



**NEAR EAST UNIVERSITY**  
**Department Of Computer Engineering**

**Com 400 Graduation Project**

**LIBRARY AUTOMATION**

**Using Visual Basic**

**Submitted To : Mr. Ümit İLHAN**

**Submitted From: MEHMET DEĞİRMENCİ - 980327**

**NICOSIA - 2004**



## Title of Contents

Acknowledgement	I
Abstract	II
Introduction	III
1. About Library	1
2. History of Visual Basic	2
2.1 Visual Basic Description	4
2.2 What we can do with Visual Basic	8
2.3 Used Components	13
2.4 Data Access Interface	17
2.4.1 Why use ADO	17
2.4.2 OLE DB Providers	17
2.5 Creativity of a new ODBC data source	18
2.6 Menu Editor	19
3. Data Base Management System.	20
3.1 Information about DBMS	20
3.2 Data Models	21
3.3 Relation Model	24
3.4 SQL	24
3.5 The Basic Structure of SQL	26
3.6 Mapping Constrains	27
3.7 Normalization Using Functional dependences	29
3.7.1 Introduction of normalization	29
3.7.2 First Normal Form	31
3.7.3 Second Normal Form	34
3.7.4 Third Normal Form	34
3.7.5 Boyce-Codd Normal Form	35
4. Implementation of circulation system of Library	36
4.1 Password menu form	36
4.2 Main Menu Form	37
4.3 Member Information Menu Form	38
4.4 Search Menu Form	40
4.5 Data Base Structure	41
5. Software Design Issues	42
5.1 Password Menu Form	42
5.2 Main Menu Form	43
5.3 Member Information Menu Form	44
5.4 Search Book Menu Form	48
5.5 Member Search Menu Form	51
6. Conclusion	52
7. References	53
8. Appendices	54

## **ACKNOWLEDGEMENT**

Firstly I would like to thank my dear parents who helped me until this moment.

Secondly I would like to thank all my instructor and all my friends.

Thirdly I want to thank all my friends who helped for this project

Expecially I want to thank my supervisor who is Mr.Ümit İLHAN for her infinite helpness while I was prepearing this project and her kinds.

## **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 library records deal with member. That is, This project is pursuing an aim of library program, that covers services needed in most library.

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 library records to make and learn library record working systems for the requirement definitions. So, I examined some library programs and met people I know who working in library .
- 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**

This project is about writing software on circulation of books in libraries by using database (Microsoft Access for preparations and SQL Server) and Visual Basic 6.0 as an interface.

A discussion is done with university's library employees to learn system of circulation of books how it is managed. Also a new discussion is done with university's computer center to get idea of designing the database. After collection the necessary information about the system, designation my own database is done with the help of my supervisor and my database teacher. There are different types of book in library, which have either ISBN or ISSN number. But all of the books have a unique dept number as a primary key. Members also have unique member number as primary key.

With using this project library employer can make registration of books and new members of library, and circulation of books between registered members. The stuff of library or member of library can learn that the book is inside or outside. If the book is inside the library, members can find the book in an easy way without wasting time. If the book is borrowed to a member, the stuff can learn that who borrowed the book, when the book was

Borrowed and when will the book be returned. TO provide the circulation of book in a good way, library needs a punishment method for late returned books.

The library can add new books, remove the unused books and can update the information about the book. Also library can add new members or remove.

In the second part of graduation project, a brief information is given about visual basic 6.0 and it's history. An explain is done about usage of visual basic 6.0 and it's used components in this project. Creation of ODBC (open database connectivity) is focused on data source with the ActiveX Dtat Objects.

In the third part, a general overview of the nature and purpose of database systems is provided. An explained is occurred how the concept of database system has developed, what the common features of database systems are, what a database does for the user. Data

model is presented, which provides a high level view of the issues in database design with the entity relationship and also concentration on the relational data model, covering the relational algebra and relational calculus. SQL (structured Query Language), with focuses on the database query and programming language, with describe data manipulation: queries, updates, insertions and deletions. Different types of normalization form using functional dependencies are explained in detail.

In the forth part, a detailed information is given about each for how of user can run this program in details.

In the fifth part, user can understand E-R diagram of the system and the main event procedures used in the project.

updates, insertions and deletions. Different types of normalization form using functional dependencies are explained in detail.

In the forth part, a detailed information is given about each for how of user can run this program in details.

In the fifth part, user can understand E-R diagram of the system and the main event procedures used in the project.

## **2. History of Visual Basic**

### **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

### **VB Version 2.0**

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

### **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



### **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

### **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 document
- 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

### **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

## **2.1. Visual Basic Discription**

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 three key concepts as below;

- Window

- Events
- 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.

## **2.2. What 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?

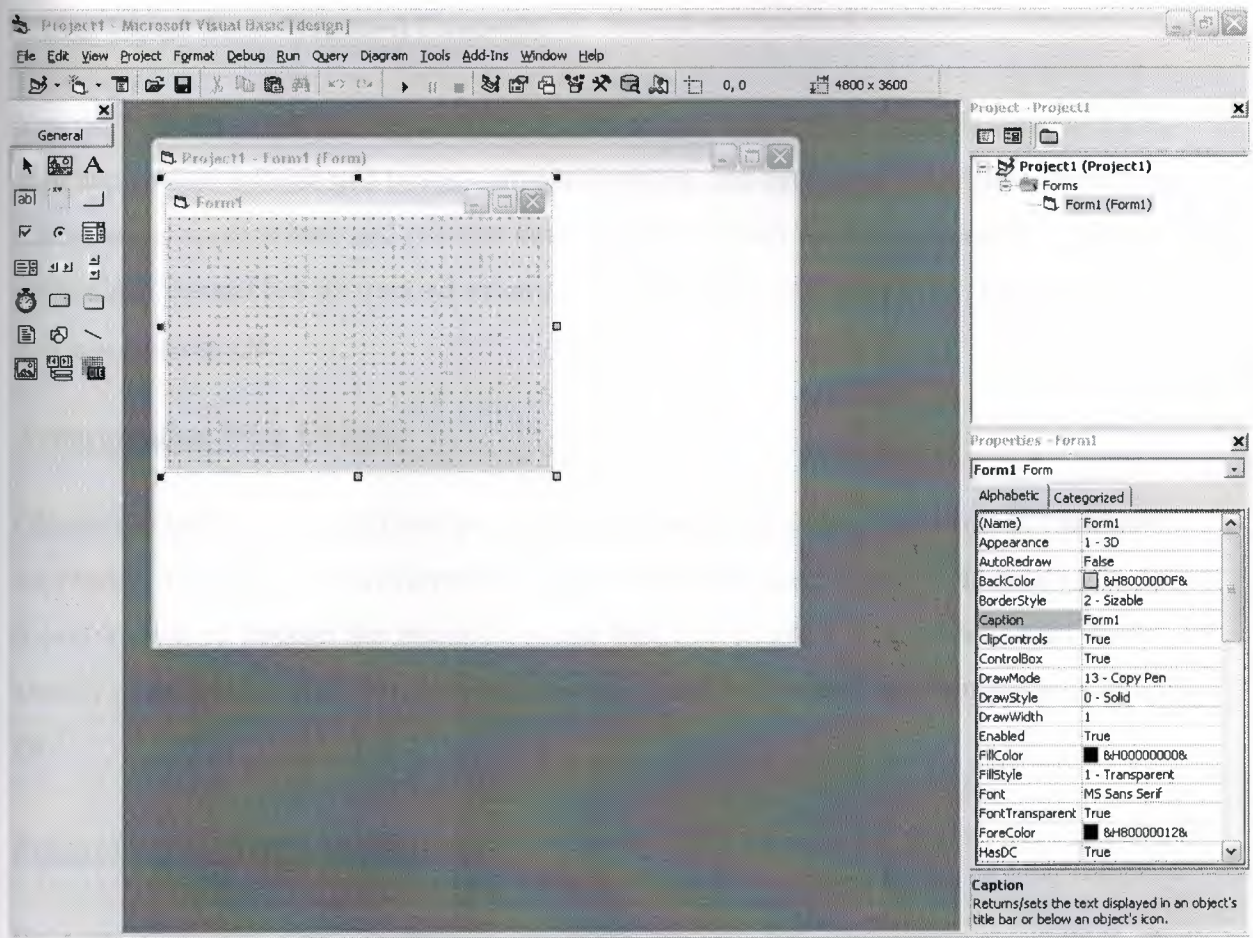


Figure 2.2.1

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.

## **2.3. Used Components**

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 to 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.

## **2.4. Data Access Interfaces**

### **2.4.1. 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.

### **2.4.2. 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.



## 2.5. Creativity of a 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.

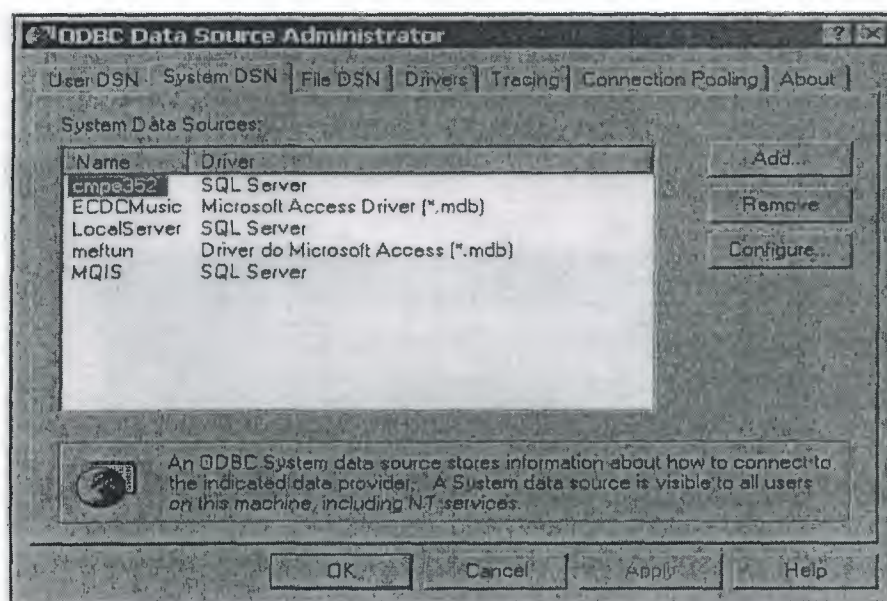


Figure 2.5.1

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.

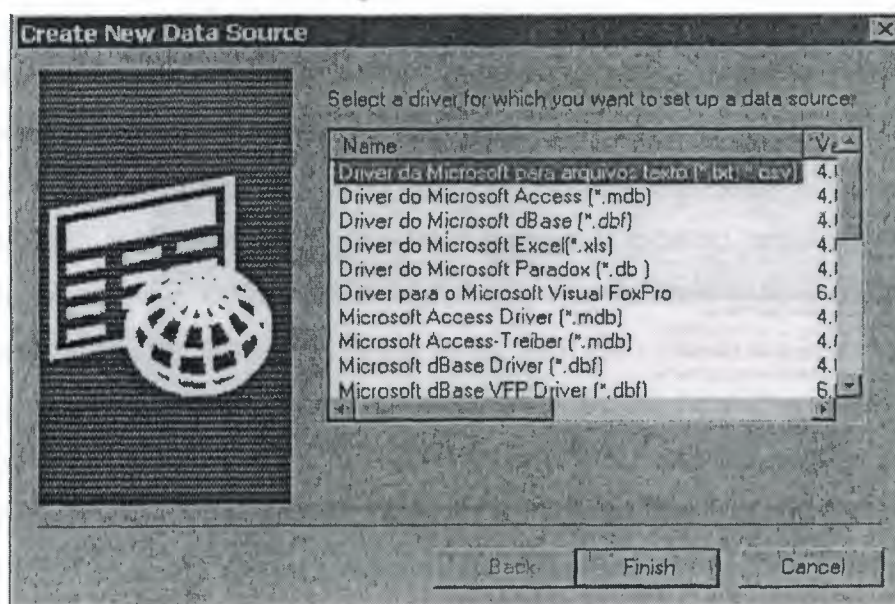


Figure 2.5.2

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.

## 2.6. 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.

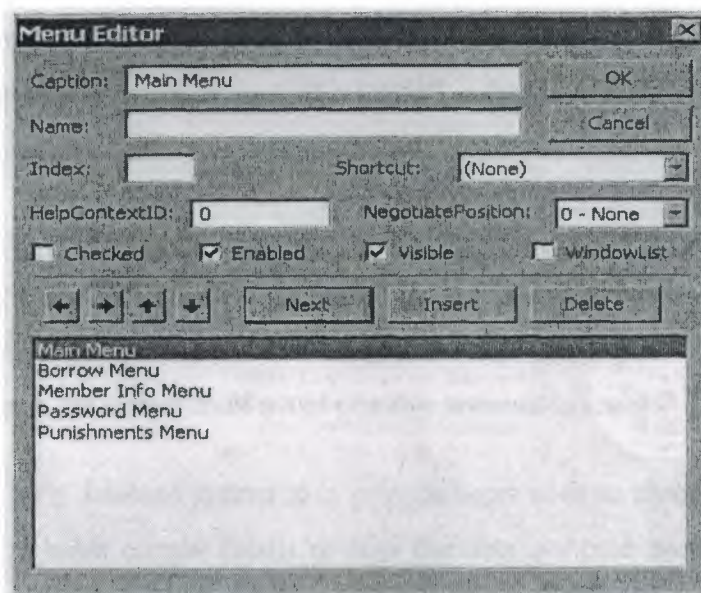


Figure 2.6.1

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.

### **3. Database Management System**

#### **3.1. 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

### **3.2. 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.

### The Select Operation

In some situations we may want to select only those tuples from a relation that satisfy a certain condition. The select operation is used to select those tuples from a relation that satisfy a certain condition. The select operation is denoted by the symbol  $\sigma$ .

### The Project Operation

If you want to know the values of some attributes of a relation, you can use the project operation. The project operation is denoted by the symbol  $\pi$ .

### Set Difference Operation

The set difference operation, denoted by the symbol  $-$ , is used to find the tuples that are in one relation but not in another. The expression  $R - S$  is a relation containing those tuples that are in  $R$  but not in  $S$ .

### Example

Suppose we have two relations,  $R$  and  $S$ , with the following tuples:

R	S
(1, 2)	(1, 2)
(2, 3)	(2, 3)
(3, 4)	(3, 4)
(4, 5)	(4, 5)
(5, 6)	(5, 6)
(6, 7)	(6, 7)
(7, 8)	(7, 8)
(8, 9)	(8, 9)
(9, 10)	(9, 10)
(10, 11)	(10, 11)

If we perform the set difference operation  $R - S$ , we will get the following result:

$R - S$
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 8)
(8, 9)
(9, 10)
(10, 11)

Since all tuples in  $R$  are also in  $S$ , the result is an empty relation.

### **3.3 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 ( $\times$ ). This operation is a binary operation, we shall use infix notation for binary operations and, thus, write the Cartesian product of relations  $r_1$  and  $r_2$  as  $r_1 \times r_2$ .

#### **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 might 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 tuples that are in one relation but not in another. The expression  $r-s$  results in a relation containing those tuples in  $r$  but not in  $s$ .

### **3.4. SQL**

SQL means "Structured Query Language". There are numerous versions 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 changed 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 parts.

### **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.

### **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.

### 3.5. 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 A1, A2,...An FROM r1,r2, m
WHERE P
```

NOTE: A<sub>i</sub> represents attribute and each r<sub>i</sub> a relation. P is a predicate.

### 3.6. 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 with at most one entity in  $A$ .

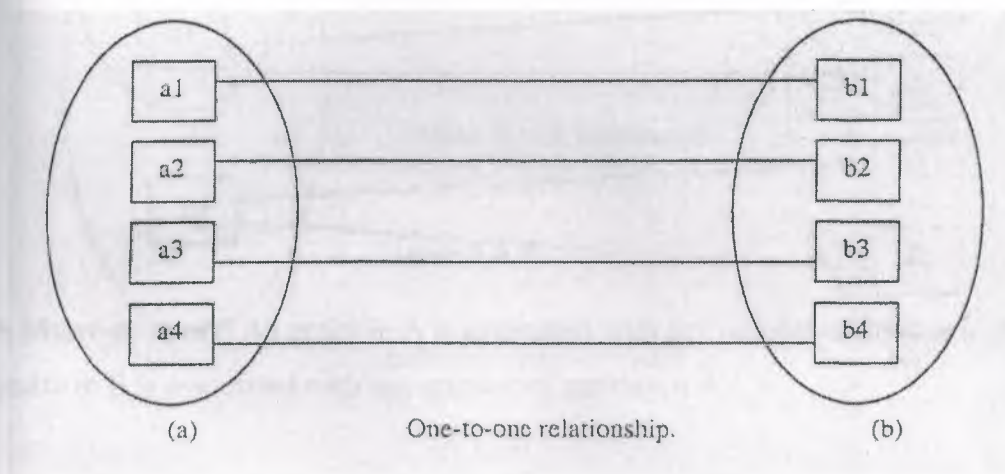


Figure 3.6.1

**One-to-many:** An entity is associated with any number of entities in  $B$ . An entity in  $B$  can be associated with at most one entity in

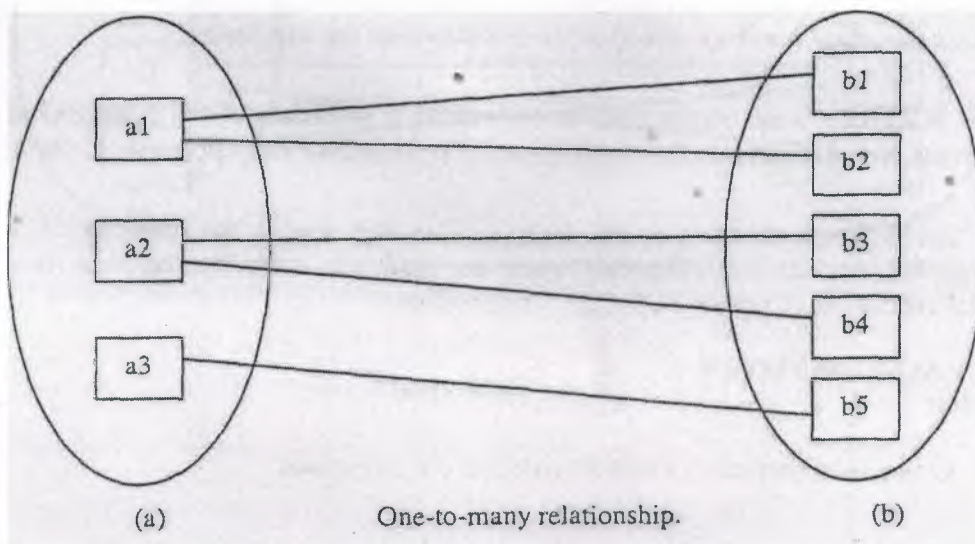


Figure 3.6.2

- > **Many-to-one:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.

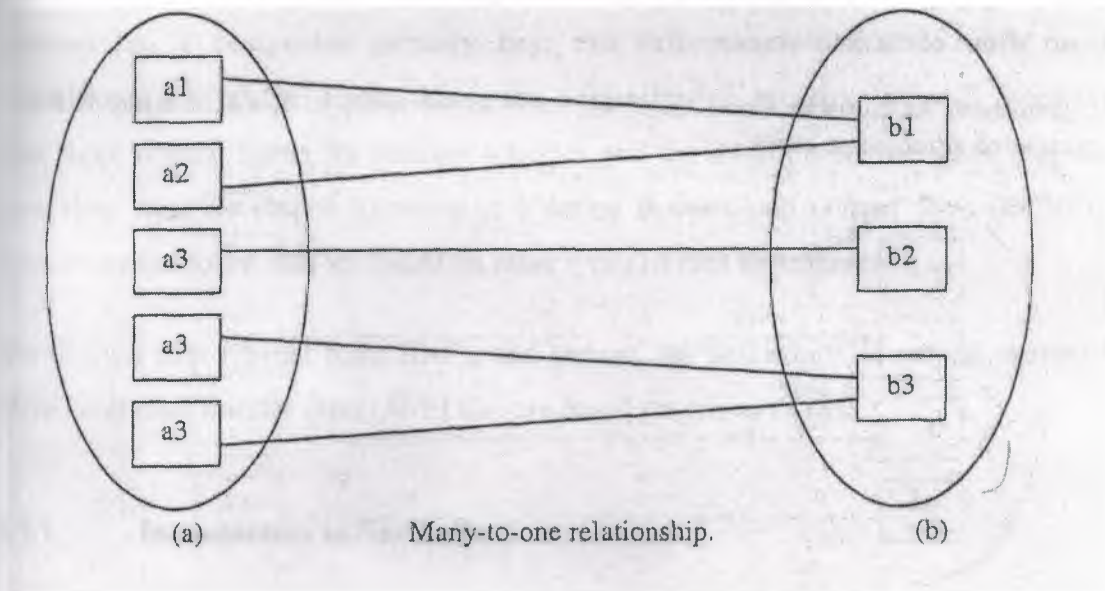


Figure 3.6.3

- > **Many-to-many:** An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

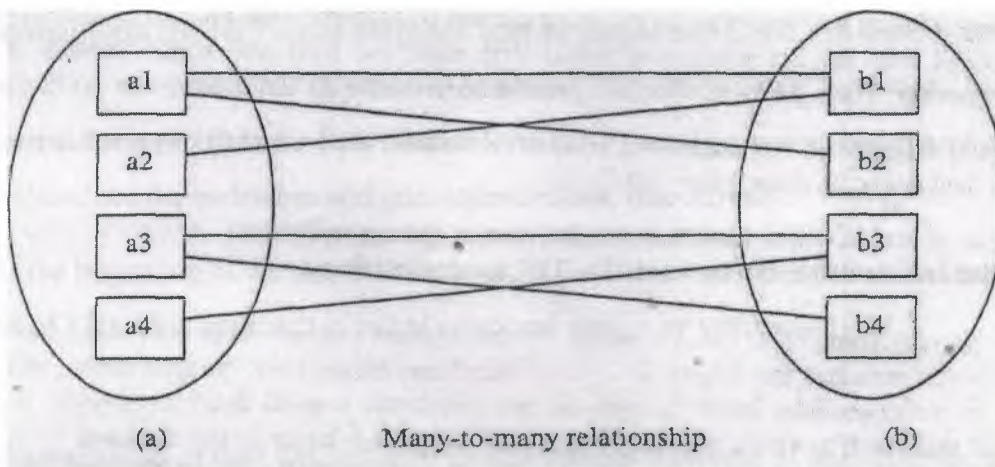


Figure 3.6.4



### **3.7. Normalization using functional dependencies**

We assume that a set of functional dependencies is given for each relation, and that each relation has a designated primary key; this information combined with the tests (conditions) for normal forms drives the normalization process. We will focus on the first three normal forms for relation schemas and the intuition behind them, and discuss how they were developed historically. I define Boyce-Codd normal form (BCNF), and further normal forms that are based on other types of data dependencies.

We discuss first normal form (1NF), and present the definitions of second normal form (2NF) and third normal form (3NF) that are based on primary keys.

#### **3.7.1. Introduction to Normalization:**

The normalization process, as first proposed by Codd (1972a), takes a relation schema through a series of tests to "certify" whether it satisfies a certain normal form. The process, which proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary, can thus be considered as relational design by analysis. Initially, Codd proposed three normal forms, which he called first, second, and third normal form. A stronger definition of 3NF—called Boyce-Codd normal form (BCNF)—was proposed later by Boyce and Codd. All these normal forms are based on the functional dependencies among the attributes of a relation. Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed, based on the concepts of multivalued dependencies and join dependencies, respectively.

At the beginning of we also discuss how 3NF relations may be synthesized from a given set of FDs. This approach is called relational design by synthesis.

Normalization of data can hence be looked upon as a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies. Unsatisfactory relation schemas that do not meet certain conditions the normal form tests are decomposed into smaller relation schemas that meet the tests and hence possess the desirable properties. Thus, the normalization procedure provides database designers with:

- A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes.
- A series of normal form tests that can be carried out on individual relation schema so that the relational database can be normalized to any desired degree.

The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized. Normal forms, when considered in isolation from other factors, do not guarantee a good database design. It is generally not sufficient to check separately that each relation schema in the database is, say, in BCNF or 3NF. Rather, the process of normalization through decomposition must also confirm the existence of additional properties that the relational schemas, taken together, should possess. These would include two properties:

- The lossless join or nonadditive join property, which guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.
- The dependency preservation property, which ensures that each functional dependency is represented in some individual relations resulting after decomposition.

The nonadditive join property is extremely critical and must be achieved at any cost, whereas the dependency preservation property, although desirable, is sometimes sacrificed.

Additional normal forms may be defined to meet other desirable criteria, based on additional types of constraints. However, the practical utility of normal forms becomes questionable when the constraints on which they are based are hard to understand or to detect by the database designers and users who must discover these constraints. Thus database design as practiced in industry today pays particular attention to normalization up to BCNF or 4NF.

Another point worth noting is that the database designers need not normalize to the highest possible normal form. Relations may be left in a lower normalization status for performance reasons. The process of storing the join of higher normal form relations as a base relation which is in a lower normal form is known as demoralization. Before proceeding further, let us look again at the definitions of keys of a relation schema. A superkey of a relation schema  $R = \{A_1, A_2, \dots, A_n\}$  is a set of attributes  $S \subset R$  with the property that no two tuples  $t_1$  and  $t_2$  in any legal relation state  $r$  of  $R$  will have  $t_1[S] = t_2[S]$ . A key  $K$  is a superkey with the additional



property that removal of any attribute from  $K$  will cause  $K$  not to be a superkey any more. The difference between a key and a superkey is that a key has to be minimal, that is, if we have a key  $K = \{A_1, A_2, \dots, A_k\}$  of  $R$ , then  $K - \{A_i\}$  is not a key of  $R$  for any  $i$ ,  $1 \leq i \leq k$ .

If a relation schema has more than one key, each is called a candidate key. One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys. Each relation schema must have a primary key.

An attribute of relation schema  $R$  is called a prime attribute of  $R$  if it is a member of some candidate key of  $R$ . An attribute is called nonprime if it is not a prime attribute that is, if it is not a member of any candidate key.

We now present the first three normal forms: 1NF, 2NF, and 3NF. These were proposed by Codd (1972a) as a sequence to achieve the desirable state of 3NF relations by progressing through the intermediate states of 1NF and 2NF if needed.

### **3.7.2. First Normal Form:**

First normal form (1NF) is now considered to be part of the formal definition of a relation in the basic (flat) relational model; historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuples must be a single value from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuples of values, or a combination of both as an attribute value for a single tuple. In other words, 1NF disallows "relations within relations" or "relations as attributes of tuples." The only attribute values permitted by 1NF are single atomic (or indivisible) values.

Consider the DEPARTMENT relation schema, whose primary key is DNUMBER, and suppose that we extend it by including the DLOCATIONS attribute as shown in Figure 1(a). We assume that each department can have a number of locations. The DEPARTMENT schema and an example extension are shown in Figure 1. As we can see, this is not in 1NF because DLOCATIONS is not an atomic attribute, as illustrated by the first tuple in Figure 1(b). There are two ways we can look at the DLOCATIONS attribute:

**\*\*** The domain of DLOCATIONS contains atomic values, but some tuples can have a set of these values. In this case, DLOCATIONS is not functionally dependent on DNUMBER.



> The domain of DLOCATIONS contains sets of values and hence is nonatomic. In this case, DNUMBER  $\rightarrow$  DLOCATIONS, because each set is considered a single member of the attribute domain.

Department

a)

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Ik		4	A

b)

DEPARTMENT		r	
DNAME	<u>DNUMBER</u>	DMGRSSN	DLOCATIONS
Research		333445S	{Beteire, Sugarisnd,
5 Administrator*		S5	Houston} {Stafford}
4 Headquarters		98765432	{Houston}

c)

DNAME	<u>DNUMBER</u>	DMGRSSN	DLOCATION
Research	5	3334-	Bella! re
Research	5	55555	Sugartar
Research	5	33344555	o

**FIGURE 1:** Normalization into 1NF. (a) Relation schema that is not in 1NF, (b) Example relation instance, (c) 1NF relation with redundancy.

In either case, the DEPARTMENT relation of Figure 1 is not in 1NF; in fact, it does not even qualify as a relation. There are three main techniques to achieve first normal form for such a relation:

1. Remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT\_LOCATIONS along with the primary key DNUMBER of DEPARTMENT. The primary key of this relation is the combination {DNUMBER, DLOCATION}, as shown in Figure 14.2. A distinct tuple in DEPT\_LOCATIONS exists for each location of a department. This decomposes the non-1NF relation into two 1NF relations.

2. Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT, as shown in Figure 1(c). In this case, the primary key becomes the combination {DNUMBER, DLOCATION}. This solution has the disadvantage of introducing redundancy in the relation.
3. If a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the DLOCATIONS attribute by three atomic attributes: DLOCATION1, DLOCATION, and DLOCATIONS. This solution has the disadvantage of introducing *null values* if most departments have fewer than three locations.

Of the three solutions above, the first is superior because it does not suffer from redundancy and it is completely general, having no limit placed on a maximum number of values. In fact, if we choose the second solution, it will be decomposed further during subsequent normalization steps into the first solution.

The first normal form also disallows multivalued attributes that are themselves composite. These are called nested relations because each tuple can have a relation within it. Figure 2 shows how the EMP\_PROJ relation could appear if nesting is allowed. Each tuple represents an employee entity, and a relation PROJS(PNUMBER, HOURS) within *each* tuple represents the employee's projects and the hours per week that employee works on each project. The schema of this EMP\_PROJ relation can be represented as follows:

EMP\_PROJ (SSN, ENAME, {PROJS(PNUMBER, HOURS)})

The set braces {} identify the attribute PROJS as multivalued, and we list the component attributes that form PROJS between parentheses ( ). Interestingly, recent research into the relational model is attempting to allow and formalize nested relations, which were disallowed early on by 1NF.

Notice that SSN is the primary key of the EMP\_PROJ relation in Figures 2(a) and (b), while PNUMBER is the partial primary key of the nested relation; that is, within each tuple, the nested relation must have unique values of PNUMBER. To normalize this into 1NF, we remove the nested relation attributes into a new relation and *propagate the primary key* into it; the



primary key of the new relation will combine the partial key with the primary key of the original relation. Decomposition and primary key propagation yield the schemas EMP\_PROJ1 and EMP\_PROJ2 shown in Figure 2(c).

This procedure can be applied recursively to a relation with multiple-level nesting to unnest the relation into a set of INF relations. This is useful in converting an unnormalized relation schema with many levels of nesting into INF relations.

### **3.7.3. Second Normal Form:**

Second normal form (2NF) is based on the concept of full functional dependency. A functional dependency  $X \longrightarrow Y$  is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute  $A \in X$ ,  $(X - \{A\})$  does not functionally determine Y. A functional dependency  $X \longrightarrow Y$  is a partial dependency if some attribute  $A \in X$  can be removed from X and the dependency still holds; that is, for some  $A \in X$ ,  $(X - \{A\}) \rightarrow Y$ .

The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key. If the primary key contains a single attribute, the test need not be applied at all. A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

If a relation schema is not in 2NF, it can be "second normalized" or "2NF normalized" into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.

### **3.7.4. Third Normal Form:**

Third normal form (3NF) is based on the concept of transitive *dependency*. A functional dependency  $X \longrightarrow Y$  in a relation schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both  $X \rightarrow Z$  and  $Z \longrightarrow Y$  hold.

Normal Form Test Remedy (Normalization) First (INF) Relation should have no nonatomic Form new relations for each attributes or nested relations.

Nonatomic attribute or nested relation Second (2NF) For relations where primary key Decompose and set up a contains multiple attributes, no new relation for each partial



nonkey attribute should be key with its dependent functionally dependent on a part of attribute(s). Make sure to the primary key. Keep a relation with the original primary-key and any attributes that are fully functionally dependent on it. Third (3NF) Relation should not have a nonkey Decompose and set up a attribute functionally determined by relation that includes the nonkey attribute(s) that another nonkey attribute (or by a set of fictionally determine(s)

nonkey attributes.) That is, there other nonkey attribute(s).Should be no transitive dependency of a nonkey attribute on the primary key.

According to Codd's original definition, a relation schema  $R$  is in 3NF if it satisfies 2NF and no nonprime attribute of  $R$  is transitively dependent on the primary key.

Table 1 informally summarizes the three normal forms based on primary keys, the tests used in each case and the corresponding "remedy" or normalization to achieve the normal form.

### **3.7.5. Boyce-Codd Normal Form:**

Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF, because every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF.

The formal definition of BCNF differs slightly from the definition of 3NF. A relation schema  $R$  is in BCNF if whenever a nontrivial functional dependency  $X \twoheadrightarrow A$  holds in  $R$  then  $X$  is a superkey of  $R$ . The only difference between the definitions of BCNF and 3NF is that condition (b) of 3NF, which allows  $A$  to be prime, is absent from BCNF.

In practice, most relation schemas that are in 3NF are also in BCNF. Only if  $X \twoheadrightarrow A$  holds in a relation schema  $R$  with  $X$  not being a superkey and  $A$  being a prime attribute will  $R$  be in 3NF but not in BCNF. Ideally, relational database design should strive to achieve BCNF or 3NF for every relation schema. Achieving the normalization status of just 1NF or 2NF is not considered adequate, as they were developed historically as stepping-stones to 3NF and BCNF. Figure 3 shows a relation TEACH with the following dependencies:

FD1: {STUDENT, COURSE}  $\twoheadrightarrow$  INSTRUCTOR

FD2: INSTRUCTOR  $\twoheadrightarrow$  COURSE

Note that {STUDENT, COURSE} is a candidate key for this relation. Hence this relation is in

3NF but not BCNF. Decomposition of this relation schema into two schemas is not straightforward because it may be decomposed in one of the three possible pairs:

All three decompositions "lose" the functional dependency FD1. The desirable decomposition out of the above three is the third one, because it will not generate spurious tuples after a join. A test to determine whether a decomposition is nonadditive (lossless). In general, a relation not in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations, as is the case in this example.

#### 4. Implementation of circulation system of library

I am going to explain the user interfaces part of my program to a user who does not know anything about the program.

##### 4.1. Password Menu Form:

In the password menu, every user must have a password to use. If a user does not have a password s/he cannot use. Firstly s/he must enter user name then must enter the correct password. After clicking the 'OK' button s/he can use program. If the user enters click the 'Cancel', the textboxes are cleaned and waiting for user to enter the username and password.




Figure 4.1.1

The username is very important because every user have different rights while using the program.



## 4.2. Main Menu Form:



The image shows a screenshot of a computer application window titled "MyBooks". The window has a menu bar with the options "MENU", "REGISTER", "PASSWORD", and "ABOUT". The main area of the window is dark grey. In the center, there is a smaller, light grey dialog box titled "Form6" with a subtitle "ADMINISTRATION". This dialog box contains two input fields: "User Name" and "Password". Below these fields are three buttons: "OK", "CHANGE PASSWORD", and "EXIT".

Figure 4.2.1



#### 4.3. Member Information Menu Form:

If the user wants to insert a new member, user must first click on the 'New Record' button to clear the text boxes and enable the 'Insert' button. After the user enters the necessary information to the text boxes, the user must click 'Insert' button to add the information to the database. While inserting a new member, user must pay attention to member number, if member number is entered then the program warns the user to change the member number.

The screenshot shows a software window titled 'Form7'. At the top, there are fields for 'DAY PERIOD', 'IN DATE', and 'DATE' (set to 11.01.2004). Below these are input fields for 'MEMBER ID' (7), 'NAME' (ALI), 'SURNAME' (USTAAGLU), 'E\_MAIL' (McLaren@msn.com), 'PHONE' (2231456789), and 'ADDRESS' (KARSIYAKA). A section labeled 'DETAILED INFORMATION' contains a 'BORROWED BOOKS' table. To the right, an 'UPDATE' button is above a table listing members. The table has columns: MEMBERID, NAME, SURNAME, and ADDRESS. Member 7 (ALI USTAAGLU) is selected. At the bottom, there are buttons: SEARCH, MEMBER PAGE, MAIN MENU, PRINT, ADD BOOK, ALL INFORMATIONS, and DELETE.

MEMBERID	NAME	SURNAME	ADDRESS
2	CELAL	kunduraci	SARIYER
3	CEMAL	kunduraci	SARIYER
4	ÖNDER	karlioglu	SARIYER
5	GÖKHAN	koc	ZEYTINBURNU
6	selale	koc	KOC_GKHN@
7	ALI	USTAAGLU	KARSIYAKA
8	goulhaz	USTAAGLU	6440/3 SOK N

Figure 4.3.1

If user clicks any command button which is on the top of the form then the members whose names are started with the selected characters are shown in the list box. when the user double clicks on any member name from the listbox, the command fields are filled. User has following rights:

**Update:** When the user clicks the 'Update' button, user can replace the old

information with the new information. While replacing the new information, user must pay attention to member number. If member number is entered before, the program forces the user to change the member number. When an update is done in any information, this updating occurs in all tables not to lose information.

**Delete:** When the user clicks the 'Delete' button, user can delete the member if the member does not have book. When a deletion occurs, deletion occurs all my tables.

**Member page:** When the user clicks the 'member page' button, the member page will come up to the screen.

**Search :** When the user click the 'search' button directly, all members will be sort in the table, but if user enters some information to the textboxes about any member then clicks to the 'search' button, then just selected member information will come to the table.

**Print:** When the user click the 'print' button then user can take a print out all over the selected member informations.

**Add Book:** When user click the 'Add Book' button then user can add selected book to selected member statistics.

**All Information:** When user click the 'All Information' button then it shows all over the infprmation on the member table, for example if member is green that user took a book or books from the library and he/she have a time to bring it back. If member is yellow that user didn't take any book from the library yet. If member is red that member took a book or books from the library and his or her limited time over which depends on the library.



#### 4.4. Search Menu Form

The user can come to the search menu only from the book menu. While passing from book menu to search menu, in the menu we can search any book from library, then we can add it to member's record or we can update book information.

If the user wants to insert a book to member's record then click on the 'Add Book into Member's Record' button then that book is added to member's record. For updating, user should click 'update' button, before that user should choose book from the table and then book information will come to the textboxes. After that, user can change book information. After that if user clicks 'update' button that information is updated. If not, nothing changes.

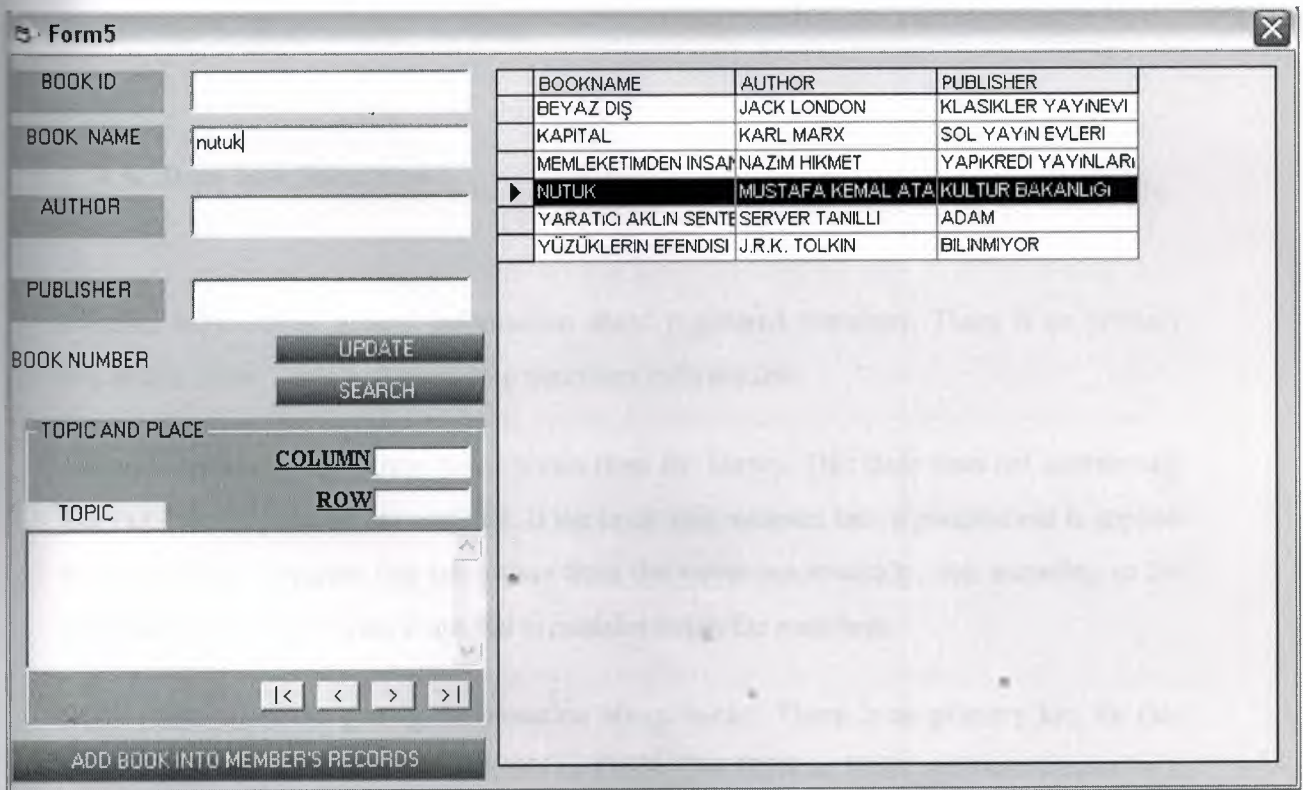


Figure 4.3.2

When the ISSN or ISBN number of book is read by the bar code or entered from the keyboard then the user has following rights:



**Update:** When the user clicks the "Update" button, user can replace the old information with the new information. While replacing the new information, user must pay attention to debt number. If debt number is entered before, the program forces the user to change the debt number. When an update is done in any information, this updating occurs in all tables not to lose information.

**Search :** When the user click the 'search' button directly, all book will be sort in the table, but if user enters some information to the textboxes about any book then clicks to the 'search' button, then just selected book information comes to the table.

**Add Book into Member's Record:** This button just for adding book to member's record. Before click this button user should select book then click this button for add that book to member's record.

#### **4.5. Data base Structure**

Member table stores general information about registered members. There is no primary key in this table. Table just including members information.

'AboutDateTable' table stores taken books from the library. This table does not contain any key but contain punishment attribute. If the book was returned late, a punishment is applied to the member. Program that takes time from the server automatically, and according to the program limitation, Program is applied to punishment to the members.

Book table stores the general information about books. There is no primary key for this table. The book must have either ISSN or ISBN. But ISSN or ISBN attribute cannot be a key because it is possible to have more than one book. So there is no reason to use primary key in this table.

## **5. Software Design Issues**

### **5.1. Password Menu Form:**

#### FormJLoadQ:

I Access from my application to the data source using ActiveConnection command, this is necessary for exchanging data. Then I opened my database to access without facing any problem. If opened before, we do not need to open so many times.

#### cypherl\_KeyPressQ:

This is for my textbox to check that the pressed key is Enter key or not if enter the program enables the other text box for writing.

#### visiblesQ:

In this function I only enabled or disabled my textboxes, command or other Graphical user interfaces according to the type of user. Unauthorized user can not use this program. Because he/she can not enter the program without correct password.

#### **Command 1 Click (index As Integer):**

According to the value of index defined in my function, my program executes different parts of program.

If the value of index is "0" then it executes the OK button to check whether the username and its password is same or not. In here I open the password table that is related with usernames and passwords of people. Then I use while loop to check the text that is written in the text box is in my database or not. If the Username is in my database I check the password that is written correctly according to that user. If the username and

password is true, then I execute the program according to the user's right.

If the value of index is "1" then it executes the cancel button to clear the fields of textboxes that was written before by the user.

If the value of index is "2" then I execute the Exit Program button to end the program.

## **5.2. Main menu Form:**

### **Form LoadQ:**

In this function I check that is an error occurs or not while the form was loading. I find the general values that are always needed by the program like a punishment value for each day or last day of the semester. Private Sub

### **Command Click (index As Integer):**

According to the value of index defined in my function, my program executes different parts of program by using the case statement.

If the value of index is "0" then my program executes the 'Member Information' button to load the member information menu.

If the value of index is "1" then my program executes the 'Punishments' button to load the Punishment menu.

If the value of index is "2" then my program executes the 'Change User' button to load the password menu.

If the value of index is "3" then my program executes the 'Exit' button to finish the execution of the program.



### **5.3. Member Information Menu Form:**

#### **Form loadQ:**

I Access from my application to the data source using ActiveConnection command, this is necessary for exchanging data. Then I opened my database to access without facing any problem. I enabled my command buttons that can be seen upper side of the form if my database involves a name that is started with same name of my command button. In here I opened my database again because if the user changed his password, he can not access the tables because the database closed before.

#### **Command1 Click (index As Integer):**

According to the value of index defined in my function, my program executes different parts of program by using the case statement.

If the value of index is "0" then my program executes the 'New Record' button to register a new user to the database as a first step. In this part I clean all the textboxes, listbox, datagrid and also I enabled or disabled some of these.

If the value of index is "1" then my program executes the 'Insert' button to add new information about the user to the 'users' table. First of all I opened my 'memberrecordtable' table for adding new information. While adding a new member, automatically member id is count up one by one. By this way every members have different member id.

If the value of index is "2" then my program executes the 'Update' button to edit the old information with the new information. I found the person that would be updated by the filter command with using 'usernum', which. Then I change the values of the attributes with the new information by using the update command. While updating the information, my database automatically checks the member id of the person is repeated or not. When I updated the person's information, I also need to update my other tables according to my new information.

If the value of index is "3" then my program executes the 'Delete' button to delete information about the person by using the delete command. First I found the person's usernum in my 'Users' table by using the filter command then I checked the usernum with my 'Transaction' table's usernum to see that person has a book or not. I count the rows of the datagrid. If the number of rows is greater or equal to 0 then this means that the user has a book. If the user has a book I can not delete the person until the book will have been returned. If I did not check this criterion, when I delete a person who has book on his/herself, the book would be lost.

If the value of index is "4" then my program executes the 'Return' button to return the book from the user. In here I first get the values from my datagrid. In for loop I first find the index my selected element. When I found the index of that element I got the kod number, which is a primary key. Then I opened my transition table to delete the book from user. If book is deleted, the book must be inserted in the returned table. Before inserting the book, I calculate the punishment if the book is returned lately and I wrote a message to the screen.

If the value of index is "5" then my program executes the 'Borrow Menu' button to load the borrow menu. While I am loading the borrow menu, I must first close my adodc connection because I will reopen this connection in the borrow menu again.

If the value of index is "6" then my program executes the 'Past' button to enable the combo box. After I enabled my combo box, I change the name of my command button name to 'Now'. Then I opened my 'GNRL' table to get the date of beginning of semester and date of ending of the semester. According to the values of these dates, I execute my SQL statement to find the user's history.

If the value of index is "7" then my program executes the 'Return Main Menu' button to load the main menu. Before I load the main menu, I must first close my connection and reopen my connection in the main menu.



### **CMDLETTER click (index As Integer):**

If one of the command buttons pressed that is defined at the top, my list box started to fill with the names whose first alphabet start with the command button's caption. I get the first character of the names from ' Users ' table in a while. I compare first character of names with the command button's caption. If they are same I insert them to the list box, otherwise I do not insert to the database and I move a head.

### **ListBoxForNames dblClickQ:**

When I double clicked the listbox, the information of that selected person is brought from database to the textboxes and to the datagrid. This listbox does not only fill with the alphabets also fill with the numbers. If it is filled with number, also the same information will be brought to the screen. For these operations, I must first find that the selected value is name or number. If selected value is number then I execute my SQL satatement, which is related with the numbers. Else I execute another SQL statement, which is related with the string expression. After the execution of the SQL statement I find the related person usernum (primary key). Then I can easily show the information easily about the selected person in my list box. To show the information I use the TextBoxDoldurma function, which is described in detail above.

### **TextBox FillingO:**

I fill the text boxes with the necessary information by using the information that is given by the user or administrator. In this function I must access my database again also I must create a new connection query according to the SQL statement. In my database I don't hide statue attribute, which is a derived attribute because I understood the statue by using the user's registered id. If the register is start with number we can easily say that the user is student. If the register-id starts with the "A" alphabet we can understand that this user is teacher else the user is a personal who is working in the library or in the university. I put the department of users by using the case statement. If the user write 25 in Department text box. The compiler automatically understands that this is the computer-engineering department. If the user want to write the correct and long name of the department, then this



is also possible by writing the long name of department.

#### **ComboPast ClickO:**

We can think this function like reminder of the past information. This is also very important because the user want to learn past information. Which book he borrowed in the first part of the related year. At that time s/he also want to learn then they returned the book. If they returned the book lately the punishment method automatically is executed.

#### **Regid ChangeQ:**

When I pressed any character, my listbox started to fill with that character plus the characters pressed before. For each pressed character I must clear my listbox then I must add the new information according to the string written in the textbox.

#### **TextBox DeletingQ:**

In this function I only clear the contents of my textboxes. This is necessary when I inserted a new person in my database or when I come from another form.

#### **TextBox EnableQ:**

In this function I enable the text boxes if I double click the list box or other special operations. This is very important while updating the information about the related subject or person.

#### **TextBox DisableO:**

In this function I disabled the text boxes if I deleted or update the file. This is also very important because this process provide user not to make mistake.

#### **TextBookNo dblClickO:**

If the user double click the ISBN text box. Then the content of the text is cleared and also the database view of the screen also changed by the value of this text book.

#### **5.4. Search book Menu Form:**

Form\_loadQ:

I reopened my database to access without facing any problem because the database closed before. I write the name, author, title, publisher and book id of the book to my textboxes, which are taken from the book information menu. Then I execute my SQL statement, which brings the information about the book to the datagrid.

#### **Private Sub Cmdsearch Clickindex As Integer):**

According to the value of index defined in my function, my program executes different parts of program by using the case statement.

If the value of index is "0" then my program executes the 'New Record' button to register a new book to the database as a first step. In this part I clean all the textboxes, listbox, datagrid and also I enabled or disabled some of these.

If the value of index is "1" then my program executes the 'Insert' button to add new information about to the book to the 'books' table. First of all I opened my 'books' table for adding new information. In my 'books' table 'Debt\_Number' is assigned as a primary key so while adding a new user the system automatically controls that if same 'Debt\_Number' is given to the two different books or not. While controlling if I found usernum as repeated, I force user to change the usernum. At a last step I insert the information, I enabled and disabled my objects in my form, which had been enabled or disabled when the 'New Record' button is pressed.

If the value of index is "2" then my program executes the 'Update' button to edit the old information with the new information. I found the book that would be updated with 'Debt\_Number' by using the filter command. Then I change the values of the attributes with the new information by using the update command. While updating the information, my database automatically checks book id of the book is repeated or not. When I updated the book's information, I also need to update my other tables according to my new information.

If the value of index is "3" then my program executes the 'Delete' button to delete information about the book by using the delete command. First I must find the book in my 'Books' table by using the filter command then I checked the 'Debt\_Number' of the book with my 'Transaction' table's 'Debt\_Number' to see that person has a book or not. I count the rows of the datagrid. If the number of rows is greater or equal to 0 then I check each book's 'Debt Number' with the attributes of 'Transaction' table. If the 'Debt\_Number' is found that this means that the book is outside can not be deleted until the book is returned. If I did not check this criterion, the book would be lost.

If the value of index is "4" then my program executes the 'Return' button to return the book from the user. In here I first get the values from my datagrid. In for loop I first find the index my selected element. Then I opened my transition table to delete the book from user. If book is deleted, the book must be inserted in the returned table. Before inserting the book, I calculate the punishment if the book is returned lately and I wrote a message to the screen.

If the value of index is "5" then my program executes the 'Borrow' button to borrow the book to the user. I found the books with using the ISBN or ISSN number, then I got the 'Debt\_Number' of each book in a loop. Then I control this 'Debt\_Number' is in 'Transaction' table or not. If 'Debt\_Number' is seen in the 'transition' table this meant that the book is outside. Then control the next book to see that the book is inside or not. This continues until the available book is found. While borrowing book to the user, the users have different rights. I separate these differences in and if statement. If the user type is teacher the return date of book is greater than student type. I also control that the return date is Saturday or Sunday. If the day is Saturday I increments two days to the return sate a final date. In a case statement I limit the rights of the users. An example if the user type is student the user can not borrow more than 3.

If the value of index is "6" then my program executes the 'Go to Member Menu' button to load the member menu. While I am loading the member menu, I must first close my adodc connection because I will reopen this connection in the member menu again.

If the value of index is "7" then my program executes the 'Back' button. This button



works like a reset operation. I execute an SQL statement to show the current member's information to the datagrid. I also enabled and disabled my textboxes and command buttons by using the my functions.

#### Listbox\_DblClickO:

When I double clicked the list box, the book, which has the selected ISBN or ISSN number is brought from database to the textboxes and to the datagrid. To bring the information to the screen I first find which one is selected in the list box in a for loop. Then I execute my SQL statement according to the selected value. In my SQL statement, I also check that the book is inside or not. If the book is not inside, an error message is brought to the screen. This error message showed that when will the book be returned and who borrowed the book? To show the information I first find the related entities of each table. After this I bring the information to the screen using the visual basic standards commands.

#### ListboxQ:

When the bar code reads the ISSN or ISBN number, this function is always executed and controls that the datagrid is empty or not. If the datagrid is empty I find the debt number of book and information about book is written to the text boxes and I enabled the borrow button. In my second if statement, I check that my ISBN text box is empty or not. If not empty then I control that the book is in datagrid or not. If the book's debt number is in the datagrid, I enabled the return button and disabled the borrow button.

#### TextBoxDoldurmaO:

This is used to write information of book to the textboxes. I find that book id in which this function is called. According to the value of book id, I execute my SQL statement to fill the text boxes. While writing to the text boxes, if the attribute of book is empty then some errors may occur. To hinder this situation, I use 'On Error Resume Next'.

#### TextboxsilmeQ:

This is used to clear the textboxes by equaling textboxes's text function to an empty state.

#### **TextboxenableQ:**

This is used to enable the textboxes by equaling textboxes's enabled function to true.

#### **TextboxdisableQ:**

This is used to enable the textboxes by equaling textboxes's enabled function to false.

### **5.5. Member Search Menu Form:**

#### **loadQ:**

I Access from my application to the data source using ActiveConnection command, this is necessary for exchanging data. Then I opened my database to access without facing any problem. If opened before, we do not need to open so many times.

#### **Cmdl Click(Index As Integer):**

According to the value of index defined in my function, my program executes different parts of program by using the case statement.

If the value of index is "0" then my program executes the 'All Information' button to bring the information about punished members to the datagrid. For this operation I wrote a SQL statement and I equalize this to adodc's record source and I refreshed the adodc.

## **6. 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 have learned Visual Programming language from my supervisor Mr. Umit ILHAN and I want to thank to him for his great helps to me while I was doing my graduation project.



## 7. References

> Visual Basic 6.0 How To Program  
H. M. Deitel, P. J. Deitel, T. R. Nieto  
1999-Prentice-Hall, Inc

Visual Basic Lecture Note: Ümit İlhan 2002-2003

> 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





## 8.1 Source Codes

### 8.1.1 MDI FORM

```
Private Sub MDIForm_Load()  
Veriac  
MDIForm1.Top = 0  
MDIForm1.Left = 0  
MDIForm1.Width = Screen.Width  
MDIForm1.Height = Screen.Height  
Form6.Show  
mnBOOK.Enabled = False  
mnMEMBER.Enabled = False  
mnPASSWORD.Enabled = False  
mnREPORT.Enabled = False  
Form6.KDCButton6.Visible = True  
Form6.KDCButton3.Visible = True  
9 Form7.Label10.Visible = False  
Form7.Label9.Visible = False  
Form7.Text8.Visible = False  
Form7.Text9.Visible = False  
Form7.Label11.Visible = False  
Form7.Label12.Visible = False  
Form7.Label13.Visible = False  
Form7.Label14.Visible = False  
Form7.Visible = False  
End Sub  
  
Private Sub MDIForm_Unload(Cancel As Integer)  
VeriKapat  
End Sub  
  
Private Sub mnBOOKCOMING_Click()  
Form8.Show  
Form1.Visible = False  
Form3.Visible = False  
Form4.Visible = False  
Form5.Visible = False  
Form6.Visible = False  
Form7.Visible = False  
Form9.Visible = False  
Form10.Visible = False  
Form11.Visible = False  
  
End Sub  
  
Private Sub mnBOOKSEARCH_Click()  
Form5.Show  
Form1.Visible = False  
Form3.Visible = False  
Form4.Visible = False  
Form8.Visible = False  
Form6.Visible = False  
Form7.Visible = False  
Form9.Visible = False  
Form10.Visible = False
```



```
'Form11.Visible = False
End Sub
```

```
Private Sub mnMEMBERSEARCH_Click()
Form7.Show
Form1.Visible = False
Form3.Visible = False
Form4.Visible = False
Form5.Visible = False
Form6.Visible = False
Form8.Visible = False
Form9.Visible = False
Form10.Visible = False
'Form11.Visible = False
End Sub
```

```
Private Sub mnNEWBOOK_Click()
Form1.Show
Form8.Visible = False
Form3.Visible = False
Form4.Visible = False
Form5.Visible = False
Form6.Visible = False
Form7.Visible = False
Form9.Visible = False
Form10.Visible = False
'Form11.Visible = False
End Sub
```

```
Private Sub mnNEWMEMBER_Click()
Form4.Show
Form1.Visible = False
Form3.Visible = False
Form8.Visible = False
Form5.Visible = False
Form6.Visible = False
Form7.Visible = False
Form9.Visible = False
Form10.Visible = False
Form11.Visible = False
End Sub
```

```
Private Sub mnNEWPASSWORD_Click()
Form6.Show
Form6.Frame3.Visible = True
Form1.Visible = False
Form3.Visible = False
Form4.Visible = False
Form5.Visible = False
Form8.Visible = False
Form7.Visible = False
Form9.Visible = False
Form10.Visible = False
Form11.Visible = False
End Sub
```

```
Private Sub mnPASSWORDCHANGE_Click()
```

```

Form6.Show
Form6.Frame2.Visible = True
Form6.Frame1.Visible = False
Form6.Frame3.Visible = False
Form6.Command6.Visible = False
Form6.KDCButton3.Visible = False
Form1.Visible = False
Form3.Visible = False
Form4.Visible = False
Form5.Visible = False
Form8.Visible = False
Form7.Visible = False
Form9.Visible = False
Form10.Visible = False
Form11.Visible = False
End Sub

```

```

Private Sub mnREPORT_Click()
Form10.Show
Form1.Visible = False
Form3.Visible = False
Form4.Visible = False
Form5.Visible = False
Form6.Visible = False
Form7.Visible = False
Form9.Visible = False
Form8.Visible = False
Form11.Visible = False
End Sub

```

### **8.1.2 Main Member Menu**

```

Private Sub KDCButton1_Click()
Form9.Visible = False
Form4.Visible = True
End Sub

```

```

Private Sub KDCButton2_Click()
Form7.Show
Form9.Visible = False
End Sub

```

```

Private Sub KDCButton3_Click()
Form9.Visible = False
End Sub

```

### 8.1.3 Book Comming form

```
Private Sub Form_Load()
```

```
Text1 = ""
```

```
Text2 = ""
```

```
Text3 = ""
```

```
Text4 = ""
```

```
Text10 = ""
```

```
Text9 = ""
```

```
Label10 = ""
```

```
Label11 = ""
```

```
End Sub
```

```
Private Sub KDCButton1_Click()
```

```
Form3.Show
```

```
Form8.Visible = False
```

```
End Sub
```

```
Private Sub KDCButton2_Click()
```

```
Form8.Visible = False
```

```
End Sub
```

```
Private Sub KDCButton3_Click()
```

```
bookgeldiAc "select * from BOOKRECORDTABLE WHERE BOOKID=" & Val(Text1) & " and  
BOOKNAME=" & UCase(Trim(Text2)) & " and AUTHOR=" & UCase(Trim(Text3)) & " and  
PUBLISHER=" & UCase(Trim(Text4)) & " and DURUM=" & "disarda" & ""
```

```
If bookgeldi.RecordCount > 0 Then
```

```
bookgeldi.MoveFirst
```

```
Label11 = bookgeldi("ROW")
```

```
Label10 = bookgeldi("COLOMN")
```

```
bookgeldi("DURUM") = "icerde"
```

```
bookgeldi.Update
```

```
bookgeldiAc "select * from ABOUTDATETABLE where MEMBERID=" & Val(Text10) & " and  
MEMBERNAME=" & UCase(Trim(Text9)) & " and OUTBOOK=" & UCase(Trim(Text2)) & ""
```

```
If bookgeldi.RecordCount > 0 Then
```

```
bookgeldi.MoveFirst
```

```
bookgeldi("COMINGDATE") = Date
```

```
bookgeldi("BOOK") = "geldi"
```

```
bookgeldi.Update
```

```
End If
```

```
End If
```

```
End Sub
```



### 8.1.4 Registered Member Form

```
Private Sub Form_Load()
```

```
Frame2.Visible = False
KDCButton2.Visible = False
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text8.Text = ""
Text9.Text = ""
Text7 = Date
List1.Clear
End Sub
```

```
Private Sub KDCButton1_Click()
```

```
Frame2.Visible = False
Command6.Visible = False
On Local Error Resume Next
If DataGrid1.Columns(0) = "" Then
MsgBox ("You have not selected any member." + Chr(10) + Chr(13) + "Please, select from MEMBER SEARCH FORM")
Exit Sub
ElseIf Text1 <> "" And Text2 <> "" And Text3 <> "" And Text4 <> "" And Text5 <> "" And Text6 <> ""
Then
```

```
memberkayitAc "select * from MEMBERRECORDTABLE where MEMBERID=" & Val(Text1)
```

```
If memberkayit.RecordCount > 0 Then
```

```
MsgBox ("There is just record has such a MEMBERID ' " & memberkayit("NAME") & " " & memberkayit("SURNAME") & " " & Chr(10) + Chr(13) + "IDs of members cannot be changed .")
Else
```

```
memberkayitAc "select * from MEMBERRECORDTABLE where MEMBERID=" & Val(DataGrid1.Columns(0))
outbookaramaAc "select * from ABOUTDATETABLE where MEMBERID=" & Val(DataGrid1.Columns(0)) & " order by OUTDATE"
memberkayit("MEMBERID") = Val(Text1)
memberkayit("NAME") = UCase(Trim(Text2))
memberkayit("SURNAME") = UCase(Trim(Text3))
memberkayit("ADDRESS") = UCase(Trim(Text6))
memberkayit("E_MAIL") = Trim(Text4)
memberkayit("TELEPHONE") = Trim(Text5)
memberkayit.Update
```

```
If outbookarama.RecordCount > 0 Then
```

```
outbookarama.MoveFirst
Do Until outbookarama.EOF
outbookarama("MEMBERID") = Val(Text1)
outbookarama("MEMBERNAME") = UCase(Trim(Text2))
outbookarama.Update
outbookarama.MoveNext
Loop
End If
```

```

End If
End If
memberkayitAc "select * from MEMBERRECORDTABLE order by MEMBERID"
Set DataGrid1.DataSource = memberkayit
End Sub

Private Sub KDCButton2_Click()
Frame2.Visible = False
Command6.Visible = False
End Sub

Private Sub KDCButton3_Click()
Label10.Visible = False
Label9.Visible = False
Text8.Visible = False
Text9.Visible = False
Label11.Visible = False
Label12.Visible = False
Label13.Visible = False
Label14.Visible = False
Frame2.Visible = False
'Command6.Visible = False
memberaramaAc "select * from MEMBERRECORDTABLE"
If memberarama.RecordCount > 0 Then
Dim SORGU As String
If Text1 = "" And Text2 = "" And Text3 = "" Then
memberaramaAc "SELECT MEMBERID,NAME,SURNAME,ADDRESS FROM
MEMBERRECORDTABLE order by MEMBERID"
Else

SORGU = "SELECT MEMBERID,NAME,SURNAME,ADDRESS from MEMBERRECORDTABLE
WHERE"

If Not Trim(Text1) = "" Then
SORGU = SORGU + " MEMBERID LIKE '" & Val(Text1) & "%' "
End If

If Not Trim(Text2) = "" Then
If Not Trim(Text1) = "" Then
SORGU = SORGU + " AND NAME LIKE '" + Trim(Text2) + "%' "
Else
SORGU = SORGU + " NAME LIKE '" + Trim(Text2) + "%' "
End If
End If

If Not Text3 = "" Then
If Trim(Text1) = "" And Trim(Text2) = "" Then
SORGU = SORGU + " SURNAME LIKE '" + Trim(Text3) + "%' "
Else
SORGU = SORGU + " AND SURNAME LIKE '" + Trim(Text3) + "%' "
End If
End If

memberaramaAc SORGU + " order by MEMBERID"

End If

```

```
Set DataGrid1.DataSource = memberarama
```

```
End If
```

```
End Sub
```

```
Private Sub KDCButton4_Click()
```

```
Form7.Label10.Visible = False
```

```
Form7.Label9.Visible = False
```

```
Form7.Text8.Visible = False
```

```
Form7.Text9.Visible = False
```

```
Label11.Visible = False
```

```
Label12.Visible = False
```

```
Label13.Visible = False
```

```
Label14.Visible = False
```

```
Frame2.Visible = False
```

```
Command6.Visible = False
```

```
Form9.Show
```

```
Form7.Visible = False
```

```
End Sub
```

```
Private Sub KDCButton5_Click()
```

```
Form7.Label10.Visible = False
```

```
Form7.Label9.Visible = False
```

```
Form7.Text8.Visible = False
```

```
Form7.Text9.Visible = False
```

```
Label11.Visible = False
```

```
Label12.Visible = False
```

```
Label13.Visible = False
```

```
Label14.Visible = False
```

```
Frame2.Visible = False
```

```
Command6.Visible = False
```

```
Form7.Visible = False
```

```
End Sub
```

```
'Private Sub Command4_Click()
```

```
'Form7.Label10.Visible = False
```

```
'Form7.Label9.Visible = False
```

```
'Form7.Text8.Visible = False
```

```
'Form7.Text9.Visible = False
```

```
'Label11.Visible = False
```

```
'Label12.Visible = False
```

```
'Label13.Visible = False
```

```
'Label14.Visible = False
```

```
'Frame2.Visible = False
```

```
'Command6.Visible = False
```

```
'Form9.Show
```

```
'Form7.Visible = False
```

```
'End Sub
```

```
Private Sub KDCButton6_Click()
```

```
Form10.RichTextBox1.Text = "MEMBER ID" + Chr(13) + Text1 + Chr(10) + Chr(10) + Chr(13) +  
"NAME" + Chr(13) + Text2 + Chr(10) + Chr(10) + Chr(13) + "SURNAME" + Chr(13) + Text3 + Chr(10)  
+ Chr(10) + Chr(13) + "E_MAIL" + Chr(13) + Text4 + Chr(10) + Chr(10) + Chr(13) +  
"PHONENUMBER" + Chr(13) + Text5 + Chr(10) + Chr(10) + Chr(13) + "ADDRESS" + Chr(13) + Text6  
+ Chr(10) + Chr(10) + Chr(13) + " BORRED BOOKS "
```

```
If List1.ListCount > 0 Then
```



```
For i = 0 To List1.ListCount
```

```
    kitap = List1.List(i)
    kitapicinraporac "select * from ABOUTDATETABLE where MEMBERID=" & Val(Text1) & " and
    MEMBERNAME=" & UCase(Trim(Text2)) & " and OUTBOOK=" & kitap & " and BOOK=" &
    "gelmedi" & ""
    If kitapicinraporac.RecordCount > 0 Then
        cikistarihi = kitapicinraporac("OUTDATE")
        gelmesigerekentarih = kitapicinraporac("INDATE")
        kitap = kitap + "    OUTDATE:" & Str(cikistarihi) + "    INDATE:" & Str(gelmesigerekentarih)
    End If
    Form10.RichTextBox1.Text = Form10.RichTextBox1.Text + Chr(13) + kitap
Next
End If
End Sub
```

```
Private Sub KDCButton7_Click()
    On Local Error Resume Next
    Frame2.Visible = False
    Command6.Visible = False
    If Not kitapekle = 1 Then
        MsgBox ("firstly, You must select ANY BOoK from BOoK SEARCH FORM")
        Exit Sub
    End If
    If DataGrid1.Columns(0) = "" Then
        MsgBox ("You have not selected any member." & Chr(10) & Chr(13) & "Please, select from MEMBER
        SEARCH FORM")
        Exit Sub
    Else
        bookvermeAc "select * from BOOKRECORDTABLE where BOOKID=" & kitapid & " and
        BOOKNAME=" & datagriddenisim & " and DURUM=" & "icerde" & ""
        If bookverme.RecordCount > 0 Then
            bookverme.MoveFirst
            Label11.Visible = True
            Label12.Visible = True
            Label13.Visible = True
            Label14.Visible = True
            Label11 = bookverme("COLOMN")
            Label12 = bookverme("ROW")
            bookverme("DURUM") = "disarda"
            bookverme.Update
```

```
        If Text9 = "" Then
            Text9 = Str(7)
        End If
        If Not Text9 = "" Then
            datekayitAc "select * from ABOUTDATETABLE"
            datekayit.AddNew
            datekayit("MEMBERID") = Val(Text1)
            datekayit("MEMBERNAME") = UCase(Trim(Text2))
            datekayit("OUTDATE") = Date
            newdate = Val(Text9) + Date
            datekayit("INDATE") = newdate
            datekayit("DAYPERIOT") = Val(Text9)
            datekayit("OUTBOOK") = UCase(Trim(datagriddenisim))
            datekayit("BOOK") = "gelmedi"
```

```

datekayit.Update
Text8 = newdate

outbookaramaAc "select * from ABOUTDATETABLE where MEMBERID=" & Val(Text1) & " and
BOOK=" + "gelmedi" + "" order by OUTDATE"
If outbookarama.RecordCount > 0 Then
outbookarama.MoveFirst
List1.Clear
Do Until outbookarama.EOF
List1.AddItem outbookarama("OUTBOOK")
outbookarama.MoveNext
Loop
End If
Else
MsgBox ("Please write DAY PERIOD.")
End If

Else
MsgBox ("All of these books are out")
End If

End If
End Sub

Private Sub KDCButton8_Click()
MSFlexGrid1.Clear
gridler
Frame2.Visible = True
'Command6.Visible = True
MSFlexGrid1.Rows = 1
i = 1

memberaramaAc "select * from MEMBERRECORDTABLE"
If memberarama.RecordCount > 0 Then
MSFlexGrid1.Rows = memberarama.RecordCount + 1
memberaramaAc "SELECT MEMBERID,NAME,SURNAME,ADDRESS FROM
MEMBERRECORDTABLE order by MEMBERID"
memberarama.MoveFirst

Do Until memberarama.EOF
datekayitAc "select * from ABOUTDATETABLE where MEMBERID=" &
memberarama("MEMBERID") & " and BOOK=" + "gelmedi" + ""

If datekayit.RecordCount >= 1 Then

datekayit.MoveFirst
Do Until datekayit.EOF

If Date > datekayit("INDATE") Then

MSFlexGrid1.TextMatrix(i, 0) = memberarama("MEMBERID")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 0
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 1) = memberarama("NAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 1

```

```

MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 2) = memberarama("SURNAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 2
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 3) = memberarama("ADDRESS")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 3
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 4) = datekayit("OUTBOOK")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 4
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 5) = datekayit("INDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 5
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 6) = datekayit("OUTDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 6
MSFlexGrid1.CellBackColor = &HFF&
MSFlexGrid1.TextMatrix(i, 7) = datekayit("DAYPERIOT")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 7
MSFlexGrid1.CellBackColor = &HFF&
If IsNull(datekayit("COMINGDATE")) Then
MSFlexGrid1.TextMatrix(i, 8) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 8
MSFlexGrid1.CellBackColor = &HFF&
Else
MSFlexGrid1.TextMatrix(i, 8) = datekayit("COMINGDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 8
MSFlexGrid1.CellBackColor = &HFF&
End If
i = i + 1
End If

If Date < datekayit("INDATE") Then

MSFlexGrid1.TextMatrix(i, 0) = memberarama("MEMBERID")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 0
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 1) = memberarama("NAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 1
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 2) = memberarama("SURNAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 2
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 3) = memberarama("ADDRESS")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 3
MSFlexGrid1.CellBackColor = &H80FFFF

```



```

MSFlexGrid1.TextMatrix(i, 4) = datekayit("OUTBOOK")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 4
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 5) = datekayit("INDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 5
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 6) = datekayit("OUTDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 6
MSFlexGrid1.CellBackColor = &H80FFFF
MSFlexGrid1.TextMatrix(i, 7) = datekayit("DAYPERIOT")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 7
MSFlexGrid1.CellBackColor = &H80FFFF
If IsNull(datekayit("COMINGDATE")) Then
MSFlexGrid1.TextMatrix(i, 8) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 8
MSFlexGrid1.CellBackColor = &HFF&
Else
MSFlexGrid1.TextMatrix(i, 8) = datekayit("COMINGDATE")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 8
MSFlexGrid1.CellBackColor = &HFF&
End If
i = i + 1
End If

datekayit.MoveNext
Loop

```

Else

```

MSFlexGrid1.TextMatrix(i, 0) = memberarama("MEMBERID")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 0
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 1) = memberarama("NAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 1
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 2) = memberarama("SURNAME")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 2
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 3) = memberarama("ADDRESS")
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 3
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 4) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 4
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 5) = ""

```

```

MSFlexGrid1.Row = i
MSFlexGrid1.Col = 5
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 6) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 6
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 7) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 7
MSFlexGrid1.CellBackColor = &H80FF80
MSFlexGrid1.TextMatrix(i, 8) = ""
MSFlexGrid1.Row = i
MSFlexGrid1.Col = 8
MSFlexGrid1.CellBackColor = &H80FF80
i = i + 1
End If

```

```

memberarama.MoveNext
Loop

```

```

End If
End Sub

```

```

Private Sub KDCButton9_Click()
On Local Error Resume Next
If Not DataGrid1.Columns(0) = "" Then
memberaramaAc "select * from MEMBERRECORDTABLE where MEMBERID=" &
Val(DataGrid1.Columns(0)) & " and NAME=" + DataGrid1.Columns(1) + " and SURNAME=" +
DataGrid1.Columns(2) + ""
outbookaramaAc "select * from ABOUTDATETABLE where MEMBERID=" &
Val(DataGrid1.Columns(0)) & " and BOOK=" + "gelmedi" + " order by OUTDATE"
cevap = MsgBox("Do you want to delete record ' " + DataGrid1.Columns(1) + " '?", vbYesNo +
vbCritical, "DELETING")
If cevap = vbYes Then
If outbookarama.RecordCount > 0 Then
If outbookarama.RecordCount > 1 Then
MsgBox ("Member has books not returned")
End If
If outbookarama.RecordCount = 1 Then
MsgBox ("Member has book not returned")
End If
outbookarama.MoveFirst
List1.Clear
Do Until outbookarama.EOF
List1.AddItem outbookarama("OUTBOOK")
outbookarama.MoveNext
Loop
GoTo skip
End If
memberarama.Delete
outbookaramaAc "select * from ABOUTDATETABLE where MEMBERID=" &
Val(DataGrid1.Columns(0)) & " order by OUTDATE"
outbookarama.MoveFirst
If outbookarama.RecordCount > 0 Then

```

```

        outbookarama.Delete
        outbookarama.MoveNext
    End If
End If

```

```

skip:
End If
End Sub

```

```

Private Sub List1_dblClick()
If Not List1.Text = "" Then
DataGrid1.BackColor = &HFFFFFF
outbookaramaAc "select * from ABOUTDATETABLE WHERE OUTBOOK=" +
List1.List(List1.ListIndex) + ""
outbookarama.MoveFirst
Do Until outbookarama.EOF
'If Not IsNull(outbookarama("COMINGDATE")) Then
If Date > outbookarama("INDATE") Then
DataGrid1.BackColor = &HFF&
End If
'End If
outbookarama.MoveNext
Loop
Set DataGrid1.DataSource = outbookarama
End If
End Sub

```

```

'Private Sub MSFlexGrid1_SelChange()
'MSFlexGrid1.FocusRect = flexFocusNone
'End Sub

```

```

Private Sub Text9_Change()
If Val(Text9) > 0 Then
Text8 = Val(Text9) + Date
End If
End Sub

```

```

Public Function gridler()
MSFlexGrid1.Cols = 9
MSFlexGrid1.FixedCols = 0
MSFlexGrid1.Row = 0
MSFlexGrid1.Col = 0
MSFlexGrid1.Text = "M.ID"
MSFlexGrid1.Col = 1
MSFlexGrid1.Text = "MEMBER NAME"
MSFlexGrid1.Col = 2
MSFlexGrid1.Text = "MEMBER SURNAME"
MSFlexGrid1.Col = 3
MSFlexGrid1.Text = "PUBLISHER"
MSFlexGrid1.Col = 4
MSFlexGrid1.Text = "BOOK"
MSFlexGrid1.Col = 5
MSFlexGrid1.Text = "INDATE"
MSFlexGrid1.Col = 6

```



```

MSFlexGrid1.Text = "OUTDATE"
MSFlexGrid1.Col = 7
MSFlexGrid1.Text = "PERIOT"
MSFlexGrid1.Col = 8
MSFlexGrid1.Text = "COMINGDATE"
MSFlexGrid1.ColWidth(0) = 500
MSFlexGrid1.ColWidth(1) = 2000
MSFlexGrid1.ColWidth(2) = 2000
MSFlexGrid1.ColWidth(3) = 2000
MSFlexGrid1.ColWidth(4) = 2000
MSFlexGrid1.ColWidth(5) = 900
MSFlexGrid1.ColWidth(6) = 900
MSFlexGrid1.ColWidth(7) = 700
MSFlexGrid1.ColWidth(8) = 1300
End Function

```

### 8.1.5 Administer form

```

Private Sub Form_Load()
Frame2.Visible = False
Frame3.Visible = False
Text4 = ""
Text5 = ""
Text6 = ""
Text7 = ""
Text8 = ""
Text1.Text = ""
Text1.ToolTipText = "Enter The Administer Name"
Text2.Text = ""
Text2.ToolTipText = "Enter The Administer Password"

```

End Sub

```

Private Sub KDCButton1_Click()
If Text9 <> "" And Text10 <> "" Then
If Text10 = Text11 Then
passwordkayitAc "select * from passwordtable"
passwordkayit.AddNew
passwordkayit("username") = Text9
passwordkayit("userpassword") = Text10
passwordkayit.Update
Else
MsgBox ("wrong confirm password")
End If

```

```

Frame3.Visible = False
Text9 = ""
Text10 = ""
Text11 = ""
End If
End Sub

```

```

Private Sub KDCButton2_Click()
passwordkayitAc "select * from passwordtable where username='" + Text4 + "' and userpassword='" +

```

```

Text5 + ""
If passwordkayit.RecordCount > 0 Then
If Text7 <> "" And Text8 <> "" Then
If Text7 = Text8 Then
passwordkayit("username") = Text6
passwordkayit("userpassword") = Text7
passwordkayit.Update
Frame2.Visible = False
Text4 = ""
Text5 = ""
Text6 = ""
Text7 = ""
Text8 = ""
KDCButton6.Visible = True
Else
MsgBox ("wrong confirm password")
End If
End If
Else
MsgBox ("wrong password or username")
End If
End Sub

```

```

Private Sub KDCButton3_Click()
Frame2.Visible = False
KDCButton6.Visible = True
KDCButton4.Visible = True
KDCButton5.Visible = True
End Sub

```

```

Private Sub KDCButton4_Click()
Frame2.Visible = True
KDCButton6.Visible = False
KDCButton4.Visible = False
KDCButton5.Visible = False
End Sub

```

```

Private Sub KDCButton5_Click()
End
End Sub

```

```

Private Sub KDCButton6_Click()
Static i As Integer
passwordkayitAc "select * from passwordtable where username='" + Text1 + "' and userpassword='" +
Text2 + "'"
If passwordkayit.RecordCount > 0 Or "programciyaizinver" = Text1 Then
MDIForm1.mnBOOK.Enabled = True
MDIForm1.mnMEMBER.Enabled = True
MDIForm1.mnPASSWORD.Enabled = True
MDIForm1.mnREPORT.Enabled = True
Form6.Visible = False
passwordkayitAc "select * from passwordtable "
passwordkayit.MoveFirst
If passwordkayit("userpassword") = Text2 Then
passwordkayitAc "select * from passwordtable"
Set Form11.DataGrid1.DataSource = passwordkayit
Form11.Show

```

```

        End If
Text1 = ""
Text2 = ""

Else
i = i + 1
If i = 3 Then
MsgBox ("Program will be shut down")
End
End If
End If

End Sub

```

### 8.1.6 Book Search form

```

Private Sub DataGrid1_DblClick()
On Local Error GoTo git
If Not DataGrid1.Columns(1) = "" Then
datagriddenisim = DataGrid1.Columns(1)
bookkaramaAc "select * from BOOKRECORDTABLE where BOOKNAME=" + datagriddenisim + ""
Y = bookkarama.RecordCount
bookkarama.MoveFirst
kitapno = bookkarama("BOOKID")
Text5 = bookkarama("TITLE")
Text6.Enabled = True
Text7.Enabled = True
Text6 = bookkarama("COLOMN")
Text7 = bookkarama("ROW")
Label8 = bookkarama("COLOMN")
Column = bookkarama("COLOMN")
Label9 = bookkarama("ROW")
Row = bookkarama("ROW")
Text1 = bookkarama("BOOKID")
Text2 = bookkarama("BOOKNAME")
Text3 = bookkarama("AUTHOR")
Text4 = bookkarama("PUBLISHER")
Label12 = bookkarama("DURUM")
booknumberAc "select * from BOOKNUMBER where BOOKNAME=" + DataGrid1.Columns(0) + ""
tane = booknumber("NUMBER")
Label10 = tane
bookkarama.MoveFirst
End If
git:
End Sub

```

```

Private Sub Form_Load()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End Sub

```



```

Private Sub KDCButton1_Click()
bookvermeAc "select * from BOOKRECORDTABLE where BOOKNAME=" + datagriddenisim + " and
DURUM=" + "icerde" + ""
If bookverme.RecordCount > 0 Then
If datagriddenisim <> "" And bookkarama("DURUM") = "icerde" Then
bookvermeAc "select * from BOOKRECORDTABLE where BOOKID=" & kitapid & " and
BOOKNAME=" + datagriddenisim + " and DURUM=" + "icerde" + ""
If bookverme.RecordCount > 0 Then
Form7.Label11.Visible = True
Form7.Label12.Visible = True
Form7.Label11 = kitapcolumn
Form7.Label12 = kitaprow
Form7.Label13.Visible = True
Form7.Show
Form5.Visible = False
kitapekle = 1
Form7.Label14.Visible = True
Form7.Label10.Visible = True
Form7.Label9.Visible = True
Form7.Text8.Visible = True
Form7.Text9.Visible = True
Else
MsgBox ("book are out")
End If
Else
MsgBox ("all these books are out")
End If
End Sub

Private Sub KDCButton2_Click()
On Local Error GoTo git
If Not DataGrid1.Columns(0) = "" Then
bookkayitAc "select * from BOOKRECORDTABLE where BOOKID=" & Val(DataGrid1.Columns(0)) &
" and BOOKNAME=" + DataGrid1.Columns(1) + ""
booknumberAc "select * from BOOKNUMBER where BOOKNAME=" + bookkayit("BOOKNAME") +
""
If bookkayit.RecordCount > 0 Then
cevap = MsgBox("There is just record has such a BOOKID.Do you want to change the record.", vbYesNo
+ vbCritical, "WARNING")
If cevap = vbYes Then
If bookkayit("BOOKNAME") <> UCase(Trim(Text2)) Then
If booknumber("NUMBER") > 1 Then
booknumber("NUMBER") = booknumber("NUMBER") - 1
booknumber.Update
ElseIf booknumber("NUMBER") = 1 Then
booknumber.Delete
End If
booknumberAc "select * from BOOKNUMBER where BOOKNAME=" + UCase(Trim(Text2)) + ""
If booknumber.RecordCount > 0 Then
booknumber("NUMBER") = booknumber("NUMBER") + 1
booknumber.Update
Else
booknumber("BOOKNAME") = UCase(Trim(Text2))
booknumber("AUTHOR") = UCase(Trim(Text3))

```

```

booknumber("PUBLISHER") = UCase(Trim(Text4))
booknumber("NUMBER") = 1
booknumber.Update
End If
End If
bookkayit("BOOKNAME") = UCase(Trim(Text2))
bookkayit("BOOKID") = UCase(Trim(Text1))
bookkayit("AUTHOR") = UCase(Trim(Text3))
bookkayit("PUBLISHER") = UCase(Trim(Text4))
bookkayit("TITLE") = UCase(Trim(Text5))
bookkayit("COLOMN") = UCase(Trim(Text6))
bookkayit("ROW") = UCase(Trim(Text7))
bookkayit("DURUM") = "icerde"
bookkayit.Update
Text6.Enabled = False
Text7.Enabled = False
End If
End If
End If
git:
End Sub

Private Sub KDCButton3_Click()
bookaramaAc "select * from BOOKRECORDTABLE"
If bookarama.RecordCount > 0 Then
Dim SORGU As String
If Text1 = "" And Text2 = "" And Text3 = "" And Text4 = "" Then
bookaramaAc "SELECT BOOKID,BOOKNAME,AUTHOR,PUBLISHER FROM
BOOKRECORDTABLE order by BOOKID"
booknumberAc "SELECT BOOKNAME,AUTHOR,PUBLISHER FROM BOOKNUMBER order by
BOOKNAME"

Else

SORGU = "SELECT BOOKID,BOOKNAME,AUTHOR,PUBLISHER from BOOKRECORDTABLE
WHERE"

If Not Trim(Text1) = "" Then
SORGU = SORGU + " BOOKID =" & Val(Text1)

End If

If Not Trim(Text2) = "" Then
If Not Trim(Text1) = "" Then
SORGU = SORGU + " AND BOOKNAME LIKE '" + Trim(Text2) + "%' "
Else
SORGU = SORGU + " BOOKNAME LIKE '" + Trim(Text2) + "%' "
End If
End If

If Not Text3 = "" Then
If Trim(Text1) = "" And Trim(Text2) = "" Then
SORGU = SORGU + " AUTHOR LIKE '" + Trim(Text3) + "%' "
Else
SORGU = SORGU + " AND AUTHOR LIKE '" + Trim(Text3) + "%' "
End If
End If

```

```

If Not Text4 = "" Then
    If Trim(Text1) = "" And Trim(Text2) = "" And Trim(Text3) = "" Then
        SORGU = SORGU + "PUBLISHER like '" + Trim(Text4) + "%'"
    Else
        SORGU = SORGU + "and PUBLISHER like '" + Trim(Text4) + "%'"
    End If
End If
bookaramaAc SORGU + " order by BOOKID"
Set DataGrid1.DataSource = bookarama
GoTo skip
End If
Y = booknumber.RecordCount
Set DataGrid1.DataSource = booknumber

End If
' If Not Trim(Text1) = "" Then
' SORGU = SORGU + " BOOKID LIKE'" + Trim(Text1) + "%' "
' End If
' If Not Trim(Text2) = "" Then
' If Not Trim(Text1) = "" Then
' SORGU = SORGU + " AND BOOKNAME LIKE'" + Trim(Text2) + "%' "
' Else
' SORGU = SORGU + " BOOKNAME LIKE'" + Trim(Text2) + "%' "
' End If
' End If
' If Not Trim(Text3) = "" Then
' If Trim(Text1) = "" And Trim(Text2) = "" Then
' SORGU = SORGU + " AUTHOR LIKE'" + Trim(Text3) + "%' "
' Else
' SORGU = SORGU + " AND AUTHOR LIKE'" + Trim(Text3) + "%' "
' End If
' End If
' If Not Text4 = "" Then
' If Trim(Text1) = "" Or Trim(Text2) = "" Or Trim(Text3) = "" Then
' SORGU = SORGU + "BOOKID like '" + Trim(Text4) + "%'"
' Else
' SORGU = SORGU + "and BOOKID like '" + Trim(Text4) + "%'"
' End If
' End If
' bulAc SORGU + " ORDER BY BOOKNAME"
' Set DataGrid1.DataSource = bul
' End If
' End If
skip:
End Sub

Private Sub SEARCH_Click()
bookaramaAc "select * from BOOKRECORDTABLE"
If bookarama.RecordCount > 0 Then
Dim SORGU As String
If Text1 = "" And Text2 = "" And Text3 = "" And Text4 = "" Then
bookaramaAc "SELECT BOOKID,BOOKNAME,AUTHOR,PUBLISHER FROM
BOOKRECORDTABLE order by BOOKID"
booknumberAc "SELECT BOOKNAME,AUTHOR,PUBLISHER FROM BOOKNUMBER order by
BOOKNAME"

```



Else

SORGU = "SELECT BOOKID,BOOKNAME,AUTHOR,PUBLISHER from BOOKRECORDTABLE  
WHERE"

If Not Trim(Text1) = "" Then  
SORGU = SORGU + " BOOKID =" & Val(Text1)

End If

If Not Trim(Text2) = "" Then  
If Not Trim(Text1) = "" Then  
SORGU = SORGU + " AND BOOKNAME LIKE '" + Trim(Text2) + "%' "  
Else  
SORGU = SORGU + " BOOKNAME LIKE '" + Trim(Text2) + "%' "  
End If  
End If

If Not Text3 = "" Then  
If Trim(Text1) = "" And Trim(Text2) = "" Then  
SORGU = SORGU + " AUTHOR LIKE '" + Trim(Text3) + "%' "  
Else  
SORGU = SORGU + " AND AUTHOR LIKE '" + Trim(Text3) + "%' "  
End If  
End If

If Not Text4 = "" Then  
If Trim(Text1) = "" And Trim(Text2) = "" And Trim(Text3) = "" Then  
SORGU = SORGU + "PUBLISHER like '" + Trim(Text4) + "%'"  
Else  
SORGU = SORGU + "and PUBLISHER like '" + Trim(Text4) + "%'"  
End If  
End If

bookaramaAc SORGU + " order by BOOKID"  
Set DataGrid1.DataSource = bookarama  
GoTo skip  
End If  
Y = booknumber.RecordCount  
Set DataGrid1.DataSource = booknumber

End If

' If Not Trim(Text1) = "" Then  
' SORGU = SORGU + " BOOKID LIKE'" + Trim(Text1) + "%' "  
' End If  
' If Not Trim(Text2) = "" Then  
' If Not Trim(Text1) = "" Then  
' SORGU = SORGU + " AND BOOKNAME LIKE'" + Trim(Text2) + "%' "  
' