NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

FLIGHT INFORMATION AND TICKET SERVICES (FITs)

Graduation Project
COM400

Student: Ülkü Esra OKUYAN (20031609)

Supervisor: Assist. Prof. Dr. Elbrus Bashir IMANOV

Nicosia - 2008

ACKNOWLEDGMENT

First of all, I would like to give my special thanks to my supervisor Assist. Prof. Dr. Elbrus Bashir IMANOV. Under the guidance of him I successfully overcome many difficulties. He always believed in me that Iwill fulfill and succeed on my project. Also I would like to express my thanks to other instructors in Computer Engineering Department for their help , clue and guideness.

Special thanks to my family ,I could never have completed my study without their endless support. And also I would like to thanks my fiancé he always with me according to my decesion and I could never forget to his support.

Finally I want to thanks all my friends and specially Yusuf ALAK, Atakan AKAR, Murat EVEREKLI, Alper KURKCU, Bilal KONUK, Cevdet DIKICILER and Berna KUCUK. They are always help me when I give up hope.I am very lucky to have such friends.

ABSTRACT

The aim of this Projects is to sell and reservation the ticket of one flight company so it has not branch office. The program was prepared by using Delphi 6 programming and using Paradox7. Delphi is a programming language that can be used with Paradox7.

This project consist of many different pages but most of them dependent each other Initially, Login form comes to secreen.Afterwards the Main Form of Flight information and ticket services(FITs) company come to secreen. The Main form that contains eight main menus.

FITs is help to agecy, the program provides easy,quick and more reliable process on company works.

TABLE OF CONTENTS

INTRODUCTION

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

Delphi language, a set of object-oriented extensions to standard Pascal, is the language of Delphi. Delphi Pascal is a high-level, compiled, strongly typed language that supports structured and object-oriented design. Its benefits include easy-to-read code, quick compilation, and the use of multiple unit files for modular programming.

The term "visual" describes Delphi very well. All of the user interface development is conducted in a What You See Is What You Get environment (WYSIWYG), which means you can create polished, user friendly interfaces in a very short time, or prototype whole applications in a few hours.

In short, Delphi includes just about everything you need to write applications that will run on an Intel platform under Windows, but if your target platform is a Silicon Graphics running IRIX, or a Sun Sparc running SOLARIS, or even a PC running LINUX, then you will need to look elsewhere for your development tools.

This specialisation on one platform and one operating system, makes Delphi a very strong tool. The code it generates runs very rapidly, and is very stable, once your own bugs have been ironed out!

This project about information of the flight and control the your work very well so we call the name is Flight Information and Ticket Services (FITs). The project consists of introduction, three chapter and conclusion.

- Chapter one describes the basic concepts of Delphi6.
- Chapter two describes the database that uses Delphi programming language.
- Chapter three shows how to program is work and explains the several important form of the FITs.

CHAPTER ONE

1. BASIC CONCEPT OF DELPHI 6

1.1. Introduction to Delphi

Delphi is a Rapid Application Development (RAD) tool for Windows, originally developed by Borland Software Corporation. Version 1.0 was first released in 1995. Delphi was originally a confidential research project at Borland which evolved into a product that was to be called AppBuilder. Delphi is a Rapid Application Development (RAD) tool for Windows, originally developed by Borland Software Corporation.

Developer Danny Thorpe chose the Delphi codename in reference to the Oracle at Delphi. One of the original goals of Delphi was to provide database connectivity to programmers as a key feature and a popular database package at the time was Oracle database.

Shortly before the first release of Borland's AppBuilder, Novell AppBuilder was released, leaving Borland in need of a new name. After much struggle, the name Delphi prevailed.

An important strength for Delphi from its birth to current versions is powerful components for database development. In fact a user can create simple database applications without writing a single line of code. As easy as it is to dive into Delphi and make something useful there is nearly unlimited power for the most advanced development projects.

1.2. What is Delphi?

Delphi is a software development package created by Borland. It is a Rapid Application Development (RAD) environment. It allows you to drag and drop components on to a blank canvas to create a program. Delphi will also allow you to use write console based DOS like programs. Delphi's most popular use is the development of desktop and enterprise database applications, but as a general-purpose development tool it is capable of, and is used for, most types of development projects.

Delphi is based around the Pascal language but is more developed object orientated derivative. Unlike Visual Basic, Delphi uses punctuation in its basic syntax to make the program easily readable and to help the compiler sort the code. Although

Delphi code is not case sensitive there is a generally accepted way of writing Delphi code. The main reason for this is so that any programmer can read your code and easily understand what you are doing, because they write their code like you write yours. The Delphi product is distributed as various suites: Personal, Professional, Enterprise (formerly Client/Server) and Architect.

1.3. History Of Delphi

Delphi was originally released in 1995 by Borland for the 16 bit Microsoft Windows 3.x Operating Environment. The following year Delphi 2 was released for 32 bit Windows. A new version has been released roughly once a year.

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

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

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

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

Delphi 8, released December 2003, was a .NET -only release that allowed developers to compile Delphi Object Pascal code into .NET Microsoft Intermediate Language|MSIL . It was also significant in that it changed its IDE for the first time, from the multiple-floating-window-on-desktop style IDE to a look and feel similar to Microsoft's Visual Studio.NET.

Although Borland fulfilled one of the biggest requests from developers (.NET support), it was criticized both for making it available too late, when a lot of former Delphi developers had already moved to C#, and for focusing so much on backward compatibility that it was not very easy to write new code in Delphi. Delphi 8 also lacked significant high-level features of the c sharp|C# language, as well as many of the more

appealing features of Microsoft's Visual Studio IDE. (There were also concerns about the future of Delphi Win32 development. Because Delphi 8 did not support Win32, Delphi 7.1 was included in the Delphi 8 package.)

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

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

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

On September 6 , 2006, The Developer Tools Group (the working name of the not yet spun off company) of Borland Software Corporation released single language versions of Borland Developer Studio, bringing back the popular "Turbo" moniker. The Turbo product set includes Turbo Delphi for Win32, Turbo Delphi for .NET, Turbo C++, and Turbo C#. Each version is available in two editions: "Explorer"—a free downloadable version—and "Professional"—a relatively cheap (US$399) version which opens access to thousands of third-party components. Unlike earlier "Personal" editions of Delphi, new "Explorer" editions can be used for commercial development.

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

1.4. The main features that distinguish Delphi

- The Pascal-based programming language
- The VCL/CLX (Visual Component Library)

- A strong emphasis on database connectivity
- A large number of third party components.
- Delegation of interface implementation to a field or property of the class
- Implementation of message handlers by tagging a method of a class with the integer constant of the message to handle
- COM independent interfaces with reference counted class implementations
- Can be compiled into native x86 code or managed .NET code

1.5. The key features of the product

- It's very easy to create forms based applications for windows, where you drag and drop controls onto forms, write a little bit of code, and you have a program. The VCL is a powerful and feature-rich component library.

- It's easy to get started, especially if you have a good book to help get you going.

- There are hundreds of thousands of third-party components (some freeware, some commercial, some open source) for Delphi. Whatever you need to do, there's a component out there to help you get your application written.

- Unlike other RAD tools, like Visual Basic 6, no runtime is needed (you can have a standalone EXE)

- Unlike Visual Basic and other RAD tools, Borland values compatibility highly, and even Delphi 1.0 applications can usually be made to work in the latest Delphi with only minor changes. If you want to know how the other half feels, check out the reactions when Microsoft completely abandoned Visual Basic 6 developers, when they created "Visual Basic .net". Essentially they killed millions of developers work, and forced them to rewrite all their code, and stopped selling and updating the old product (Visual Basic 6) without offering any reasonable upgrade path.

- You can write code in Delphi that targets classic Windows environments from Windows 95 and up, or the latest ".net" environments, with the same syntax, and the same powerful libraries.

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

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

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

Some projects to which Delphi is suited:

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

This is not intended to be an exhaustive list, more an indication of the depth and breadth of Delphi's applicability. Because it is possible to access any and all of the Windows API, and because if all else fails, Delphi will allow you to drop a few lines of assembler code directly into your ordinary Pascal instructions, it is possible to do more or less anything. Delphi can also be used to write Dynamically Linked Libraries (DLLs) and can call out to DLLs written in other programming languages without difficulty.

Because Delphi is based on the concept of self contained Components (elements of code that can be dropped directly on to a form in your application, and exist in object form, performing their function until they are no longer required), it is possible to build applications very rapidly. Because Delphi has been available for quite some time, the number of pre-written components has been increasing to the point that now there is a component to do more or less anything you can imagine. The job of the programmer has

become one of gluing together appropriate components with code that operates them as required.

1.7. Components

At the core of Delphi is it's Object Pascal compiler but Delphi is a RAD tool for it's Integrated Development Environment (IDE).

The IDE is where the developer spends most of his programming time. It contains an editor for working on Delphi units as well as a visual forms designer that generates code automatically. The IDE is a two way tool which means that the developer can make changes to the visual forms or the underlying code.

Another key part of Delphi is the included object library known as the Visual Component Library(VCL). Many VCL objects are available on the Component Palette in the IDE for visual development.

Various versions of Delphi include various utilities for resource management, image development, database access and development.

1.8. Advantages and Disadvantages of Delphi

1.8.1. Advantages

Delphi exhibits the following advantages:

- Rapid Application Development (RAD)
- Based on a well-designed language - high-level and strongly typed, with low-level escapes for experts
- A large community on Usenet and the World Wide Web (e.g. news://newsgroups.borland.com and Borland's web access to Delphi)
- Can compile to a single executable, simplifying distribution and reducing DLL versioning issues
- Many VCL and third-party components (usually available with full source code) and tools (documentation, debug tools, etc.)
- Quick optimizing compiler and ability to use assembler code
- Multiple platform native code from the same source code
- High level of source compatibility between versions

- Cross Kylix - a third-party toolkit which allows you to compile native Kylix/Linux applications from inside the Windows Delphi IDE, hence easily enabling dual-platform development and deployment
- Cross FBC - a sister project to CrossKylix, which enables you to cross-compile your Windows Delphi applications to multi-platform targets - supported by the Free Pascal compiler - without ever leaving the Delphi IDE
- Class helpers to bridge functionality available natively in the Delphi RTL, but not available in a new platform supported by Delphi
- The language's object orientation features only class- and interface-based Polymorphism in object-oriented programming|polymorphism

1.8.2. Disadvantages

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

1.9. Products developed with Delphi

- There are many products developed with Delphi. The most well-known ones are:
- Borland products: Borland Delphi, Borland C++ Builder, Borland JBuilder versions 1 & 2
- Database management: MySQL Tools (Administrator, Query Browser, Migration Toolkit)
- Image viewers: FastStone Image Viewer, FuturixImager

- Internet messaging: Skype (VoIP and IM), The Bat! (e-mail client), PopTray (e-mail check tool), FeedDemon (RSS/Atom feed viewer), XanaNews (newsgroup reader), Xnews (newsgroup reader)
- Engineering Software: Altium Designer/Protel (Electronics Design)
- Music production: FL Studio (formerly FruityLoops)

- Software development: Dev-C++ (IDE), DUnit, Help & Manual (help system authoring), Inno Setup (installer engine)
- Web authoring: Macromedia HomeSite (HTML editor), TopStyle Pro (CSS editor), Macromedia Captivate (screencast)
- Web browsers (MSIE shells): Avant Browser, Netcaptor
- Utilities: Spybot - Search & Destroy, Ad-Aware (anti-spyware), Total Commander (file manager), Copernic Desktop Search, PowerArchiver

1.10. Delphi 6 Editions

There are 3 editions in Delphi 6 :

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

This Program will concentrate on the Enterprise edition..

1.11. Delphi 6 Archite

Delphi 6 Architect is designed for professional enterprise developers who need to adapt quickly to changing business rules and manage sophisticated applications that

synchronize with multiple database schemas. Delphi 2006 Architect includes an advanced ECO III framework that allows developers to rapidly deploy scalable external facing Web applications with executable state diagrams, object-relational mapping, and transparent persistence.

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

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

1.12. Installation of Delphi 6

To install Delphi 6 Enterprise, run INSTALL.EXE (default location C:\Program Files\Borland Delphi) and follow the installation instructions.We are prompted to select a product to install, you only have one choice "Delphi 6":

Figure 1.1 The Select Page For Start Installation

While the setup runs, you'll need to enter your serial number (Figure 1.2) and the authorization key (the two you got from inside a Cd rom driver).

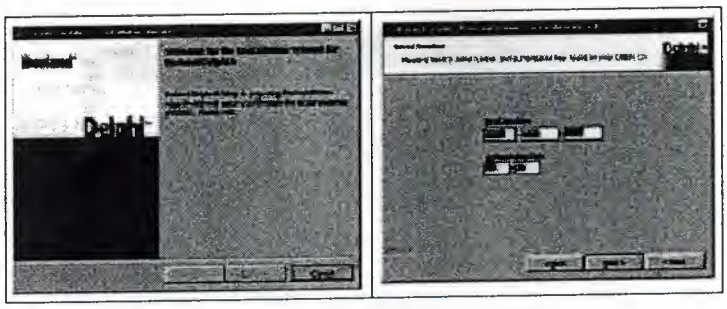

Figure 1.2 Serial Number And Authorization Screen

Later, the License Agreement screen will popup:

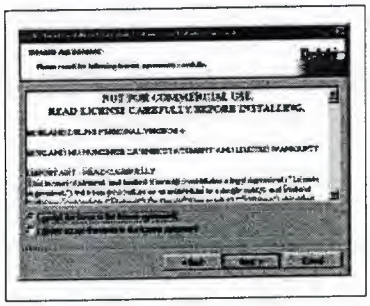

Figure 1.3 Lisanse Agreement Screen

After that, you have to pick the Setup Type, choose Typical. This way Delphi 6 Enterprise will be installed with the most common options. The next screen prompts you to choose the Destination folder (Figure 1.4).

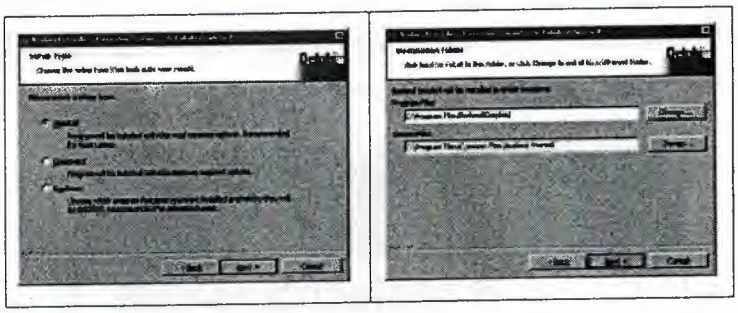

Figure 1.4. SetUp Type and Destination Folder Screen

At the end of the installation process, the set-up program will create a sub menu in the Programs section of the Start menu (Figure 1.5), leading to the main Delphi 6 Enterprise program plus some additional tools.

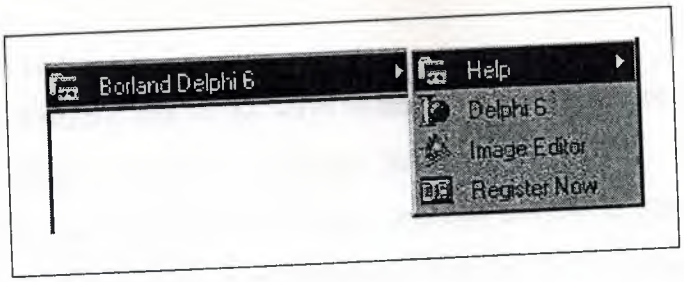

Figure 1.5. Start Menu Screen

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

1.13. 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).

1.13.1. Running Delphi For The First Time

You can start Delphi in a similar way to most other Windows applications: (Figure 1.6)

- Choose Programs | Borland Delphi 6 | Delphi 6 from the Windows Start menu
- Choose Run from the Windows Start menu and type Delphi32

  Double-click Delphi32.exe in the $(DELPHI)\Bin folder. Where $(DELPHI) if a folder where Delphi was installed. The default is C:\Program Files\Borland\Delphi6.
- Double-click the Delphi icon on the Desktop (if you've created a shortcut)

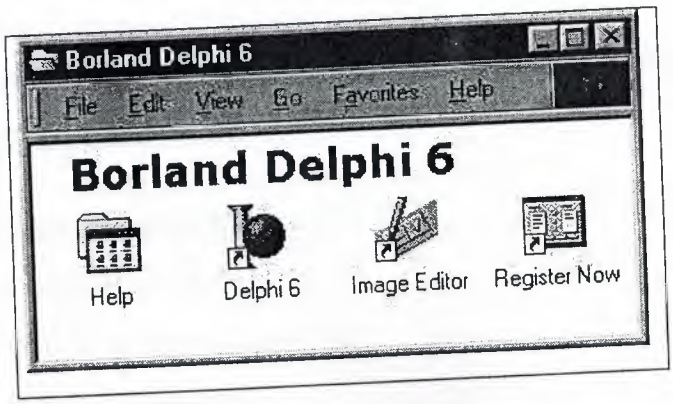

Figure 1.6. Borland Delphi 6 Folder

1.13.2. The Delphi IDE

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

When Delphi starts (it could even take one full minute to start - depending on your hardware performance) you are presented with the IDE (Figure 1.7): the user interface where you can design, compile and debug your Delphi projects.

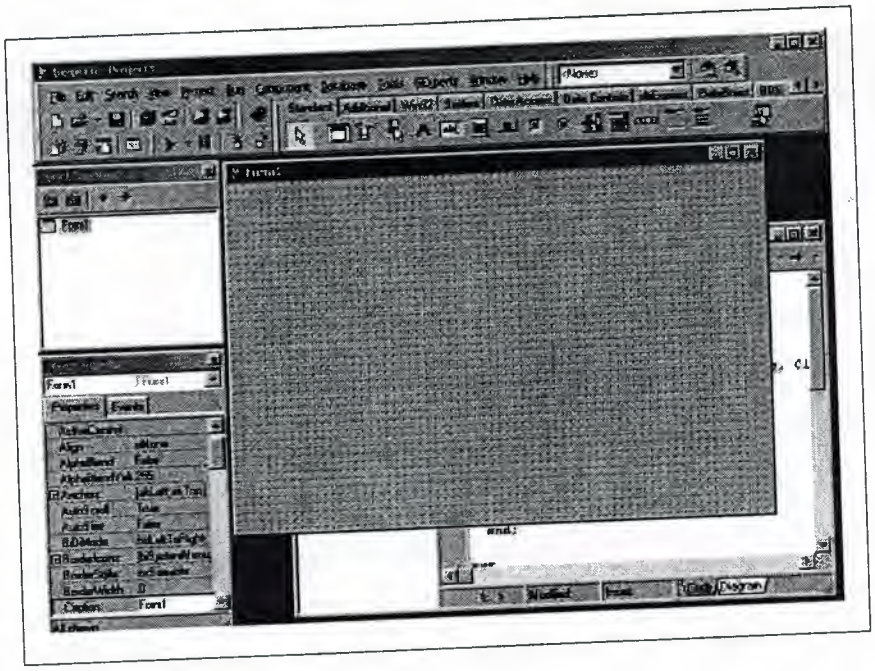

Figure 1.7. IDE

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

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

- A syntax sensitive program file editor
- A rapid optimising compiler
- Built in debugging /tracing facilities
- A visual interface developer
- Syntax sensitive help files
- Database creation and editing tools

- Image/Icon/Cursor creation / editing tools
- Version Control CASE tools

1.13.3. The Menus and Toolbar

The main window, positioned on the top of the screen, contains the main menu, toolbar and Component palette (Figure 1.8).

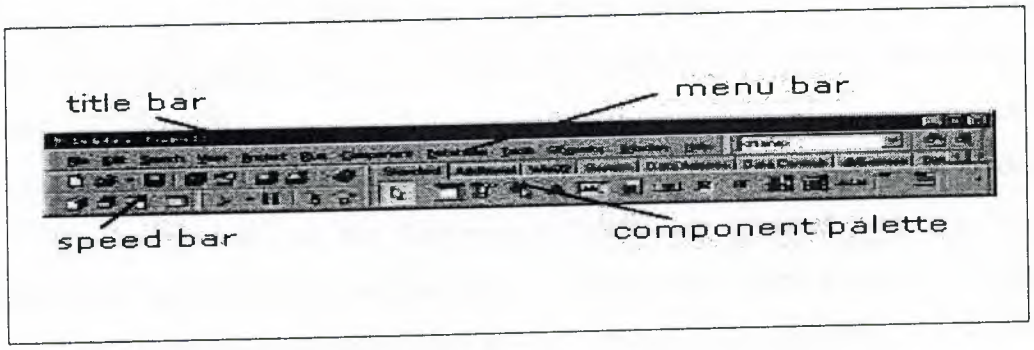

Figure 1.8. Menu ,Title , Speed Bar & Component Palette

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

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

1.13.4. The Component Palette

You are probably familiar with the fact that any window in a standard Windows application contains a number of different (visible or not to the end user) objects, like: buttons, text boxes, radio buttons, check boxes etc. In Delphi programming terminology such objects are called controls (or components).Components are the building blocks of every Delphi application. To place a component on a window you drag it from the component palette (Figure 1.9). Each component has specific attributes that enable you to control your application at design and run time.

