NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Bus Terminal Automation

Graduation Project
COM- 400

Student: Tuncay Tokgöz (20000878)

Supervisor: Mr. Okan Donangil

Nicosia – 2005

TABLE OF CONTENTS

ACKNOWLEDGEMENT

"First, I would like to thank my supervisor Mr. Okan DONANGİL for his invaluable advice and belief in my work and myself over the course of this Graduation Project.

Second, I would like to express my gratitude to Near East University for the scholarship that made the work possible.

Third, I thank my family for their constant encouragement and support during the preparation of this project.

Finally, I would also like to thank all my friends for their advice and support."

ABSTRACT

As the information age has effected every aspect of our life, the need for computerizing many information systems has raised.

Once of the important branches that are effected by information revolution is the computer programming languages.

Project is written using Visual Basic 6.0 programming language and used Microsoft Access Database language for databases. Visual Basic is one of the best and easy programming languages.

Aim of this project is to control bus terminal to give the information to the passengers.That is,This project is pursuing an aim of bus terminal program, such as vehicle, driver, company informations, to help applied used to give a direction.

Before coming to this point, this project has gone through some important steps;

- First one was that I had to have some knowleges about how terminal control to make and learn terminal control working systems for the requirement definitions.
- Second step was to design and to put in order informations about the program.
- The later steps were steps of the implementation of the designed information on computer by using Visual Basic Language.

INTRODUCTION

The Bus Terminal Control System is aimed at solving integral problems related with the user's information, management and operation of a bus station.

This program is able to manage all the station information (administration, buses, companies...), to be informed humans., to display publicity on the information, to generate automatically a human that provides information exits, to take any kind of information from any terminal, to recognise the number plates of the vehicles.

Companies it collects administrative information about the different companies that operate at the bus-station. Buses it gathers information about registration, driver, regular assigned line, etc. for every bus that operates at the station, as well as about what company it belongs to. Platforms it allows to customise each station according to the number and type of platforms that conform each bus-station. It provides the time when it is being used and the present state of each one.

Destinations it provides information about each city to which a line is bound for or a city through which the buses runs. It gathers information about the pronunciation of this city, etc. It gathers information about all the arrivals, exits and schedule of each bus or vehicle at the station or at a specific platform that it has taken. With this information it is possible to be invoiced to each company according to the real use of the station, to the time of maintaining real information on the platform. It allows to define each existing line with specific information about the different schedules from each one of them according dhddthth to the day. The schedules can also be made according to the specifically for each company and the route for each line with the cities thourgh which it runs.

Visual Basic is a Microsoft Windows programming Language.Visual Basic programs are created in an Integrated Development Environment (IDE) . The IDE allows the programmer to create , run and debug Visual Basic programs conveniently. IDEs allow a programmer to create working programs in a fraction of the time that it would normally take to code programs without using IDEs. The process of rapidly creating an application is typically referred to as Rapid Application Development(RAD). Visual Basic is the world's most widely used RAD language.

Visual Basic is derived from the BASIC programming language. Visual Basic is a distinctly different language providing powerfull features such as graphical user interfaces, even handling, access to the Win32 API, object-oriented features, error handling, structured programming, and much more.

The Visual Basic IDE allows Windows programs to be created without the need for the programmer to be a Windows programming export.

Microsoft provides several version of Visual Basic, namely the Learning Edition , the Professional Edition and the Enterprice Edition. The Learning Edition provides fundemantal programming capabilities than the Learning Edition and is the choice of many programmers to write Visual Basic applications. The Enterprice Edition is used for developing large-scale computing systems that meet the needs of substandial organizations.

Visual Basic is an interpreted language. However , the professional and Enterprice Edition allows Visual Basic code to be compiled to native code.

Visual Basic evolved from BASIC(Beginner's All purpose Symbolic Instruction Code). Basic was developed in the mid 1960's by Professors John Kemeny and Thomas Kurtz of Darthmouth College as a language for writing simple programs. BASIC's primary purpose was to help people learn how to program.

The widespread use of BASIC with various types of computers (sometimes called hardware platforms) led to many enhancements to the language. With the development of the Microsoft windows graphical user interface (GUI) in the late 1980s and the early 1990s, the natural evolution of BASIC was Visual Basic, which was created by Microsoft Corporation in 1991.

Until Visual Basic appeared, develoing Microsoft Windows-based applications was a diffucult and cumbersome process. Visual Basic greatly simplifies Windows application development. Since 1991 six versions have been released, with the latest-Visual Basic 6-appearing in september 1998.

After a brief explanation about the Visual Basic 6.0 and the developing layers, I hope that you will find the necessary information that you need all about the Visual Basic even if you are a text based programmer.

1.HISTORY of VISUAL BASIC

Microsoft first released Visual Basic in 1987. It was the first visual development tool from Microsoft, and it was to compete with C, C++, Pascal and other well-known programming languages. From the start, Visual Basic wasn't a hit. It wasn't until release 2.0 that people really discovered the potential of the language, and with release 3.0 it had become the fastest-growing programming language on the market.

Below is the order and the approximate year in which a new version of Visual Basic was released:

VB was introduced in 1991 as Version 1.0

- Very simple controls (controls nuts and bolts of your project, we use controls to get user input and to display output)
  - Text box controls
  - List box controls
  - Combo box controls and a few
- No DBMS features
- Only sequential and random files

1992,VB Version 2.0

- Increased controls
- Feature of DBMS
- Paradox (only level of module)

1993,VB Version 3.0

- More powerful DBMS features
- No need standard module of DBMS
- Data controls are used
- OLE 1.0(Object linking and embedding) feature

1996,VB Version 4.0

- Ability to generate 32-bit applications for both windows95 & Windows NT
- Use OLE technology of Microsoft
- Use some of the techniques of OOP and class modules are introduced
- The ability to extend the VB programming environment. Create or use third party tools into the VB environment
- Conditional compilation to allow you to do multi platform development more easily

1997,VB Version 5.0

- Compilation of native code, p-code
- Create it's own Active-X controls
- Multiple projects
- Design and application for Internet and Intranet environment with Active-X documents.
- New function of code editor
- Downloadable Internet controls
- Visual Models
- Object base data storage- repository
- Has dynamic Linked Library (DDL), to combine VB with another programming languages such as C
- You could also create your own OLE controls in C and use them in VB

1998,VB Version 6.0

- Native Code Compiler

  Create applications, and both client and server-side components that are optimized for throughput by the world-class Visual C++ 6.0 optimized native-code compiler
- ADO (ActiveX Data Objects)

  Visual Basic 6.0 introduces ADO as the powerful new standard for data access, Included OLE DB drivers include SQL server 6.5+, Oracle 7.3.3+, Microsoft Access, ODBC, and SNA server
- Integrated Professional Visual Database Tools

Visual Basic 6.0 provides a complete set of tools for integrating databases with any application.

- Automatic data binding
- Data environment designer
- Data Report designer
- Visual Basic Web Class Designer
- Dynamic HTML Page Designer
-

1.1VISUAL BASIC DISCRIPTION

Visual Basic is a high level programming language evolved from the earlier dos version called basic. Besic means Beginners' Allpurpose Symbolic Instruction Code. It is a fairly easy programming language to learn. The codes look a bit like English Language. Different software companies produced different version of basic, such as Microsoft QBASIC, QUICKBASIC, GWBASIC ,IBM BASICA and so on.

Visual babasic is a visual and events driven Programming Language.These are the main divergence from the old basic. In basic, programming is done in a text-only environment and the program is executed sequentially. In visual basic, programming is done in a graphical environment. Because users may click on a certain object randomly, so each object has to be programmed indepently to be able to response to those actions(events).Therefore, a visual basic program is made up of many subprograms, each has its own program codes, and each can be excecuted indepently and at the same time each can be linked together in one way or another

Today's most popular operating system for PC's is Widows 98, and also it's an environment that most of the software in the world needs an environment of Windows 98 in order to operate or run. Nearly it became an international standard to make the programs, software to be able to run on Windows 98. So from these points we did a software package that should run on Windows 98. In order to make the programs to run in Windows 98 needs an interface program, which is MS Visual Basic 6.0, which is the most popular Visual Programming language in world for making programs for Windows 98 environment.

Visual Basic is Windows development language, that's why you must be familiar with the Windows environment. The "Visual" part of the "Visual Basic" refers to the method used to

create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply drag and drop rebuilt objects into place on screen. If you've ever used a drawing program such as Paint, you already have most of the skills necessary to create an effective user interface.

The "Basic" part refers to the BASIC (Beginners Ail-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

Windows involves two key concepts as below;

- Window
- Events and Messages

Window:

A window is a simply rectangular region with its own boundaries.

Examples of windows are:

- An Explorer window in windows 95
- A document window in word processor
- Dialog box that pop up window and reminds you of an appointment
- A command button
- Icons
- Text boxes
- Option boxes
- Menu bars

Microsoft Windows Operating system manages all of these many windows by assigning each one unique id number. The system continually monitors each of these windows for signs of activity or events.

Events and messages:

An event is an action recognized by a form or control. Events can occur through user action (response) such as a mouse click or a key press using objects of window (through programmatic control), or even as a result of another windows action.

Event-Driven applications execute Basic code in response to an event. Each form and control in VB has a predefined set of events. If one of these events occurs and there is a user code in the associated event procedure, VB invokes that code. For example most object recognize a Click event. If user clicks a form (object), code in the form's Click event procedure is executed. Each time an event occurs, it causes a message to be sent to the operating system. The system processes the message and broadcast it to the other windows. Each window can take the appropriate action based on its own instructions from dealing with that particular message.

Fortunately, Visual Basic insulates you from having to deal with all of the low-level message handling. Many of the messages are handled automatically by VB. This allows you to quickly create powerful application without having to deal with necessary details.
Programs in conventional programming languages run from the top down. For older programming languages, execution starts from the first line and moves with the flow of the program to different parts as needed.

Visual Basic program usually works completely different. The code doesn't follow a predefined path. It executes different code section in response to events. The core of a Visual Basic programs is a set of independent pieces of code that are activated by, and so respond to, only the event they have been told to recognize.

The programming code in VB that tells your program how to respond to events (event procedure) .An event procedure is a body of code that is only executed in response to an external event.

Your code can also trigger events during execution. It is for this reason that is important to understand the event-driven model and keep it mind when designing VB applications.

The fastest and easiest way to create applications for Microsoft Windows Whether you are an experienced professional or brand new to Windows programming, Visual Basic provides you with a complete set of tools to simplify rapid application development.

The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system .Applications Edition included in Microsoft Excel, Microsoft Access, and many other Windows applications uses the same language. Whether your goal is to create a small utility for yourself or your work group, a large enterprise-wide system, or even distributed applications spanning the globe via the Internet, Visual Basic has the tools you need.

- Data access features allow you to create databases and front-end applications for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.
- ActiveX technologies allow you to use the functionality provided by other applications, such as Microsoft Word, which is a word processor, Microsoft Excel spreadsheet, and other Windows applications. You can even automate applications and objects created using the Professional or Enterprise editions of Visual Basic.
- Internet capabilities make it easy to provide access to documents and applications across the Internet from within your application.
- Your finished application is a true .exe file that uses a run-time dynamic-link library (DLL) that you can freely distribute.

System Requirements for Visual Basic:

Following hardware and software is required for Visual Basic applications:

- Microsoft Windows NT 3.51 or later, or Microsoft Windows 95 or later.
- 80486 or higher microprocessor.
- VGA or higher-resolution screen supported by Microsoft Windows.
- 8 MB of RAM for applications. (This will vary, depending on the specific type libraries or DLLs you include with your applications.)
- 16 MB of RAM for the Visual Basic development environment.

Project limitations:

A single project can contain up to 32,000 identifiers, which include, but are not limited to, forms, controls, modules, variables, constants, procedures, functions, and objects. Variable names in Visual Basic can be no longer than 255 characters, and the names of forms, controls, modules, and classes cannot be longer than 40 characters. Visual Basic imposes no limit on the actual number of distinct objects in a project

Form Structure:

While many of the files in a typical Visual Basic project are in a binary format and are readable only by specific processes and functions of Visual Basic or your application, the form (.Frm) and project (.vbp) files are saved as ASCII text. These are readable in a text viewer (Notepad for instance).

Visual Basic form (.frm) files are created and saved in ASCII format. The structure of a form consists of:

- The version number of the file format.
- A block of text containing the form description.
- A set of form attributes.
- The Basic code for the form.

The form description contains the property settings of the form. Blocks of text that define the properties of controls on the form are nested within the form. Controls contained within other controls have their properties nested within the text of the container.

1.2WHAT WE CAN DO WITH VISUAL BASIC

Creating User Interface:

The user interface is perhaps the most important part of an application; it's certainly the most visible. To users, the interface is the application; they probably aren't aware of the code that is executing behind the scenes. No matter how much time and effort you put into writing and optimizing your code, the usability of your application depends on the interface.

When you design an application, a number of decisions need to be made regarding the interface. Should you use the single-document or multiple-document style? How many different forms will you need? What commands will your menus include, and will you use toolbars to duplicate menu functions? What about dialog boxes to interact with the user? How much assistance do you need to provide?

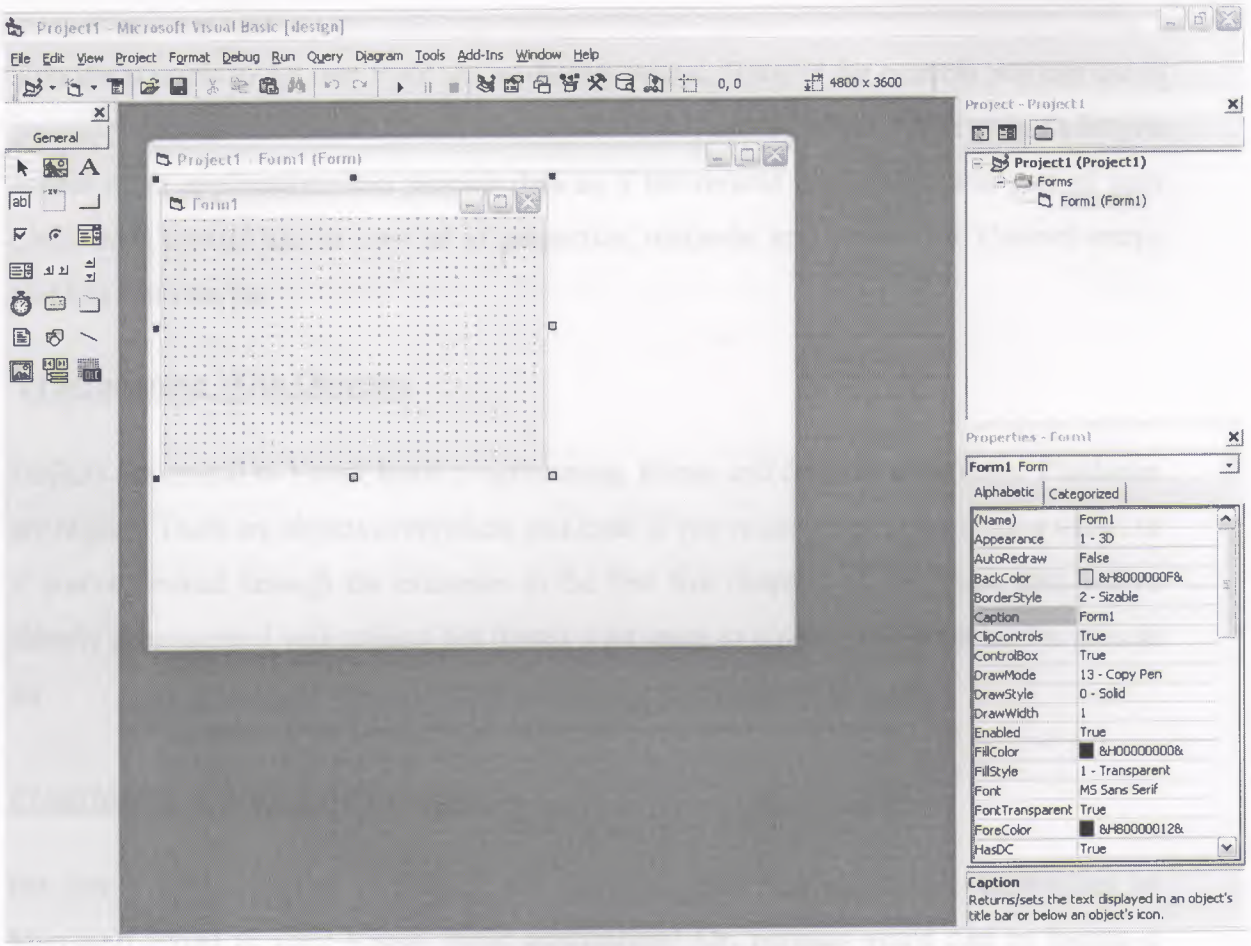


Figure 2.2.1

Before you begin designing the user interface, you need to think about the purpose of the application. The design for a primary application that will be in constant use should be different from one that is only used occasionally for short periods of time. An application with the primary purpose of displaying information has different requirements than one used to gather information.

The intended audience should also influence your design. An application aimed at a beginning user demands simplicity in its design, while one for experienced users may be more complex. Other applications used by your target audience may influence their expectations for an application's behavior. If you plan on distributing internationally, language and culture must be considered part of your design.

Using Visual Basic Standard Controls:

You use controls to get user input and to display output. Some of the controls you can use in your applications include text boxes, command buttons, and list boxes. Other controls let you access other applications and process data as if the remote application was part of your code. Each control has its own set of properties, methods, and events. Ex: Control arrays, text box controls, etc.

Programming With Objects:

Objects are central to Visual Basic programming. Forms and controls are objects. Databases are objects. There are objects everywhere you look. If you've used Visual Basic for a while, or if you've worked through the examples in the first five chapters of this book, then you've already programmed with objects but there's a lot more to objects than what you've seen so far.

Programming With Components:

Do you sometimes need to provide the same analysis and calculation capabilities as Microsoft Excel in your Visual Basic application? Or, perhaps you'd like to format a document using Microsoft Word formatting tools, or store and manage data using the Microsoft Jet database engine. Even better, would you like to be able to create or buy standard components, and then use them in multiple applications without having to modify them? All this and more can be accomplished by building your applications using ActiveX components. An ActiveX component is a reusable piece of programming code and data made up of one or more objects created using ActiveX technology. Your applications can use existing components, such as those included in Microsoft Office applications, code components, ActiveX documents, or ActiveX controls (formerly called OLE controls) provided by a variety of vendors. Or, if you have the Visual Basic, Professional or

Enterprise Edition, you can create your own ActiveX controls. For components that support object linking and embedding, you can insert objects into your application without writing any code by using the component's visual interface. You can insert an OLE-enabled object into your application by using the OLE container control or by adding the object's class to the Toolbox. To fully understand ActiveX components, you should first be familiar with how to work with classes, objects, properties, and methods, which are explained in "Programming with Objects."

Responding to mouse and keyboard Event:

Your Visual Basic applications can respond to a variety of mouse events and keyboard events. For example, forms, picture boxes, and image controls can detect the position of the mouse pointer, can determine whether a left or right mouse button is being pressed, and can respond to different combinations of mouse buttons and SHIFT, CTRL, or ALT keys. Using the key events, you can program controls and forms to respond to various key actions or interpret and process ASCII characters.

In addition, Visual Basic applications can support both event-driven drag-and-drop and OLE drag-and-drop features. You can use the Drag method with certain properties and events to enable operations such as dragging and dropping controls. OLE drag and drop gives your applications all the power you need to exchange data throughout the Windows environment and much of this technology is available to your application without writing code.

You can also use the mouse or keyboard to manage the processing of long background tasks, which allows your users to switch to other applications or interrupt background processing.

Working with Texts and Graphics

Visual Basic includes sophisticated text and graphics capabilities for use in your applications. If you think of text as a visual element, you can see that; size, shape and color can be used to enhance the information presented. Just as a newspaper uses headlines, columns and bullets to break the words into bite-sized chunks, text properties can help you emphasize important concepts and interesting details.

Visual Basic also provides graphics capabilities allowing you great flexibility in design, including the addition of animation by displaying a sequence of images.

Debugging your code and handling Errors:

No matter how carefully crafted your code, errors can (and probably will) occur. Ideally, Visual Basic procedures wouldn't need error-handling code at all. Unfortunately, sometimes files are mistakenly deleted, disk drives run out of space, or network drives disconnect unexpectedly. Such possibilities can cause run-time errors in your code. To handle these errors, you need to add error-handling code to your procedures.

Sometimes errors can also occur within your code; this type of error is commonly referred to as a *bug*. Minor bugs: for example, a cursor that doesn't behave as expected can be frustrating or inconvenient. More severe bugs can cause an application to stop responding to commands, possibly requiring the user to restart the application, losing whatever work hasn't been saved. The process of locating and fixing bugs in your application is known as *debugging*. Visual Basic provides several tools to help analyze how your application operates. These debugging tools are particularly useful in locating the source of bugs, but you can also use the tools to experiment with changes to your application or to learn how other applications work.

Accessing Data:

Almost all applications require some form of data storage and manipulation, and Visual Basic provides a number of tools to meet these needs, including the data control and data-bound controls, data access objects, remote data objects, and the remote data control.

Designing for Performance and Compatibility:

In an ideal world, every user of your applications would have a computer with the fastest possible processor, plenty of memory, unlimited drive space, and a blazingly fast network connection. Reality dictates that for most users, the actual performance of an application will be constrained by one or more of the above factors. As you create larger and more sophisticated applications, the amount of memory the applications consume and the speed with which they execute become more significant. You may decide you need to *optimize* your application by making it smaller and by speeding calculations and displays.

As you design and code your application, there are various techniques that can be used to optimize the performance. Some techniques can help to make your application faster; others

can help to make it smaller. In this chapter I will explain some of the more common optimization tricks that you can use in your own applications.

Visual Basic shares most of its language features with Visual Basic for Applications, which is included in Microsoft Office and many other applications. Visual Basic, Scripting Edition (VBScript), a language for Internet scripting, is also a subset of the Visual Basic language. If you're also developing in Visual Basic for Applications or VBScript, you'll probably want to share some of your code between these languages.

International Issues:

If you are planning to distribute your Visual Basic application to an international market, you can reduce the amount of time and code necessary to make your application as functional in its foreign market as it is in its domestic market. This chapter introduces key concepts and definitions for developing international applications with Visual Basic, presents a localization model, and emphasizes the advantages of designing software for an international market.

Distributing Your Application:

Once you have created a Visual Basic application, you may want to distribute it to others. You can freely distribute any application you create with Visual Basic to anyone who uses Microsoft Windows. If you are going to distribute your application, you will need to write or use a setup program that installs your application onto a user's machine.

1.3.USED COMPENENTS

I am going to explain some components, which are used for this project. All this components contains its own .OCX files, so user can register this files to use new components. Following components are commonly used for projects and also they used for "Periodics Control System" and "Book Control System" programs that which were written by me.

Command Button:

Most Visual Basic applications have command buttons that allow the user to simply click them to perform actions. When the user chooses the button, it not only carries out the appropriate action, it also looks as if it's being pushed in and released. Whenever the user clicks a button, the Click event procedure is invoked. You place code in the Click event procedure to perform any action you choose.

Label:

A label control displays text that the user cannot directly change. You can use labels Ito identify controls, such as text boxes and scroll bars that do not have their own caption property. The actual text displayed in a label is controlled by the Caption property, which can be set at design time in the Properties window or at run time by assigning it in code. By default, the caption is the only visible part of the label control. However, if you set the BorderStyle property to one (which you can do at design time), the label appears with a border giving it a look similar to a text box. You can also change the appearance of the label by setting the BackColor, BackStyle, ForeColor, and Font properties.

Text Box:

Text boxes are versatile controls that can be used to get input from the user or to display text. Text boxes should not be used to display text that you don't want the user to change, unless you've set the Locked property to true. The actual text displayed in a text box is controlled by the Text property. It can be set in three different ways: at design time in the Property window, at run time by setting it in code or by input from the user at run time. The current contents of a text box can be retrieved at run time by reading the Text property.

Option Button:

Option buttons present a set of two or more choices to the user. Unlike check boxes, however, option buttons should always work as part of a group; selecting one option button immediately clears all the other buttons in the group. Defining an option button group tells the user, "Here is a set of choices from which you can choose one and only one."

List Box:

List boxes and combo boxes present a list of choices to the user. By default, the choices are displayed vertically in a single column, although you can set up multiple columns as well. If the number of items exceeds what can be displayed in the combo box or list box, scroll bars automatically appear on the control. The user can then scroll up and down or left to right through the list.

Timer:

Timer is used to make some operation in a specific time interval. Time interval can be adjusted from the properties of the timer.

Microsoft DataGrid 6.0:

The DataGrid Displays and enables data manipulation of a series of rows and columns representing records and fields from a Recordset object. The DataGrid control's Columns collection's Count property and the Recordset object's RecordCount property to determine the number of columns and rows in the control. A DataGrid control can have as many rows as the system resources can support and up to 32767 columns.

Microsoft Ado data control 6. 0:

In Visual Basic, three data access interfaces are available to you: ActiveX Data Objects (ADO), Remote Data Objects (RDO), and Data Access Objects (DAO). A data access interface is an object model that represents various facets of accessing data. Using Visual Basic, you can programmatically control the connection, statement builders, and returned data for use in any application.

Frame:

A Frame control provides an identifiable grouping for controls. You can also use a Frame to subdivide a form functionally—for example, to separate groups of OptionButton controls.

1.4.DATA INTERFACES

1.4.1.ActiveX Data Objects (ADO)

The ADO Data control uses Microsoft ActiveX Data Objects (ADO) to quickly create connections between data-bound controls and data providers. Data-bound controls are any controls that feature a DataSource property. Data providers can be any source written to the OLE DB specification. You can also easily create your own data provider using Visual Basic's class module.

Although you can use the ActiveX Data Objects directly in your applications, the ADO Data control has the advantage of being a graphic control (with Back and Forward buttons) and an easy-to-use interface that allows you to create database applications with a minimum of code.

Several of the controls found in Visual Basic's Toolbox can be data-bound, including the CheckBox, ComboBox, Image, Label, ListBox, PictureBox, and TextBox controls. Additionally, Visual Basic includes several data-bound ActiveX controls such as the DataGrid, DataCombo, Chart, and DataList controls. You can also create your own data-bound ActiveX controls, or purchase controls from other vendors.

1.4.2.OLE DB Providers

OLE DB is a new low-level interface that introduces a "universal" data access paradigm. That is, OLE DB is not restricted to ISAM, Jet, or even relational data sources, but is capable of dealing with any type of data regardless of its format or storage method. In practice, this versatility means you can access data that resides in an Excel spreadsheet, text files, or even on a mail server such as Microsoft Exchange.

In Visual Basic 6.0, you leverage the flexibility of OLE DB through ADO, the programmer interface to OLE DB. You can even create your own OLE DB Providers in Visual Basic.

OLE DB is not designed to be accessed directly from Visual Basic due to its complex interfaces. Instead ActiveX Data Objects (ADO) encapsulates and exposes virtually all of OLE DB's functionality.

1.5.CREATIVITY OF NEW ODBC DATA SOURCE

In the control panel you must first enter Administrative tools. In Administrative rule you must double click the Data sources (ODBC). After this you will see the below figure. You must select your data source if created before. If you can not see your data source then you must add new data source by clicking Add command button.

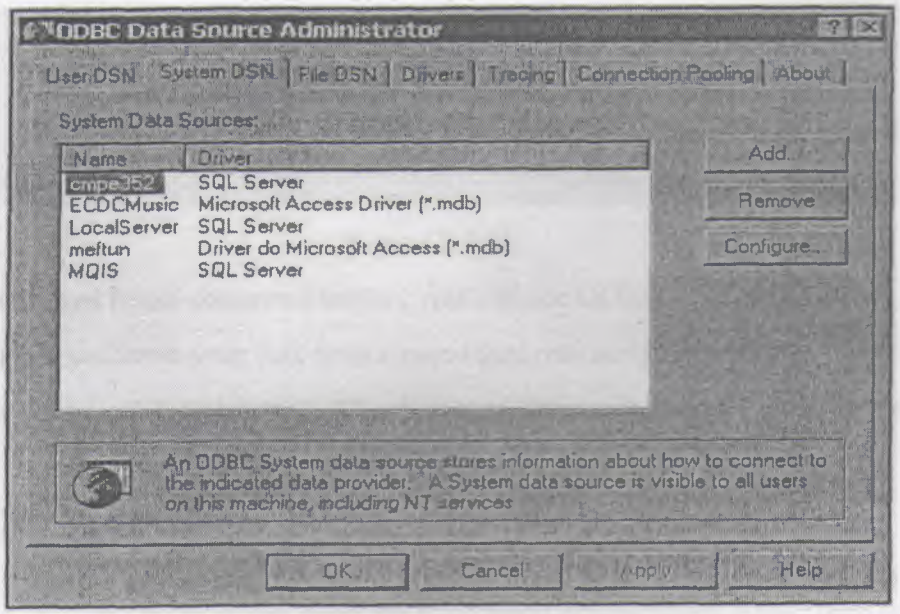


Figure 2.5.1

When you clicked add command button, you will see Create New Data Source window. In this window you select your driver, which you want to set up a data source. Then you must press Final command to finish creation of data source.

Figure 2.5.2

When you clicked finish command button, you will see ODBC Microsoft Access Setup window. In this window you write your data source name then you assign the directory of your data source by clicking select command button. After this step you finished your ODBC connection.

1.6.MENU EDITOR

Firstly, to display the menu editor, from the tools menu choose Menu Editor then you will face with the Menu editor window shown blown.

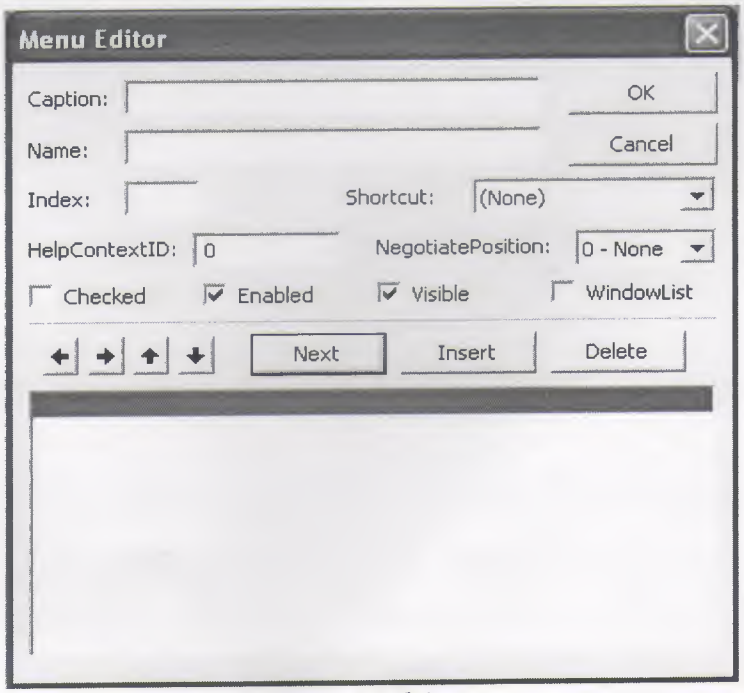


Figure 1.6.1

While most menu control properties can be set using the Menu Editor; all menu properties are also available in the properties window. You should normally create a menu in the menu editor; however, to quickly change a single property, you could use the properties window.

If you want your application to provide a set of commands to users, menus offer a convenient and consistent way to group commands and an easy way for users to access them. The menu bar appears immediately below the title bar on the form and contains one or more menu titles. When you click a menu title, a menu containing a list of menu items drop down. Menu items can include commands, separator bars and sub menu titles. Each menu items, the user sees corresponds to a menu control you define in the menu editor. To make your application easier to use, you should group menu items according to their function.

Some menu items perform an action directly; for example, the exit menu item in the file menu closes application. Other menu items display a dialog box - a window that requires the user to supply information need by the application to perform action. These menu items should be followed by an ellipses (..). For example, when you choose Save As., from the file menu, the save file appears in the dialog box.

2.DATABASE MANAGEMENT SYSTEM

2.1.INFORMATION ABOUT DBMS

Databases Management System (DBMS) consists of a collection of interrelated data and collection of programs to access that data. The data contains information about one particular enterprise. The primary goal of a DBMS is to provide an environment, which is both convenient and efficient to use in retrieving and storing information.

Database systems are designed to manage large bodies of information. The management of data involves both the definition of structures for the manipulating of information. In addition the database system crashes or attempts at authorized access. If data is to be shared among several users, the system must avoid possible anomalous results.

A major purpose of a database system is to provide users with an abstract view of the data. That is the system hides certain details of how the data is stored and maintained. This is accomplished by defining three levels of abstraction.

> The Physical Level
> The Conceptual Level
> Level The view Level

Underlying the structure of a database is the data model, collection of conceptual tools for describing data, data relationships, data semantics, and data constraints. The various data models that have been proposed, is divided into three different groups:

1 - Object- Based Logical Model

2- Record Based Logical Model

3- Physical Data Models

Database change over time as information is inserted and deleted. The Collection of information stored in the database at a particular moment in time is called an instance of the database. The overall design of the database is called the database scheme. The ability to modify scheme definition in one level without affecting scheme definition in the next-higher level is called data independence. There are two levels of data dependencies,

1- Physical Data independencies

2- Logical Data independencies

A database scheme is specified by a set of definitions, which are expressed by a data definition language (DDL). DDL statements are compiled into a set of tables, which are stored in special file called the data dictionary, which contains metadata. A data manipulating language (DML) is a language that enables users to access or manipulate data. There are basically two *types:* procedural DML's which require a user to specify what data is needed and how to get it and nonprocedural DML's which require a user to specify what data is needed without specifying how to get it

2.2.DATA MODELS

Underlying the structure of a database is the concept of data model, a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. The various data models that have been proposed fall into three different groups: object-based logical models, record-based logical models, and physical data models.

Object-Based Logical Models:

Object-based logical models are used in describing data at the conceptual and view levels. They are characterized by the fact that they provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. There are many different models, and more likely to come. Some of the more widely known ones are:

- The entity-relationship model.
- The object-oriented model.
- The binary model.
- The semantic data model.
- The functional data model.

The Entity-Relationship Model:

The entity-relationship (E-R) data model is based on a perception of a real world, which consists of a collection of basic objects called entities, and relationships among these objects. An entity is an object that is distinguishable from other objects by a specific set of attributes. For example, the attributes number and balance describe one particular account in a bank. A relationship is an

association among several entities. For example, a CustAcct relationship associates a customer with each account that she or he has. The set of all entities of the same type and relationships of the same type are termed an entity set and relationship set, respectively.

In addition to entities and relationships, the E-R model represents certain constraints to which the contents of a database must conform. One important constraint is mapping cardinalities, which express the number of entities to which another entity can be associated via a relationship set.

The overall logical structure of a database can be expressed graphically by an E-R diagram, which consists of the following components:

- Rectangles, which represent entity sets.
- Ellipses, which represent attributes
- Diamonds, which represent relationship among entity sets.
- Lines, which link attributes to entity sets and entity sets to relationships.

Each component is labeled with the entity or relationship it represents.
To illustrate, consider part of a database banking system consisting of customers and the accounts that they have.

From a historical Perspective, the relational data model is relatively new. The first database systems are based on either the network model or the hierarchical model. Those two older models are tied more closely to the underlying implementation of the database than is the relational model. The relational model has established itself as the primary data model of the commercial data processing in systems for computer-aided design and other environments.

Relational Algebra:

The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are select, project, Cartesian product, rename, union, and set difference. In addition to fundamental operations, there are several other operations, namely, set intersection, natural join, division and assignment.

2.3 RELATION MODEL

The Cartesian Product Operation :

In order to combine information from several relations. One operation that allows us to do that is the Cartesian product operation, denoted by a cross (x). This operation is a binary operation, we shall use infix notation for binary operations and, thus, write the Cartesian product of relations rl and r2 as rl x r2.

The Rename Operation:

In some queries we introduced the convention of naming attributes by relation_name. attribute_name in order to eliminate possible ambiguity. Another form of potential ambiguity arises when the same relation appears more than once in a query.

The Union Operation:

If you consider a query that night be posed by a bank's advertising department: "Find all customers if the Lefkosa branch." That is, find everyone who has a loan, an account, or both, then we use the union operator.

Set Difference Operation:

The set difference operation, denoted by -, allows us to find topless that are in one relation but not in another. The expression r-s results in a relation containing those topless in r but not in s.

2.4.SQL (STRUCTURED QUERY LANGUAGE)

SQL means "Structured Query Language". There are numerous version of SQL. The original version was developed at IBM's San Jose Research Laboratory. This language originally called Sequel was implemented as part of the system R Project in early 1970's, the Sequel language has evolved since then, and its name change to SQL. Although the product version of SQL differs in several language details, the differences are for the most part, minor. The SQL language has several a parts.

SQL provides a data definition language (DDL), a data manipulation language (DML) and a set of Data Control commands. Although there are some areas of overlap, the DDL commands allow you to create and define new databases, fields, and indexes. The DML commands let you build queries to sort, filter, and extract data from the database. Data control statements control the execution of DML and DDL SQL statements to ensure that data stored in a relational database remains consistent and secure in a multi-user environment. Most applications that call the ODBC-API almost exclusively use DML commands, of which the SELECT statement is the most frequently used. Thus, this chapter will have an emphasis on explaining the SELECT statement. Detailed descriptions if other DML statements should be investigated from other sources at the reader's discretion.

A typical DBMS allows users to store, access, and modify data in an organized, efficient way. Originally, the users of DBMSs were programmers. Accessing the stored data required writing a program in a programming language such as COBOL. While these programs were often written to present a relatively friendly interface to a non-technical user, access to the data itself required the services of a knowledgeable programmer. Casual access to the data was not practical.

Users were not entirely happy with this situation. While they could access data, it often required convincing a DBMS programmer to write special software. For example, if a sales department wanted to see the total sales in the previous month by each of its salespeople, and it wanted this information ranked in order by each salesperson's

length of service in the company, it had two choices. Either a program already existed that allowed the information to be accessed in exactly this way, or the department had to ask a programmer to write such a program. In many cases, this was more work than it was worth, and it was always an expensive solution for one-time, or ad-hoc, inquiries. As more and more users wanted easy access, this problem grew larger and larger

Data Definition Language (DDL):

The SQL DDL provides commands for defining relations schemes, deleting relations, creating indices and modifying relations.

A database scheme is specified by a set of definitions, which are expressed by a special language called a data definition language (DDL). The result of compilation of DDL statements is a set of tables, which are stored in a special file called data dictionary (or directory).

A data directory is a file that contains metadata; that is, "data about data." This file is consulted before actual data is read or modified in the database system.

The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called a data storage and definition language. The result of compilation of these definitions is a set of instructions to specify the implementation details of the database schemes that are usually hidden from the users.

DDL statements in SQL are expressions built mainly around the following commands:

Statements	Usage
CREATE	Used to create new tables, fields, views and indexes.
DROP and DELETE	Used to delete tables and indexes from the database.
ALTER	Used to modify tables by adding fields or changing field definitions
GRANT and REVOKE	Use to handle privileges

Almost all data definition language statements are data source specific. For example, the CREATE TABLE statement does not use the standard ODBC data types. Instead, it uses data source-specific data types. Because of this, the syntax is not provided for the data definition language statements supported by ODBC. Therefore, they will not be discussed in this manual.

Interactive data manipulating language (DML):

The SQL DML includes a query language based on both the relational algebra and the tuple relational calculus. It includes also commands to insert, delete, and modify tuples in the database.

By data manipulation we mean:

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of data stored in the database.

At the physical level, we must define algorithm that allow for efficient access to data. At higher levels of abstraction, an example is placed on ease of use. The goal is to provide for efficient human interaction with the system.

A data manipulation language (DML) is language that enables users to access or manipulate data as organized by the appropriate data model. There are basically two types:

> Procedural DMLs require a user to specify what data is needed and how to get it.

> Nonprocedural DMLs require a user to specify what data is needed without specifying how to get it.

Nonprocedural DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, these languages may generate code which is not as efficient as that produced by procedural languages.

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval called a query language. Although technically incorrect, it is

common practice to use the terms query language and data manipulation language synonymously.

DML statements are expressions built around the following commands.

Statement	Usage
SELECT	Used to query the database for records that satisfy specific criteria
INSERT	Used to load batches of data into the database in a single operation
UPDATE	Used to change the values of particular records and fields
DELETE	Used to remove records from a database table

2.5.THE BASIC STRUCTURE OF SQL

The basic structure of SQL Expression consists of 3 clauses: Select, From and Where.

** The SELECT clause corresponds to the projection operation of the relational algebra. It used to list the attributes desired in the result of a query.

** The FROM clause corresponds to the Cartesian product operation of the relation algebra. It lists the relation to be scanned in the evaluation of the expression.

** The WHERE clause corresponds to the selection predicate of the relational algebra. It consists of a predicate involving attributes of the relations that appear in the From clause.

The different meaning of the "SELECT" in SQL and in the relational algebra is an unfortunate historical fact. We emphasize the different interpretation here to minimize potential confusion. A typical SQL query has the form:

SELECT Al, A2,...An FROM rl,r2,m
WHERE P

NOTE: Ai represents attribute and each rl a relation. P is a predicate.

As can be seen, the ODBC interface standardizes database access by following a set of guidelines:

- ODBC is a call-level interface. To solve the problem of how applications access multiple DBMSs using the same source code, ODBC defines a standard CLI. This contains all of the functions in the CLI specifications from X/Open and ISO/IEC and provides additional functions commonly required by applications. A different library, or driver, is required for each DBMS that supports ODBC. The driver implements the functions in the ODBC API. To use a different driver, the application does not need to be recompiled or relinked. Instead, the application simply loads the new driver and calls the functions in it. To access multiple DBMSs simultaneously, the application loads multiple drivers. How drivers are supported is operating system – specific. For example, on the Windows operating system, drivers are dynamic-link libraries (DLLs).

- ODBC defines a standard SQL grammar. In addition to a standard call-level interface, ODBC defines a standard SQL grammar. This grammar is based on the X/Open SQL CAE specification. Differences between the two grammars are minor and primarily due to the differences between the SQL grammar required by embedded SQL (X/Open) and a CLI (ODBC). There are also some extensions to the grammar to expose commonly available language features not covered by the X/Open grammar. Applications can submit statements using ODBC or DBMS-specific grammar. If a statement uses ODBC grammar that is different from DBMS-specific grammar, the driver converts it before sending it to the data source. However, such conversions are rare because most DBMSs already use standard SQL grammar.

- ODBC provides a Driver Manager to manage simultaneous access to multiple DBMSs. Although the use of drivers solves the problem of accessing multiple DBMSs simultaneously, the code to do this can be complex. Applications that are designed to work with all drivers cannot be statically linked to any drivers. Instead, they must load drivers at run time and call the functions in them through a table of function pointers. The situation becomes more complex if the application uses multiple drivers simultaneously. Rather than forcing each application to do this, ODBC provides a Driver Manager. The Driver Manager implements all of the ODBC functions — mostly as pass-through calls to ODBC functions in drivers — and is statically linked to the application or loaded by the application at run time. Thus, the application calls ODBC functions by name in the Driver Manager, rather than by pointer in each driver. When an application needs a particular driver, it first requests a connection handle with which to identify the driver, and then requests that the Driver Manager load the driver. The Driver Manager loads the driver and stores the address of each function in the driver. To call an ODBC function in the driver, the application calls that function in the Driver Manager and passes the connection handle for the driver. The Driver Manager then calls the function using the address it stored earlier.

- ODBC exposes a significant number of DBMS features but does not require drivers to support all of them. If ODBC exposed only features that are common to all DBMSs, it would be of little use. After all, the reason so many different DBMSs exist today is that they have different features. If ODBC exposed every feature that is available in any DBMS, it would be impossible for drivers to implement. Instead, ODBC exposes a significant number of features — more than are supported by most DBMSs — but requires drivers to implement only a subset of those features. Drivers implement the remaining features only if they are supported by the underlying DBMS or if they choose to emulate them. Thus, applications can be written to exploit the features of a single DBMS, as exposed by its driver, to use only those features used by all DBMSs, or to check for support of a particular feature and react accordingly. In this way, an application can determine what features a driver and DBMS support, ODBC provides two functions (SQLGetInfo and SQLGetFunctions) that return general information about the driver and DBMS capabilities, and a list of functions the driver supports. ODBC also defines API and SQL grammar conformance levels, which specify broad ranges of features supported by the driver.

2.6.MAPPING CONSTRAINTS

Mapping cardinalities are most useful in describing binary relationship sets, although occasionally they contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

One-to-one: An entity in A associated with at most one entity in B, and an entity in B is associated at most one entity in A.

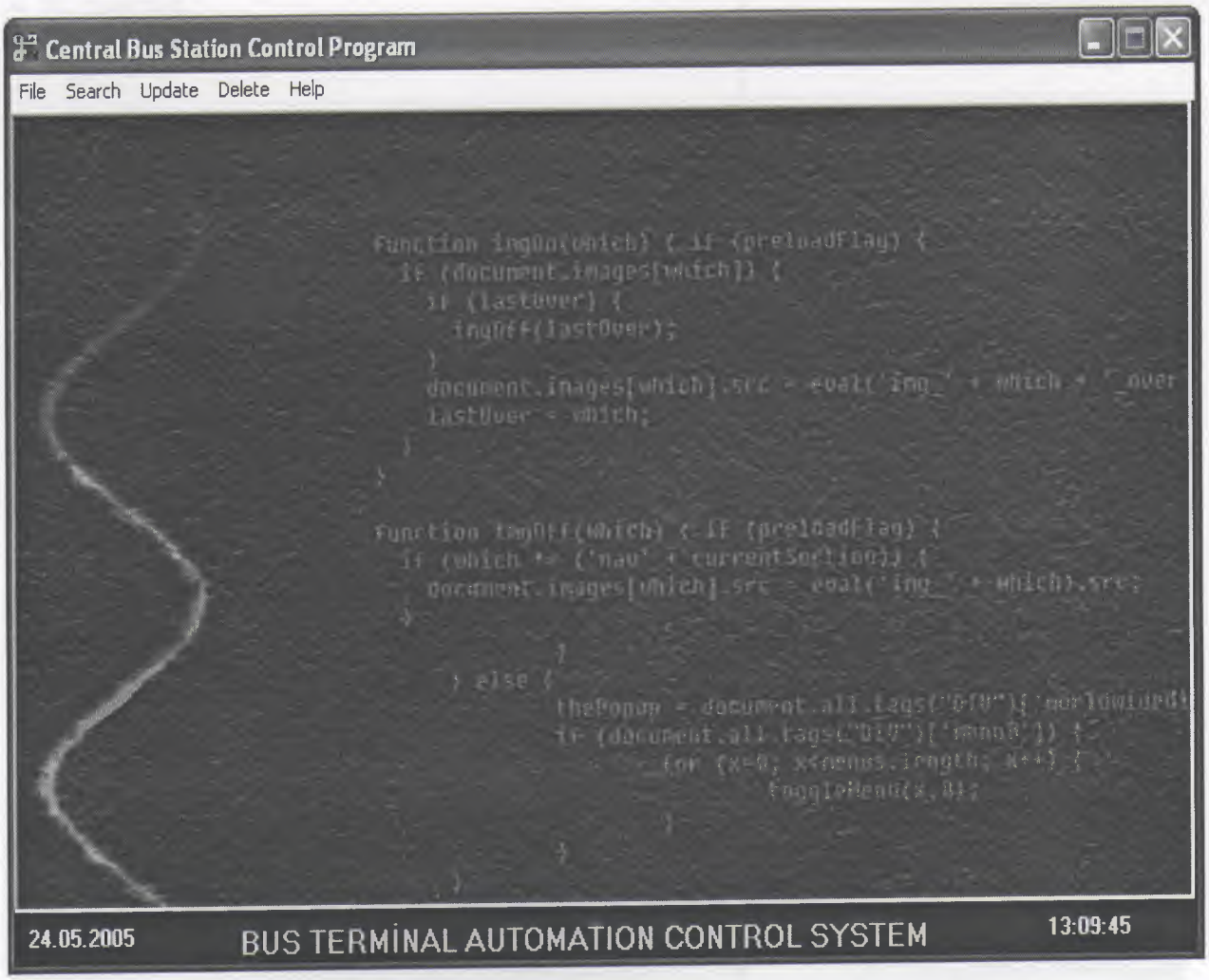
3.SOFTWARE STRUCTURE AND DETAIL

3.1.MAIN MENU

This form is main form which is appearing during program running and includes sub menus as new company , search ,update, delete, etc. on its top. On the main menu , we reach sub forms by using these sub menus.

When program is started, mission of main menu is to run "Insert, Delete or Select Database " functions in module to provide or to remove connecting or between program and database .

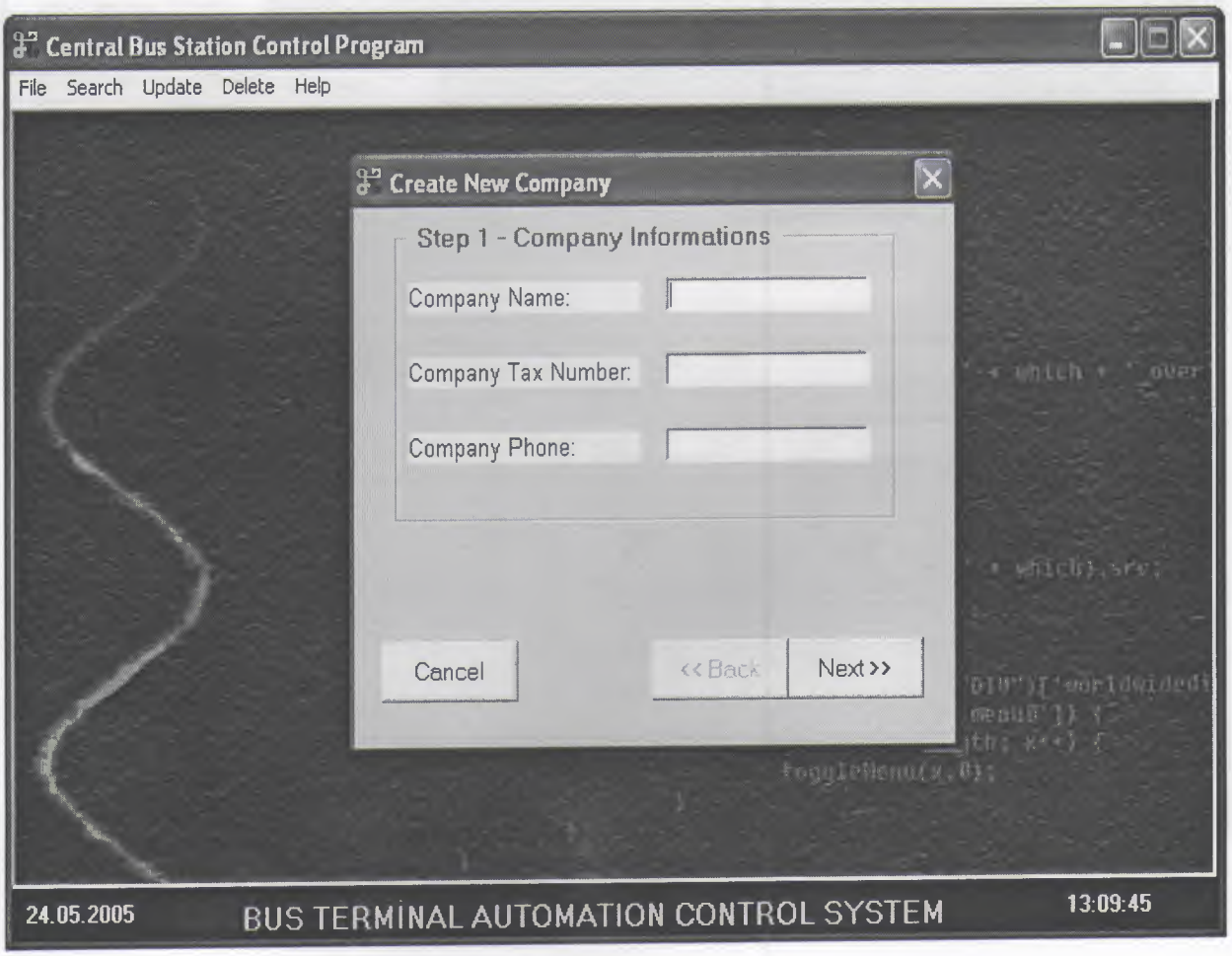
The form appearance and codes of the Main form following down.



3.1.1.NEW ENTRY MENU

This form is the create new entry form of the program. We can create new company informations by using this form. In the form, information about company as name, tax number, Phone, destination, price (student, normal, soldier), vehicle plate number, capacity and driver name, driving license number, platform number, time and date etc. is entered and this forms includes a section to give information about driver, vehicle of companys. When click on this section, company form appears in order to select company existing in creates.

The new company menu is following. This menu was named as create new company in codes of the program

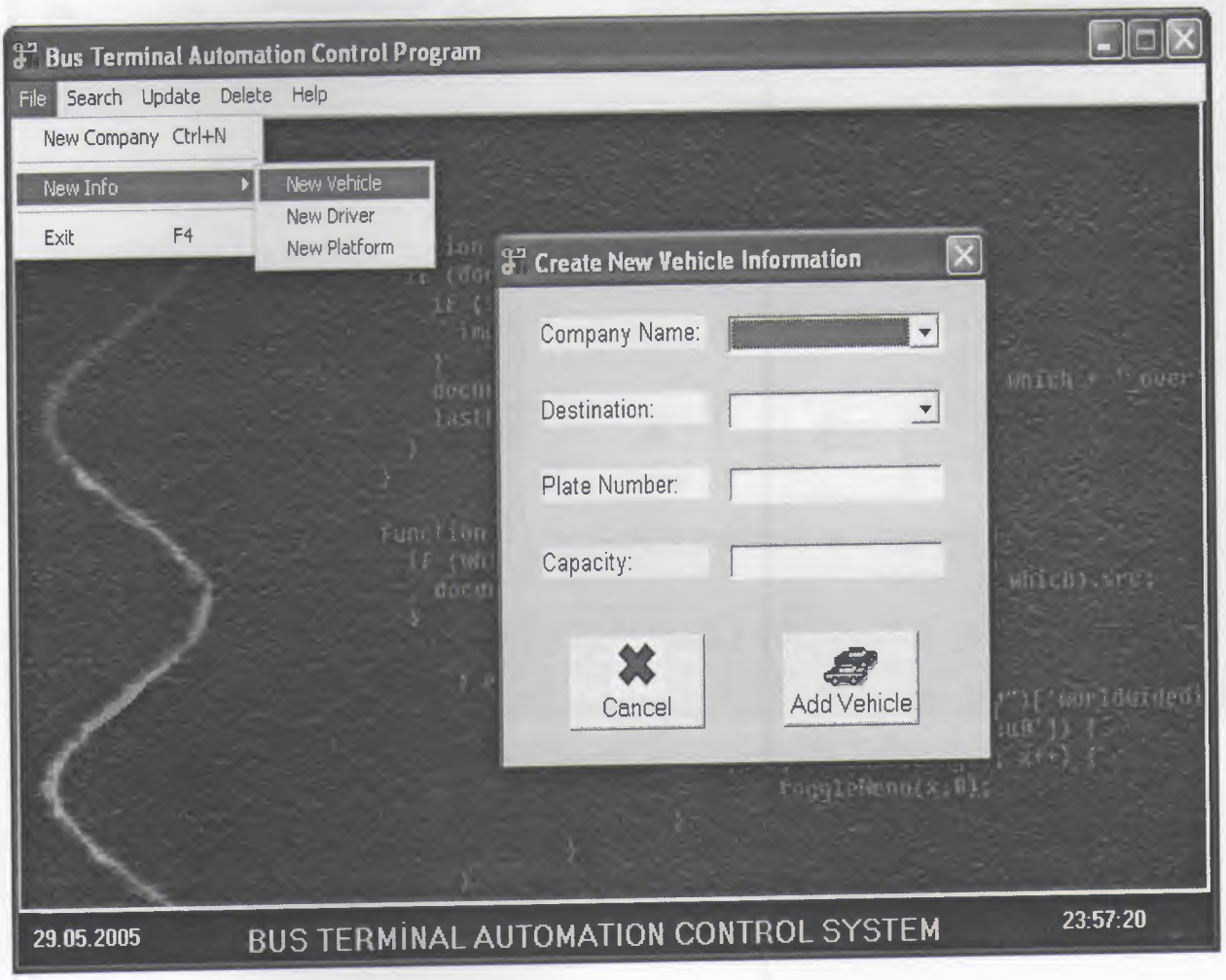


3.1.2.NEW INFO MENU

3.1.2.1.NEW VEHİCLE MENU FORM:

This form is the create a new vehicle form of the program. If the user wants to insert a new vehicle. We can create new vehicle informations by using this form. In the form, information about company as name, destination, plate number and capacity is entered and save a database.

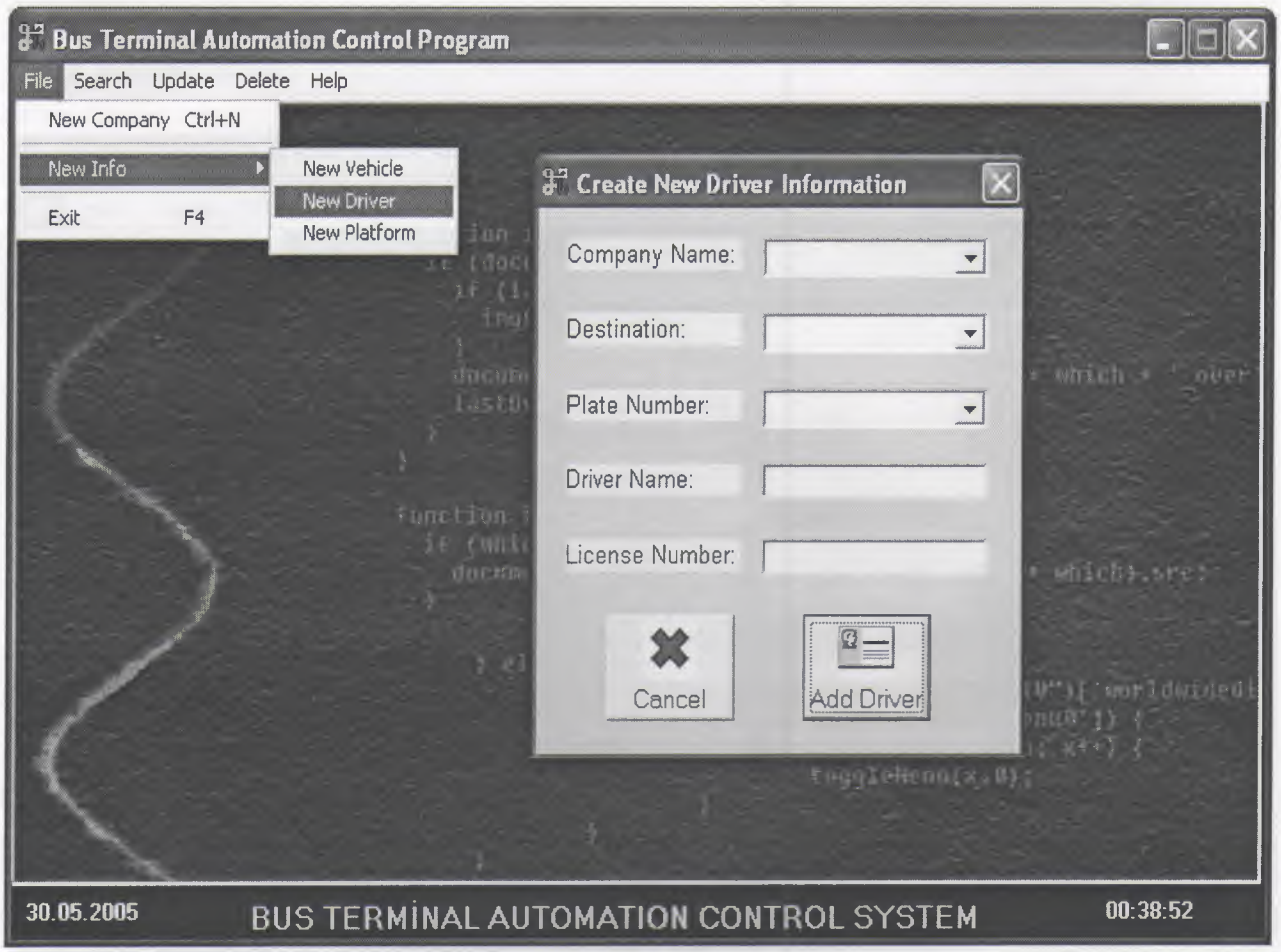
When the user click on the add vehicle button, create a new vehicle in a company.



3.1.2.2.NEW DRİVER MENU FORM:

This form is the create a new driver form of the program. If the user wants to insert a new driver. We can create new driver informations by using this form. In the form, information about company as name, destination, plate number, driver name and license number is entered and save a database.

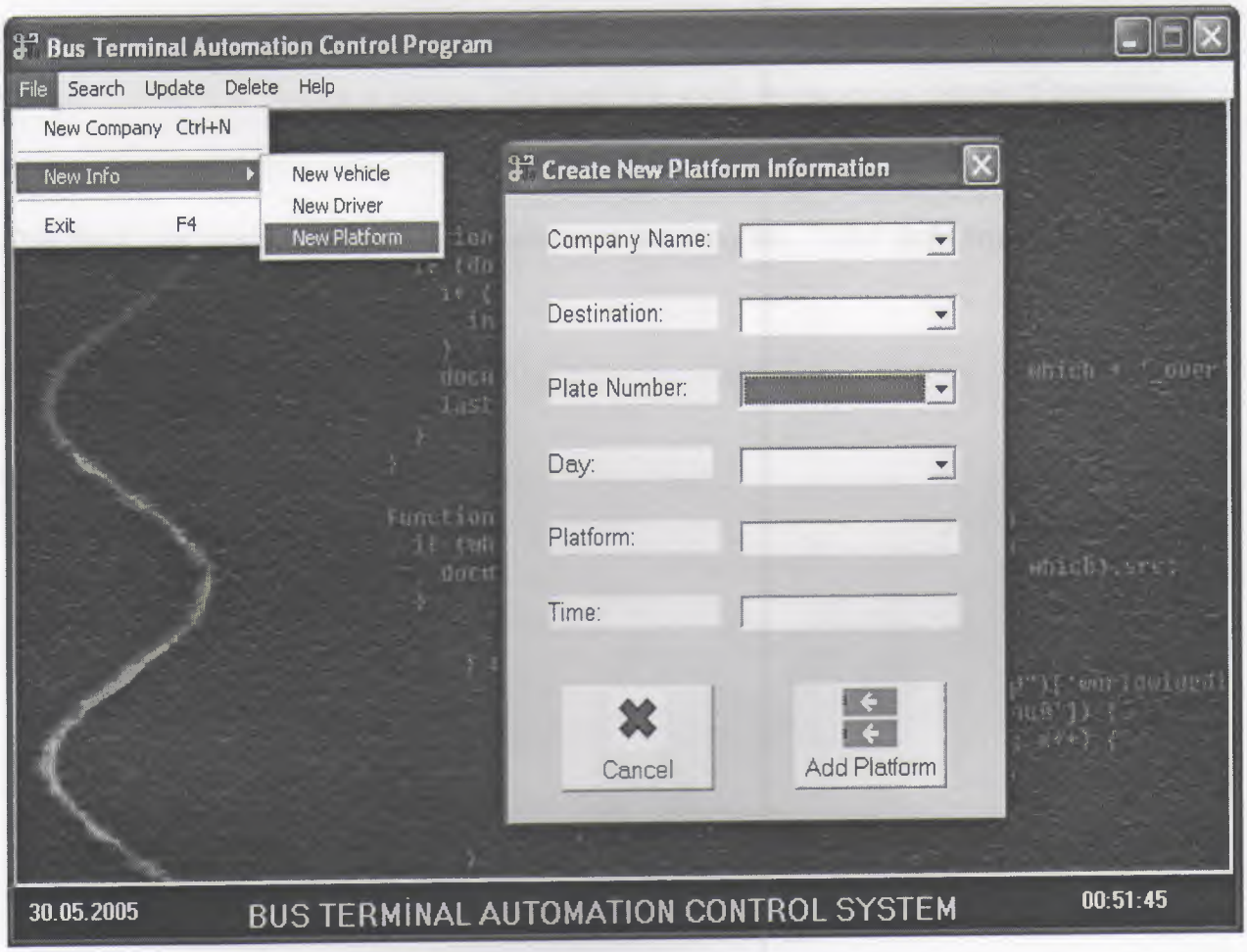
When the user click on the add driver button, create a new driver in a company.



3.1.2.3.NEW PLATFORM MENU FORM:

This form is the create new platform form of the program. If the user wants to insert a new platform. We can create new platform informations by using this form. In the form, information about company as name, destination, plate number, day, time and platform is entered and save a database.

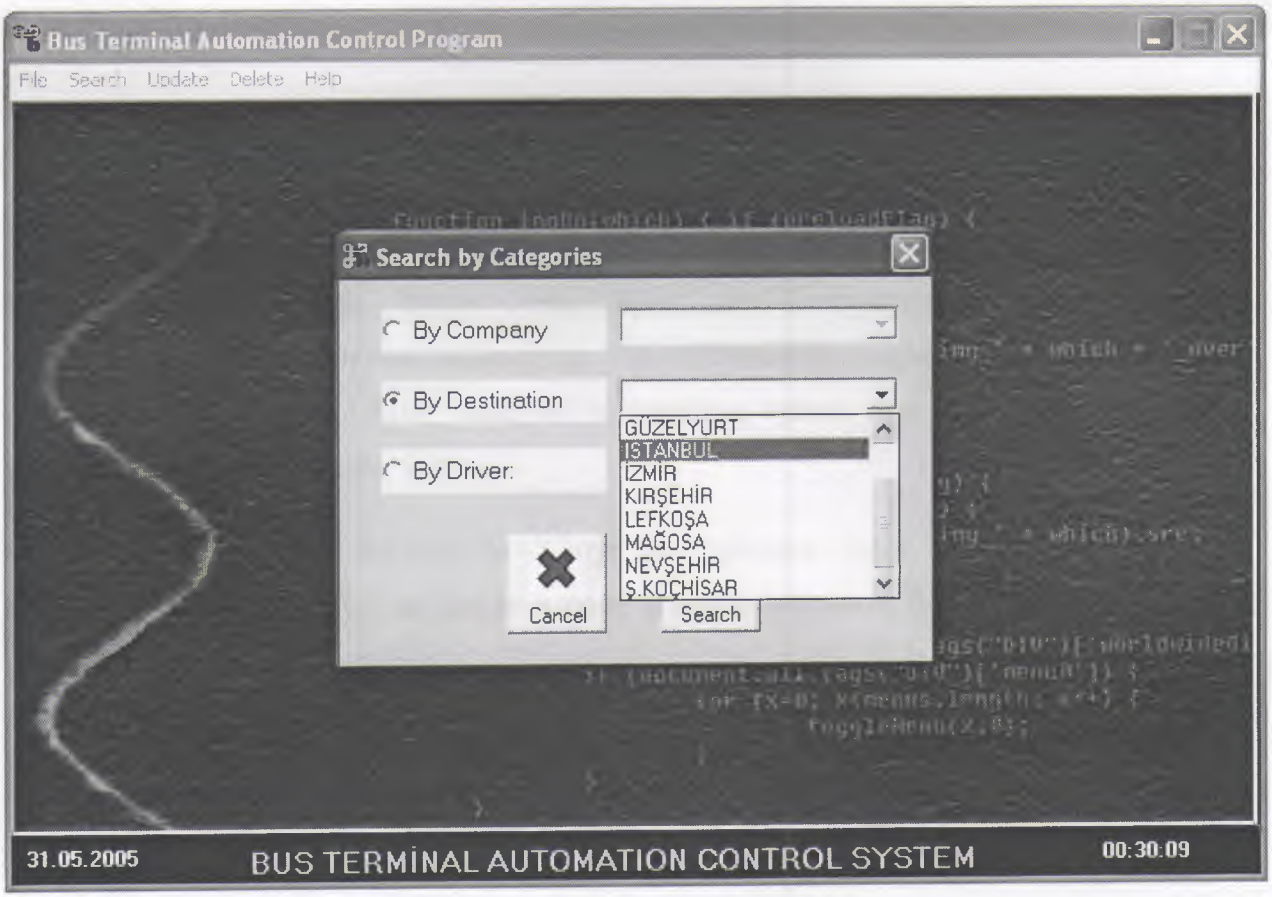
When the user click on the add platform button, create a new platform in a company.

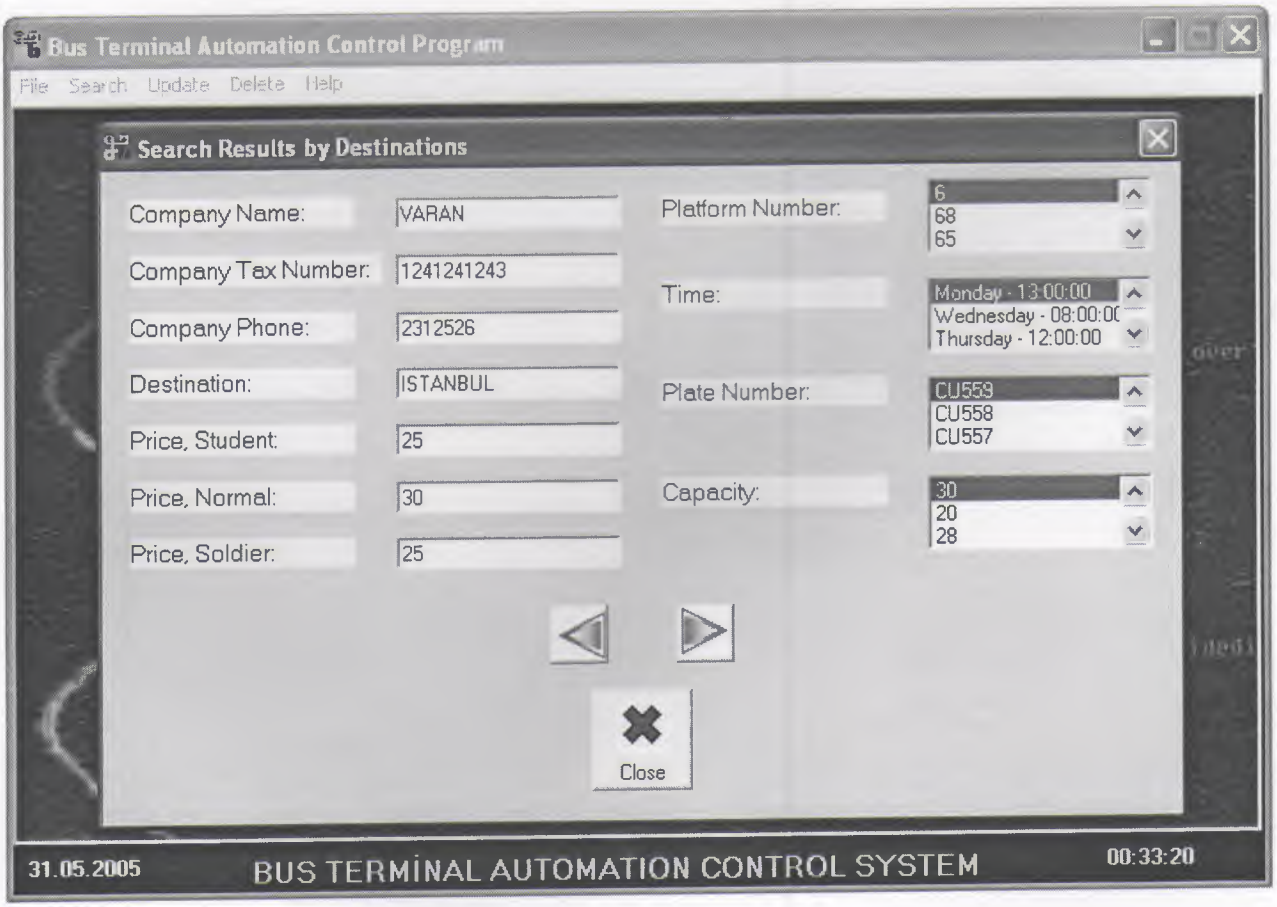


3.2.SEARCH MENU

This form name is search form of the program. This form is used to see all bus companys or ones that are searched by using destination, driver, company, sections in company. When mark to sections belong to companys in this form, we can pass to form that includes detail informations of company name, tax number, phone, price, time, platform number, plate number, capacity and history of company. If to mark a destination we can pass to form that includes detail informations of destination, company name, tax number, phone, price, platform number, time, plate number, capacity and history of company. If to mark a driver we can pass to form that includes detail informations of driver name, company name, license number and history of driver. This form was used to see easier names of companys, destination, driver.

This form was named as search bycategories in the program. Codes and form appreance of this search form is following down.


Bus Terminal Automation Control Program
File Search Update Delete Help
Search by Categories
By Company
By Destination
By Driver:
GÜZELYURT
İSTANBUL
İZMİR
KIRŞEHİR
LEFKOŞA
MAĞOSA
NEVŞEHİR
Ş.KOÇHİSAR
Cancel
Search
31.05.2005
BUS TERMİNAL AUTOMATION CONTROL SYSTEM
00:30:09


Bus Terminal Automation Control Program
File Search Update Delete Help
Search Results by Destinations
Company Name:
VARAN
Company Tax Number:
1241241243
Company Phone:
2312526
Destination:
İSTANBUL
Price, Student:
25
Price, Normal:
30
Price, Soldier:
25
Platform Number:
6
68
65
Time:
Monday - 13:00:00
Wednesday - 08:00:00
Thursday - 12:00:00
Plate Number:
CU559
CU558
CU557
Capacity:
30
20
28
Close
31.05.2005
BUS TERMİNAL AUTOMATION CONTROL SYSTEM
00:33:20

3.3.UPDATE MENU

3.3.1.UPDATE COMPANY MENU

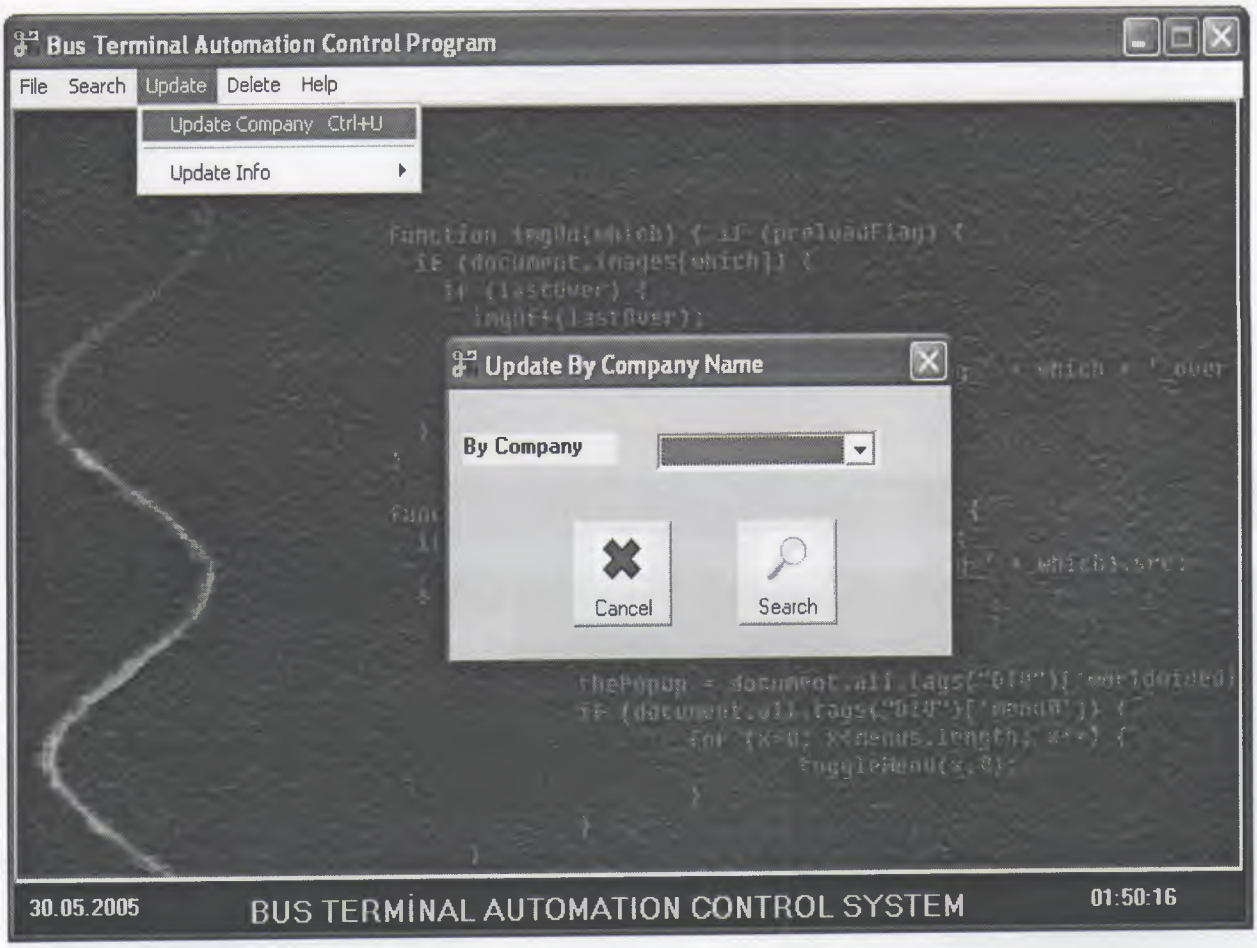
This form name is update company form of the program. This form is used to see all bus companys. That are searched by using company. Before user to choose by company name then click the search button here open the company information form. Which changes the informations is changing. This informations company name, tax number, phone, destination, price. After click the information button, user can replace the old information with the new information.

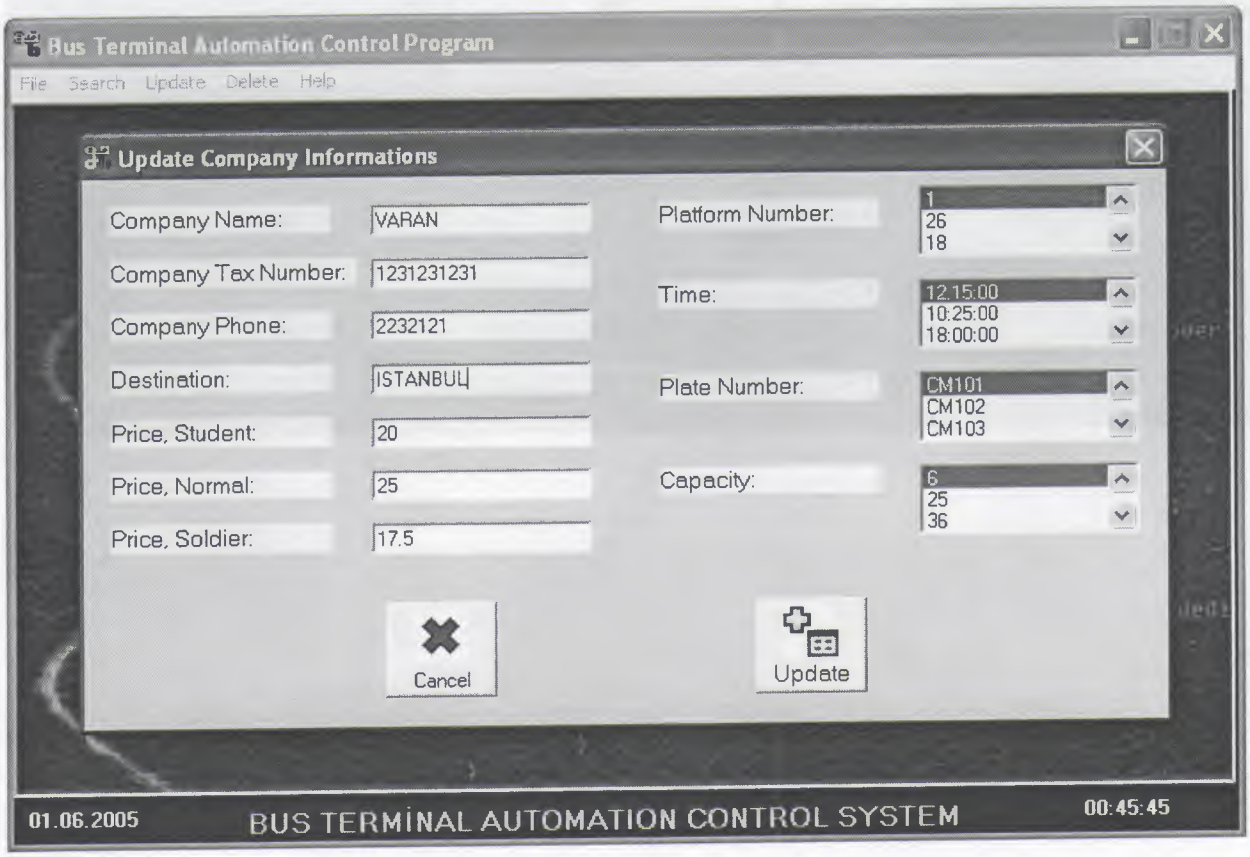
The program for the user to change the company name, tax number, phone, destination and price. When an update is done in any information, this updating occurs in all tables not to lose information.

3.3.2.UPDATE DRIVER MENU

Form name is update driver form of the program. Before choose to by driver name and then click the search button. In front of open the update driver form. When the user clicks the "Update' button, user can replace the old information with the new information.

When an update is done in any information, this updating occurs in all tables not to lose information.

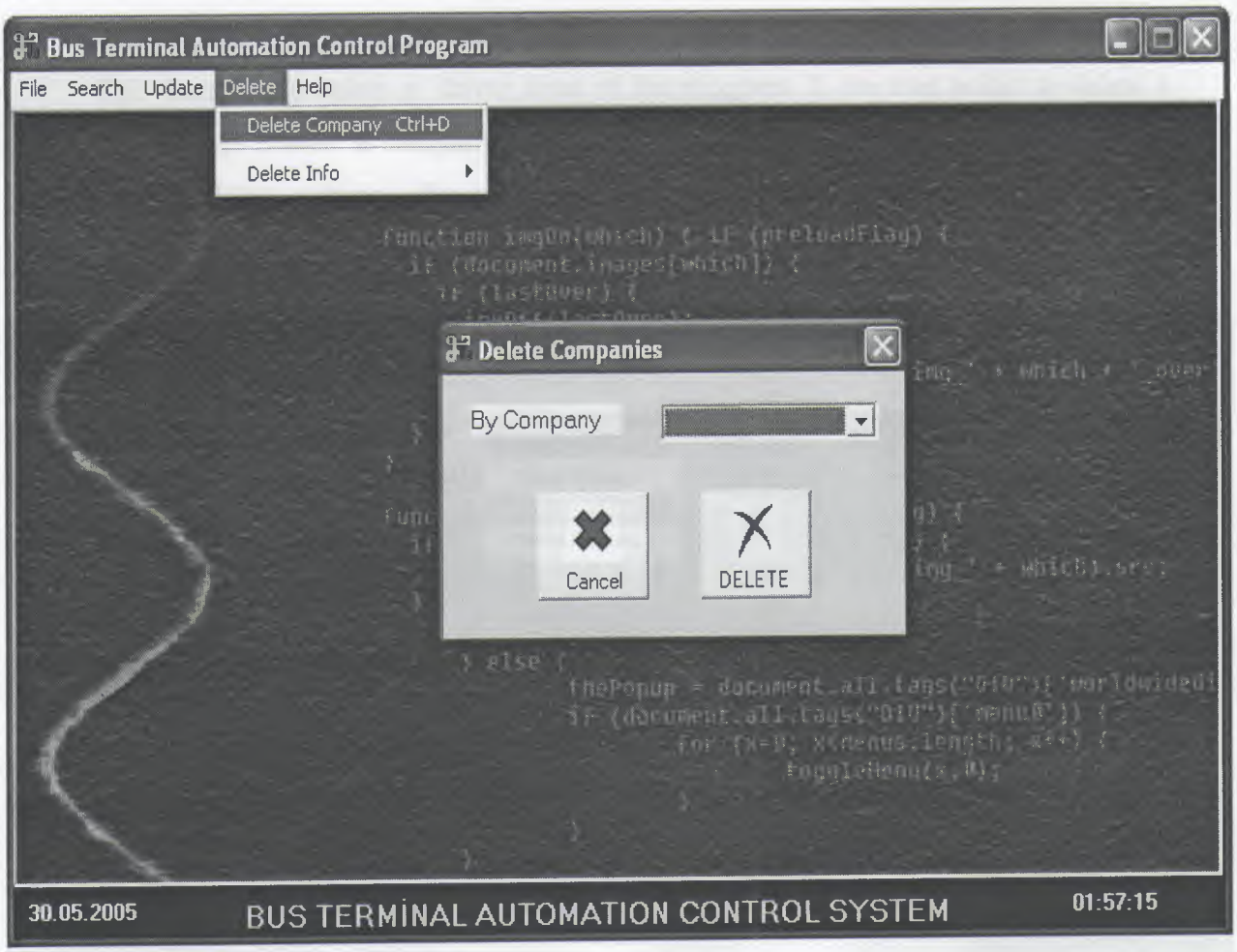

Bus Terminal Automation Control Program
File Search Update Delete Help
Update Company Ctrl+U
Update Info
Update By Company Name
By Company
Cancel
Search
30.05.2005
BUS TERMİNAL AUTOMATION CONTROL SYSTEM
01:50:16


Bus Terminal Automation Control Program
File Search Update Delete Help
Update Company Informations
Company Name: VARAN
Company Tax Number: 1231231231
Company Phone: 2232121
Destination: ISTANBUL
Price, Student: 20
Price, Normal: 25
Price, Soldier: 17.5
Platform Number: 1 26 18
Time: 12:15:00 10:25:00 18:00:00
Plate Number: CM101 CM102 CM103
Capacity: 6 25 36
Cancel
Update
01.06.2005
BUS TERMİNAL AUTOMATION CONTROL SYSTEM
00:45:45

3.4.DELETE MENU

3.4.1.DELETE COMPANY MENU

This form is the delete company form of the program. We can delete company informations by using this form. Before user to choose by company name. When the user click to 'Delete' button delete a all company information. When a deletion occurs, deletion occurs all data tables.



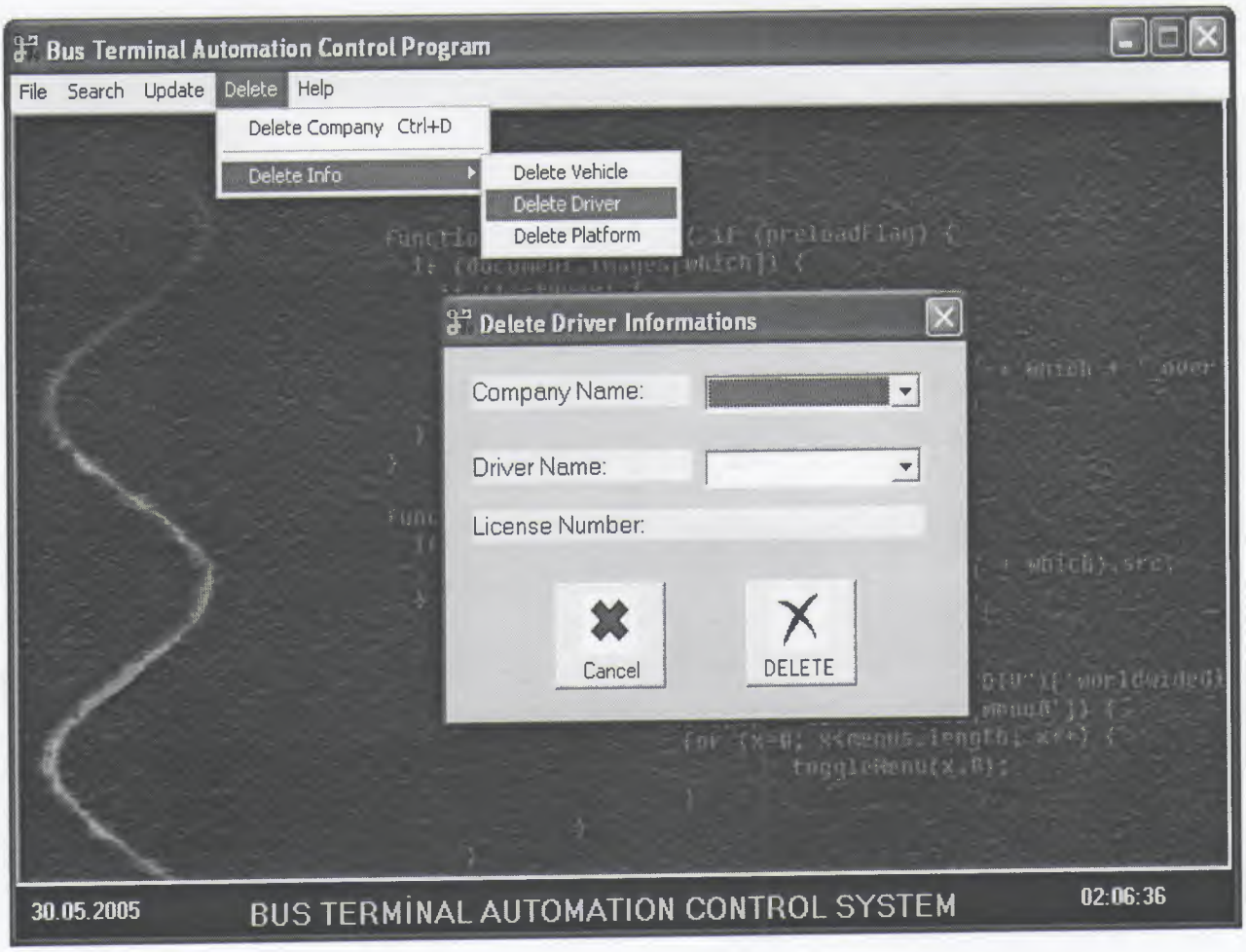
3.4.2.DELETE INFO MENU

3.4.2.1.DELETE VEHİCLE MENU FORM:

This form is the delete a vehicle form of the program. If the user wants to delete a driver. Firstly user choose a company name and plate number after when the user click to 'Delete' button delete a vehicle information. When a deletion occurs, deletion occurs all data tables.

3.4.2.2.DELETE DRIVER MENU FORM:

This form is the delete a driver form of the program. If the user wants to delete a driver. Firstly user choose a company name and driver name after when the user click to 'Delete' button delete a driver information. When a deletion occurs, deletion occurs all data tables.



3.4.2.3.DELETE PLATFORM MENU FORM:

This form is the delete a platform form of the program. If the user wants to delete a platfom. Firstly user choose a company name and platform number after when the user click to 'Delete' button delete a platform information. When a deletion occurs, deletion occurs all data tables.

Conclusion

Nowadays, windows oriented programs became more popular and flexible. Visual Basic 6.0 is one of the best well-known programming language based on window's environment. That's why I prefer this project. Now I can understand why these programming languages are very popular. Even I do not have experience with Visual Basic, this project did not become difficult to me. Visual Basic 6.0 has lots of help than other programming languages.

In my project, I have used important components of Visual Basic 6.0. Therefore I learned these components very well. Now I can use these components of Visual Basic 6.0 in an efficient manner. Also I have learned how to use new data access logic, which is ActiveX Data Objects (ADO). Additionally, I have used a database in my project. So I have gained many practices, experiences and knowledge of database. As known, database is very important topic for software programmers.

Finally, most important thing is for me that I have learned how to prepare an individual software project by using Visual Basic 6.0 to real life problems. After I have started my projects, I saw that you could face with unexpected real life problems. These real life problems are very different from the courses problem. This project became a good exercise to me for the real life and I used the things in my project that I learned from courses as theoretically.

I want to thank firstly my supervisor Mr. Okan DONANGİL and to him for his great helps to me while I was doing my graduation project.

References

- Ihsan Karagülle ; Zeydin Pala (1999). Microsoft Visual Basic 6.0 Pro. Istanbul. Türkmen Press.
- Visual Basic 6.0 How To Program H. M. Deitel, P. J. Deitel, T. R. Nieto 1999 Prentice-Hall, Inc
- Introduction to Oracle: SQL and PL/SQL Neena Kochhar, Ellen Gravina, Priya Nathan July 1999-Jerry Brosnan
- Database System Peter Roob 1993-Wads Worth Publishing
- Visual Basic Lecture Note: Ümit İlhan 2003-2004 Fall Semester

APPENDIX

All codes in this project are in this section.

CODES OF THE PROGRAM

Codes of Main Menu

```
Private Sub Form_Unload(Cancel As Integer)
   End
End Sub

Private Sub mnu_Delete_Company_Click()
   frmDeleteCompany.Show
End Sub

Private Sub mnu_Delete_Info_Driver_Click()
   frmDeleteInfoDriver.Show
End Sub

Private Sub mnu_Delete_Info_Platform_Click()
   frmDeleteInfoPlatform.Show
End Sub

Private Sub mnu_Delete_Info_Vehicle_Click()
   frmDeleteInfoVehicle.Show
End Sub

Private Sub mnu_Exit_Click()
   End
End Sub

Private Sub mnu_New_Company_Click()
   frmNewEntry.Show
End Sub

Private Sub mnu_New_Info_Driver_Click()
   frmNewDriver.Show
End Sub
```

```
Private Sub mnu_New_Info_Platform_Click()
  frmNewPlatform.Show
End Sub

Private Sub mnu_New_Info_Time_Click()
  frmNewTime.Show
End Sub

Private Sub mnu_New_Info_Vehicle_Click()
  frmNewVehicle.Show
End Sub

Private Sub mnu_Search_Company_Click()
  frmSearch.Show
End Sub

Private Sub mnu_Update_Company_Click()
  frmUpdateCompany.Show
End Sub

Private Sub mnu_Update_Info_Driver_Click()
  frmUpdateDriver.Show
End Sub

Private Sub mnuExit_Click()
   End
End Sub


Private Sub mnuAbout_Click()
   frmAbout.Show
End Sub


Private Sub Timer1_Timer()
   Label1.Caption = Date
   Label2.Caption = Time
End Sub
```

Codes of New Entry Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim Step As Integer
```

```
Dim SQL As String
Dim MaxCompanyID As Long
Dim DayNumber As Integer

Private Sub cmdBack_Click()
On Error GoTo ErrorHandler
frameInfo(Step).Visible = False 'O anki step kapanır.
frameInfo(Step - 1).Visible = True 'Bir önceki step açılır.
Step = Step - 1 'Step azaltılır.
If Step = 1 Then cmdBack.Enabled = False '1. stepte ise back tuşu kapanır.
If Not Step = 5 Then
   cmdNext.Enabled = True
   cmdFinish.Visible = False
End If

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdCancel_Click()
Unload Me
End Sub


Private Sub cmdFinish_Click() 'Finish tuşuna basıldığında çalışır ve kaydı gerçekleştirir.
On Error GoTo ErrorHandler

Select Case cmbDay.Text
   Case "Monday"
      DayNumber = 1
   Case "Tuesday"
      DayNumber = 2
   Case "Wednesday"
      DayNumber = 3
   Case "Thursday"
      DayNumber = 4
   Case "Friday"
      DayNumber = 5
   Case "Saturday"
      DayNumber = 6
   Case "Sunday"
      DayNumber = 7
End Select

SQL = "SELECT Name FROM tblCompany"
If RS.State = 0 Then
```

```
    RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
  Else
    Set RS = Conn.Execute(SQL)
  End If
  Do While Not RS.EOF 'Döngü ile tüm şirketleri kontrol ediyor.
    If txt1Name.Text = RS!Name Then 'Eğer aynı adlı bir şirket zaten varsa kaydı
  gerçekleştirmiyor ve mesaj veriyor.
      MsgBox "This company name is already in database.", vbExclamation + vbOKOnly
      Exit Sub
    End If
    RS.MoveNext
  Loop

  'Step 1, Creating Company Informations
  SQL1 = "INSERT INTO tblCompany (Name,Phone,TaxNo) VALUES ('" & txt1Name.Text
  & "', '" & txt1Phone.Text & "', '" & txt1TaxNo.Text & "')"
  Conn.Execute (SQL1)
  SQLRead1 = "SELECT MAX(ID) AS MaxID FROM tblCompany"
  If RS.State = 0 Then
    RS.Open SQLRead1, Conn, adOpenStatic, adLockReadOnly
  Else
    Set RS = Conn.Execute(SQLRead1)
  End If
  MaxCompanyID = RS("MaxID")

  'MaxID ile bir önceki işlemde yaratılan kaydın ID nosunu alıyor.
  'Bunu sonraki kayıtta relation için ikinci tabloya yazıyor.
  'MAX(FieldName): Verilen sütundaki maximum değeri alır.
  'AS FieldName: RS'te FieldName olarak kullanılır. RS!FieldName=RS(Max("ID")) gibi.
  'Aynı şekilde SUM gibi fonksiyonlar da vardır. SUM(Field1,Field2) iki sütunun o anki
  değerlerini toplar.

  'Step 2, Creating Destination Informations
  SQL2 = "INSERT INTO tblDestination (CompanyID,ToWhere,PriceSt,PriceNorm,PriceSol)
  VALUES (" & MaxCompanyID & ", '" & txt2Dest.Text & "', '" & txt2PriceSt.Text & "', '" &
  txt2PriceNorm.Text & "', '" & txt2PriceSol.Text & "')"
  Conn.Execute (SQL2)
  SQLRead2 = "SELECT MAX(ID) AS MaxID FROM tblDestination"
  Set RS = Conn.Execute(SQLRead2)
  MaxDestID = RS("MaxID")

  'Step 3, Creating Vehicle Informations
  SQL3 = "INSERT INTO tblVehicle (DestinationID,VehicleNumber,Capacity) VALUES (" &
  MaxDestID & ",'" & txt5Plate.Text & "', '" & txt5Capacity.Text & "')"
   Conn.Execute (SQL3)
   SQLRead3 = "SELECT MAX(ID) AS MaxID FROM tblVehicle"
   Set RS = Conn.Execute(SQLRead3)
   MaxVehicleID = RS("MaxID")

   'Step 4, Creating Driver Informations
```

```
SQL4 = "INSERT INTO tblDriver (VehicleID,DriverName,LicenseNo) VALUES (" &
MaxVehicleID & ", '" & txt3Name.Text & "', '" & txt3LicenseNo.Text & "')"
Conn.Execute (SQL4)

'Step 5, Creating Platform Informations
SQL5 = "INSERT INTO tblGo (VehicleID,Platform,GoingDay,GoingTime) VALUES (" &
MaxVehicleID & ", '" & txt4PlatformNo.Text & "', '" & CInt(DayNumber) & "', '" &
txt4Time.Text & "')"
Conn.Execute (SQL5)

MsgBox "New entry saved.", vbInformation + vbOKOnly
cmdNew.Visible = True
cmdBack.Enabled = False
cmdNext.Enabled = False
cmdFinish.Visible = False

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdNext_Click()
On Error GoTo ErrorHandler
frameInfo(Step).Visible = False 'Backtekilerin tam tersi!
frameInfo(Step + 1).Visible = True
Step = Step + 1
If Step = 5 Then
   cmdNext.Enabled = False
   cmdFinish.Visible = True
End If
If Not Step = 1 Then cmdBack.Enabled = True

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
frameInfo(1).Visible = True
Step = 1
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
```

```
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub Form_Unload(Cancel As Integer) 'Formun kapanması olayı.
On Error GoTo ErrorHandler
If RS.State = 1 Then RS.Close 'RS açıksa kapatır.
If Conn.State = 1 Then Conn.Close 'Conn açıksa kapatır.

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdNew_Click()
Unload Me
Me.Show
End Sub
```

Codes of New Vehicle Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT DISTINCT tblDestination.ToWhere FROM tblDestination INNER JOIN
tblCompany ON tblDestination.CompanyID = tblCompany.ID WHERE
tblCompany.Name='" & cmbCompany.Text & "'"
Set RS = Conn.Execute(SQL)
cmbDestination.Clear
Do While Not RS.EOF
   cmbDestination.AddItem RS("ToWhere")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
```

```
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub

Private Sub cmdAdd_Click()
On Error GoTo ErrorHandler
SQL = "SELECT ID FROM tblDestination WHERE ToWhere = '" & cmbDestination.Text &
"'"
Set RS = Conn.Execute(SQL)
SQL = "INSERT INTO tblVehicle (DestinationID,VehicleNumber,Capacity) VALUES (" &
RS("ID") & ", '" & txtPlateNo.Text & "', " & txtCapacity.Text & ")"
Conn.Execute (SQL)
MsgBox "Vehicle added.", vbInformation + vbOKOnly
Unload Me
Me.Show

ErrorHandler:
If Err.Number <> 0 Then
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub


Private Sub cmdCancel_Click()
Unload Me
End Sub


Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF
    cmbCompany.AddItem RS("Name")
    RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
```

```
End If
End Sub
```

Codes of New Driver Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT DISTINCT tblDestination.ToWhere FROM tblDestination INNER JOIN
tblCompany ON tblDestination.CompanyID = tblCompany.ID WHERE
tblCompany.Name='" & cmbCompany.Text & "'"
Set RS = Conn.Execute(SQL)
cmbDest.Clear
Do While Not RS.EOF
   cmbDest.AddItem RS("ToWhere")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmbDest_Click()
On Error GoTo ErrorHandler
SQL = "SELECT DISTINCT tblVehicle.VehicleNumber FROM tblVehicle INNER JOIN
tblDestination ON tblVehicle.DestinationID = tblDestination.ID WHERE
tblDestination.ToWhere='" & cmbDest.Text & "'"
Set RS = Conn.Execute(SQL)
cmbPlate.Clear
Do While Not RS.EOF
   cmbPlate.AddItem RS("VehicleNumber")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdAdd_Click()
```

```
On Error GoTo ErrorHandler
SQL = "SELECT ID FROM tblVehicle WHERE VehicleNumber = '" & cmbPlate.Text & "'"
Set RS = Conn.Execute(SQL)
SQL = "INSERT INTO tblDriver (VehicleID,DriverName,LicenseNo) VALUES (" &
RS("ID") & ", '" & txtDriver.Text & "', " & txtLicense.Text & ")"
Conn.Execute (SQL)
MsgBox "Driver added.", vbInformation + vbOKOnly
Unload Me
Me.Show

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub cmdCancel_Click()
Unload Me
End Sub


Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF
   cmbCompany.AddItem RS("Name")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub
```

Codes of New Platform Menu

```
Dim Conn As New ADODB.Connection
```

```
Dim RS As New ADODB.Recordset


Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT DISTINCT tblDestination.ToWhere FROM tblDestination INNER JOIN
tblCompany ON tblDestination.CompanyID = tblCompany.ID WHERE
tblCompany.Name='" & cmbCompany.Text & "'"
Set RS = Conn.Execute(SQL)
cmbDest.Clear
Do While Not RS.EOF
   cmbDest.AddItem RS("ToWhere")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmbDest_Click()
On Error GoTo ErrorHandler
SQL = "SELECT DISTINCT tblVehicle.VehicleNumber FROM tblVehicle INNER JOIN
tblDestination ON tblVehicle.DestinationID = tblDestination.ID WHERE
tblDestination.ToWhere='" & cmbDest.Text & "'"
Set RS = Conn.Execute(SQL)
cmbPlate.Clear
Do While Not RS.EOF
   cmbPlate.AddItem RS("VehicleNumber")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdAdd_Click()
On Error GoTo ErrorHandler
Select Case cmbDay.Text
   Case "Monday"
      DayNumber = 1
   Case "Tuesday"
      DayNumber = 2
```

```
    Case "Wednesday"
        DayNumber = 3
    Case "Thursday"
        DayNumber = 4
    Case "Friday"
        DayNumber = 5
    Case "Saturday"
        DayNumber = 6
    Case "Sunday"
        DayNumber = 7
End Select

SQL = "SELECT ID FROM tblVehicle WHERE VehicleNumber = '" & cmbPlate.Text & "'"
Set RS = Conn.Execute(SQL)
SQL = "INSERT INTO tblGo (VehicleID,Platform,GoingDay,GoingTime) VALUES (" &
RS("ID") & ", " & txtPlatform.Text & ", '" & DayNumber & "', '" & txtTime.Text & "')"
Conn.Execute (SQL)
MsgBox "Platform added.", vbInformation + vbOKOnly
Unload Me
Me.Show

ErrorHandler:
If Err.Number <> 0 Then
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub


Private Sub cmdCancel_Click()
Unload Me
End Sub


Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF
    cmbCompany.AddItem RS("Name")
    RS.MoveNext
Loop
```

```
ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub
```

Codes of Search Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmdSearch_Click()
'Seçilen kategoriye göre farklı pencereler geliyor.
'Company kategorisi seçiliyse frmSearchResultsByCompany penceresi açılır.
'Destination kategorisi seçiliyse frmSearchResultsByDest penceresi açılır.
'TaxNo kategorisi seçiliyse frmSearchResultsByTaxNo penceresi açılır.

If frmSearch.optCompany.Value = True Then frmSearchResultsByCompany.Show
If frmSearch.optDest.Value = True Then frmSearchResultsByDest.Show
If frmSearch.optDriver.Value = True Then frmSearchResultsByTaxNo.Show
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
'On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

'Form yüklendiğinde, comboboxların içini databasedeki verilere göre doldurur.
SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF
   cmbCompany.AddItem RS("Name")
   RS.MoveNext
Loop

SQL = "SELECT DISTINCT ToWhere FROM tblDestination"
```

```
Set RS = Conn.Execute(SQL)
cmbDest.Clear
Do While Not RS.EOF
  cmbDest.AddItem RS("ToWhere")
  RS.MoveNext
Loop

SQL = "SELECT DISTINCT DriverName FROM tblDriver"
Set RS = Conn.Execute(SQL)
cmbDriver.Clear
Do While Not RS.EOF
  cmbDriver.AddItem RS("DriverName")
  RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
  MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
  Unload Me
End If
End Sub

Private Sub optCompany_Click()
'DisableAll fonksiyonu çağrılır ve yalnızca Company kategorisi açılır.
Call DisableAll
If optCompany.Value = True Then cmbCompany.Enabled = True
End Sub

Private Sub optDest_Click()
'DisableAll fonksiyonu çağrılır ve yalnızca Destination kategorisi açılır.
Call DisableAll
If optDest.Value = True Then cmbDest.Enabled = True
End Sub

Private Sub optDriver_Click()
'DisableAll fonksiyonu çağrılır ve yalnızca TaxNo kategorisi açılır.
Call DisableAll
If optDriver.Value = True Then cmbDriver.Enabled = True
End Sub

Private Sub DisableAll()
'Bu fonksiyon tüm comboboxları kullanıma kapatır. Ve sonra yalnızca gerekli olanı tekrar
açılır.
cmbCompany.Enabled = False
cmbDest.Enabled = False
cmbDriver.Enabled = False
End Sub
```

Codes of Search Result By Company

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim QryCrt As String

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
'On Error GoTo ErrorHandler
Dim DayArray
DayArray = Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday")

Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

QryCrt = frmSearch.cmbCompany.Text

SQL = "SELECT * FROM tblCompany INNER JOIN " & _
      "(tblDestination INNER JOIN " & _
      "(tblVehicle INNER JOIN " & _
      "tblGo ON tblGo.VehicleID = tblVehicle.ID) " & _
      "ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblCompany.Name='" & QryCrt & "'"

'Arama kategorisine göre tüm fieldlar doldurulur.

RS.Open SQL, Conn, adOpenKeyset, adLockOptimistic

txtName.Text = RS("Name")
txtTaxNo.Text = RS("TaxNo")
txtPhone.Text = RS("Phone")
txtDest.Text = RS("ToWhere")
txtPriceSt.Text = RS("PriceSt")
txtPriceNorm.Text = RS("PriceNorm")
txtPriceSol.Text = RS("PriceSol")

If Not RS.BOF Then RS.MoveFirst
```

```
lstPlatform.Clear
Do While Not RS.EOF
   lstPlatform.AddItem RS("Platform")
   RS.MoveNext
Loop

If Not RS.BOF Then RS.MoveFirst
lstTime.Clear
Do While Not RS.EOF
   lstTime.AddItem DayArray(RS("GoingDay") - 1) & " - " & RS("GoingTime")
   RS.MoveNext
Loop

If Not RS.BOF Then RS.MoveFirst
lstPlate.Clear
Do While Not RS.EOF
   lstPlate.AddItem RS("VehicleNumber")
   RS.MoveNext
Loop

If Not RS.BOF Then RS.MoveFirst
lstCapacity.Clear
Do While Not RS.EOF
   lstCapacity.AddItem RS("Capacity")
   RS.MoveNext
Loop

'Verilen şirkette araç kaydı yoksa (RecordCount=0) başka bir SQL çalışıyor.
'Bu SQL, araçları işe karıştırmıyor, yalnızca şirket adı ve istikamet bilgisi verilir.

If RS.RecordCount < 1 Then
   SQL = "SELECT * FROM tblCompany INNER JOIN tblDestination ON
tblDestination.CompanyID = tblCompany.ID WHERE tblCompany.Name='" & QryCrt & "'"
   Set RS = Conn.Execute(SQL)

   txtName.Text = RS("Name")
   txtTaxNo.Text = RS("TaxNo")
   txtPhone.Text = RS("Phone")
   txtDest.Text = RS("ToWhere")
   txtPriceSt.Text = RS("PriceSt")
   txtPriceNorm.Text = RS("PriceNorm")
   txtPriceSol.Text = RS("PriceSol")
End If

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)
If RS.State = 1 Then RS.Close
Conn.Close
End Sub


Private Sub lstCapacity_Click()
'ListIndex: lstboxtaki seçili olan değerin liste indexindeki değerini verir. 1. eleman 0 2.
eleman 1  ..... 5. eleman 4 tür.
lstPlatform.ListIndex = lstCapacity.ListIndex
lstTime.ListIndex = lstCapacity.ListIndex
lstPlate.ListIndex = lstCapacity.ListIndex
End Sub

Private Sub lstPlate_Click()
lstPlatform.ListIndex = lstPlate.ListIndex
lstTime.ListIndex = lstPlate.ListIndex
lstCapacity.ListIndex = lstPlate.ListIndex
End Sub

Private Sub lstPlatform_Click()
lstTime.ListIndex = lstPlatform.ListIndex
lstPlate.ListIndex = lstPlatform.ListIndex
lstCapacity.ListIndex = lstPlatform.ListIndex
End Sub

Private Sub lstTime_Click()
lstPlatform.ListIndex = lstTime.ListIndex
lstPlate.ListIndex = lstTime.ListIndex
lstCapacity.ListIndex = lstTime.ListIndex
End Sub
```

Codes of Result By Destinations

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim RS2 As New ADODB.Recordset
Dim QryCrt As String
Dim CompanyID As Integer
Dim DestinationID As Integer

Private Sub cmdNext_Click()
```

```
On Error Resume Next
If Not RS.EOF And RS.RecordCount > 1 Then
   RS.MoveNext
   SQL = "SELECT
tblCompany.Name,tblDestination.ToWhere,tblGo.GoingDay,tblGo.Platform,tblGo.GoingTim
e,tblVehicle.VehicleNumber,tblVehicle.Capacity FROM tblCompany INNER JOIN " & _
         "(tblDestination INNER JOIN (tblVehicle " & _
         "INNER JOIN tblGo ON tblVehicle.ID = tblGo.VehicleID)" & _
         "ON tblVehicle.DestinationID = tblDestination.ID)" & _
         "ON tblDestination.CompanyID = tblCompany.ID " & _
            "WHERE tblDestination.ToWhere = '" & RS("ToWhere") & "' " & _
            "AND tblCompany.Name = '" & RS("Name") & "'"

   Set RS2 = Conn.Execute(SQL)
   Call FillTop(True)
   Call FillBottom(True)
End If
End Sub


Private Sub cmdPrev_Click()
On Error Resume Next
If Not RS.BOF And RS.RecordCount > 1 Then
   RS.MovePrevious
   SQL = "SELECT
tblCompany.Name,tblDestination.ToWhere,tblGo.GoingDay,tblGo.Platform,tblGo.GoingTim
e,tblVehicle.VehicleNumber,tblVehicle.Capacity FROM tblCompany INNER JOIN " & _
         "(tblDestination INNER JOIN (tblVehicle " & _
         "INNER JOIN tblGo ON tblVehicle.ID = tblGo.VehicleID)" & _
         "ON tblVehicle.DestinationID = tblDestination.ID)" & _
         "ON tblDestination.CompanyID = tblCompany.ID " & _
            "WHERE tblDestination.ToWhere = '" & RS("ToWhere") & "' " & _
            "AND tblCompany.Name = '" & RS("Name") & "'"

   Set RS2 = Conn.Execute(SQL)
   Call FillTop(True)
   Call FillBottom(True)
End If
End Sub


Private Sub Command1_Click()
Unload Me
End Sub


Private Sub Form_Load()
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Set RS2 = New ADODB.Recordset
```

```
If Not Conn.State = 1 Then
   Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
   Conn.ConnectionString = App.Path & "/data/data.mdb"
   Conn.Open
End If

QryCrt = frmSearch.cmbDest.Text

Call FillTop(False)
Call FillBottom(False)

End Sub

Private Sub FillTop(isNext)
On Error GoTo ErrorHandler
If isNext = False Then
   SQL = "SELECT DISTINCT * FROM tblCompany INNER JOIN " & _
         "tblDestination " & _
         "ON tblDestination.CompanyID = tblCompany.ID " & _
            "WHERE tblDestination.ToWhere='" & frmSearch.cmbDest.Text & "'"

   If Not RS.State = 1 Then
      RS.Open SQL, Conn, adOpenKeyset, adLockOptimistic
   Else
      Set RS = Conn.Execute(SQL)
   End If
End If

   txtName.Text = RS("Name")
   CompanyID = RS("tblCompany.ID")
   txtTaxNo.Text = RS("TaxNo")
   txtPhone.Text = RS("Phone")
   txtDest.Text = RS("ToWhere")
   DestinationID = RS("tblDestination.ID")
   txtPriceSt.Text = RS("PriceSt")
   txtPriceNorm.Text = RS("PriceNorm")
   txtPriceSol.Text = RS("PriceSol")

ErrorHandler:
If Err.Number <> 0 And Not Err.Number = 3021 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   'Unload Me
End If
End Sub

Private Sub FillBottom(isNext)
'On Error GoTo ErrorHandler
Dim DayArray
```

```
DayArray = Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday")

If isNext = False Then
   SQL = "SELECT
tblGo.GoingDay,tblGo.Platform,tblGo.GoingTime,tblVehicle.VehicleNumber,tblVehicle.Cap
acity FROM tblCompany INNER JOIN " & _
         "(tblDestination INNER JOIN (tblVehicle " & _
         "INNER JOIN tblGo ON tblVehicle.ID = tblGo.VehicleID)" & _
         "ON tblVehicle.DestinationID = tblDestination.ID)" & _
         "ON tblDestination.CompanyID = tblCompany.ID " & _
            "WHERE tblDestination.ToWhere = '" & QryCrt & "' " & _
            "AND tblCompany.Name = '" & txtName.Text & "'"
   'txtName.Text = SQL
   If Not RS2.State = 1 Then
      RS2.Open SQL, Conn, adOpenKeyset, adLockOptimistic
   Else
      Set RS2 = Conn.Execute(SQL)
   End If
End If

If Not RS2.BOF Then RS2.MoveFirst
lstPlatform.Clear
Do While Not RS2.EOF
   lstPlatform.AddItem RS2("Platform")
   RS2.MoveNext
Loop

If Not RS2.BOF Then RS2.MoveFirst
lstTime.Clear
Do While Not RS2.EOF
   lstTime.AddItem DayArray(RS2("GoingDay") - 1) & " - " & RS2("GoingTime")
   RS2.MoveNext
Loop

If Not RS2.BOF Then RS2.MoveFirst
lstPlate.Clear
Do While Not RS2.EOF
   lstPlate.AddItem RS2("VehicleNumber")
   RS2.MoveNext
Loop

If Not RS2.BOF Then RS2.MoveFirst
lstCapacity.Clear
Do While Not RS2.EOF
   lstCapacity.AddItem RS2("Capacity")
   RS2.MoveNext
Loop

ErrorHandler:
```

```
If Err.Number <> 0 And Not Err.Number = 3021 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   'Unload Me
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
If RS.State = 1 Then RS.Close
If RS2.State = 1 Then RS2.Close
If Conn.State = 1 Then Conn.Close
End Sub

Private Sub lstCapacity_Click()
'ListIndex: lstboxtaki seçili olan değerin liste indexindeki değerini verir. 1. eleman 0 2.
eleman 1  ..... 5. eleman 4 tür.
lstPlatform.ListIndex = lstCapacity.ListIndex
lstTime.ListIndex = lstCapacity.ListIndex
lstPlate.ListIndex = lstCapacity.ListIndex
End Sub


Private Sub lstPlate_Click()
lstPlatform.ListIndex = lstPlate.ListIndex
lstTime.ListIndex = lstPlate.ListIndex
lstCapacity.ListIndex = lstPlate.ListIndex
End Sub

Private Sub lstPlatform_Click()
lstTime.ListIndex = lstPlatform.ListIndex
lstPlate.ListIndex = lstPlatform.ListIndex
lstCapacity.ListIndex = lstPlatform.ListIndex
End Sub

Private Sub lstTime_Click()
lstPlatform.ListIndex = lstTime.ListIndex
lstPlate.ListIndex = lstTime.ListIndex
lstCapacity.ListIndex = lstTime.ListIndex
End Sub
```

Codes of Result By Driver

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim QryCrt As String
```

```
Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

QryCrt = frmSearch.cmbDriver.Text

SQL = "SELECT tblCompany.Name,tblDriver.DriverName,tblDriver.LicenseNo FROM
tblCompany INNER JOIN " & _
    "(tblDestination INNER JOIN " & _
    "(tblVehicle INNER JOIN " & _
    "tblDriver ON tblDriver.VehicleID = tblVehicle.ID) " & _
    "ON tblVehicle.DestinationID = tblDestination.ID) " & _
    "ON tblDestination.CompanyID = tblCompany.ID " & _
      "WHERE tblDriver.DriverName='" & QryCrt & "'"

RS.Open SQL, Conn, adOpenKeyset, adLockOptimistic
txtName.Text = RS("Name")
txtDriver.Text = RS("DriverName")
txtLicense.Text = RS("LicenseNo")

If txtName.Text = "" Or txtName.Text = Null Then
  SQL = "SELECT * FROM tblDriver WHERE DriverName='" & QryCrt & "'"
  Set RS = Conn.Execute(SQL)
  txtDriver.Text = RS("DriverName")
  txtLicense.Text = RS("LicenseNo")
End If

ErrorHandler:
If Err.Number <> 0 Then
  MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
  Unload Me
End If
End Sub


Private Sub Form_Unload(Cancel As Integer)
If RS.State = 1 Then RS.Close
Conn.Close
End Sub
```

Codes of Update Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim QryCrt As String

Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
QryCrt = frmUpdateCompany.cmbCompany.Text

SQL = "SELECT * FROM tblCompany INNER JOIN " & _
      "(tblDestination INNER JOIN " & _
      "(tblVehicle INNER JOIN " & _
      "tblGo ON tblGo.VehicleID = tblVehicle.ID) " & _
      "ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblCompany.Name='" & QryCrt & "'"

If RS.State = 1 Then RS.Close

RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic

'Tekrar yazılan bilgiler veritabanına işlenir ve değişiklikler RS.Update komutu ile
gerçekleştirilir.

RS!Name = txtName.Text
RS!ToWhere = txtDest.Text
RS!TaxNo = txtTaxNo.Text
RS!Phone = txtPhone.Text
RS!PriceSt = txtPriceSt.Text
RS!PriceNorm = txtPriceNorm.Text
RS!PriceSol = txtPriceSol.Text
RS.Update
RS.Close
MsgBox "Company informations updated.", vbInformation + vbOKOnly
Unload Me

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
'On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
```

```
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

QryCrt = frmUpdateCompany.cmbCompany.Text

SQL = "SELECT * FROM tblCompany INNER JOIN tblDestination ON
tblDestination.CompanyID = tblCompany.ID WHERE tblCompany.Name = '" & QryCrt &
"'"
RS.Open SQL, Conn, adOpenKeyset, adLockOptimistic
'Set RS = Conn.Execute(SQL)

txtName.Text = RS("Name")
txtTaxNo.Text = RS("TaxNo")
txtPhone.Text = RS("Phone")
txtDest.Text = RS("ToWhere")
txtPriceSt.Text = RS("PriceSt")
txtPriceNorm.Text = RS("PriceNorm")
txtPriceSol.Text = RS("PriceSol")

SQL = "SELECT * FROM tblCompany INNER JOIN " & _
    "(tblDestination INNER JOIN " & _
    "tblVehicle " & _
    "ON tblVehicle.DestinationID = tblDestination.ID) " & _
    "ON tblDestination.CompanyID = tblCompany.ID " & _
      "WHERE tblCompany.Name='" & QryCrt & "'"
Set RS = Conn.Execute(SQL)

If Not RS.BOF Then RS.MoveFirst
lstPlate.Clear
Do While Not RS.EOF
  lstPlate.AddItem RS("VehicleNumber")
  RS.MoveNext
Loop

If Not RS.BOF Then RS.MoveFirst
lstCapacity.Clear
Do While Not RS.EOF
  lstCapacity.AddItem RS("Capacity")
  RS.MoveNext
Loop

SQL = "SELECT * FROM tblCompany INNER JOIN " & _
    "(tblDestination INNER JOIN " & _
    "(tblVehicle INNER JOIN " & _
    "tblGo ON tblGo.VehicleID = tblVehicle.ID) " & _
    "ON tblVehicle.DestinationID = tblDestination.ID) " & _
    "ON tblDestination.CompanyID = tblCompany.ID " & _
      "WHERE tblCompany.Name='" & QryCrt & "'"
```

```
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly

'Mevcut şirketler yazılıyor.

cmbCompany.Clear
Do While Not RS.EOF
   cmbCompany.AddItem RS("Name")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub
```

Codes of Update Driver Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmdSearch_Click()
frmUpdateInfoDriver.Show
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT DriverName FROM tblDriver"
```

```
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly

'Mevcut sürücüler yazılıyor.

cmbCompany.Clear
Do While Not RS.EOF
   cmbCompany.AddItem RS("DriverName")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub
```

Codes of Update Info Driver Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmdUpdate_Click()
RS.Close
SQL = "SELECT * FROM tblDriver WHERE DriverName = '" & CStr(Me.Caption) & "'"
RS.Open SQL, Conn, adOpenDynamic, adLockPessimistic
RS("DriverName") = txtDriver.Text
RS("LicenseNo") = txtLicense.Text
RS.Update
MsgBox "Driver informations updated.", vbInformation + vbOKOnly
Unload Me

'SQL = "UPDATE tblDriver SET DriverName = '" & txtDriver.Text & "' AND LicenseNo =
'" & txtLicense.Text & "' WHERE DriverName = '" & Me.Caption & "'"
'Set RS = Conn.Execute(SQL)
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
```

```
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

QryCrt = frmUpdateDriver.cmbCompany.Text

SQL = "SELECT tblCompany.Name,tblDriver.DriverName,tblDriver.LicenseNo FROM
tblCompany INNER JOIN " & _
      "(tblDestination INNER JOIN " & _
      "(tblVehicle INNER JOIN " & _
      "tblDriver ON tblDriver.VehicleID = tblVehicle.ID) " & _
      "ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblDriver.DriverName='" & QryCrt & "'"

RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
txtName.Text = RS("Name")
txtDriver.Text = RS("DriverName")
txtLicense.Text = RS("LicenseNo")

Me.Caption = RS("DriverName")

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub Form_Unload(Cancel As Integer)
If RS.State = 1 Then RS.Close
Conn.Close
End Sub
```

Codes of Delete Company Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim ErrorHandler As Label


Private Sub cmdDelete_Click()
On Error GoTo ErrorHandler 'Hata oluşursa ErrorHandler ile başlayan bölüm çalışır ve
programın bu bölümünün çalışması durur.
SQL = "SELECT ID FROM tblCompany WHERE Name='" & cmbCompany.Text & "'"
Set RS = Conn.Execute(SQL)
CompanyID = RS("ID")
SQL = "SELECT ID FROM tblDestination WHERE CompanyID=" & CompanyID
Set RS = Conn.Execute(SQL)
```

```
DestinationID = RS("ID")
SQL = "SELECT ID FROM tblVehicle WHERE DestinationID=" & DestinationID
Set RS = Conn.Execute(SQL)
VehicleID = RS("ID")

Conn.Execute ("DELETE FROM tblDestination WHERE CompanyID=" & CompanyID)
Conn.Execute ("DELETE FROM tblVehicle WHERE DestinationID=" & DestinationID)
Conn.Execute ("DELETE FROM tblGo WHERE VehicleID=" & VehicleID)
Conn.Execute ("DELETE FROM tblDriver WHERE VehicleID=" & VehicleID)

SQL = "DELETE FROM tblCompany WHERE Name='" & cmbCompany.Text & "'"
Conn.Execute (SQL) 'Seçili şirketi siler.
MsgBox "Company deleted.", vbInformation + vbOKOnly
Unload Me

ErrorHandler:
If Err.Number <> 0 Then 'Hata varsa! Err.Number = 0 ise hata yok demektir.
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
'          Hata açıklaması. X işareti    Yalnızca OK düğmesi              Hata Numarası
   Unload Me
End If
End Sub


Private Sub Command1_Click()
Unload Me
End Sub


Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear 'Şirket listesini temizliyor.
Do While Not RS.EOF 'Şirket listesi oluşturuyor.
   cmbCompany.AddItem RS("Name") 'Şirket kaydını ekliyor.
   RS.MoveNext 'Bir sonraki kayda gidiyor.
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
```

```
End Sub
```

Codes of Delete Info Vehicle Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT * FROM tblCompany INNER JOIN " & _
      "(tblDestination INNER JOIN " & _
      "tblVehicle ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblCompany.Name='" & cmbCompany.Text & "'"

If RS.State = 0 Then
   RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
Else
   Set RS = Conn.Execute(SQL)
End If

cmbVehicle.Clear
Do While Not RS.EOF
   cmbVehicle.AddItem RS("VehicleNumber")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub cmbVehicle_Click()
On Error GoTo ErrorHandler
SQL = "SELECT Capacity FROM tblVehicle WHERE VehicleNumber = '" &
cmbVehicle.Text & "'"
If RS.State = 0 Then
   RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
Else
   Set RS = Conn.Execute(SQL)
End If
lblCapacity.Caption = RS!Capacity

ErrorHandler:
```

```
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdDelete_Click()
On Error GoTo ErrorHandler
SQL = "DELETE FROM tblVehicle WHERE VehicleNumber='" & cmbVehicle.Text & "'"
Conn.Execute (SQL)
MsgBox "Vehicle deleted.", vbInformation + vbOKOnly
Unload Me

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF
   cmbCompany.AddItem RS("Name")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub
```

Codes of Delete Info Driver Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT * FROM tblCompany INNER JOIN " & _
      "(tblDestination INNER JOIN " & _
      "(tblVehicle INNER JOIN " & _
      "tblDriver ON tblDriver.VehicleID = tblVehicle.ID) " & _
      "ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblCompany.Name='" & cmbCompany.Text & "'"

If RS.State = 0 Then 'State: Durum.
   RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic ' RS kapalıysa açıyor.
Else
   Set RS = Conn.Execute(SQL) 'Açıksa yalnızca çalıştırıyor.
End If

cmbDriver.Clear
Do While Not RS.EOF
   cmbDriver.AddItem RS("DriverName")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub

Private Sub cmbDriver_Click()
On Error GoTo ErrorHandler
SQL = "SELECT LicenseNo FROM tblDriver WHERE DriverName = '" & cmbDriver.Text
& "'"
If RS.State = 0 Then 'Duruma göre açar ya da çalıştırır.
   RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
Else
   Set RS = Conn.Execute(SQL)
End If
lblLicense.Caption = RS!LicenseNo 'Ehliyet no yazdırılıyor.

ErrorHandler:
If Err.Number <> 0 Then
```

```
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub

Private Sub cmdDelete_Click()
On Error GoTo ErrorHandler
SQL = "DELETE FROM tblDriver WHERE LicenseNo='" & lblLicense.Caption & "'"
Conn.Execute (SQL) 'Sürücü siliniyor.
MsgBox "Driver deleted.", vbInformation + vbOKOnly
Unload Me

ErrorHandler:
If Err.Number <> 0 Then
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo ErrorHandler
Set Conn = New ADODB.Connection
Set RS = New ADODB.Recordset
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
Conn.ConnectionString = App.Path & "/data/data.mdb"
Conn.Open

SQL = "SELECT DISTINCT Name FROM tblCompany"
RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
cmbCompany.Clear
Do While Not RS.EOF 'Şirket listesi oluşturuluyor.
    cmbCompany.AddItem RS("Name")
    RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
    MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
    Unload Me
End If
End Sub
```

Codes of Delete Info Platform Menu

```
Dim Conn As New ADODB.Connection
Dim RS As New ADODB.Recordset

Private Sub cmbCompany_Click()
On Error GoTo ErrorHandler
SQL = "SELECT * FROM tblCompany INNER JOIN " & _
    "(tblDestination INNER JOIN " & _
    "(tblVehicle INNER JOIN " & _
      "tblGo ON tblGo.VehicleID = tblVehicle.ID) " & _
      "ON tblVehicle.DestinationID = tblDestination.ID) " & _
      "ON tblDestination.CompanyID = tblCompany.ID " & _
        "WHERE tblCompany.Name='" & cmbCompany.Text & "'"

' Inner Join iki tabloyu birleştirir.
' Inner Join kullanılırsa iki ayrı bağlantı oluşturmaya gerek kalmaz.
' SELECT * FROM tabloAdı INNER JOIN ikinci tablo adı ON Gerekli Koşul
' şeklinde kullanılır.
' Koşul: Vehicle.CompanyID = Company.ID şeklinde.

If RS.State = 0 Then 'Duruma göre rs açılıyor. ya da çalıştırılıyor.
   RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
Else
   Set RS = Conn.Execute(SQL)
End If

cmbPlatform.Clear
Do While Not RS.EOF
   cmbPlatform.AddItem RS("Platform")
   RS.MoveNext
Loop

ErrorHandler:
If Err.Number <> 0 Then
   MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
   Unload Me
End If
End Sub


Private Sub cmdDelete_Click()
On Error GoTo ErrorHandler
SQL = "SELECT tblGo.* FROM tblCompany INNER JOIN " & _
    "(tblDestination INNER JOIN " & _
    "(tblVehicle INNER JOIN " & _
    "tblGo ON tblGo.VehicleID = tblVehicle.ID) " & _
    "ON tblVehicle.DestinationID = tblDestination.ID) " & _
    "ON tblDestination.CompanyID = tblCompany.ID " & _
      "WHERE tblCompany.Name='" & cmbCompany.Text & "' AND " & _
```

```
        "tblGo.Platform=" & cmbPlatform.Text

    RS.Close
    RS.Open SQL, Conn, adOpenKeyset, adLockPessimistic
    RS.Delete
    MsgBox "Platform deleted.", vbInformation + vbOKOnly
    Unload Me

    ErrorHandler:
    If Err.Number <> 0 Then
       MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
       Unload Me
    End If
    End Sub


    Private Sub Command1_Click()
    Unload Me
    End Sub


    Private Sub Form_Load()
    On Error GoTo ErrorHandler
    Set Conn = New ADODB.Connection
    Set RS = New ADODB.Recordset
    Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
    Conn.ConnectionString = App.Path & "/data/data.mdb"
    Conn.Open

    SQL = "SELECT DISTINCT Name FROM tblCompany"
    RS.Open SQL, Conn, adOpenStatic, adLockReadOnly
    cmbCompany.Clear
    Do While Not RS.EOF
       cmbCompany.AddItem RS("Name")
       RS.MoveNext
    Loop
    ErrorHandler:
    If Err.Number <> 0 Then
       MsgBox Err.Description, vbCritical + vbOKOnly, "Error Occured, Code:" & Err.Number
       Unload Me
    End If
    End Sub
```

Codes of Splash

```
Private Sub Command1_Click()
   Unload Me
   frmMain.Show
```

```
End Sub

Private Sub Command2_Click()
End
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
  Unload Me
End Sub


Private Sub Form_Load()
   lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
End Sub
```

Codes of About Menu

```
' Reg Key Security Options...
Const KEY_ALL_ACCESS = &H2003F


' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                    ' Unicode nul terminated string
Const REG_DWORD = 4                 ' 32-bit number


Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"


Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal
hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As
Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef
lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Private Sub Form_Load()
   lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
   lblTitle.Caption = App.Title
End Sub


Private Sub cmdSysInfo_Click()
     Call StartSysInfo
```

```
End Sub

Private Sub cmdOK_Click()
    Unload Me
End Sub

Public Sub StartSysInfo()
  On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String


    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO,
gREGVALSYSINFO, SysInfoPath) Then
    ' Try To Get System Info Program Path Only From Registry...
    ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC,
gREGVALSYSINFOLOC, SysInfoPath) Then
        ' Validate Existance Of Known 32 Bit File Version
        If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
            SysInfoPath = SysInfoPath & "\MSINFO32.EXE"


        ' Error - File Can Not Be Found...
        Else
            GoTo SysInfoErr
        End If
    ' Error - Registry Entry Can Not Be Found...
    Else
        GoTo SysInfoErr
    End If


    Call Shell(SysInfoPath, vbNormalFocus)


    Exit Sub
SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub
```

```
Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As
String, ByRef KeyVal As String) As Boolean
    Dim i As Long                           ' Loop Counter
    Dim rc As Long                          ' Return Code
    Dim hKey As Long                        ' Handle To An Open Registry Key
    Dim hDepth As Long                      '
    Dim KeyValType As Long                  ' Data Type Of A Registry Key
    Dim tmpVal As String                    ' Tempory Storage For A Registry Key
Value
    Dim KeyValSize As Long                  ' Size Of Registry Key Variable
    '------------------------------------------------------------
    ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
    '------------------------------------------------------------
    rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open
Registry Key


    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError      ' Handle Error...


    tmpVal = String$(1024, 0)               ' Allocate Variable Space
    KeyValSize = 1024                       ' Mark Variable Size


    '------------------------------------------------------------
    ' Retrieve Registry Key Value...
    '------------------------------------------------------------
    rc = RegQueryValueEx(hKey, SubKeyRef, 0, KeyValType, tmpVal, KeyValSize)    '
Get/Create Key Value


    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError      ' Handle Errors


    tmpVal = VBA.Left(tmpVal, InStr(tmpVal, VBA.Chr(0)) - 1)
    '------------------------------------------------------------
    ' Determine Key Value Type For Conversion...
    '------------------------------------------------------------
    Select Case KeyValType                  ' Search Data Types...
    Case REG_SZ                             ' String Registry Key Data Type
        KeyVal = tmpVal                     ' Copy String Value
    Case REG_DWORD                          ' Double Word Registry Key Data Type
        For i = Len(tmpVal) To 1 Step -1        ' Convert Each Bit
            KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))   ' Build Value Char. By
Char.
        Next
        KeyVal = Format$("&h" + KeyVal)         ' Convert Double Word To String
    End Select
```

```
    GetKeyValue = True                      ' Return Success
    rc = RegCloseKey(hKey)                  ' Close Registry Key
    Exit Function                           ' Exit

GetKeyError:    ' Cleanup After An Error Has Occured...
    KeyVal = ""                             ' Set Return Val To Empty String
    GetKeyValue = False                     ' Return Failure
    rc = RegCloseKey(hKey)                  ' Close Registry Key
End Function
```