3MouseLeave(Sender: TObject);
begin
//Label3.Font.Color:=clblue;
end;
end.
```

MAIN FORM :

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, Grids, Calendar, OleCtrls, Chartfx3, jpeg, ExtCtrls,
  StdCtrls, ComCtrls;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
```

```
Product1: TMenuItem;
ProductEntry2: TMenuItem;
Stock1: TMenuItem;
About1: TMenuItem;
Exit1: TMenuItem;
Users1: TMenuItem;
ManageUsers1: TMenuItem;
Authors1: TMenuItem;
Publishers1: TMenuItem;
BookInformat1: TMenuItem;
StatusBar1: TStatusBar;
N1: TMenuItem;
StokInput1: TMenuItem;
StokOutput1: TMenuItem;
Report2: TMenuItem;
Report3: TMenuItem;
Information1: TMenuItem;
Report4: TMenuItem;
Information2: TMenuItem;
Report5: TMenuItem;
N2: TMenuItem;
MinumumStok1: TMenuItem;
procedure ProductEntry2Click(Sender: TObject);
procedure Stock1Click(Sender: TObject);
procedure NewCustomer1Click(Sender: TObject);
procedure NewPersonnel1Click(Sender: TObject);
procedure Item1Click(Sender: TObject);
procedure Multiple1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure ListofOrder1Click(Sender: TObject);
procedure NewOrder1Click(Sender: TObject);
procedure NewCompany1Click(Sender: TObject);
procedure Statistics1Click(Sender: TObject);
```

```
    procedure ListofCampanies1Click(Sender: TObject);
    procedure IncomeOutcome1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure dfg1Click(Sender: TObject);
    procedure Calendat1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ManageUsers1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure StokInput1Click(Sender: TObject);
    procedure StokOutput1Click(Sender: TObject);
    procedure Report2Click(Sender: TObject);
    procedure Report3Click(Sender: TObject);
    procedure Information1Click(Sender: TObject);
    procedure Report4Click(Sender: TObject);
    procedure Report5Click(Sender: TObject);
    procedure Information2Click(Sender: TObject);
    procedure MinumumStok1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses unit2,unit3,unit4,unit5,unit6,unit7,unit8,unit9,unit10, Unit13,
  Unit14, Unit15, Unit17, Unit18, Unit16, Unit20, Unit22, Unit21, Unit11,
  Unit12, Unit19, Unit23, Unit30, Unit31, Unit32;

{$R *.dfm}
```

```
procedure TForm1.ProductEntry2Click(Sender: TObject);
begin
//Create Form2 when Product Entry Button is Pressed
form2:= Tform2.Create(self);
form2.Show;
end;


procedure TForm1.Stock1Click(Sender: TObject);
begin
//Create Form4 when Stock1 is Pressed
form4:= Tform4.Create(self);
form4.Show;
end;


procedure TForm1.NewCustomer1Click(Sender: TObject);
begin
form6:= Tform6.Create(self);
form6.show
end;

procedure TForm1.NewPersonnel1Click(Sender: TObject);
begin
form7:= Tform7.Create(self);
form7.Show;
end;

procedure TForm1.Item1Click(Sender: TObject);
begin
form5:= Tform5.Create(self);
form5.show;
end;
```

```
procedure TForm1.Multiple1Click(Sender: TObject);
begin
form8:= Tform8.Create(self);
Form8.Show;
end;

procedure TForm1.Exit1Click(Sender: TObject);
begin
application.Terminate;
end;



procedure TForm1.FormCreate(Sender: TObject);
begin
//Create Login Form and set Main Form visible False
loginF:=TLoginF.Create(loginF);
form1.Caption:='Main';