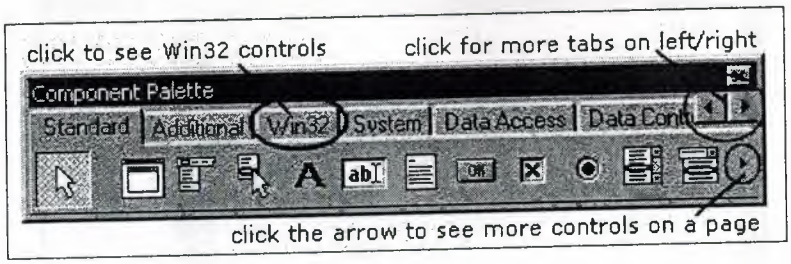

Figure 1.9. Component Palatte

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

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

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

1.13.5. The Code Editor

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

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

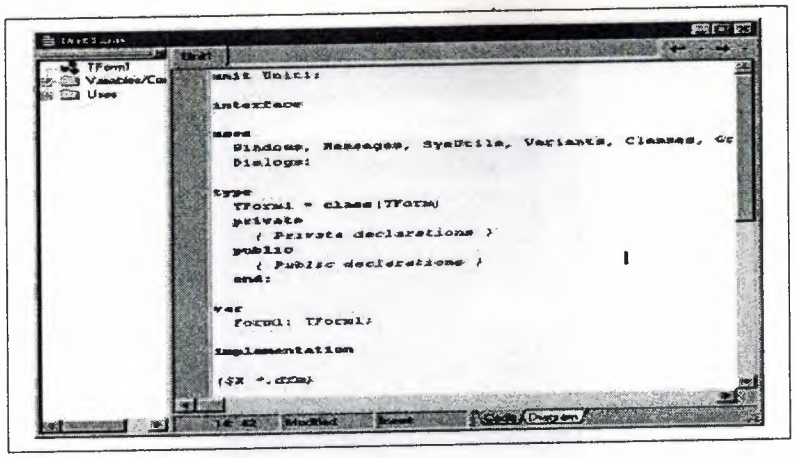

Figure 1.10. Code Editor Window

If the Form1 is the active window and you press [F12], the Code Editor window will be placed on top (Figure 1.10). As you design user interface of your application, Delphi automatically generates the underlying Object Pascal code. More lines will be added to this window as you add your own code that drives your application. This window displays code for the current form (Form1); the text is stored in a (so-called) unit - Unit1. You can open multiple files in the Code Editor. Each file opens on a new page of the Code editor, and each page is represented by a tab at the top of the window.

1.13.6. The Object Inspector

Each component and each form, has a set of properties – such as color, size, position, caption – that can be modified in the Delphi IDE or in your code, and a collection of events – such as a mouse click, keypress, or component activation – for which you can specify some additional behavior. The Object Inspector (Figure 1.11)displays the properties and events (note the two tabs) for the selected component and allows you to change the property value or select the response to some event.

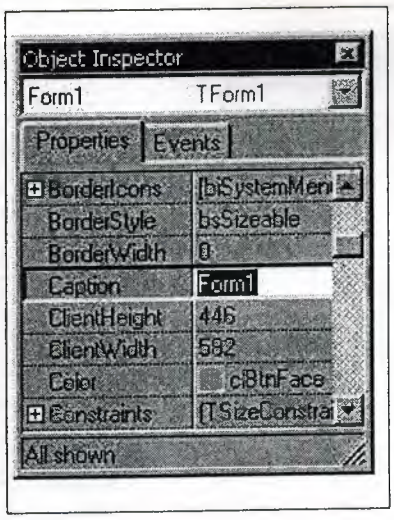

Figure 1.11. Object Inspector

For example, each form has a Caption (the text that appears on it's title bar). To change the caption of Form1 first activate the form by clicking on it. In the Object Inspector find the property Caption (in the left column), note that it has the 'Form1' value (in the right column).

To change the caption of the form simply type the new text value, like 'My Form' (without the single quotes). When you press [Enter] the caption of the form will change to My Form.

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

1.13.7. The Object Tree View

Above the Object Inspector you should see the Object TreeView window (Figure 1.12). For the moment it's display is pretty simple. As you add components to the form, you'll see that it displays a component's parent-child relationships in a tree diagram. One of the great features of the Object TreeView is the ability to drag and drop components in order to change a component container without losing connections with other components.

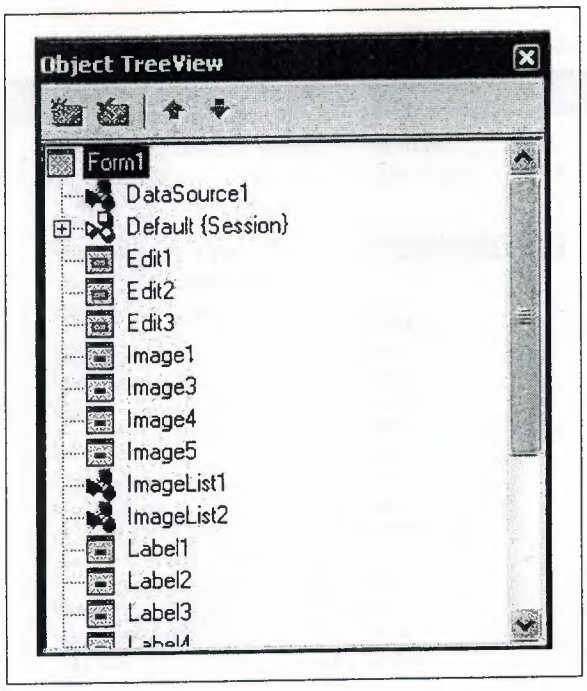

Figure 1.12. Object Tree View

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

1.13.8. Class Completion

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

To configure Class Completion, choose Tools|Environment Options and click the Explorer tab (Figure 1.13).

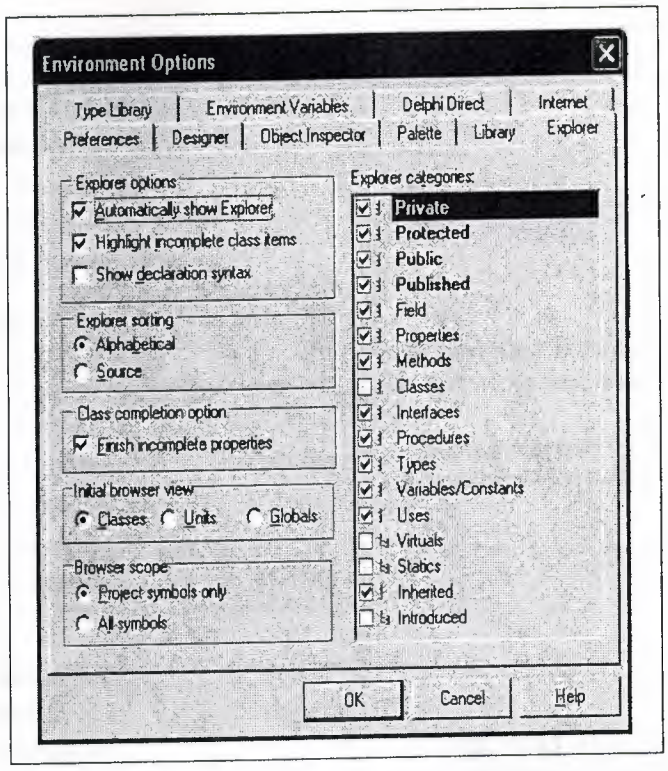

Figure 1.13. Class

1.13.9. Debugging applications

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

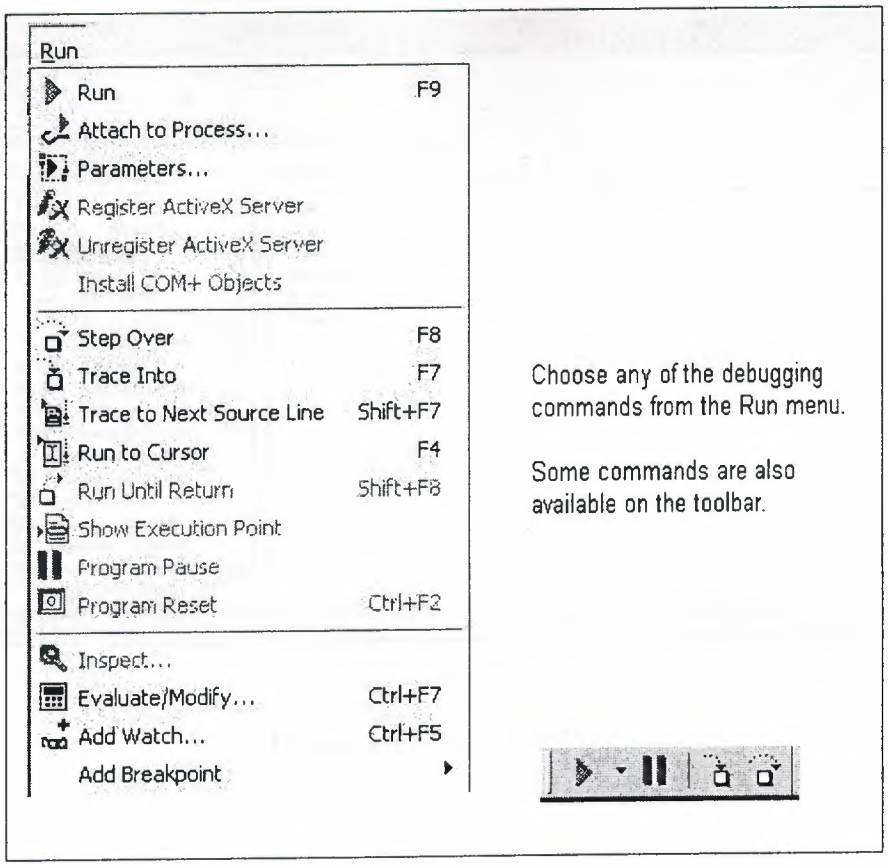

Figure1.14. Debugger

To use the debugger (Figure1.14), you must compile your program with debug information. Choose Project|Options, select the Compiler page, and check Debug Information. Then you can begin a debugging session by running the program from the IDE. To set debugger options, choose Tools|Debugger Options.

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

1.13.10. Exploring databases

The SQL Explorer (or Database Explorer in some editions of Delphi) lets you work directly with a remote database server during application development (Figure 1.15). For example, you can create, delete, or restructure tables, and you can import constraints while you are developing a database application.

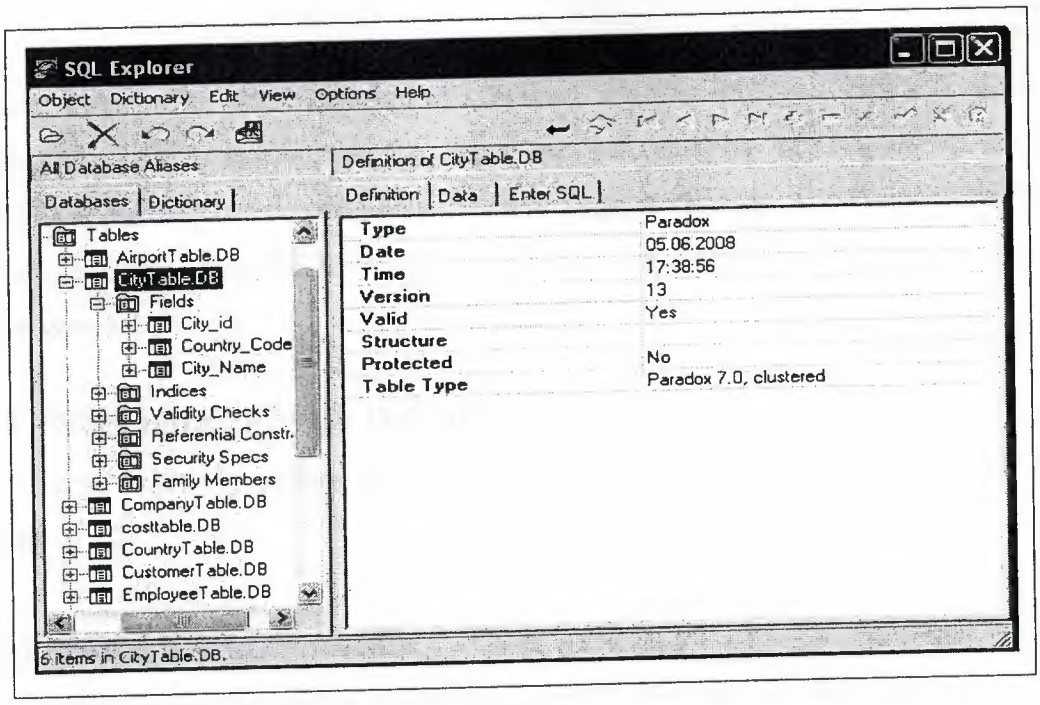

Figure 1.15. SQL Explorer

1.14. Templates and the Object Repository

The Object Repository contains forms, dialog boxes, data modules, wizards, DLLs, sample applications, and other items that can simplify development. Choose File|New to display the New Items dialog when you begin a project (Figure 1.16). Check the Repository to see if it contains an object that resembles one you want to create.

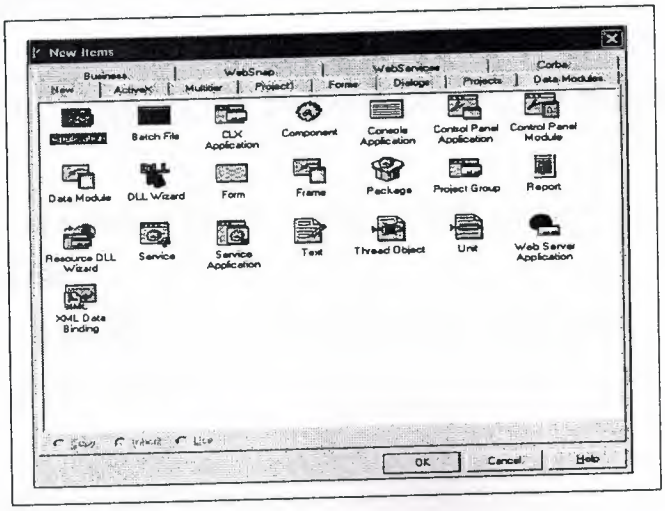

Figure 1.16. New Item

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

1.15. Programming With Delphi

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

1.15.1.Starting a New Application

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

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

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

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

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

Each form has its own unit and form files.

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

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

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

When you open a new project, Delphi displays the project's main form, named Form1 by default (Figure 1.17). You'll create the user interface and other parts of your application by placing components on this form.

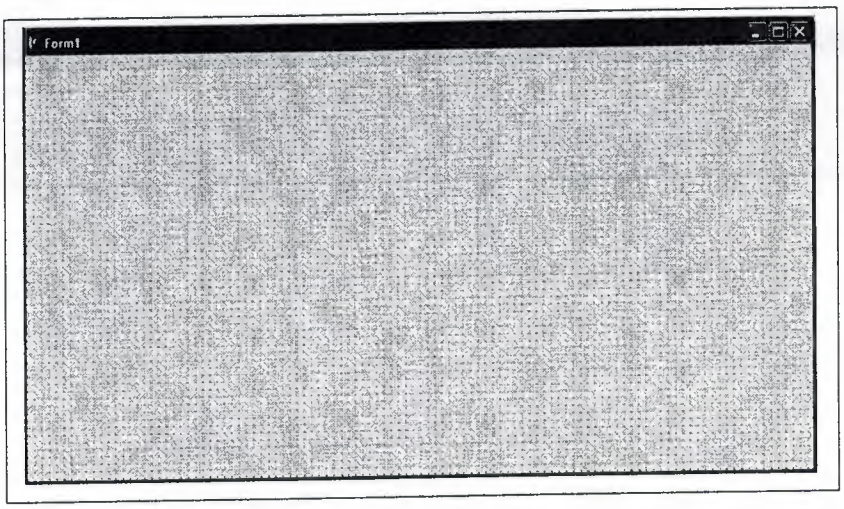

Figure 1.17. Form Screen

The default form has maximize , minimize buttons and a close button , and a control menu.

Next to the form, you'll see the Object Inspector, which you can use to set property values for the form and components you place on it.

The drop-down list at the top of the Object Inspector shows the current selected object.when an object is sellected the Object Inspector show its properties.

1.15.1.1. Setting Property Values

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

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

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

1.15.2. Adding objects to the form

The Component palette represents components by icons grouped onto tabbed pages (Figure 1.18). Add a component to a form by selecting the component on the palette, then clicking on the form where you want to place it. You can also double-click a component to place it in the middle of the form.

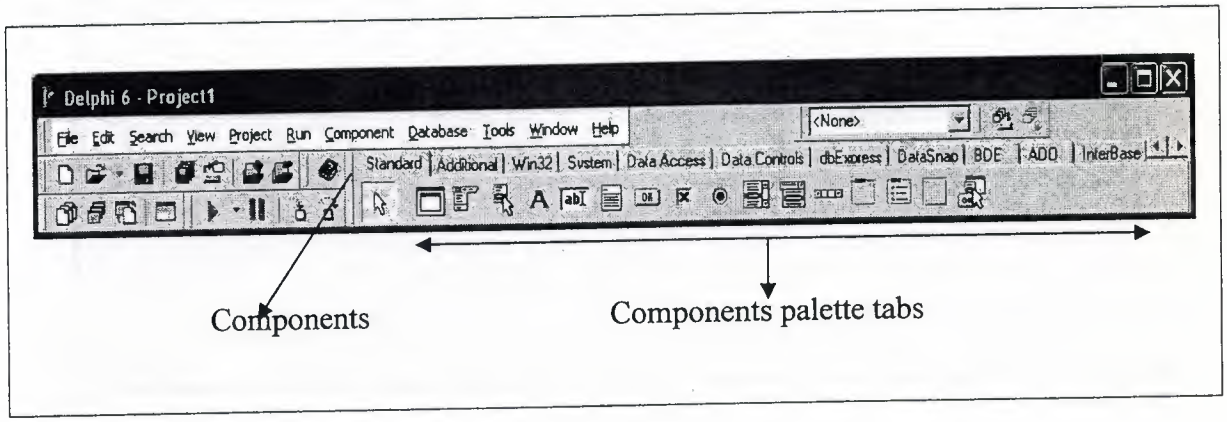

Figure 1.18. Standard Components Palette

1.15.3. Add a Query and a StatusBar to the form

Drop a Table component onto the form. Click the BDE tab on the Component palette (Figure 1.19). To find the *Query* component, point at an icon on the palette for a moment; Delphi displays a Help hint showing the name of the component.

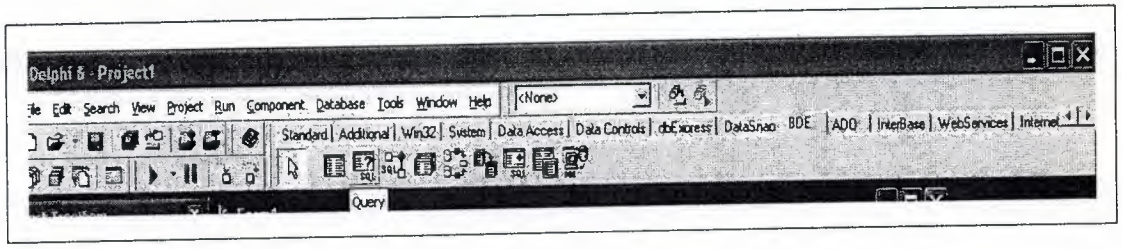

Figure 1.19. BDE Component palette

When you find the Query component, click it once to select it, then click on the form to place the component. The Query component is nonvisual, so it doesn't matter where you put it (Figure 1.20). Delphi names the object Query1 by default. (When you

point to the component on the form, Delphi displays its name--Query1--and the type of object it is--TQuery.)

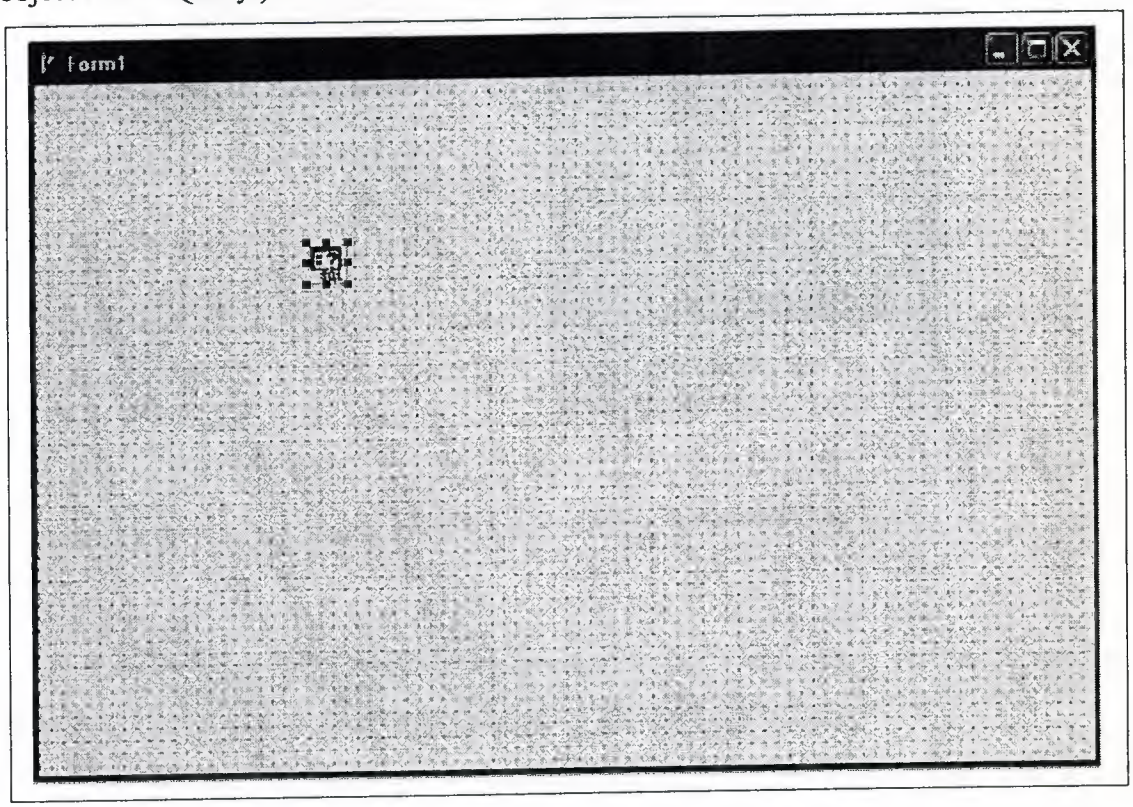

Figure 1.20. Query In The Form

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

Set the DatabaseName property of Query1 to AIRPORT. (AIRPORT is an alias to the sample database that you're going to use.)

Select Query1 on the form, then choose the DatabaseName property in the Object Inspector. Select AIRPORT from the drop-down list (Figure 1.21).

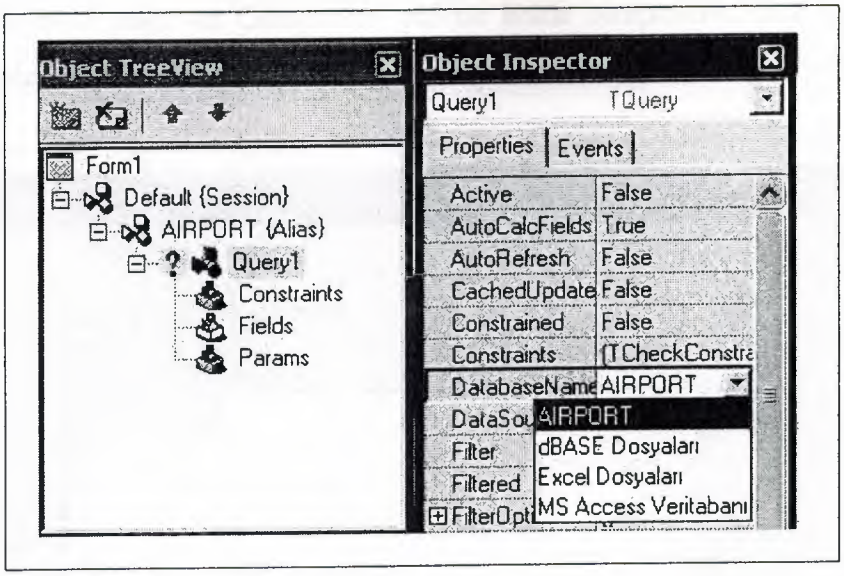

Figure 1.21. Select DatabaseName

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

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

1.15.4. Connecting to a Database

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

1. From the Data Access page of the Component palette, drop a DataSource component onto the form. The DataSource component is nonvisual, so it doesn't matter where you put it on the form. Set its DataSet property to Query1.

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

If necessary, you can enlarge the form by dragging its lower right corner. Your form should now resemble the following figure 1.22 :

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

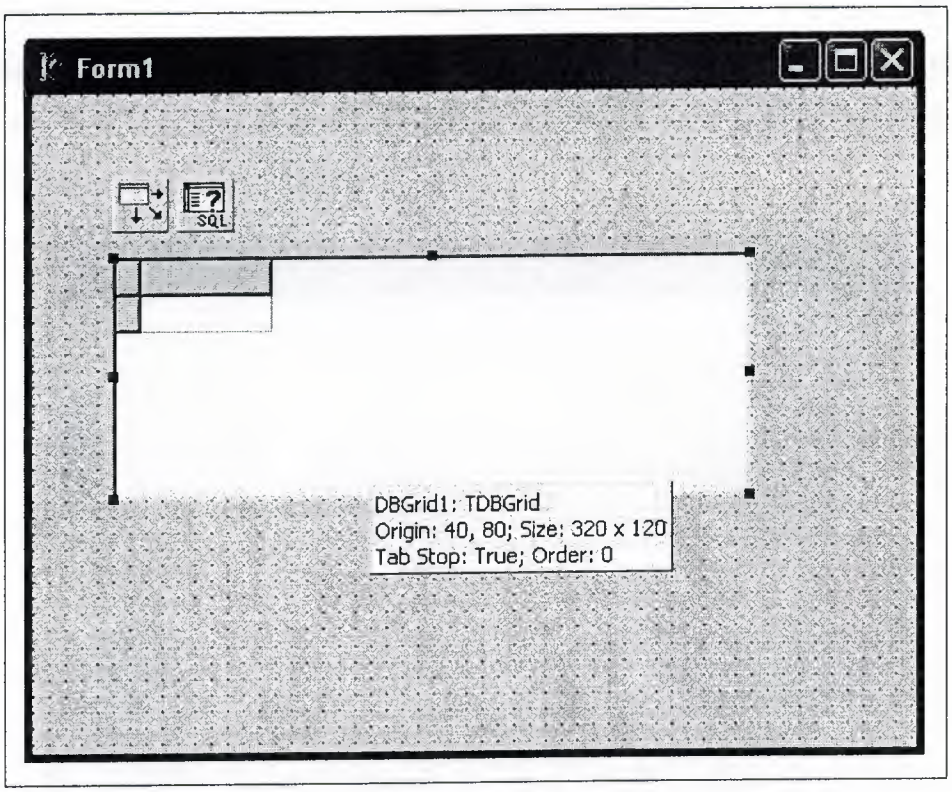

Figure 1.22. DBGrid In The Form

3. Set DBGrid properties to align the grid with the form. Double-click Anchors in the Object Inspector to display akLeft, akTop, akRight, and akBottom; set them all to True.

4. Set the DataSource property of DBGrid to DataSource1 (the default name of the DataSource component you just added to the form).

Now you can finish setting up the *Query1* object you placed on the form earlier.

5. Select the Query1 object on the form, Then double click SQL from properties part , Now you should write sql clause 'select * from <tablename>' (select * from Costtable) , then click ok button leave the sql secreen, after that set the RequestLive to True finally set Active to True top of the properties part .

When you set Active to True, the grid fills with data from the AIRPORT.DB database table (Figure 1.23). If the grid doesn't display data, make sure you've correctly set the properties of all the objects on the form, as explained in the instructions above. (Also verify that you copied the sample database files into your ...\Borland Shared\Data directory when you installed Delphi.)

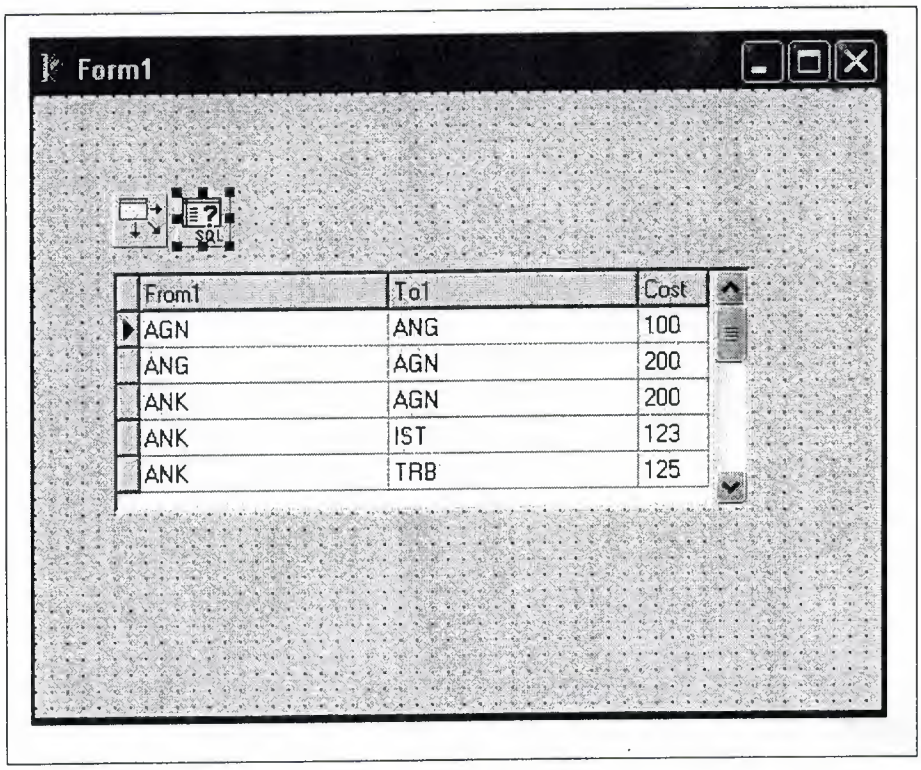

Figure 1.23. Show Table

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

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

case, a Query). Finally, the dataset (Query1) points to an actual database table as shown (AIRPORT).

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

CHAPTER 2

2. DATABASE CONCEPTS OF DELPHI 6

Delphi enables you to create robust database applications quickly and easily. Delphi database applications can work directly with desktop databases like Paradox, dBASE,the Local InterBase Server, and ODBC data sources. The Delphi Client/Server edition also works with remote database servers such as Oracle, Sybase, Microsoft SQL Server, Informix, InterBase, and ODBC data sources. Delphi client applications can be scaled easily between mission critical network-based client/server databases, and local databases on a single machine.

This chapter introduces Delphi's database tools, including the Data Access and Data Controls component pages, the Fields Editor, the Database Desktop, and the Database Forms Expert.

2.1. Database Application in Delphi

A Delphi database application is built using Delphi database development tools, Delphi data-access components, and data-aware GUI components. A database application uses Delphi components to communicate with the Borland Database Engine (BDE), which in turn communicates with databases. The following figure 2.1. illustrates the relationship of Delphi tools and Delphi database applications to the BDE and data sources:

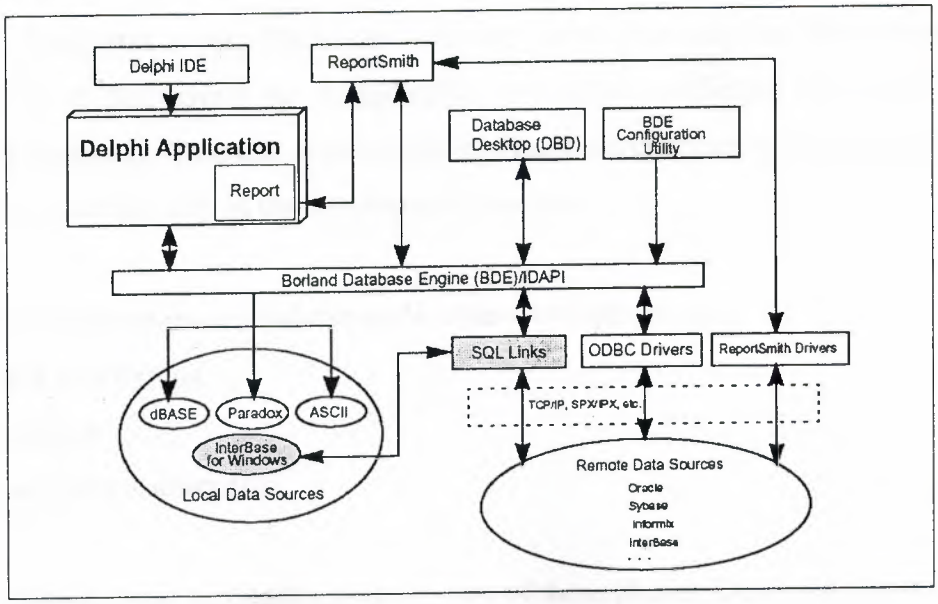

Figure 2.1. Delphi Database Architecture

The following table summarizes Delphi's database features (Table.2.1):

TOOL	PURPOSE
Report Smith	Create,view,and print report
Borland Database Engine(BDE)	Access data form file based Paradox and dBASE tables,and from local InterBase Sever database
BDE Configuration Utility	Create and manage database connection aliases used by the BDE
Local InterBase Server	Provides a single-user,multi-instance desktop SQL server for building and testing Delphi applications,before scaling them up to a production database,such as oracle,sybase,informix,or Interbase on a remote server
InterBase SQL Link	Native driver that connect Delphi applications to the local interbase server.

Table.2.1. Database Features Summary

2.1.1. Database application development cycle

The goal of database application development is to build a product which meets end users' long-term needs. While this goal may seem obvious, it is important not to lose sight of it throughout the complexities and often conflicting demands of the development process. To create a successful application it is critical to define the end users' needs in detail early in the development process.

The three primary stages of database application development are ;

- Design and prototyping
- Implementation
- Deployment and maintenance

There are database and application tasks in each of these phases. Depending on the size

and scope of the development project, the database and application tasks may be performed by different individuals or by the same individual. Often, one team or individual will be responsible for the database tasks of the project, and another team or individual will be responsible for the application tasks.

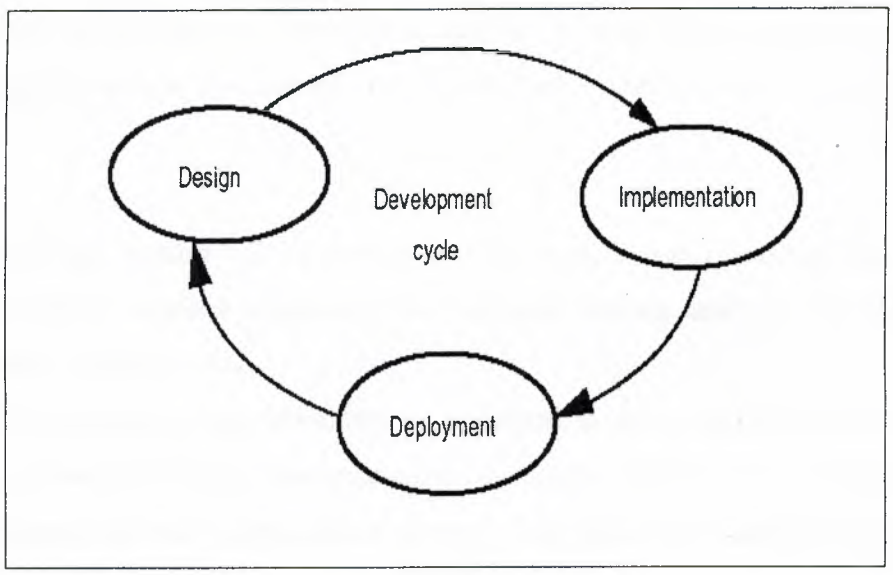

Figure 2.2 Deployment Cycle

2.1.1.1. Design phase

he design phase begins with requirements definition. In consultation with nowledgeable end users, define the functional specifications for the database and applications. Determine which aspects of the functional requirements will be mplemented in the database design, and which aspects will be implemented in the applications.

For client/server applications, often certain functions can be performed either by the server or by the application; for example, a complex mathematical transform function could be performed either by the client application or by a stored procedure on the server. The hardware deployment configuration will generally determine whether such functions are best performed on the server or client. For example, if the client platforms are expected to be low-end desktop PCs, and the server platform is expected to be a high-end workstation, then it will probably be best to run computation-intensive functions on the server. If the hardware configuration changes, then it is possible tomove the function between client and server in a later iteration.

2.1.1.2. Implementation phase

The implementation phase, you use Delphi to build and test the application conceived n the design phase. During the implementation phase, you should use a duplicate data source, that is, a data source that has the same essential structure as the production database, but with a small subset of representative data. It is not recommended to develop an application against a production database, since the untested application may corrupt the data or otherwise interfere with normal database activities.

If your application will ultimately be deployed to use a desktop data source, make copies of the required tables with the Database Desktop, and populate them with representative "dummy" data.

If the application will ultimately be deployed to use a remote data source (an SQL server), then you can take two approaches during the implementation phase:

- Develop and test the application against a non-production database on the Local InterBase Server.
- Develop and test the application against a non-production database on the server.

The first approach has the advantage that is isolated on the development platform(s), and so will not interfere with other server activities. It will not consume server resources or increase network traffic. Its primary disadvantage is that only standard SQL server features can be used and tested during this phase, if you are using a server other than InterBase for the deployed application.

The second approach enables you to surface all server-specific features, but will consume network and server resources during testing. This approach can be dangerous, since it is conceivable that a programmer error could cause a server to crash during testing.

2.1.1.3. Deployment phase

In the deployment phase, the client/server application is put to the acid test: it is handed over to end users. To ensure that the application's basic functionality is error-free, deploy a prototype application before attempting to deploy a production application.

Since the ultimate judges of an application's efficacy are its users, developers must be prepared to incorporate changes to applications arising from their suggestions, changing business needs, and for general enhancement (for example, for usability).Sometimes application changes may require changes to the database, and conversely, changes to the database may require application changes. For this reason, application developers and database developers should work together closely during this phase.As features and enhancements are incorporated into the application, the application moves iteratively closer to completion.

2.1.2. Deploying an application

Deploying an application means giving it to the end users, and providing the necessary software they need to use the application in a production environment. Non-database applications require only an .EXE file to run—Delphi applications do not require a run time interpreter or DLL.

Typically, when deploying a database application, you will create a package that includes all the files that end users need to run the application and access data sources. These files include

- The application .EXE file and .DLL files (if any)
- Required ancillary files (for example, a README file or .HLP files for online help)
- BDE support for database access (desktop or server)
- ReportSmith Runtime for running and printing reports
- If the application uses VBX controls, include each VBX along with BIVBX11.DLL

If you are distributing the files on disks, you will generally want to compress them with a standard file compression utility, and provide the utility on the disk. You may also want to build a simple installation application to install the files for your users. For complex applications, you may want to use one of the many commercially-available installation programs.

2.1.2.1. Deploying BDE support

When you deploy a database application, you must ensure that the client platform has the correct version of the BDE installed. Delphi includes Redistributable BDE (Table 2.2) , with its own installation utility, that can you can redistribute with your applications. When you deploy an application, simply include a copy of the Redistributable BDE disk.

The Delphi license agreement requires you to make all the files in Redistributable BDE available to your application users. This requirement enables users to install the new version of the BDE for Delphi without interfering with existing Paradox and Dbase applications. You can advise your users to save disk space and install only the drivers required to run your application, but you must still distribute all the files in the Redistributable BDE (Table 2.2).

FILE NAME	DESCRIPTION
IDAPI01.DLL	BDE API DLL
IDBAT01.DLL	BDE Batch utilies DLL
IDQRY01.DLL	BDE Query DLL
IDASCII01.DLL	BDE ASCII Driver DLL
IDPDX01.DLL	BDE Paradox Driver DLL
IDDBAS01.DLL	BDE dBASE Driver DLL
IDR10009.DLL	BDE Resources DLL
ILD01.DLL	Language Driver DLL
IDODBC01.DLL	BDE ODBC Socket DLL
ODBC.NEW	Microsoft ODBC Driver Manager DLL, version 2.0
ODBCINST.	NEW Microsoft ODBC Driver installation DLL, version 2.0
TUTILITY.DLL	BDE Tutility DLL
BDECFG.EXE	BDE Configuration Utility
BDECFG.HLP	BDE Configuration Utility Help
IDAPI.CFG	BDE (IDAPI) Configuration File

Table 2.2. Redistributable Borland Database Engine files

2.2. Using data access components and tools

This section describes how to use key Delphi features and tools when building database applications, including:

- The TSession component.
- Dataset components (TTable and TQuery), their properties, and their methods.
- TDataSource components, their properties, and their methods.
- TField objects, their properties, and their methods.
- The Fields Editor to instantiate and control TField objects.
- TReport and TBatchMove components.

2.2.1. Database components hierarchy

The Delphi database component hierarchy (Figure 2.3) is important to show the properties,
methods, and events inherited by components from their ancestors. The most important database components are

- *TSession*, a global component created automatically at run time. It is not visible on forms either at design time or run time.

- *TDatabase*, component that provides an additional level of control over server logins,transaction control, and other database features. It appears on the Data Access component page.

- *TDataSet* and its descendents, TTable and TQuery, collectively referred to as dataset components. TTable and TQuery components appear on the Data Access component page.

- *TDataSource*, a conduit between dataset components and data-aware components. It appears on the Data Access component page.

- *TFields*, components corresponding to database columns, created either dynamically by Delphi at run time or at design time with the Fields Editor. Data controls use them to access data from a database. In addition, you can define calculated fields whose values are calculated based on the values of one or more database columns.

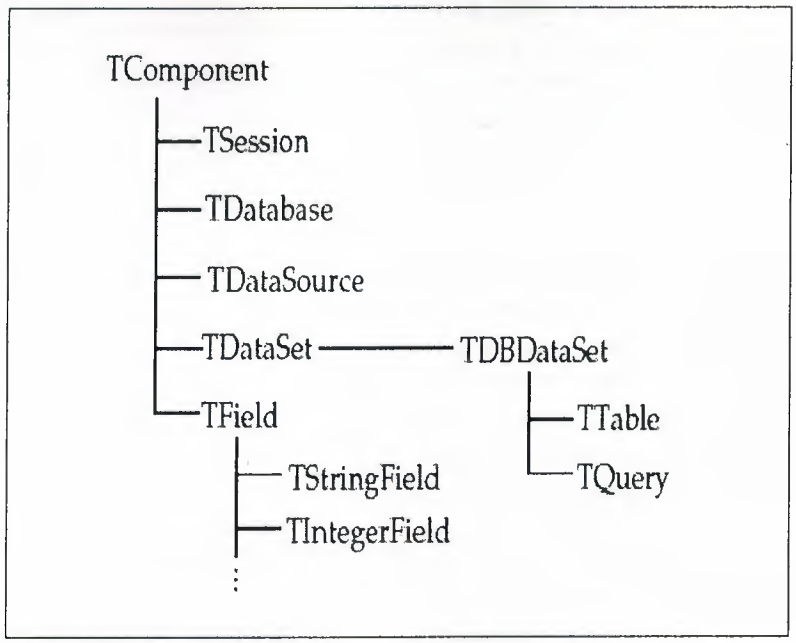

Figure 2.3. Delphi Data Access components hierarchy

2.3. Using Data Controls

To display and edit data from a database, use the components on the Data Controls page of the Component palette (Figure 2.4). Data controls include components such as TDBGrid for displaying and editing all specified records and fields in a table, and TDBNavigator for navigating among records, deleting records, and posting records when they change.

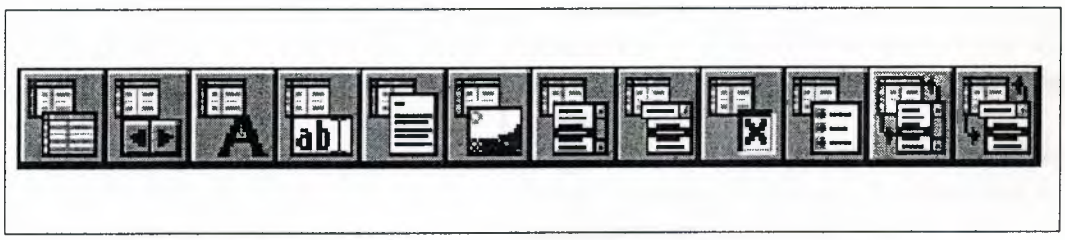

Figure 2.4. Data Controls Component palette

The following table (Table 2.2) summarizes the data controls in order from left to right as they appear on the Component palette:

DATA CONTROL	DESCRIPTION
TDBGrid	Displays information from a data source in a spreadsheet-like grid. Columns in the grid can be specified at design time using the Fields Editor or at run time (dynamically bound).
TDBNavigator	Provides buttons for navigating through data obtained from a data source. At design time, you can choose to include one or more buttons to navigate through records, update records, post records, and refresh the data from the data source
TDBText	Displays data from a specific column in the current data record.
TDBEdit	Uses an edit box to display and edit data from a specific column in the current data record.
TDBMemo	Displays memo-type database data. Memo fields can contain multiple lines of text or can contain BLOB (binary large object) data.
TDBImage	Displays graphic images and BLOB data from a specific column in the current data record.
TDBListBox	Displays a list of items from which a user can update a specific column in the current data record.
TDBComboBox	Combines a TDBEdit control with an attached list. The application user can update a specific column in the current data record by typing a value or by choosing a value from the drop-down list.

Table 2.2. Data controls

2.4. Using SQL in applications

SQL (Structured Query Language) is an industry-standard language for database operations. Delphi enables your application to use SQL syntax directly through the TQuery component. Delphi applications can use SQL to access data from:

- Paradox or dBASE tables, using local SQL. The allowable syntax is a sub-set of ANSIstandard SQL and includes basic SELECT, INSERT, UPDATE, and DELETE statements. For more information on local SQL syntax, see Appendix C, "Using local SQL."

- Databases on the Local InterBase Server. Any statement in InterBase SQL is allowed.For information on syntax and limitations, see the InterBase Language Reference.

- Databases on remote database servers (Delphi Client/Server only). You must have installed the appropriate SQL Link. Any standard statement in the server's SQL is allowed. For information on SQL syntax and limitations, see your server documentation.

Delphi also supports heterogeneous queries against more than one server or table type (for example, data from an Oracle table and a Paradox table).

2.4.1. Using TQuery

TQuery is a dataset component, and shares many characteristics with TTable, "Using data access components and tools." In addition, Tquery enables Delphi applications to issue SQL statements to a database engine (either the BDE or a server SQL engine).

The SQL statements can be either static or dynamic, that is, they can be set at design time or include parameters that vary at run time.

2.4.1.1. When to use TQuery

For simple database operations, TTable is often sufficient and provides portable database access through the BDE. However, TQuery provides additional capabilities

that TTable does not. Use TQuery for:

- Multi-table queries (joins).
- Complex queries that require sub-SELECTs.
- Operations that require explicit SQL syntax.

TTable does not use SQL syntax; TQuery uses SQL, which provides powerful relational capabilities but may increase an application's overall complexity. Also, use of nonstandard (server-specific) SQL syntax may decrease an application's portability among servers.

2.4.1.2. How to use TQuery

To access a database, set the DatabaseName property to a defined BDE alias, a directory path for desktop database files, or a file name for a server database. If the application has a TDatabase component, DatabaseName can also be set to a local alias that it defines.

To issue SQL statements with a TQuery component:

- Assign the TQuery component's SQL property the text of the SQL statement. You can do this:
  - ✓ At design time, by editing the TQuery's SQL property in the Object Inspector,choosing the SQL property, and entering the SQL statements in the String List Editor dialog box. With Delphi Client/Server, you can also use the Visual Query Builder to construct SQL syntax.
  - ✓ At run time, by closing any current query with Close, clearing the SQL property with Clear, and then specifying the SQL text with the Add method.
- Execute the statement with the TQuery component's Open or ExecSQL method. Use Open for SELECT statements. Use ExecSQL for all other SQL statements. The differences between Open and ExecSQL are discussed in a subsequent section.

- To use a dynamic SQL statement, use the Prepare method, provide parameters and then call Open or ExecSQL. Prepare is not required, but will improve performance for dynamic queries executed multiple times.

The following diagram (Figure 2.5) illustrates the lifetime of a TQuery component and the methods
used to work with it:

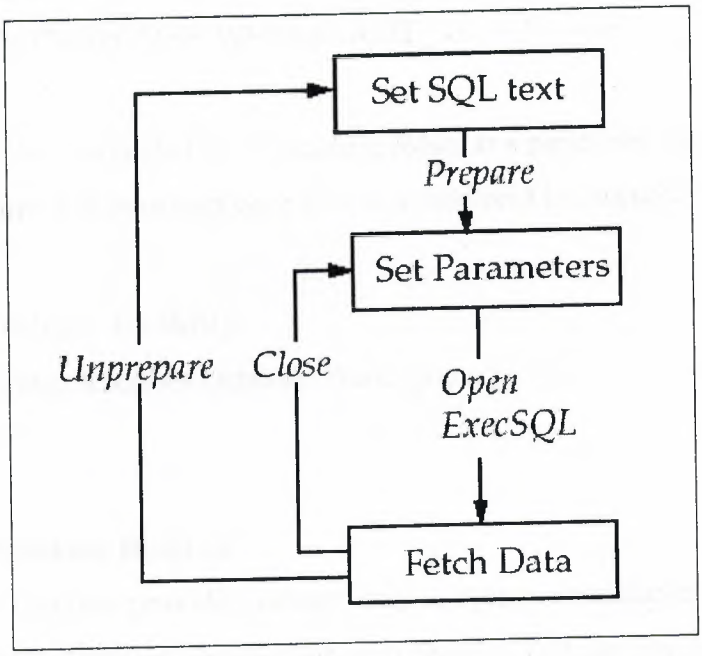

Figure 2.5. TQuery methods and flow

2.4.2. The SQL property

The SQL property contains the text of the SQL statement to be executed by a Query component. This property is of type TStrings, which means that it is a series of strings in a list. The list acts very much as if it were an array, but it is actually a special class with unique capabilities. For more information on TStrings, see the online VCL reference.

A Query component can execute two kinds of SQL statements:

- Static SQL statements
- Dynamic SQL statements

A static SQL statement is fixed at design time and does not contain any parameters or

variables. For example, this statement is a static SQL statement:

SELECT * FROM CUSTOMER WHERE CUST_NO = 234

A dynamic SQL statement, also called a parameterized statement, includes parameters for column or table names. For example, this is a dynamic SQL statement:

SELECT * FROM CUSTOMER WHERE CUST_NO = :Number

The variable Number, indicated by the leading colon, is a parameter which must be provided at run time and may vary each time the statement is executed.

2.5. Using Database Desktop

This appendix describes Database Desktop and provides a synopsis of Database Desktop features.

2.5.1. What is Database Desktop?

Database Desktop provides an easy way to create, restructure, and query tables to help you develop database applications with Delphi. You can use Database Desktop either as a standalone application on a single computer running Windows or as a multiuser application on a network.

2.5.2. Starting Database Desktop

To start Database Desktop, double-click the Database Desktop icon in the Delphi program group or choose File|Run in the Program Manager and run DBD.EXE.

CHAPTER3

3. FLIGHT INFORMATION AND TICKET SERVICES

3.1. User Login

This program begin the login form (Figure 3.1) that includes username, password and forget password.The aim of this login form is protect program from the non-users.

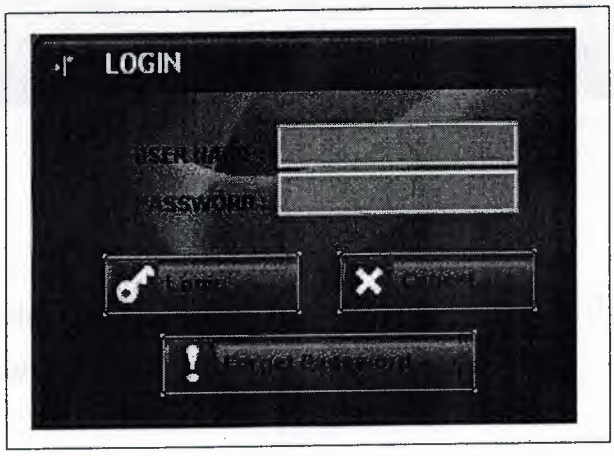

Figure 3.1. Login Form

The first user name , password and Secret answer gives from the programmer.User Name is Admin and password 12345. if entering any user name or password moreover that, you will see this error (Figure 3.2) ;

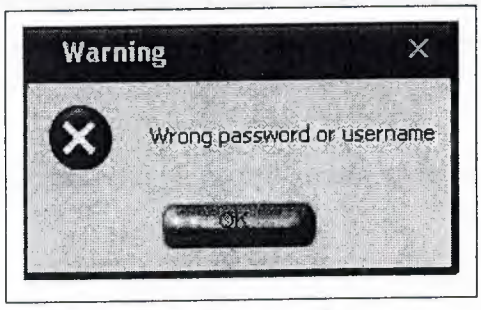

Figure 3.2. Warning Message

If you forget the password , Click the Forget Password button then you will see the this box (Figure 3.4) ;

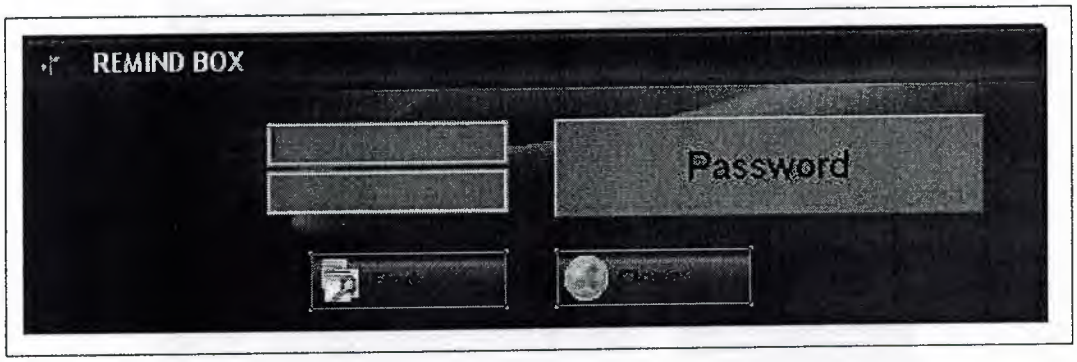

Figure 3.3. Remaind Box

You shoul enter user name and secret answer then click the find button for the remaind your password.

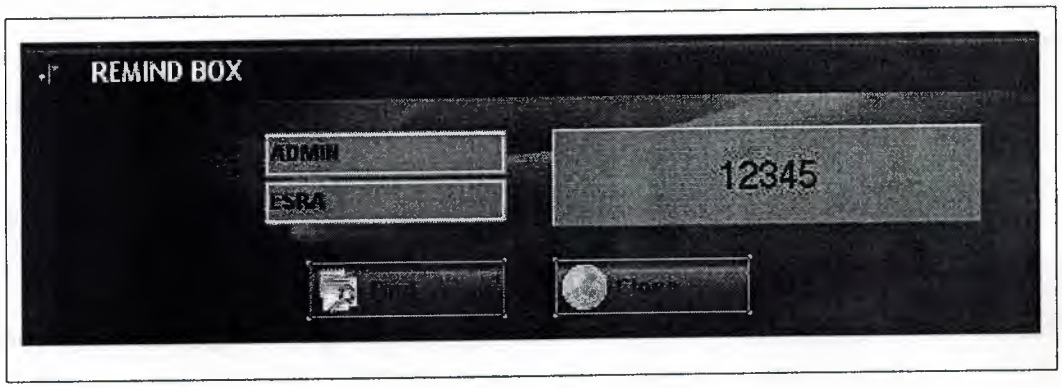

Figure 3.4. Show Password

After enter the correct username and password , main menu will open which has 5 main process;

3.2. New Information

This menu is included four adding process we shows one of them;

When you click File→ New→ Add City . City form (Figure 3.5) will come in the new form.

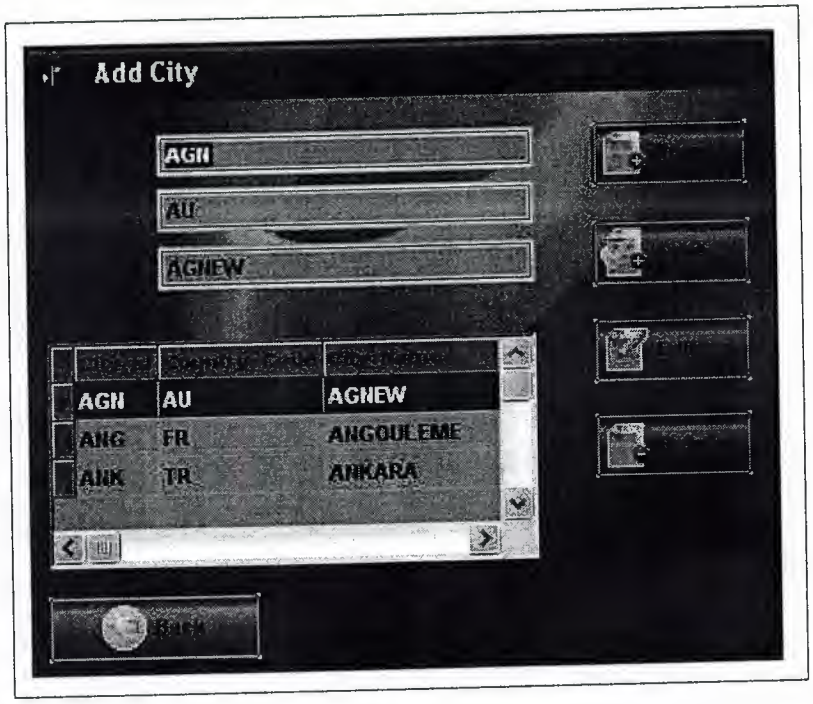

Figure 3.5. Add City

You can save , delete , edit information at this form. We save several place in the program, if you need without these add with new button.After arrange the place you are ready to sell or to reservation of ticket.

3.3. Customer Data

This form included for the register new customer you can not sell/register the ticket before the register (Figure 3.6).

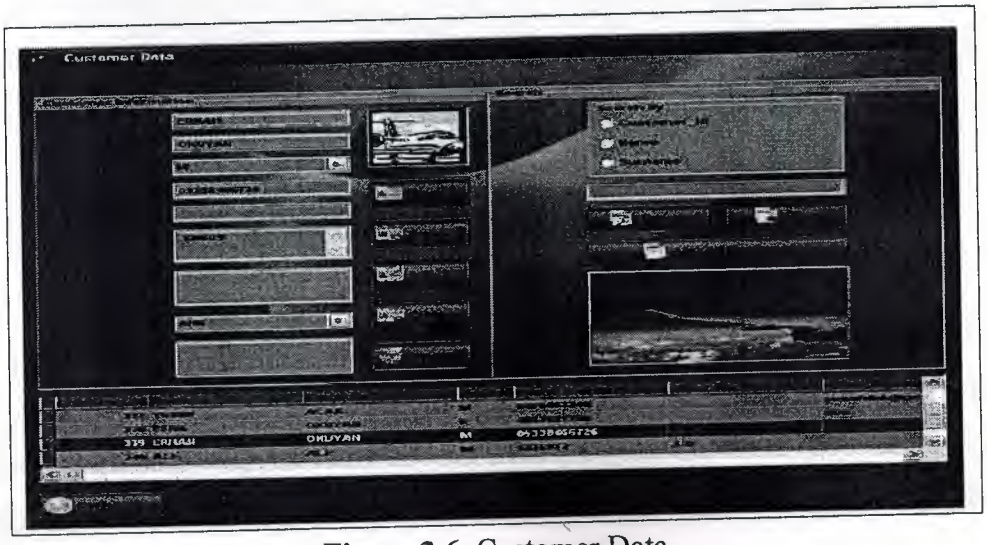

Figure 3.6. Customer Data

Also this form included search part.If you want to search information of the customer , you can search by customer id , customer name or customer surname.When you click the search button you will see the customer information in the table included as the search word.

3.4. Ticket Services

This form includen two part ticket reservation and ticket selling you can reserve the ticket or directly sell.

3.4.1 Reservation

You can arrange the flight with the customer information then click the reservation button (Figure 3.7) , the program will be automatically give option date. Customer have to buy before the option date.Option date shows the date which is two days before the departure date.

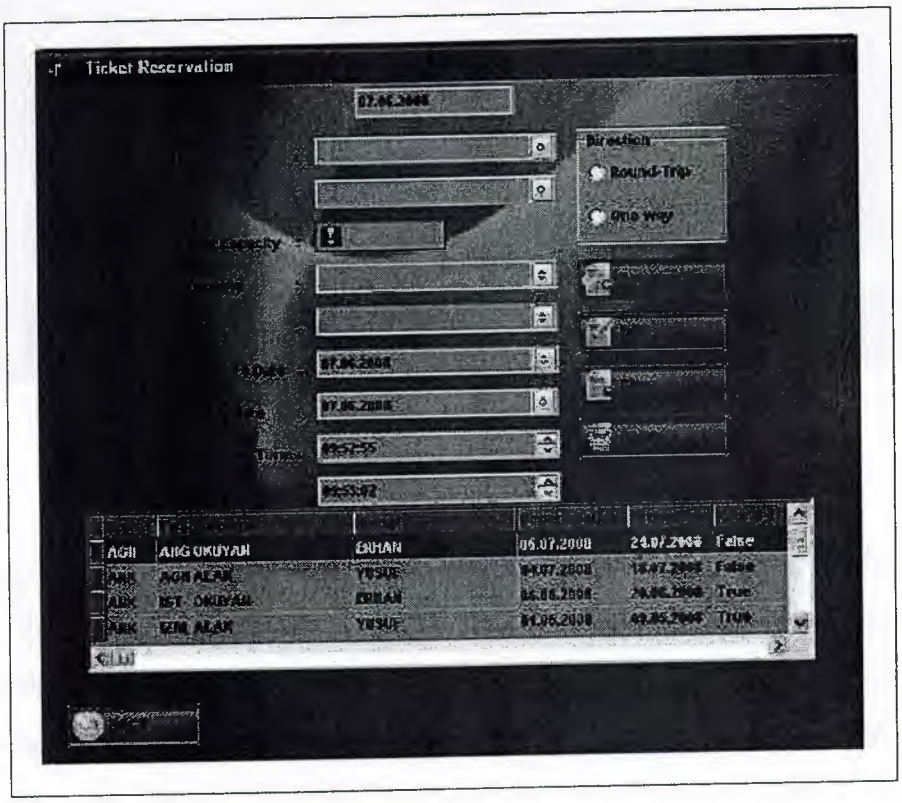

Figure 3.7. Reservation

You can see the flight capacity after from and to selected if capacity is zero, you can not this flight for the reservation if not click the reservation button then you can see the table your arrange if any wrong double click from the table then re-arrange whatever you want to change.After re-arrange click the edit button for the save your work.

3.4.2 Selling

You can arrange the flight with the customer information then click the sell button , the program will be automaticly save your work.

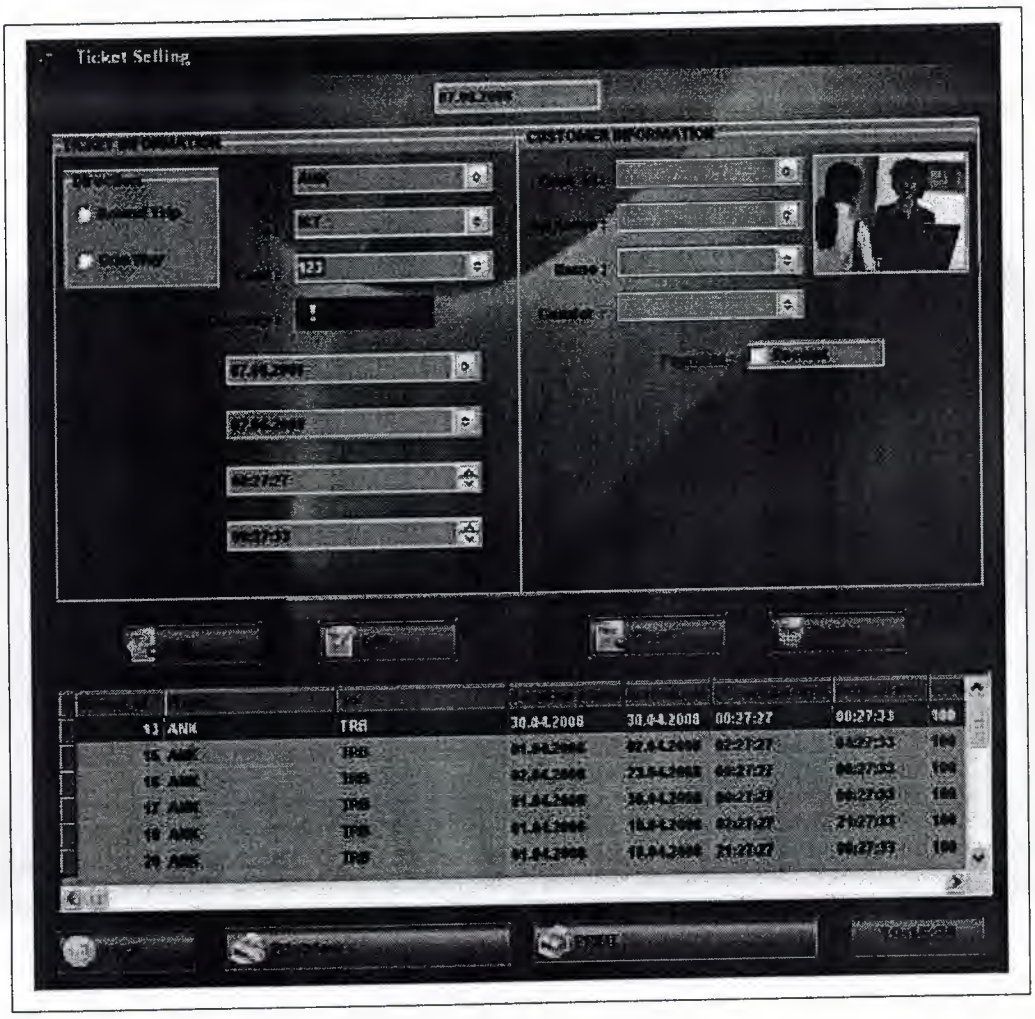

Figure 3.8. Selling

Again you have to check flight capacity before sell the ticket if flight capacit is zero this means there is no place as shown the figure 3.8. Capacity is not zero you can sell the ticket after arrange the work. Before click the sell button you should tick the payment box otherwise you can not sell the ticket. Payment box is control box if customer give the ticket price click if not program not save your work even click the sell button after all this you can sell the ticket with help the table.Select customer from the table and Double click now check your work if there is any problem or not. If it is ok click printview button for the ticket or click print button directly printing. You can

solve any problem with edit button just call the whatever you want from the table then click the edit button after re-arrange data.

3.5. Company Data

This form (Figure 3.9) is very important and every programs should has as company data because we want to know what we are doing at the job in the beginning till now.

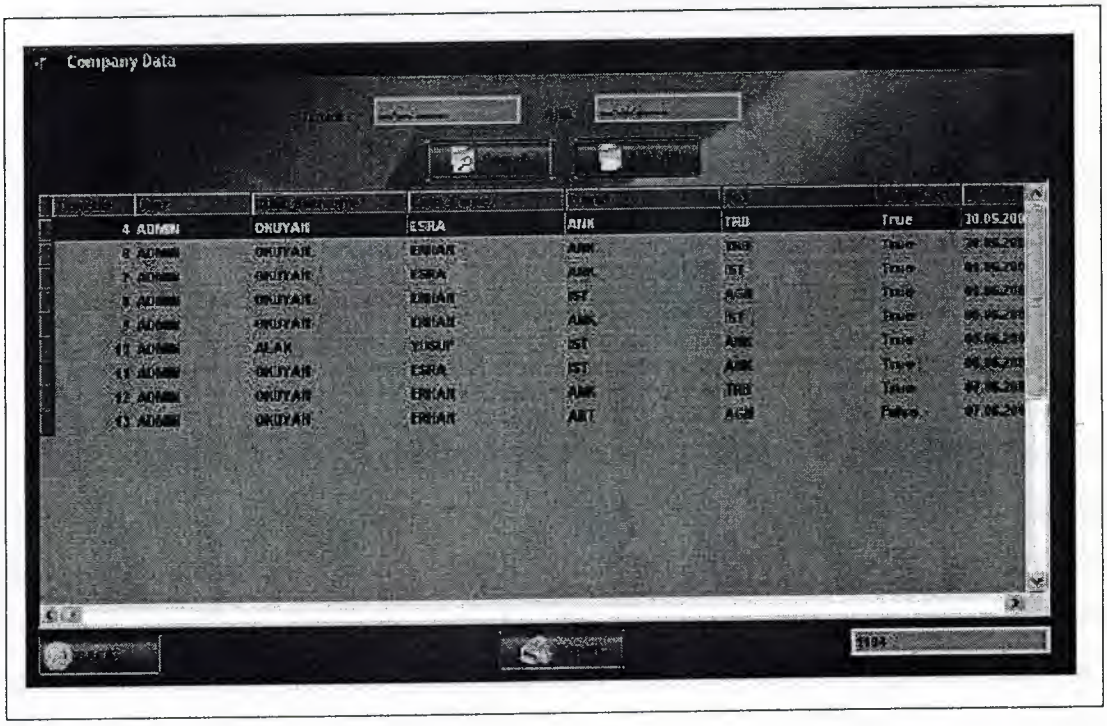

Figure 3.9. Company Data

In this form saving data while you are sell the ticket with the sell dates. If you wan to see specific dates write top of the form in the box then click the search button for the filter whatever you select or if you can see from the first sell dates till the last sell date with click the list all button. End of the table you can see the sum of the price with specific dates or all dates. Then you can print the information with report button.

3.6. User Register

This form for the new user registration.You can declare the new user with register form (Figure 3.10).

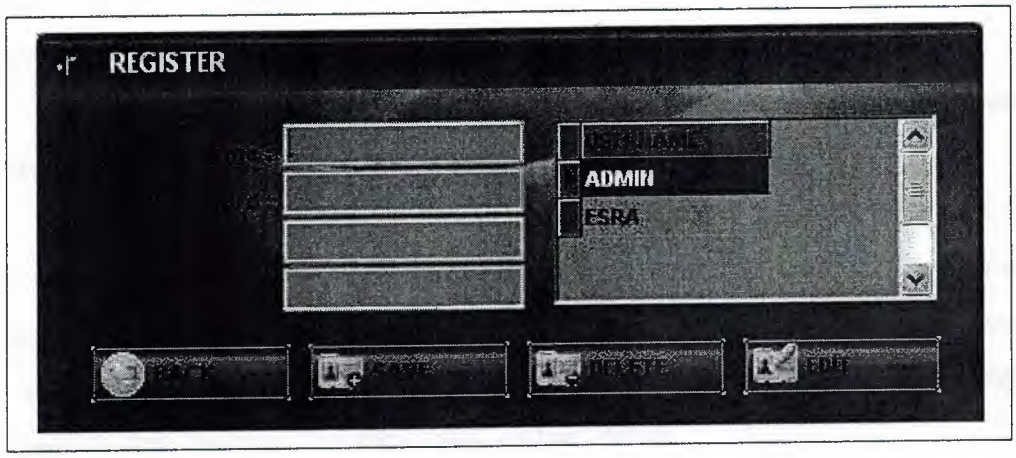

Figure 3.10. Register

If you want to declare new user for the enter the program you can do with the form. Enter the new user name and declare the password for that user as you can see after step is re-password you should write the same password to this box whatever you declare password box. This is control while the saving user. After enter that you declare secret answer for the that user , it is necessary when the user is forget the password. When you finish click the save button for the save your work.Now new user can enter the program with password.

CONCLUSION

After making so many reasources about Delphi programming language investigating through internet to make this project, I learned many things about Delphi, bacause I obliged to finish my project and everything had to be done by myself.

This can be used easily for each user and program can be record customer information and so many things.

The operation structures of this program could be explain briefly; as follows when user executes program,first login secreen appears.In this screen user enter the username and password to use the FITs (Flight Information and Ticket Services).So user must have a valid user name and password. Also user must have appropriate privileges on data base; such as view, add, update, delete.

When the username and password correct, user meets the Main menu screen. User can do several works in the program such as Ticket reservation, Ticket selling , Adding new flight.

In the future other options could be add to the program and also for the future implementations the current program can be developed using different program languages.

REFERENCES

[1] Ihsan KARAGULLE & Zeydin PALA, Borland Delphi ile Veritabanı, Türkmen kitabevi , Istanbul , 2001.

[2] Memik YANIK, Borland Delphi6, Beta A.Ş., Istanbul, 2002.

[3] Guide for some code. "http://www.delphiturkiye.com/ipdb.htm"

[4] Guide for Airport telphone number.
"http://www.neredennereye.com/bilgi/havaalanlari".

[5] Guide for some code. "http://www.programlama.com".

[6] Guide for some information. "http://www.zamansiz.com/sitemap/f-411.html".

APPENDIX

Program code

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, jpeg, ExtCtrls, StdCtrls, WinSkinData, ComCtrls ,shellapi,
  Mask, DBCtrls, LbButton;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    icketSell1: TMenuItem;
    reservation1: TMenuItem;
    flightinformation1: TMenuItem;
    icketprice1: TMenuItem;
    icketSell2: TMenuItem;
    Reservation2: TMenuItem;
    Add1: TMenuItem;
    Exit1: TMenuItem;
    Placeinfo1: TMenuItem;
    Flightinfo1: TMenuItem;
    CompanyData1: TMenuItem;
    Help1: TMenuItem;
    Setting1: TMenuItem;
    ExchangeRate1: TMenuItem;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
AddCity1: TMenuItem;
AddCost1: TMenuItem;
SkinData1: TSkinData;
AddCountry1: TMenuItem;
MonthCalendar1: TMonthCalendar;
Label4: TLabel;
EmployeeInfo1: TMenuItem;
USERREGISTER1: TMenuItem;
Bevel1: TBevel;
Bevel2: TBevel;
Bevel3: TBevel;
LbButton1: TLbButton;
AddFlight1: TMenuItem;
LbButton2: TLbButton;
procedure icketSell2Click(Sender: TObject);
procedure exit2Click(Sender: TObject);
procedure AddCity1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure AddCost1Click(Sender: TObject);
procedure AddCountry1Click(Sender: TObject);
procedure Flightinfo1Click(Sender: TObject);
procedure Placeinfo1Click(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure icketprice1Click(Sender: TObject);
procedure CompanyData1Click(Sender: TObject);
procedure Reservation2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure ExchangeRate1Click(Sender: TObject);
procedure EmployeeInfo1Click(Sender: TObject);
procedure Help1Click(Sender: TObject);
procedure USERREGISTER1Click(Sender: TObject);
procedure LbButton1Click(Sender: TObject);
procedure AddFlight1Click(Sender: TObject);
private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit2, Unit5, Unit3, Unit6, Unit4, Unit7, Unit8, Unit9, Unit10, Unit12,
  Unit13, Unit17, Unit16, Unit19, Unit20;

{$R *.dfm}

procedure TForm1.icketSell2Click(Sender: TObject);
begin
form4.show;
form1.Enabled:=false;
end;

procedure TForm1.exit2Click(Sender: TObject);
begin
form1.Close;
end;

procedure TForm1.AddCity1Click(Sender: TObject);
begin
form3.Show;
form1.Enabled:=false;
end;

procedure TForm1.Exit1Click(Sender: TObject);
var
```

```
a:word;
begin
a:=application.MessageBox('Are You Sure to leaving the system?','WARNING',36);
if(a=IDYES)then
begin
form1.close;
form16.close;
end;
end;

procedure TForm1.AddCost1Click(Sender: TObject);
begin
form6.Show;
form1.Enabled:=false;
end;

procedure TForm1.AddCountry1Click(Sender: TObject);
begin
Form7.show;
form1.Enabled:=false;
end;

procedure TForm1.Flightinfo1Click(Sender: TObject);
begin
form8.show;
form1.Enabled:=false;
end;

procedure TForm1.Placeinfo1Click(Sender: TObject);
begin
form9.show;
form1.Enabled:=false;
end;
```

```
procedure TForm1.FormActivate(Sender: TObject);
begin
form8.Close;
form2.Close;
form1.MonthCalendar1.Date:=date;
end;

procedure TForm1.icketprice1Click(Sender: TObject);
begin
form5.Show;
form1.Enabled:=false;
end;

procedure TForm1.CompanyData1Click(Sender: TObject);
begin
form10.Show;
form1.Enabled:=false;
end;

procedure TForm1.Reservation2Click(Sender: TObject);
begin
form2.Show;
form1.Enabled:=false;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

procedure TForm1.ExchangeRate1Click(Sender: TObject);
begin
ShellExecute(Handle, 'open', 'http://www.tcmb.gov.tr/kurlar/today.html', nil, nil,
sw_ShowMaximized);
```

```
end;

procedure TForm1.EmployeeInfo1Click(Sender: TObject);
begin
form13.show;
form1.Enabled:=false;
end;

procedure TForm1.Help1Click(Sender: TObject);
begin
form12.show;
form1.Enabled:=false;
end;

procedure TForm1.USERREGISTER1Click(Sender: TObject);
begin
Form17.show;
form1.Enabled:=false;
end;

procedure TForm1.LbButton1Click(Sender: TObject);
begin
form19.show;
form1.Enabled:=false;
end;

procedure TForm1.AddFlight1Click(Sender: TObject);
begin
Form20.show;
Form1.Enabled:=false;
end;

end.
```

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, LbButton, DB, DBTables, Grids, DBGrids, StdCtrls, DBCtrls,
  ComCtrls, jpeg, ExtCtrls, Mask, BDE, LbStaticText;

type
  TForm2 = class(TForm)
    LbButton1: TLbButton;
    DepartureDate: TDateTimePicker;
    DepartureTime: TDateTimePicker;
    ArrivalTime: TDateTimePicker;
    TodayDate1: TEdit;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    from1: TComboBox;
    surname: TComboBox;
    to1: TComboBox;
    name: TComboBox;
    Query2: TQuery;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    Query3: TQuery;
```

```
ArrivalDate: TDateTimePicker;
Label8: TLabel;
Query1: TQuery;
RadioGroup1: TRadioGroup;
Query1From1: TStringField;
Query1To1: TStringField;
Query1Surname: TStringField;
Query1Name: TStringField;
Query1DepartureDate: TDateField;
Query1ArrivalDate: TDateField;
Query1RoundTrip: TBooleanField;
Query1DepartureTime: TTimeField;
Query1ArrivalTime: TTimeField;
Query1TodayDay: TDateField;
Query1OptionDate: TDateField;
Timer1: TTimer;
LbButton5: TLbButton;
LbButton2: TLbButton;
Bevel1: TBevel;
Query2From1: TStringField;
Query2To1: TStringField;
Query2Cost: TStringField;
Query3Customer_id: TAutoIncField;
Query3Name: TStringField;
Query3Surname: TStringField;
Query3Gender: TStringField;
Query3Phone_Number: TStringField;
Query3Other_Number: TStringField;
Query3EMail_Address: TStringField;
Query3Address: TStringField;
Query3City: TStringField;
Query3Detail: TStringField;
Label9: TLabel;
LbStaticText1: TLbStaticText;
```

```
  DataSource4: TDataSource;
  Query4: TQuery;
  DBGrid1: TDBGrid;
  DataSource1: TDataSource;
  procedure LbButton1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure FormActivate(Sender: TObject);
  procedure from1Change(Sender: TObject);
  procedure surnameChange(Sender: TObject);
  procedure RadioGroup1Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
  procedure LbButton5Click(Sender: TObject);
  procedure LbButton2Click(Sender: TObject);
  procedure LbButton3Click(Sender: TObject);
  procedure LbButton4Click(Sender: TObject);
  procedure DBGrid1DblClick(Sender: TObject);
  procedure to1Change(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form2: TForm2;
 direction:string;

implementation

uses Unit1, Unit6;

{$R *.dfm}

procedure TForm2.LbButton1Click(Sender: TObject);
```

```
begin
form1.Enabled:=true;
form1.show;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form2.Left:=1;
form2.Top:=46;
form2.ClientHeight:=508;
form2.ClientWidth:=620;
end;

procedure TForm2.FormActivate(Sender: TObject);
begin
from1.Items.Clear;
surname.Items.Clear;

query2.First;
query3.First;
while not(query3.Eof) do begin
surname.Items.Add(query3.Fields[2].AsString);
query3.Next;
end;

while not(query2.Eof) do begin
from1.Items.Add(query2.Fields[0].AsString);
query2.Next;
end;
form2.DepartureDate.Date:=Date;
form2.ArrivalDate.Date:=Date;
end;
```

```
procedure TForm2.from1Change(Sender: TObject);
begin
to1.Items.Clear;
query2.First;
while not(query2.Eof) do begin
 if (query2.Fields[0].AsString=from1.Text) then
  Begin
   to1.Items.Add(query2.Fields[1].AsString);
  end;
   query2.Next;
 end;
end;

procedure TForm2.surnameChange(Sender: TObject);
begin
name.Items.Clear;
query3.First;
while not(query3.Eof) do begin
if (query3.Fields[2].AsString=surname.Text) then begin
name.Items.Add(query3.Fields[1].AsString);
end;
query3.Next;
end;
end;

procedure TForm2.RadioGroup1Click(Sender: TObject);
begin
if (form2.RadioGroup1.ItemIndex=0) then direction:='True';
if (form2.RadioGroup1.ItemIndex=1) then direction:='False';
end;

procedure save;
var
option : TDate;
```

```
begin
    option:=form2.DepartureDate.Date;
    form2.Query1.Fields[0].AsString:=form2.From1.Text;
    form2.Query1.Fields[1].AsString:=form2.to1.Text;
    form2.Query1.Fields[4].AsString:=datetostr(form2.DepartureDate.Date);
    form2.Query1.Fields[5].AsString:=datetostr(form2.ArrivalDate.Date);
    form2.Query1.Fields[7].AsString:=timetostr(form2.DepartureTime.Time);
    form2.Query1.Fields[8].AsString:=timetostr(form2.ArrivalTime.Time);
    form2.Query1.Fields[3].AsString:=form2.name.Text;
    form2.Query1.Fields[2].AsString:=form2.Surname.Text;
    form2.Query1.Fields[9].AsString:=datetostr(date);
    form2.Query1.Fields[6].AsString:=direction;
    form2.Query1.Fields[10].AsString:=DateToStr(option-2);
end;

procedure read;
begin
form2.From1.Text:=form2.Query1.Fields[0].AsString;
form2.to1.Text:=form2.Query1.Fields[1].AsString;
form2.Name.Text:=form2.Query1.Fields[3].AsString;
form2.Surname.Text:=form2.Query1.Fields[2].AsString;
form2.TodayDate1.Text:=datetostr(form2.Query1.Fields[9].AsDateTime);
direction:=form2.Query1.Fields[6].AsString;
form2.DepartureDate.Date:=strtodate(form2.Query1.Fields[4].AsString);
form2.ArrivalDate.Date:=strtodate(form2.Query1.Fields[5].AsString);
form2.DepartureTime.Time:=strtotime(form2.Query1.Fields[7].AsString);
form2.ArrivalTime.Time:=strtotime(form2.Query1.Fields[8].AsString);
end;

procedure clear;
begin
form2.From1.Text:='';
form2.to1.Text:='';
form2.Name.Text:='';
```

```
form2.Surname.Text:='';
form2.TodayDate1.Text:='';
direction:='';
form2.RadioGroup1.ItemIndex:=-1;
form2.LbStaticText1.Caption:='';
end;


procedure TForm2.Timer1Timer(Sender: TObject);
begin
form2.TodayDate1.Text:=datetostr(date);
end;

procedure TForm2.LbButton5Click(Sender: TObject);
begin
clear;
if direction='' then
form2.RadioGroup1.ItemIndex:=-1;

end;

function
bul(from1:string;to1:string;name:string;surname:string;ddate:string;adate:string):boolea
n;
begin
bul:=false;
form2.Query1.First;
while not form2.Query1.eof do
if (from1=form2.Query1.Fields[0].asstring)and
  (to1=form2.Query1.Fields[1].asstring)  and
  (name=form2.Query1.Fields[3].asstring) and
  (ddate=form2.Query1.Fields[4].asstring) and
  (adate=form2.Query1.Fields[5].asstring) and
  (surname=form2.Query1.Fields[2].asstring)then
```

```
begin
bul:=true;
exit;
end
else
form2.Query1.Next;
end;

procedure TForm2.LbButton2Click(Sender: TObject);
begin
if (query4.Fields[3].AsString='0') then begin
application.MessageBox('This flight is full!','Warning',16);
end
else begin
if not
bul(from1.Text,to1.Text,name.Text,surname.Text,datetostr(form2.DepartureDate.Date),
datetostr(form2.ArrivalDate.Date)) then begin
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
begin
if (form2.RadioGroup1.ItemIndex<>-1) then begin
if(timetostr(form2.DepartureTime.Time)<>timetostr(form2.ArrivalTime.Time)) then
begin
query1.Insert;
save;
query1.Post;
clear;
if (direction='') then form2.RadioGroup1.ItemIndex:=-1;
end else application.MessageBox('The Departure Time and Arrival Time must be
different.','Warning',16);
end;
end;
end;
query4.Edit;
query4.Fields[3].AsString:=inttostr((strtoint(LbStaticText1.Caption))-1);
```

```
query4.Post;
end;
end;

procedure TForm2.LbButton3Click(Sender: TObject);
begin
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
 begin
  query1.edit;
  save;
  query1.Post;
  clear;
 if (direction='') then form2.RadioGroup1.ItemIndex:=-1;
 end;
 end;

procedure TForm2.LbButton4Click(Sender: TObject);
begin
if
bul(from1.Text,to1.Text,name.Text,surname.Text,datetostr(form2.DepartureDate.Date),
datetostr(form2.ArrivalDate.Date)) then begin
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
begin
query1.Delete;
clear;
if (direction='') then form2.RadioGroup1.ItemIndex:=-1;
end;
end;
end;

procedure TForm2.DBGrid1DblClick(Sender: TObject);
var
a,b,c,d,e,f:string;
begin
```

```
a:=dbgrid1.Fields[0].AsString;
b:=dbgrid1.Fields[1].AsString;
c:=dbgrid1.Fields[3].AsString;
d:=dbgrid1.Fields[2].AsString;
e:=dbgrid1.Fields[4].AsString;
f:=dbgrid1.Fields[5].AsString;

if bul(a,b,c,d,e,f) then begin
clear;
read;
if (direction='True') then form2.RadioGroup1.ItemIndex:=0;
if (direction='False') then form2.RadioGroup1.ItemIndex:=1;
if (direction='') then form2.RadioGroup1.ItemIndex:=-1;
end;

end;

procedure TForm2.to1Change(Sender: TObject);
begin
query4.Close;
query4.SQL.Clear;
query4.SQL.Add('select * from flighttable where from1 like
'+#39+(from1.Text)+'%'+#39'and to1 like'+#39+(to1.Text)+'%'+#39);
query4.Open;
LbStaticText1.Caption:=query4.Fields[3].AsString;
end;

end.
```

```
unit Unit3;

interface

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

type
  TForm3 = class(TForm)
    DBEdit1: TDBEdit;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    DBGrid1: TDBGrid;
    Query1: TQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    DBEdit2: TDBEdit;
    DataSource4: TDataSource;
    DataSource5: TDataSource;
    DataSource6: TDataSource;
    Image1: TImage;
    Label2: TLabel;
    Label1: TLabel;
    DBEdit3: TDBEdit;
    Label3: TLabel;
    procedure SpeedButton5Click(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
```

```
  procedure LbButton2Click(Sender: TObject);
  procedure LbButton3Click(Sender: TObject);
  procedure LbButton4Click(Sender: TObject);
  procedure Query1AfterPost(DataSet: TDataSet);
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form3: TForm3;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm3.SpeedButton5Click(Sender: TObject);
begin
form1.show;
form3.Hide;
end;

procedure TForm3.LbButton5Click(Sender: TObject);
begin
form1.Enabled:=true;
form1.Show;
form3.close;
end;

procedure TForm3.LbButton1Click(Sender: TObject);
```

```
begin
dbedit1.Text:='';
dbedit2.Text:='';
dbedit1.SetFocus;
Query1.Insert;
end;

procedure TForm3.LbButton2Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') and (dbedit3.Text<>'')then begin
if (dbedit2.Text<>Query1.Fields[0].AsString)then begin
Query1.Post;
end;
end;
end;

procedure TForm3.LbButton3Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query1.Edit;
dbedit1.SetFocus;
end;
end;
procedure TForm3.LbButton4Click(Sender: TObject);
var
a:word;
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
a:=application.MessageBox('Your Entry Will Be Deleted.Are You Sure?','Warning',36);
if(a=IDYES)then
begin
Query1.Delete;
end;
end;
```

```
end;

procedure TForm3.Query1AfterPost(DataSet: TDataSet);
begin
try
 DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
 On EDatabaseError do
  ShowMessage('Query Save Error!...');
end;
end;

procedure TForm3.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form3.Left:=1;
form3.Top:=46;
form3.ClientHeight:=287;
form3.ClientWidth:=375;
end;

end.
```

```
unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Mask, DBCtrls, DB, Grids, DBGrids, ExtCtrls, jpeg,
  Buttons, DBTables, LbButton, ComCtrls ,BDE, LbStaticText;

type
  TForm4 = class(TForm)
    GroupBox1: TGroupBox;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    CheckBox1: TCheckBox;
    GroupBox2: TGroupBox;
    DepartureDate: TDateTimePicker;
    ArrivalDate: TDateTimePicker;
    DepartureTime: TDateTimePicker;
    ArrivalTime: TDateTimePicker;
    LbButton5: TLbButton;
    Image2: TImage;
    Bevel3: TBevel;
    Image3: TImage;
    Image1: TImage;
    Bevel2: TBevel;
    Label8: TLabel;
    Label5: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Bevel4: TBevel;
    Image4: TImage;
    Label9: TLabel;
```

```
Label7: TLabel;
Label6: TLabel;
Label12: TLabel;
Label11: TLabel;
Label10: TLabel;
Label1: TLabel;
LbButton7: TLbButton;
LbButton8: TLbButton;
LbStaticText1: TLbStaticText;
Timer1: TTimer;
TodayDate1: TEdit;
From1: TComboBox;
To1: TComboBox;
Cost: TComboBox;
Surname: TComboBox;
Name: TComboBox;
Gender: TComboBox;
DataSource1: TDataSource;
Query1: TQuery;
DataSource2: TDataSource;
Query2: TQuery;
Query3: TQuery;
DataSource3: TDataSource;
RadioGroup1: TRadioGroup;
LbButton6: TLbButton;
DBGrid1: TDBGrid;
Query4: TQuery;
DataSource4: TDataSource;
Query5: TQuery;
DataSource5: TDataSource;
Query1Travel_id: TAutoIncField;
Query1From1: TStringField;
Query1To1: TStringField;
Query1DepartureDate: TDateField;
```

```
Query1ArrivalDate: TDateField;
Query1DepartureTime: TTimeField;
Query1ArrivalTime: TTimeField;
Query1Cost: TStringField;
Query1Name: TStringField;
Query1Surname: TStringField;
Query1Gender: TStringField;
Query1TodayDay: TDateField;
Query1Roundtrip: TBooleanField;
Timer2: TTimer;
Bevel1: TBevel;
custid: TComboBox;
Label4: TLabel;
Query3Customer_id: TAutoIncField;
Query3Name: TStringField;
Query3Surname: TStringField;
Query3Gender: TStringField;
Query3Phone_Number: TStringField;
Query3Other_Number: TStringField;
Query3EMail_Address: TStringField;
Query3Address: TStringField;
Query3City: TStringField;
Query3Detail: TStringField;
Query7: TQuery;
DataSource7: TDataSource;
Label14: TLabel;
LbStaticText2: TLbStaticText;
procedure FormActivate(Sender: TObject);
procedure LbButton2Click(Sender: TObject);
procedure To1Change(Sender: TObject);
procedure From1Change(Sender: TObject);
procedure SurnameChange(Sender: TObject);
procedure NameChange(Sender: TObject);
procedure RadioGroup1Click(Sender: TObject);
```

```
    procedure LbButton6Click(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure LbButton7Click(Sender: TObject);
    procedure LbButton8Click(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure custidChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;
  direction:string;

implementation

uses Unit1, Unit11, Unit8, Unit21, Unit10, Unit16;

{$R *.dfm}

procedure TForm4.FormActivate(Sender: TObject);
begin
LbStaticText1.Caption:=TimeToStr(Now);
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
from1.Items.Clear;
surname.Items.Clear;
```

```
custid.Items.Clear;

query2.First;
query3.First;
while not(query3.Eof) do begin
custid.Items.Add(query3.Fields[0].AsString);
query3.Next;
end;
while not(query2.Eof) do begin
from1.Items.Add(query2.Fields[0].AsString);
query2.Next;
end;
form4.DepartureDate.Date:=Date;
form4.ArrivalDate.Date:=Date;
end;

procedure TForm4.LbButton2Click(Sender: TObject);
begin
form1.Enabled:=true;
form1.Show;
form4.close;
end;

procedure TForm4.To1Change(Sender: TObject);
begin
cost.Items.Clear;
query2.First;
while not(query2.Eof) do begin
if (query2.Fields[0].AsString=from1.Text) and (query2.Fields[1].AsString=to1.Text)
then
begin
cost.Items.Add(query2.Fields[2].AsString)
end;
query2.Next;
```

```
end;
query7.Close;
query7.SQL.Clear;
query7.SQL.Add('select * from flighttable where from1 like
'+#39+(from1.Text)+'%'+#39'and to1 like'+#39+(to1.Text)+'%'+#39);
query7.Open;
LbStaticText2.Caption:=query7.Fields[3].AsString;
end;
procedure TForm4.From1Change(Sender: TObject);
begin
to1.Items.Clear;
query2.First;
while not(query2.Eof) do begin
//SHOWMESSAGE( query2.Fields[1].AsString);
if (query2.Fields[0].AsString=from1.Text) then
Begin
to1.Items.Add(query2.Fields[1].AsString);
end;
query2.Next;
end;

end;

procedure TForm4.SurnameChange(Sender: TObject);
begin
name.Items.Clear;
query3.First;
while not(query3.Eof) do begin
if (query3.Fields[0].AsString=custid.Text)and
(query3.Fields[2].AsString=surname.Text) then begin
Name.Items.Add(query3.Fields[1].AsString);
end;
query3.Next;
end;
```

```
end;

procedure TForm4.NameChange(Sender: TObject);
begin
gender.Items.Clear;
query3.First;
while not(query3.Eof) do begin
if (query3.Fields[2].AsString=surname.Text)and
(query3.Fields[1].AsString=name.Text)  then begin
Gender.Items.Add(query3.Fields[3].AsString);
end;
query3.Next;
end;

end;

procedure TForm4.RadioGroup1Click(Sender: TObject);
begin
if (form4.RadioGroup1.ItemIndex=0) then direction:='True';
if (form4.RadioGroup1.ItemIndex=1) then direction:='False';

end;

procedure save;
begin
   form4.Query1.Fields[1].AsString:=form4.From1.Text;
   form4.Query1.Fields[2].AsString:=form4.to1.Text;
   form4.Query1.Fields[3].AsString:=datetostr(form4.DepartureDate.Date);
   //SHOWMESSAGE(datetostr(form4.DepartureDate.Date));
   form4.Query1.Fields[4].AsString:=datetostr(form4.ArrivalDate.Date);
  // SHOWMESSAGE(datetostr(form4.ArrivalDate.Date));
   form4.Query1.Fields[5].AsString:=timetostr(form4.DepartureTime.Time);
 //     SHOWMESSAGE(timetostr(form4.DepartureTime.Time));
```

```
  form4.Query1.Fields[6].AsString:=timetostr(form4.ArrivalTime.Time);
//SHOWMESSAGE(timetostr(form4.ArrivalTime.Time));
  form4.Query1.Fields[7].AsString:=form4.Cost.Text;
  form4.Query1.Fields[8].AsString:=form4.Name.Text;
  form4.Query1.Fields[9].AsString:=form4.Surname.Text;
  form4.Query1.Fields[10].AsString:=form4.Gender.Text;
  form4.Query1.Fields[11].AsString:=datetostr(date);
  form4.Query1.Fields[12].AsString:=direction;
end;
procedure read;
begin
form4.From1.Text:=form4.Query1.Fields[1].AsString;
form4.to1.Text:=form4.Query1.Fields[2].AsString;
form4.Cost.Text:=form4.Query1.Fields[7].AsString;
form4.Name.Text:=form4.Query1.Fields[8].AsString;
form4.Surname.Text:=form4.Query1.Fields[9].AsString;
form4.Gender.Text:=form4.Query1.Fields[10].AsString;
form4.TodayDate1.Text:=datetostr(form4.Query1.Fields[11].AsDateTime);
direction:=form4.Query1.Fields[12].AsString;
form4.DepartureDate.Date:=strtodate(form4.Query1.Fields[3].AsString);
form4.ArrivalDate.Date:=strtodate(form4.Query1.Fields[4].AsString);
form4.DepartureTime.Time:=strtotime( form4.Query1.Fields[5].AsString);
form4.ArrivalTime.Time:=strtotime(form4.Query1.Fields[6].AsString);
end;

procedure clear;
begin
form4.From1.Text:='';
form4.to1.Text:='';
form4.Cost.Text:='';
form4.custid.Text:='';
form4.Name.Text:='';
form4.Surname.Text:='';
form4.Gender.Text:='';
```

```
form4.TodayDate1.Text:='';
direction:='';
form4.RadioGroup1.ItemIndex:=-1;
form4.CheckBox1.Checked:=false;
Form4.LbStaticText2.Caption:='';
end;

procedure TForm4.LbButton6Click(Sender: TObject);
begin
clear;
if direction='' then form4.RadioGroup1.ItemIndex:=-1;

end;

procedure TForm4.LbButton1Click(Sender: TObject);
begin
clear;
end;

function
bul(from1:string;to1:string;name:string;surname:string;ddate:string;adate:string):boolea
n;
begin
bul:=false;
form4.Query1.First;
while not form4.Query1.eof do
if (from1=form4.Query1.Fields[1].asstring)and
  (to1=form4.Query1.Fields[2].asstring)  and
  (name=form4.Query1.Fields[8].asstring) and
  (ddate=form4.Query1.Fields[3].asstring) and
  (adate=form4.Query1.Fields[4].asstring) and
  (surname=form4.Query1.Fields[9].asstring)then
begin
bul:=true;
```

```
exit;
end
else
form4.Query1.Next;
end;

procedure TForm4.LbButton5Click(Sender: TObject);
begin
if (query7.Fields[3].AsString='0') then begin
application.MessageBox('This flight is full!','Warning',16);
end
else begin
if not
bul(from1.Text,to1.Text,name.Text,surname.Text,datetostr(form4.DepartureDate.Date),
datetostr(form4.ArrivalDate.Date)) then begin
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
begin
if (form4.CheckBox1.Checked) and (form4.RadioGroup1.ItemIndex<>-1) then begin
if(timetostr(form4.DepartureTime.Time)<>timetostr(form4.ArrivalTime.Time)) then
begin
query1.Insert;
save;
query1.Post;
clear;
if form10.Query1.State in[dsEdit,dsInsert]then
begin
Form10.Query1.Fields[2].AsString:=Form4.Query1.Fields[9].AsString;
Form10.Query1.Fields[3].AsString:=Form4.Query1.Fields[8].AsString;
Form10.Query1.Fields[4].AsString:=Form4.Query1.Fields[1].AsString;
Form10.Query1.Fields[5].AsString:=Form4.Query1.Fields[2].AsString;
Form10.Query1.Fields[6].AsString:=Form4.Query1.Fields[12].AsString;
Form10.Query1.Fields[7].AsString:=Form4.Query1.Fields[11].AsString;
Form10.Query1.Fields[8].AsString:=Form4.Query1.Fields[7].AsString;
Form10.Query1.Post;
```

```
 end
else
 begin
  Form10.Query1.Insert;
  Form10.Query1.Fields[1].AsString:=Form16.Query1.Fields[0].AsString;
  Form10.Query1.Fields[2].AsString:=Form4.Query1.Fields[9].AsString;
  Form10.Query1.Fields[3].AsString:=Form4.Query1.Fields[8].AsString;
  Form10.Query1.Fields[4].AsString:=Form4.Query1.Fields[1].AsString;
  Form10.Query1.Fields[5].AsString:=Form4.Query1.Fields[2].AsString;
  Form10.Query1.Fields[6].AsString:=Form4.Query1.Fields[12].AsString;
  Form10.Query1.Fields[7].AsString:=Form4.Query1.Fields[11].AsString;
  Form10.Query1.Fields[8].AsString:=Form4.Query1.Fields[7].AsString;
  Form10.Query1.Post;
 end;
if (direction='') then form4.RadioGroup1.ItemIndex:=-1;
end else showmessage('The Departure Time and Arrival Time must be different.');
end;
end;
end;
query7.Edit;
query7.Fields[3].AsString:=inttostr((strtoint(LbStaticText2.Caption))-1);
query7.Post;
end;
end;

procedure TForm4.Timer1Timer(Sender: TObject);
begin
form4.TodayDate1.Text:=datetostr(date);
end;

procedure TForm4.LbButton7Click(Sender: TObject);
begin
//if not bul(from1.Text,to1.Text,name.Text,surname.Text) then begin
```

```
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
begin
query1.edit;
save;
query1.Post;
clear;
if (direction='') then form4.RadioGroup1.ItemIndex:=-1;
//end;
end;

end;

procedure TForm4.LbButton8Click(Sender: TObject);
begin
if
bul(from1.Text,to1.Text,name.Text,surname.Text,datetostr(form4.DepartureDate.Date),
datetostr(form4.ArrivalDate.Date)) then begin
if (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and (name.Text<>'') then
begin
query1.Delete;
clear;
if (direction='') then form4.RadioGroup1.ItemIndex:=-1;
end;
end;

end;

procedure TForm4.DBGrid1DblClick(Sender: TObject);
var
a,b,c,d,e,f:string;
begin
a:=dbgrid1.Fields[1].AsString;
b:=dbgrid1.Fields[2].AsString;
c:=dbgrid1.Fields[8].AsString;
```

```
d:=dbgrid1.Fields[9].AsString;
e:=dbgrid1.Fields[3].AsString;
f:=dbgrid1.Fields[4].AsString;

if  bul(a,b,c,d,e,f) then begin
clear;
read;
if (direction='True') then form4.RadioGroup1.ItemIndex:=0;
if (direction='False') then form4.RadioGroup1.ItemIndex:=1;
if (direction='') then form4.RadioGroup1.ItemIndex:=-1;
end;

end;

procedure TForm4.LbButton3Click(Sender: TObject);
begin
if (direction<>'') and (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and
(name.Text<>'')then  begin
//showmessage(direction);
if (direction='True') then begin
form4.Query4.SQL.Clear;
form4.Query4.SQL.Text:='select * from AirportTable where
City_id='+#39+(to1.Text)+#39;
form4.Query4.Open;
form4.Query5.SQL.Clear;
form4.Query5.SQL.Text:='select * from AirportTable where
City_id='+#39+(from1.Text)+#39;
form4.Query5.Open;
form11.QuickRep1.Preview;
end else begin
form4.Query4.SQL.Clear;
form4.Query4.SQL.Text:='select * from AirportTable where
City_id='+#39+(to1.Text)+#39;
form4.Query4.Open;
```

```
form21.QuickRep1.Preview;
end;
end;
end;

procedure TForm4.LbButton4Click(Sender: TObject);
begin