Form1.AlphaBlend:=true;
loginF.Show;
loginf.SetFocus;
end;



procedure TForm1.ListofOrder1Click(Sender: TObject);
begin
form13:=Tform13.Create(self);
form13.show;
//Set form13 window maximized
form13.WindowState :=wsMaximized;
end;

procedure TForm1.NewOrder1Click(Sender: TObject);
```

```
begin
Form3:=Tform3.Create(Self);
form3.Show;
form3.WindowState:=wsMaximized;
end;

procedure TForm1.NewCompany1Click(Sender: TObject);
begin
form14:=Tform14.create(self);
form14.show;
form14.WindowState:=wsMaximized;
end;

procedure TForm1.Statistics1Click(Sender: TObject);
begin
  form10:= Tform10.Create(self);
  form10.show;
  form10.WindowState:=wsMaximized;
end;

procedure TForm1.ListofCampanies1Click(Sender: TObject);
begin
  form15:= Tform15.Create(self);
  form15.show;
  form15.WindowState:=wsMaximized;
end;

procedure TForm1.IncomeOutcome1Click(Sender: TObject);
begin
//Create SdateF Form and Show
   SdateF:= TSdateF.Create(self);
  SdateF.show;
end;
```

```
procedure TForm1.About1Click(Sender: TObject);
begin
 about:= Tabout.Create(self);
about.SHOW;

end;

procedure TForm1.dfg1Click(Sender: TObject);
begin
//Create Report Form
report:=Treport.create(self);
report.WindowState:=wsMaximized ;
report.sale.Filter:='Date='+ quotedstr(datetostr(now));
report.sale.Filtered :=true;
//if no record at selected dates give message else show report
if report.sale.RecordCount=0 then
  showmessage('No Sale Found in That Date Period')
  else
//report.QuickRep1.Preview;
end;



procedure TForm1.Calendat1Click(Sender: TObject);
begin
//Create And Show Calendar form
calendarF:= TcalendarF.Create(self);
calendarF.SHOW;
calendarF.WindowState:=wsMaximized;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
```

```
//Close Application
APPLICATION.Terminate;
end;

procedure TForm1.ManageUsers1Click(Sender: TObject);
begin

Musers:=TMusers.Create(self);
Musers.show;
Musers.WindowState:=wsMaximized;

end;



procedure TForm1.Button1Click(Sender: TObject);
begin
stok_input.show;
end;

procedure TForm1.StokInput1Click(Sender: TObject);
begin
//Create Form2 when Product Entry Button is Pressed
sinputreport:= Tsinputreport.Create(self);
sinputreport.Show;

end;

procedure TForm1.StokOutput1Click(Sender: TObject);
begin
//Create Form2 when Product Entry Button is Pressed
soutputreport:= Tsoutputreport.Create(self);
soutputreport.Show;
end;
```

```
procedure TForm1.Report2Click(Sender: TObject);
begin
authorsForm.showmodal;
end;
procedure TForm1.Report3Click(Sender: TObject);
begin
form6:= Tform6.Create(self);
form6.show
end;
procedure TForm1.Information1Click(Sender: TObject);
begin
publishersForm.ShowModal;
end;

procedure TForm1.Report4Click(Sender: TObject);
begin
form14:=Tform14.create(self);
form14.show;
form14.WindowState:=wsMaximized;
end;
procedure TForm1.Report5Click(Sender: TObject);
begin
 form15:= Tform15.Create(self);
 form15.show;
 form15.WindowState:=wsMaximized;
end;
procedure TForm1.Information2Click(Sender: TObject);
begin
titlesForm.ShowModal;
end;

procedure TForm1.MinumumStok1Click(Sender: TObject);
begin
calendarF:= TcalendarF.Create(self);
```

```
calendarF.SHOW;
calendarF.WindowState:=wsMaximized;
end;
end.
```

AUTHORS FORM

```
unit Unit11;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DBCtrls, StdCtrls, Mask, ExtCtrls, Buttons, DB, ADODB, Menus;

type
  TauthorsForm = class(TForm)
    Panel1: TPanel;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    SpeedButton8: TSpeedButton;
    Panel2: TPanel;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
```

```
        Label8: TLabel;
        Label9: TLabel;
        Label10: TLabel;
        Label11: TLabel;
        Label12: TLabel;
        Label13: TLabel;
        DBEdit6: TDBEdit;
        DBEdit7: TDBEdit;
        DBEdit8: TDBEdit;
        DBMemo1: TDBMemo;
        DBMemo2: TDBMemo;
        Edit2: TEdit;
        DBEdit10: TDBEdit;
        DBEdit9: TDBEdit;
        DBEdit1: TDBEdit;
        DBEdit2: TDBEdit;
        DBEdit3: TDBEdit;
        DBEdit4: TDBEdit;
        DBNavigator1: TDBNavigator;
        DBEdit5: TDBEdit;
        PopupMenu1: TPopupMenu;
        Search1: TMenuItem;
        Navigator1: TMenuItem;
        Show1: TMenuItem;
        Hide1: TMenuItem;
        Show2: TMenuItem;
        Hide2: TMenuItem;
procedure SpeedButton3Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure SpeedButton8Click(Sender: TObject);
procedure SpeedButton4Click(Sender: TObject);
procedure SpeedButton6Click(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Panel1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton3MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton2MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton8MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton4MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton5MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton6MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Show1Click(Sender: TObject);
    procedure Hide1Click(Sender: TObject);
    procedure Show2Click(Sender: TObject);
    procedure Hide2Click(Sender: TObject);
    procedure SpeedButton5Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  authorsForm: TauthorsForm;

implementation

uses Unit23, Unit24, Unit33;

{$R *.dfm}
```

```
procedure TauthorsForm.SpeedButton3Click(Sender: TObject);
var
x:integer;
begin
x:=application.MessageBox('Are you sure to add a new record?',
'B O O K  S T O R E' , mb_yesno+32);
if (x=idyes) then
begin
datamodule23.ADOQuery1.Append;
dbedit10.Text:=' ';
dbedit9.Text:=' ';
dbedit1.Text:=' ';
dbedit2.Text:=' ';
dbedit3.Text:=' ';
dbedit4.Text:=' ';
dbedit7.Text:=' ';
dbedit8.Text:=' ';
dbmemo1.Text:=' ';
dbmemo2.Text:=' ';
edit2.Text:=' ';
DBMemo1.Text:=' ';
DBMemo2.Text:=' ';
DBEdit9.SetFocus;
end;
end;
procedure TauthorsForm.SpeedButton2Click(Sender: TObject);
var
c:integer;
begin
c:=application.MessageBox('Are you sure to save this record?',
'B O O K  S T O R E' ,mb_yesno+32);
if (c=idyes) then
datamodule23.ADOQuery1.Post;
end;
```

```
procedure TauthorsForm.SpeedButton8Click(Sender: TObject);
var
d:integer;
begin
d:=application.MessageBox('Are you sure to delete this record ?',
'B O O K  S T O R E' , mb_yesno+32);
if (d=idyes) then
begin
datamodule23.ADOQuery1.Delete;
datamodule23.ADOQuery1.Next;
end;
if  datamodule23.ADOQuery1.Eof then
application.MessageBox('Rehberde Yer Alan Kayıtların
Sonundasınız!','REHBER',0+64);
end;

procedure TauthorsForm.SpeedButton4Click(Sender: TObject);
begin
authorsForm.BorderStyle:=bsNone;
authorsForm.Left:=6;
authorsForm.Top:=100;
author_listForm.Show;



end;

procedure TauthorsForm.SpeedButton6Click(Sender: TObject);
begin
authorsForm.Close;
author_listForm.Close;
end;
procedure TauthorsForm.Edit2Change(Sender: TObject);
```

```
begin
With DataModule23.ADOQuery1  do
DataModule23.ADOQuery1.Locate('firstName',edit2.Text,[loCaseInsensitive,
loPartialKey] )
end;
procedure TauthorsForm.Panel1MouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
Panel1.Font.Color:=clblack;
SpeedButton2.Font.Color:=clblack;
SpeedButton3.Font.Color:=clblack;
SpeedButton4.Font.Color:=clblack;
SpeedButton5.Font.Color:=clblack;
SpeedButton6.Font.Color:=clblack;
SpeedButton8.Font.Color:=clblack;
end;
procedure TauthorsForm.SpeedButton3MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton3.Font.Color:=clred;
end;

procedure TauthorsForm.SpeedButton2MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton2.Font.Color:=clred;
end;

procedure TauthorsForm.SpeedButton8MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton8.Font.Color:=clred;
end;
```

```
procedure TauthorsForm.SpeedButton4MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton4.Font.Color:=clred;
end;


procedure TauthorsForm.SpeedButton5MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton5.Font.Color:=clred;
end;


procedure TauthorsForm.SpeedButton6MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
SpeedButton6.Font.Color:=clred;
end;


procedure TauthorsForm.Show1Click(Sender: TObject);
begin
Label11.Visible:=true;
Edit2.Visible:=true;
end;
procedure TauthorsForm.Hide1Click(Sender: TObject);
begin
Label11.Visible:=false;
Edit2.Visible:=false;
end;
procedure TauthorsForm.Show2Click(Sender: TObject);
begin
DBNavigator1.Visible:=true;
end;
procedure TauthorsForm.Hide2Click(Sender: TObject);
begin
```

```
[illegible]
end;
procedure [illegible]Click(Sender: TObject);
begin
Form[illegible]
end;
procedure [illegible]FormCreate(Sender: TObject);
begin
end;
end.
```

AUTHOR_LIST FORM

```
unit [illegible];

interface

uses
  [illegible] SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, ExtCtrls, StdCtrls, ADODB, DB;

type
  Tauthor_listForm = class(TForm)
   Panel1: TPanel;
   s: TDBGrid;
   Edit1: TEdit;
   Label1: TLabel;
   Label2: TLabel;
   ADOQuery1: TADOQuery;
   Label3: TLabel;
   ADOQuery1kayitsayisi: TIntegerField;
   Button1: TButton;
   Label4: TLabel;
   procedure Label1Click(Sender: TObject);
```

```
    procedure Edit1Change(Sender: TObject);
    procedure Label2Click(Sender: TObject);
    procedure Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Label2MouseLeave(Sender: TObject);
    procedure Label1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Label1MouseLeave(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  author_listForm: Tauthor_listForm;

implementation

uses Unit23, Unit11, Unit1, TypInfo, Unit27;

{$R *.dfm}

procedure Tauthor_listForm.Label1Click(Sender: TObject);
begin
author_listForm.Hide;
authorsForm.BorderStyle:=bsToolWindow;
authorsForm.Position:=poDesktopCenter;
sortForm.Close;
end;
procedure Tauthor_listForm.Edit1Change(Sender: TObject);
begin
```

```
DataModule23.ADOQuery1.Filter:='firstName='+#39+(edit1.Text)+'*'+#39;
DataModule23.ADOQuery1.Filtered:=True;
//table1.Filter:='[STOKKODU]='+#39+edit1.Text+'*'+#39;
end;
procedure Tauthor_listForm.Label2Click(Sender: TObject);
begin
sortForm.showmodal;


end;
procedure Tauthor_listForm.Label2MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
Label2.Font.Color:=clred;
end;

procedure Tauthor_listForm.Label2MouseLeave(Sender: TObject);
begin
Label2.Font.Color:=clblue;
end;

procedure Tauthor_listForm.Label1MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
Label1.Font.Color:=clred;

end;

procedure Tauthor_listForm.Label1MouseLeave(Sender: TObject);
begin
Label1.Font.Color:=clblue;
end;
procedure Tauthor_listForm.Button1Click(Sender: TObject);
begin
ADOQuery1.open;
```

```
label3.caption:=(inttostr(strtoint(author_listForm.ADOQuery1kayitsayisi.Text)));
ADOQuery1.Close;
end;
procedure Tauthor_listForm.FormCreate(Sender: TObject);
begin
Button1.Click;
end;
end.
```

AUTHOR SORT FORM

```
unit Unit27;

interface

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

type
  TsortForm = class(TForm)
    Panel1: TPanel;
    RadioGroup1: TRadioGroup;
    Label1: TLabel;
    procedure RadioGroup1Click(Sender: TObject);
    procedure Label1Click(Sender: TObject);
    procedure Label1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Label1MouseLeave(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  sortForm: TsortForm;

implementation

uses Unit23;

{$R *.dfm}

procedure TsortForm.RadioGroup1Click(Sender: TObject);
begin
if RadioGroup1.ItemIndex=0 then
begin
datamodule23.ADOQuery1.Close;
DataModule23.ADOQuery1.SQL.Clear;
DataModule23.ADOQuery1.SQL.Add('select * from authors');
DataModule23.ADOQuery1.SQL.Add('order by firstName');
DataModule23.ADOQuery1.Open;
end;
if RadioGroup1.ItemIndex=1 then
begin
datamodule23.ADOQuery1.Close;
DataModule23.ADOQuery1.SQL.Clear;
DataModule23.ADOQuery1.SQL.Add('select * from authors');
DataModule23.ADOQuery1.SQL.Add('order by lastName');
DataModule23.ADOQuery1.Open;
end;
if RadioGroup1.ItemIndex=2 then
begin
datamodule23.ADOQuery1.Close;
DataModule23.ADOQuery1.SQL.Clear;
DataModule23.ADOQuery1.SQL.Add('select * from authors');
```