if (direction<>'') and (from1.Text<>'') and (to1.Text<>'') and (surname.Text<>'') and
(name.Text<>'')then  begin
//showmessage(direction);
if (direction='True') then begin
form4.Query4.SQL.Clear;
form4.Query4.SQL.Text:='select * from AirportTable where
City_id='+#39+(from1.Text)+#39;
form4.Query4.Open;
form4.Query5.SQL.Clear;
form4.Query5.SQL.Text:='select * from AirportTable where
City_id='+#39+(to1.Text)+#39;
form4.Query5.Open;
form11.QuickRep1.Print;
end else begin
form4.Query4.SQL.Clear;
form4.Query4.SQL.Text:='select * from AirportTable where
City_id='+#39+(from1.Text)+#39;
form4.Query4.Open;
form21.QuickRep1.Print;
end;
end;

end;

procedure TForm4.Timer2Timer(Sender: TObject);
begin
LbStaticText1.Caption := TimeToStr(Now);
```

```
end;

procedure TForm4.FormCreate(Sender: TObject);
begin
form4.Left:=1;
form4.Top:=46;
form4.ClientHeight:=658;
form4.ClientWidth:=706;
end;

procedure TForm4.custidChange(Sender: TObject);
begin
Surname.Items.Clear;
query3.First;
while not(query3.Eof) do begin
if (query3.Fields[0].AsString=custid.Text) then begin
Surname.Items.Add(query3.Fields[2].AsString);
end;
query3.Next;
end;
end;

end.
```

```
unit Unit5;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, DBCtrls, DB, DBTables, Grids, DBGrids, StdCtrls,
  Buttons, Mask, jpeg, LbButton ,BDE;
type
  TForm5 = class(TForm)
    GroupBox1: TGroupBox;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    GroupBox2: TGroupBox;
    RadioGroup1: TRadioGroup;
    Edit1: TEdit;
    DBMemo1: TDBMemo;
    DBComboBox1: TDBComboBox;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    LbButton6: TLbButton;
    LbButton7: TLbButton;
    LbButton8: TLbButton;
    Query1: TQuery;
    DBMemo2: TDBMemo;
    DataSource2: TDataSource;
    DBMemo3: TDBMemo;
    DBComboBox2: TDBComboBox;
    DataSource3: TDataSource;
```

```
LbButton9: TLbButton;
DBGrid1: TDBGrid;
DataSource1: TDataSource;
Image3: TImage;
Image4: TImage;
Label1: TLabel;
Label10: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label9: TLabel;
Bevel2: TBevel;
Image2: TImage;
Image5: TImage;
Image1: TImage;
Bevel1: TBevel;
Bevel3: TBevel;
Bevel4: TBevel;
procedure Button2Click(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure SpeedButton3Click(Sender: TObject);
procedure LbButton2Click(Sender: TObject);
procedure LbButton3Click(Sender: TObject);
procedure LbButton4Click(Sender: TObject);
procedure LbButton5Click(Sender: TObject);
procedure LbButton6Click(Sender: TObject);
procedure LbButton7Click(Sender: TObject);
procedure LbButton8Click(Sender: TObject);
procedure LbButton1Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure LbButton9Click(Sender: TObject);
```

```
  procedure Query1AfterPost(DataSet: TDataSet);
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form5: TForm5;

implementation

uses Unit2, Unit3, Unit4, Unit1;

{$R *.dfm}

procedure TForm5.Button2Click(Sender: TObject);
begin
edit1.Text:='';
radiogroup1.ItemIndex:=-1;
end;

procedure TForm5.FormActivate(Sender: TObject);
begin
edit1.Text:='';
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

procedure TForm5.SpeedButton3Click(Sender: TObject);
begin
query1.Edit;
dbedit2.SetFocus;
end;
```

```
procedure TForm5.LbButton2Click(Sender: TObject);
begin

dbedit2.Text:='';
dbedit3.Text:='';
dbedit4.Text:='';
dbedit5.Text:='';
dbedit2.SetFocus;
query1.Insert;
end;

procedure TForm5.LbButton3Click(Sender: TObject);
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
query1.Post;
end;
end;

procedure TForm5.LbButton4Click(Sender: TObject);
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
query1.edit;
dbedit2.SetFocus;
end;
end;
procedure TForm5.LbButton5Click(Sender: TObject);
begin
dbcombobox1.Text:='';
dbcombobox2.Text:='';
dbedit2.Text:='';
dbedit3.Text:='';
dbedit4.Text:='';
dbedit5.Text:='';
```

```
dbmemo1.Text:='';
dbmemo2.Text:='';
dbmemo3.Text:='';
end;

procedure TForm5.LbButton6Click(Sender: TObject);
var
a:word;
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
if query1.Eof = true then
ShowMessage('There is no data to delete.')
else
query1.Delete;
end;
end;
end;

procedure TForm5.LbButton7Click(Sender: TObject);
begin
edit1.Clear;
query1.Refresh;
radiogroup1.ItemIndex:=-1;
end;

procedure TForm5.LbButton8Click(Sender: TObject);
begin
if(radiogroup1.ItemIndex=0)then
begin
query1.Close;
```

```
query1.SQL.Clear;
query1.SQL.Add('select * from CustomerTable where Customer_id='+(edit1.Text));
query1.Open;
end;
if(radiogroup1.ItemIndex=1)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from CustomerTable where Name
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
if(radiogroup1.ItemIndex=2)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from CustomerTable where Surname
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
end;

procedure TForm5.LbButton1Click(Sender: TObject);
begin
form1.Enabled:=true;
form1.Show;
form5.Hide;
dbedit2.Text:='';
dbedit3.Text:='';
dbedit4.Text:='';
dbedit5.Text:='';
dbcombobox1.ItemIndex:=-1;
dbcombobox2.ItemIndex:=-1;
dbmemo1.Text:='';
```

```
dbmemo2.Text:='';
dbmemo3.Text:='';
end;

procedure TForm5.FormShow(Sender: TObject);
begin
  dbCombobox2.Clear;
  Form3.Query1.First;
 while not Form3.Query1.Eof do
  begin
     dbComboBox2.Items.Add(Form3.Query1.FieldByName('City_id').AsString);
     Form3.Query1.Next;
  end;
end;

procedure TForm5.LbButton9Click(Sender: TObject);
begin
query1.SQL.Clear;
query1.SQL.Add('select * from CustomerTable order by name ASC');
query1.Open;
end;

procedure TForm5.Query1AfterPost(DataSet: TDataSet);
begin
try
  DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
  On EDatabaseError do
    ShowMessage('Query Save Error!...');
end;
end;

procedure TForm5.FormCreate(Sender: TObject);
begin
```

```
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form5.Left:=1;
form5.Top:=46;
form5.ClientHeight:=630;
form5.ClientWidth:=755;
end;
end.

unit Unit6;

interface

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

type
  TForm6 = class(TForm)
    DBGrid1: TDBGrid;
    Query1: TQuery;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DataSource2: TDataSource;
    DBEdit4: TDBEdit;
    LbButton6: TLbButton;
    DataSource3: TDataSource;
    DataSource1: TDataSource;
    Image1: TImage;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    DataSource4: TDataSource;
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure LbButton6Click(Sender: TObject);
    procedure Query1AfterPost(DataSet: TDataSet);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form6: TForm6;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm6.LbButton2Click(Sender: TObject);
begin
dbedit2.Text:='';
dbedit3.Text:='';
dbedit2.SetFocus;
Query1.Insert;
```

```
end;

procedure TForm6.LbButton3Click(Sender: TObject);
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
Query1.Post;
end;
end;

procedure TForm6.LbButton4Click(Sender: TObject);
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
Query1.Edit;
dbedit2.SetFocus;
end;
end;

procedure TForm6.LbButton5Click(Sender: TObject);
var
a:word;
begin
if (dbedit3.Text<>'') and (dbedit2.Text<>'')then begin
a:=application.MessageBox('Your Entry Will Be Deleted.Are You Sure?','Warning',36);
if(a=IDYES)then
begin
Query1.Delete;
end;
end;
end;

procedure TForm6.LbButton1Click(Sender: TObject);
begin
form6.close;
form1.Enabled:=true;
```

```
Form1.show;
end;

procedure TForm6.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];

end;

procedure TForm6.LbButton6Click(Sender: TObject);
begin
form6.DBEdit2.Text:='';
form6.DBEdit3.Text:='';
form6.DBEdit4.Text:='';
end;

procedure TForm6.Query1AfterPost(DataSet: TDataSet);
begin
try
 DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
 On EDatabaseError do
  ShowMessage('Query Save Error!...');
end;
end;
procedure TForm6.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form6.Left:=1;
form6.Top:=46;
form6.ClientHeight:=318;
form6.ClientWidth:=381;
end;
end.
```

```
unit Unit7;

interface

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

type
  TForm7 = class(TForm)
    DataSource1: TDataSource;
    Query1: TQuery;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    DataSource2: TDataSource;
    DBGrid1: TDBGrid;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    procedure LbButton5Click(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Query1AfterPost(DataSet: TDataSet);
    procedure FormCreate(Sender: TObject);
  private
```

```
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form7: TForm7;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm7.LbButton5Click(Sender: TObject);
begin
form1.Enabled:=true;
Form1.Show;
form7.close;
end;

procedure TForm7.LbButton1Click(Sender: TObject);
begin
dbedit1.Text:='';
dbedit2.Text:='';
dbedit1.SetFocus;
Query1.Insert;
end;

procedure TForm7.LbButton2Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
Query1.Post;
end;
```

```
end;

procedure TForm7.LbButton3Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
Query1.Edit;
dbedit1.SetFocus;
end;
end;

procedure TForm7.LbButton4Click(Sender: TObject);
var
a:word;
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
a:=application.MessageBox('Your Entry Will Be Deleted.Are You Sure?','Warning',36);
if(a=IDYES)then
begin
Query1.Delete;
end;
end;
end;

procedure TForm7.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

procedure TForm7.Query1AfterPost(DataSet: TDataSet);
begin
try
  DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
  On EDatabaseError do
```

```
  ShowMessage('Query Save Error!...');
end;
end;

procedure TForm7.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form7.Left:=1;
form7.Top:=46;
form7.ClientHeight:=295;
form7.ClientWidth:=385;
end;

end.

unit Unit8;

interface

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

type
 TForm8 = class(TForm)
  DataSource1: TDataSource;
  Query1: TQuery;
  DataSource2: TDataSource;
  DataSource3: TDataSource;
  LbButton1: TLbButton;
  DBEdit1: TDBEdit;
  DBEdit2: TDBEdit;
  StaticText1: TStaticText;
```

```
RadioGroup1: TRadioGroup;
DBGrid1: TDBGrid;
LbButton2: TLbButton;
LbButton3: TLbButton;
Edit1: TEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit6: TDBEdit;
DBEdit7: TDBEdit;
LbButton4: TLbButton;
LbButton5: TLbButton;
LbButton6: TLbButton;
LbButton7: TLbButton;
Image2: TImage;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label6: TLabel;
Image1: TImage;
Bevel1: TBevel;
Bevel2: TBevel;
LbButton8: TLbButton;
Image3: TImage;
Bevel3: TBevel;
procedure FormActivate(Sender: TObject);
procedure LbButton1Click(Sender: TObject);
procedure LbButton4Click(Sender: TObject);
procedure LbButton5Click(Sender: TObject);
procedure LbButton6Click(Sender: TObject);
procedure LbButton7Click(Sender: TObject);
procedure Query2AfterPost(DataSet: TDataSet);
procedure Query1AfterPost(DataSet: TDataSet);
procedure FormCreate(Sender: TObject);
```

```
  procedure DBEdit1Change(Sender: TObject);
  procedure LbButton2Click(Sender: TObject);
  procedure LbButton3Click(Sender: TObject);
  procedure LbButton8Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form8: TForm8;

implementation

uses Unit1,Unit6,Unit3;

{$R *.dfm}

procedure TForm8.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
Edit1.Text:='';
end;

procedure TForm8.LbButton1Click(Sender: TObject);
begin
form8.Hide;
form1.Enabled:=true;
form1.show;
end;

procedure TForm8.LbButton4Click(Sender: TObject);
begin
```

```
dbedit1.Text:='';
dbedit2.Text:='';
dbedit3.Text:='';
dbedit4.Text:='';
dbedit6.Text:='';
dbedit7.Text:='';
dbedit1.SetFocus;
query1.Insert;
end;

procedure TForm8.LbButton5Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
query1.edit;
dbedit1.SetFocus;
end;
end;

procedure TForm8.LbButton6Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
if Query1.State in[dsEdit,dsInsert]then
 begin
  query1.Post;
  end;
 end;
end;

procedure TForm8.LbButton7Click(Sender: TObject);
var
a:word;
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'')then begin
```

```
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
query1.Delete;
end;
end;
end;

procedure TForm8.Query2AfterPost(DataSet: TDataSet);
begin
try
 DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
 On EDatabaseError do
  ShowMessage('Query Save Error!...');
end;
end;

procedure TForm8.Query1AfterPost(DataSet: TDataSet);
begin
try
 DBISaveChanges((DataSet As TBDEDataSet).Handle)
except
 On EDatabaseError do
  ShowMessage('Query Save Error!...');
end;
end;

procedure TForm8.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
LbButton6.Enabled:=false;
LbButton7.Enabled:=false;
```

```
form8.Left:=1;
form8.Top:=46;
form8.ClientHeight:=660;
form8.ClientWidth:=818;
end;

procedure TForm8.DBEdit1Change(Sender: TObject);
begin
if(dbedit1.Text<>'')then
begin
Lbbutton6.Enabled:=true;
LbButton7.Enabled:=true;
end
else if(dbedit1.Text='')then
begin
Lbbutton6.Enabled:=false;
LbButton7.Enabled:=false;
end;
end;

procedure TForm8.LbButton2Click(Sender: TObject);
begin
if(radiogroup1.ItemIndex=0)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from AirportTable where Airport_Code
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
if(radiogroup1.ItemIndex=1)then
begin
query1.Close;
query1.SQL.Clear;
```

```
query1.SQL.Add('select * from AirportTable where Country_Code
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
if(radiogroup1.ItemIndex=2)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from AirportTable where TlfCode
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
if(radiogroup1.ItemIndex=3)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from AirportTable where Airport_name
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
if(radiogroup1.ItemIndex=4)then
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from AirportTable where City_id
like'+#39+(edit1.Text)+'%'+#39);
query1.Open;
end;
end;

procedure TForm8.LbButton3Click(Sender: TObject);
begin
edit1.Text:='';
RadioGroup1.ItemIndex:=-1;
```

```
end;

procedure TForm8.LbButton8Click(Sender: TObject);
begin
if Query1.State in[dsEdit,dsInsert]then
 begin
  query1.Cancel;
 end;
end;

end.
unit Unit9;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, LbButton, jpeg, ExtCtrls, DB, DBTables, Grids, DBGrids, StdCtrls;

type
  TForm9 = class(TForm)
    LbButton1: TLbButton;
    Image1: TImage;
    DBGrid1: TDBGrid;
    Query1: TQuery;
    DataSource1: TDataSource;
    citycode: TComboBox;
    cityname: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    detail: TMemo;
    Label3: TLabel;
    Image2: TImage;
    Bevel1: TBevel;
```

```
    DataSource2: TDataSource;
    Query2: TQuery;
    Query2City_id: TStringField;
    Query2Country_Code: TStringField;
    Query2City_Name: TStringField;
    Query1City_Code: TStringField;
    Query1City_Name: TStringField;
    Query1Detail: TStringField;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    procedure LbButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure citycodeChange(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form9: TForm9;

implementation

uses Unit1;
```

```
{$R *.dfm}

procedure TForm9.LbButton1Click(Sender: TObject);
begin
form1.Enabled:=true;
form1.show;
end;

procedure TForm9.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form9.Left:=1;
form9.Top:=46;
detail.Text:='';

end;

procedure TForm9.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];

query2.First;
while not(query2.Eof) do begin
citycode.Items.Add(query2.Fields[0].AsString);
query2.Next;
end;

end;

procedure TForm9.citycodeChange(Sender: TObject);
begin
cityname.Items.Clear;
query2.First;
while not(query2.Eof) do begin
```

```
if (query2.Fields[0].AsString=citycode.Text) then
 Begin
  cityname.Items.Add(query2.Fields[2].AsString);
 end;
  query2.Next;
 end;
end;

procedure save;
begin
form9.Query1.Fields[0].AsString:=form9.citycode.Text;
form9.Query1.Fields[1].AsString:=form9.cityname.Text;
form9.Query1.Fields[2].AsString:=form9.detail.Text;
end;

procedure read;
begin
form9.citycode.Text:=form9.Query1.Fields[0].AsString;
form9.cityname.Text:=form9.Query1.Fields[1].AsString;
form9.detail.Text:=form9.Query1.Fields[2].AsString;
end;

procedure clear;
begin
Form9.citycode.Text:='';
Form9.cityname.Text:='';
Form9.detail.Text:='';
end;

function bul(citycode:string;cityname:string;detail:string):boolean;
begin
bul:=false;
form9.Query1.First;
while not form9.Query1.eof do
```

```
if (citycode=form9.Query1.Fields[0].asstring)and
  (cityname=form9.Query1.Fields[1].asstring)  and
  (detail=form9.Query1.Fields[2].asstring)then
begin
bul:=true;
exit;
end
else
form9.Query1.Next;
end;

procedure TForm9.LbButton2Click(Sender: TObject);
begin
if not bul(citycode.Text,cityname.Text,detail.Text) then
 begin
 if (citycode.Text<>'') and (cityname.Text<>'') and (detail.Text<>'') then
  begin
     Query1.Insert;
     save;
     query1.Post;
     clear;
     Query1.Refresh;
  end;
 end;
end;

procedure TForm9.LbButton4Click(Sender: TObject);
begin
clear;
end;

procedure TForm9.LbButton3Click(Sender: TObject);
begin
if (citycode.Text<>'') and (cityname.Text<>'') and (detail.Text<>'') then
```

```
  end;
 end;

 end.


 unit Unit10;

 interface

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

 type
  TForm10 = class(TForm)
   DBGrid4: TDBGrid;
   LbButton1: TLbButton;
   Image1: TImage;
   Label1: TLabel;
   Edit1: TEdit;
   DataSource1: TDataSource;
   Query1: TQuery;
   Query1TicketId: TAutoIncField;
   Query1User: TStringField;
   Query1Cust_Surname: TStringField;
   Query1Cust_Name: TStringField;
   Query1From1: TStringField;
   Query1To1: TStringField;
   Query1RoundTrip: TBooleanField;
   Query1SellDate: TDateField;
   Query1TicketPrice: TStringField;
   Label2: TLabel;
```

```
    MaskEdit1: TMaskEdit;
    MaskEdit2: TMaskEdit;
    LbButton2: TLbButton;
    Label3: TLabel;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    procedure LbButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Query1AfterPost(DataSet: TDataSet);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form10: TForm10;

implementation

uses Unit1, Unit15;

{$R *.dfm}

procedure TForm10.LbButton1Click(Sender: TObject);
begin
form1.Enabled:=true;
form1.show;
end;
```

```
procedure TForm10.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form10.Left:=1;
form10.Top:=46;
form10.ClientHeight:=481;
form10.ClientWidth:=834;
end;

procedure TForm10.Query1AfterPost(DataSet: TDataSet);
var
sum : Integer;
begin
 sum:=0;
 query1.First;
 while not Query1.Eof do
  begin
   sum:=sum+Query1.Fields[8].AsInteger;
   Query1.Next;
  end;
  Edit1.Text:=IntToStr(sum);
end;

procedure TForm10.LbButton2Click(Sender: TObject);
var
sum : Integer;
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from tickettable where SellDate between
'+#39+(MaskEdit1.Text)+#39' and '+#39+(MaskEdit2.Text)+#39);
query1.Open;
 sum:=0;
 query1.First;
```

```
 while not Query1.Eof do
  begin
   sum:=sum+Query1.Fields[8].AsInteger;
   Query1.Next;
  end;
  Edit1.Text:=IntToStr(sum);
end;

procedure TForm10.LbButton3Click(Sender: TObject);
begin
form15.QuickRep1.Preview;
end;

procedure TForm10.LbButton4Click(Sender: TObject);
var
sum : Integer;
begin
query1.Close;
query1.SQL.Clear;
query1.SQL.Add('select * from tickettable order by TicketId ASC');
query1.Open;
 sum:=0;
 query1.First;
 while not Query1.Eof do
  begin
   sum:=sum+Query1.Fields[8].AsInteger;
   Query1.Next;
  end;
  Edit1.Text:=IntToStr(sum);
end;

procedure TForm10.FormActivate(Sender: TObject);
begin
edit1.Text:='';
```

```
end;

end.

unit Unit11;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, QuickRpt, QRCtrls, ExtCtrls, DB, DBTables;

type
  TForm11 = class(TForm)
    QuickRep1: TQuickRep;
    PageHeaderBand1: TQRBand;
    DetailBand1: TQRBand;
    QRDBText1: TQRDBText;
    QRLabel3: TQRLabel;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRLabel1: TQRLabel;
    QRDBText4: TQRDBText;
    QRLabel2: TQRLabel;
    QRDBText5: TQRDBText;
    QRLabel4: TQRLabel;
    QRDBText6: TQRDBText;
    QRLabel5: TQRLabel;
    QRDBText7: TQRDBText;
    QRLabel6: TQRLabel;
    QRDBText8: TQRDBText;
    QRLabel7: TQRLabel;
    QRDBText9: TQRDBText;
    QRLabel8: TQRLabel;
```

```
    QRLabel9: TQRLabel;
    QRLabel12: TQRLabel;
    QRLabel13: TQRLabel;
    QRDBText14: TQRDBText;
    QRLabel11: TQRLabel;
    QRLabel14: TQRLabel;
    QRLabel16: TQRLabel;
    QRDBText12: TQRDBText;
    QRLabel17: TQRLabel;
    QRDBText16: TQRDBText;
    QRDBText18: TQRDBText;
    QRDBText19: TQRDBText;
    QRDBText23: TQRDBText;
    QRDBText22: TQRDBText;
    QRDBText10: TQRDBText;
    QRDBText11: TQRDBText;
    QRLabel10: TQRLabel;
    QRDBText15: TQRDBText;
    QRDBText13: TQRDBText;
    QRDBText17: TQRDBText;
    QRLabel15: TQRLabel;
    QRBand1: TQRBand;
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form11: TForm11;

implementation
```

```
uses unit4;
{$R *.dfm}

procedure TForm11.FormActivate(Sender: TObject);
begin
form11.Color:=clGradientActiveCaption;
end;

procedure TForm11.FormCreate(Sender: TObject);
begin
form11.Left:=1;
form11.Top:=46;
end;

end.

unit Unit12;

interface

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

type
 TForm12 = class(TForm)
  LbButton1: TLbButton;
  Image1: TImage;
  Image2: TImage;
  Bevel1: TBevel;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
```

```
    Label4: TLabel;
    Label5: TLabel;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    Timer1: TTimer;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form12: TForm12;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm12.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form12.Left:=1;
form12.Top:=46;
form12.ClientHeight:=324;
form12.ClientWidth:=393;
end;
```

```
procedure TForm12.FormActivate(Sender: TObject);
begin
form12.Color:=clGradientActiveCaption;
end;

procedure TForm12.LbButton1Click(Sender: TObject);
begin
form1.show;
form1.Enabled:=true;
end;

procedure TForm12.LbButton2Click(Sender: TObject);
begin
Application.HelpCommand(HELP_INDEX,0);
end;

procedure TForm12.Timer1Timer(Sender: TObject);
begin
form12.LbButton3.Caption:=TimeToStr(Now);
end;

end.


unit Unit13;

interface

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

```
type
  TForm13 = class(TForm)
    LbButton1: TLbButton;
    Query1: TQuery;
    DBGrid1: TDBGrid;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBComboBox1: TDBComboBox;
    DBComboBox2: TDBComboBox;
    DataSource1: TDataSource;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    LbButton6: TLbButton;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    LbButton7: TLbButton;
    procedure LbButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure LbButton6Click(Sender: TObject);
    procedure LbButton7Click(Sender: TObject);
  private
```

```
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form13: TForm13;

implementation

uses Unit1, Unit14;

{$R *.dfm}

procedure TForm13.LbButton1Click(Sender: TObject);
begin
form1.Enabled:=true;
form13.Hide;
form1.show;
end;

procedure TForm13.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu];
form13.Left:=1;
form13.Top:=46;
form13.ClientHeight:=525;
form13.ClientWidth:=669;
end;

procedure TForm13.LbButton2Click(Sender: TObject);
begin
DBEdit1.Text:='';
DBEdit2.Text:='';
```

```
DBEdit3.Text:='';
DBEdit4.Text:='';
DBComboBox1.ItemIndex:=-1;
DBComboBox2.ItemIndex:=-1;
Query1.Insert;
DBEdit1.SetFocus;
end;

procedure TForm13.LbButton3Click(Sender: TObject);
begin
Query1.Edit;
end;

procedure TForm13.LbButton4Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query1.Post;
end;
end;

procedure TForm13.LbButton5Click(Sender: TObject);
begin
DBEdit1.Text:='';
DBEdit2.Text:='';
DBEdit3.Text:='';
DBEdit4.Text:='';
DBComboBox1.ItemIndex:=-1;
DBComboBox2.ItemIndex:=-1;
end;

procedure TForm13.LbButton6Click(Sender: TObject);
var
a:word;
begin
```

```
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
query1.Delete;
end;
end;
end;

procedure TForm13.LbButton7Click(Sender: TObject);
begin
Form14.show;
form13.Enabled:=false;
end;

end.

unit Unit14;

interface

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

type
 TForm14 = class(TForm)
  LbButton1: TLbButton;
  LbButton2: TLbButton;
  LbButton3: TLbButton;
  LbButton4: TLbButton;
  LbButton5: TLbButton;
```

```
Query1: TQuery;
Query2: TQuery;
DBGrid1: TDBGrid;
DataSource1: TDataSource;
DataSource2: TDataSource;
DBGrid2: TDBGrid;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
Image1: TImage;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
RadioGroup1: TRadioGroup;
Bevel1: TBevel;
LbButton6: TLbButton;
LbButton7: TLbButton;
LbButton8: TLbButton;
LbButton9: TLbButton;
LbButton10: TLbButton;
LbButton11: TLbButton;
LbButton12: TLbButton;
LbButton13: TLbButton;
procedure LbButton1Click(Sender: TObject);
procedure LbButton2Click(Sender: TObject);
procedure LbButton4Click(Sender: TObject);
procedure LbButton3Click(Sender: TObject);
procedure LbButton5Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure RadioGroup1Click(Sender: TObject);
procedure LbButton6Click(Sender: TObject);
procedure LbButton10Click(Sender: TObject);
```

```
  procedure LbButton7Click(Sender: TObject);
  procedure LbButton8Click(Sender: TObject);
  procedure LbButton9Click(Sender: TObject);
  procedure LbButton11Click(Sender: TObject);
  procedure LbButton12Click(Sender: TObject);
  procedure LbButton13Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form14: TForm14;

implementation

uses Unit13;

{$R *.dfm}

procedure TForm14.LbButton1Click(Sender: TObject);
begin
Form13.Enabled:=true;
form14.Hide;
end;

procedure TForm14.LbButton2Click(Sender: TObject);
begin
DBEdit1.Text:='';
DBEdit2.Text:='';
DBEdit3.Text:='';
DBEdit4.Text:='';
DBEdit1.SetFocus;
```

```
Query1.Insert;
Query2.Insert;
end;

procedure TForm14.LbButton4Click(Sender: TObject);
begin
Query1.Post;
Query2.Post;
end;

procedure TForm14.LbButton3Click(Sender: TObject);
begin
Query1.edit;
Query2.edit;
DBEdit1.SetFocus;
end;

procedure TForm14.LbButton5Click(Sender: TObject);
var
a:word;
begin
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
query1.Delete;
query2.Delete;
end;
end;

procedure TForm14.FormCreate(Sender: TObject);
begin
form14.Left:=1;
form14.Top:=46;
```

```
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form14.ClientHeight:=419;
form14.ClientWidth:=441;
end;

procedure TForm14.RadioGroup1Click(Sender: TObject);
begin
if(radiogroup1.ItemIndex=0)then
begin
LbButton2.Show;
LbButton3.Show;
LbButton4.Show;
LbButton5.Show;
LbButton6.Hide;
LbButton7.Hide;
LbButton8.Hide;
LbButton9.Hide;
LbButton10.Hide;
LbButton11.Hide;
LbButton12.Hide;
LbButton13.Hide;
Label1.Enabled:=True;
Label2.Enabled:=True;
Label3.Enabled:=True;
Label4.Enabled:=True;
DBEdit1.Enabled:=True;
DBEdit2.Enabled:=True;
DBEdit3.Enabled:=True;
DBEdit4.Enabled:=True;
end;
if(radiogroup1.ItemIndex=1)then
begin
LbButton2.Hide;
LbButton3.Hide;
```

```
LbButton4.Hide;
LbButton5.Hide;
LbButton10.Hide;
LbButton11.Hide;
LbButton12.Hide;
LbButton13.Hide;
LbButton6.Show;
LbButton7.Show;
LbButton8.Show;
LbButton9.Show;
Label1.Enabled:=True;
Label2.Enabled:=True;
Label3.Enabled:=False;
Label4.Enabled:=False;
DBEdit1.Enabled:=True;
DBEdit2.Enabled:=True;
DBEdit3.Enabled:=False;
DBEdit4.Enabled:=False;
end;
if(radiogroup1.ItemIndex=2)then
begin
LbButton2.Hide;
LbButton3.Hide;
LbButton4.Hide;
LbButton5.Hide;
LbButton6.Hide;
LbButton7.Hide;
LbButton8.Hide;
LbButton9.Hide;
LbButton10.Show;
LbButton11.Show;
LbButton12.Show;
LbButton13.Show;
Label1.Enabled:=False;
```

```
Label2.Enabled:=False;
Label3.Enabled:=True;
Label4.Enabled:=True;
DBEdit1.Enabled:=False;
DBEdit2.Enabled:=False;
DBEdit3.Enabled:=true;
DBEdit4.Enabled:=true;
end;
end;

procedure TForm14.LbButton6Click(Sender: TObject);
begin
DBEdit1.Text:='';
DBEdit2.Text:='';
DBEdit1.SetFocus;
Query1.Insert;
end;

procedure TForm14.LbButton10Click(Sender: TObject);
begin
DBEdit3.Text:='';
DBEdit4.Text:='';
DBEdit3.SetFocus;
Query2.Insert;
end;

procedure TForm14.LbButton7Click(Sender: TObject);
begin
Query1.Post;
end;

procedure TForm14.LbButton8Click(Sender: TObject);
begin
Query1.edit;
```

```
end;

procedure TForm14.LbButton9Click(Sender: TObject);
var
a:word;
begin
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
query1.Delete;
query2.Delete;
end;
end;

procedure TForm14.LbButton11Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query2.Post;
end;
end;
procedure TForm14.LbButton12Click(Sender: TObject);
begin
Query2.Edit;
end;

procedure TForm14.LbButton13Click(Sender: TObject);
var
a:word;
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
a:=application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
```

```
begin

query1.Delete;
query2.Delete;
end;
end;
end;
end.

unit Unit15;

interface

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

type
  TForm15 = class(TForm)
    QuickRep1: TQuickRep;
    ColumnHeaderBand1: TQRBand;
    DetailBand1: TQRBand;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRDBText4: TQRDBText;
    QRDBText5: TQRDBText;
    QRDBText6: TQRDBText;
    QRDBText7: TQRDBText;
    QRDBText8: TQRDBText;
    QRDBText9: TQRDBText;
    QRDBText10: TQRDBText;
    QRDBText11: TQRDBText;
    QRLabel1: TQRLabel;
```

```
  QRLabel2: TQRLabel;
  QRLabel3: TQRLabel;
  QRLabel4: TQRLabel;
  QRLabel5: TQRLabel;
  QRLabel6: TQRLabel;
  QRLabel7: TQRLabel;
  QRLabel8: TQRLabel;
  procedure FormActivate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form15: TForm15;

implementation

{$R *.dfm}

procedure TForm15.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

end.

unit Unit16;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
Dialogs, StdCtrls, LbButton, DB, DBTables, jpeg, ExtCtrls;

type
 TForm16 = class(TForm)
  LbButton1: TLbButton;
  LbButton2: TLbButton;
  Edit1: TEdit;
  Edit2: TEdit;
  LbButton3: TLbButton;
  Query1: TQuery;
  DataSource1: TDataSource;
  Image1: TImage;
  Label1: TLabel;
  Label2: TLabel;
  procedure LbButton1Click(Sender: TObject);
  procedure LbButton2Click(Sender: TObject);
  procedure LbButton3Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure FormActivate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form16: TForm16;

implementation

uses Unit1, Unit18, Unit10;

{$R *.dfm}
```

```
function bul(a:string;b:string):boolean;
begin
bul:=false;
form16.Query1.First;
while not form16.Query1.eof do
if (a=form16.Query1.Fields[0].asstring)and (b=form16.Query1.Fields[1].asstring)then
begin
bul:=true;
exit;
end
else
form16.Query1.Next;
end;

procedure TForm16.LbButton1Click(Sender: TObject);
begin
if bul(edit1.Text,edit2.Text) then begin
        form1.show;
        form16.Visible:=false;
        form10.Query1.Insert;
        form10.Query1.Fields[1]:=form16.Query1.Fields[0];
        end
        else
        application.MessageBox('Wrong password or username','Warning',16);
end;

procedure TForm16.LbButton2Click(Sender: TObject);
begin
form16.Close;
end;

procedure TForm16.LbButton3Click(Sender: TObject);
begin
form18.show;
```

```
form16.enabled:=false;
end;

procedure TForm16.FormCreate(Sender: TObject);
begin
form16.ClientHeight:=164;
form16.ClientWidth:=267;
end;

procedure TForm16.FormActivate(Sender: TObject);
begin
edit1.SetFocus;
end;

end.

unit Unit17;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, StdCtrls, Buttons, DB, DBTables, jpeg, ExtCtrls,
  LbButton;

type
  TForm17 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Query1: TQuery;
    DataSource1: TDataSource;
    Edit3: TEdit;
    Edit4: TEdit;
    DBGrid1: TDBGrid;
```

```
    Image1: TImage;
    Label1: TLabel;
    Label4: TLabel;
    Label3: TLabel;
    Label2: TLabel;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    procedure DBGrid1DblClick(Sender: TObject);
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form17: TForm17;

implementation

uses Unit1;

{$R *.dfm}

PROCEDURE save;
begin
with form17 do begin
query1.Fields[0].AsString:=edit1.Text;
```

```
query1.Fields[1].AsString:=edit3.Text;
query1.Fields[2].AsString:=edit4.Text;
end;
end;
PROCEDURE read;
begin
with form17 do begin
edit1.Text:=query1.Fields[0].AsString;
edit4.Text:=query1.Fields[2].AsString;
end;
end;
PROCEDURE clear;
begin
with form17 do begin
edit1.Text:='';
edit3.Text:='';
edit4.Text:='';
edit2.Text:='';
end;
end;
function bul(a:string):boolean;
begin
bul:=false;
form17.Query1.First;
while not form17.Query1.eof do
if (a=form17.Query1.Fields[0].asstring)then
begin
bul:=true;
exit;
end
else
form17.Query1.Next;
end;
procedure TForm17.LbButton2Click(Sender: TObject);
```

```
begin
if(edit1.Text<>'')then begin
if bul(edit1.Text) then begin
query1.Delete;
clear;
query1.Refresh;
end;
end;
end;

procedure TForm17.LbButton3Click(Sender: TObject);
begin
if(edit1.Text<>'')and(edit2.text<>'')and(edit3.text<>'')and(edit3.text<>'') then begin
  query1.Edit;
  save;
  query1.Post;
  clear;
  query1.Refresh;
end;
end;

procedure TForm17.LbButton4Click(Sender: TObject);
begin
form17.Hide;
form1.Show;
form1.enabled:=true;
end;

procedure TForm17.DBGrid1DblClick(Sender: TObject);
begin
if bul(dbgrid1.Fields[0].AsString) then read;
end;

procedure TForm17.LbButton1Click(Sender: TObject);
```

```
begin
if(edit1.Text<>'')and(edit2.text<>'')and(edit3.text<>'')and(edit4.text<>'') then begin
if not bul(edit1.Text) then begin
if(edit2.Text=edit3.Text) then begin
query1.Insert;
save;
query1.Post;
clear;
query1.Refresh;
end;
end;
end;
end;
procedure TForm17.FormCreate(Sender: TObject);
begin
form17.ClientHeight:=169;
form17.ClientWidth:=470;
end;

end.

unit Unit18;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, ExtCtrls, DB, DBTables, jpeg, LbButton;

type
TForm18 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Query1: TQuery;
```

```
    DataSource1: TDataSource;
    Panel1: TPanel;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form18: TForm18;

implementation

uses Unit16;

{$R *.dfm}
function bul(a:string;b:string):boolean;
begin
bul:=false;
form18.Query1.First;
while not form18.Query1.eof do
if (a=form18.Query1.Fields[0].asstring)and (b=form18.Query1.Fields[2].asstring)then
begin
bul:=true;
exit;
```

```
end
else
form18.Query1.Next;
end;
procedure TForm18.LbButton1Click(Sender: TObject);
begin
if bul(edit1.Text,edit2.Text) then
panel1.Caption:=query1.Fields[1].AsString
else
application.MessageBox('Wrong username and Secret','Warning',16);
end;

procedure TForm18.LbButton2Click(Sender: TObject);
begin
form18.Close;
form16.enabled:=true;
form16.Show;
end;

procedure TForm18.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

procedure TForm18.FormCreate(Sender: TObject);
begin
form18.ClientHeight:=120;
form18.ClientWidth:=510;
end;

end.

unit Unit19;
```

```
interface

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

type
  TForm19 = class(TForm)
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit7: TDBEdit;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton5: TLbButton;
    Image1: TImage;
    Label9: TLabel;
    Label8: TLabel;
    Label7: TLabel;
    Label6: TLabel;
    Label5: TLabel;
    Label11: TLabel;
    Label10: TLabel;
    DataSource1: TDataSource;
    Query1: TQuery;
    DBEdit6: TDBEdit;
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form19: TForm19;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm19.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form19.ClientHeight:=382;
form19.ClientWidth:=286;
form19.Left:=432;
form19.Top:=152;
end;

procedure TForm19.FormActivate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
end;

procedure TForm19.LbButton5Click(Sender: TObject);
begin
Form19.close;
form1.Show;
Form1.Enabled:=true;
end;
```

```
procedure TForm19.LbButton3Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
if Query1.State in[dsEdit,dsInsert]then
Query1.Post;
end;
end;

procedure TForm19.LbButton2Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query1.edit;
DBEdit1.SetFocus;
end;
end;

end.

unit Unit20;

interface

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

type
 TForm20 = class(TForm)
  DataSource1: TDataSource;
  DBEdit2: TDBEdit;
  DBEdit3: TDBEdit;
  DBEdit4: TDBEdit;
```

```
    DBEdit5: TDBEdit;
    LbButton1: TLbButton;
    LbButton2: TLbButton;
    LbButton3: TLbButton;
    LbButton4: TLbButton;
    LbButton5: TLbButton;
    Image1: TImage;
    Label5: TLabel;
    Label4: TLabel;
    Label3: TLabel;
    Label2: TLabel;
    Label1: TLabel;
    DataSource2: TDataSource;
    Image2: TImage;
    Image3: TImage;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Query1: TQuery;
    DBGrid1: TDBGrid;
    DBEdit1: TDBEdit;
    procedure LbButton1Click(Sender: TObject);
    procedure LbButton2Click(Sender: TObject);
    procedure LbButton3Click(Sender: TObject);
    procedure LbButton4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LbButton5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form20: TForm20;
```

```
implementation

uses Unit6, Unit1;

{$R *.dfm}

procedure TForm20.LbButton1Click(Sender: TObject);
begin
DBEdit1.Text:='';
DBEdit2.Text:='';
DBEdit3.Text:='';
DBEdit4.Text:='';
DBEdit5.Text:='';
DBEdit1.SetFocus;
Query1.Insert;
end;

procedure TForm20.LbButton2Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query1.Edit;
DBEdit1.SetFocus;
end;
end;

procedure TForm20.LbButton3Click(Sender: TObject);
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
Query1.Post;
Query1.Refresh;
end;
end;
procedure TForm20.LbButton4Click(Sender: TObject);
```

```
var
a:word;
begin
if (dbedit1.Text<>'') and (dbedit2.Text<>'') then begin
a:=Application.MessageBox('Your Information Will Be Deleted.Are You
Sure?','WARNING',36);
if(a=IDYES)then
begin
Query1.Delete;
Query1.Refresh;
end;
end;
end;
procedure TForm20.FormCreate(Sender: TObject);
begin
borderIcons:=borderIcons-[bisystemmenu,bimaximize,biminimize];
form20.Left:=1;
form20.Top:=46;
form20.ClientHeight:=344;
form20.ClientWidth:=486;
end;

procedure TForm20.LbButton5Click(Sender: TObject);
begin
Form1.Enabled:=true;
Form1.Show;
Form20.Hide;
end;

end.

unit Unit21;

interface
```

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, QRCtrls, QuickRpt, ExtCtrls;

type
  TForm21 = class(TForm)
    QuickRep1: TQuickRep;
    PageHeaderBand1: TQRBand;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel12: TQRLabel;
    QRDBText18: TQRDBText;
    QRDBText19: TQRDBText;
    DetailBand1: TQRBand;
    QRDBText1: TQRDBText;
    QRLabel3: TQRLabel;
    QRDBText4: TQRDBText;
    QRDBText5: TQRDBText;
    QRLabel4: TQRLabel;
    QRDBText6: TQRDBText;
    QRLabel5: TQRLabel;
    QRDBText7: TQRDBText;
    QRLabel6: TQRLabel;
    QRDBText8: TQRDBText;
    QRLabel7: TQRLabel;
    QRDBText9: TQRDBText;
    QRLabel8: TQRLabel;
    QRLabel9: TQRLabel;
    QRLabel13: TQRLabel;
    QRDBText14: TQRDBText;
    QRLabel11: TQRLabel;
```

```
      QRLabel14: TQRLabel;
      QRLabel16: TQRLabel;
      QRDBText12: TQRDBText;
      QRLabel17: TQRLabel;
      QRDBText16: TQRDBText;
      QRBand1: TQRBand;
    private
      { Private declarations }
    public
      { Public declarations }
    end;

  var
    Form21: TForm21;

  implementation

  {$R *.dfm}

  end.

  unit Unit21;

  interface

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

  type
    TForm21 = class(TForm)
      QuickRep1: TQuickRep;
      PageHeaderBand1: TQRBand;
      QRDBText2: TQRDBText;
```

```
  QRDBText3: TQRDBText;
  QRLabel1: TQRLabel;
  QRLabel2: TQRLabel;
  QRLabel12: TQRLabel;
  QRDBText18: TQRDBText;
  QRDBText19: TQRDBText;
  DetailBand1: TQRBand;
  QRDBText1: TQRDBText;
  QRLabel3: TQRLabel;
  QRDBText4: TQRDBText;
  QRDBText5: TQRDBText;
  QRLabel4: TQRLabel;
  QRDBText6: TQRDBText;
  QRLabel5: TQRLabel;
  QRDBText7: TQRDBText;
  QRLabel6: TQRLabel;
  QRDBText8: TQRDBText;
  QRLabel7: TQRLabel;
  QRDBText9: TQRDBText;
  QRLabel8: TQRLabel;
  QRLabel9: TQRLabel;
  QRLabel13: TQRLabel;
  QRDBText14: TQRDBText;
  QRLabel11: TQRLabel;
  QRLabel14: TQRLabel;
  QRLabel16: TQRLabel;
  QRDBText12: TQRDBText;
  QRLabel17: TQRLabel;
  QRDBText16: TQRDBText;
  QRBand1: TQRBand;
private
  { Private declarations }
public
  { Public declarations }
```

```
 end;

var
 Form21: TForm21;

implementation

{$R *.dfm}

end.
```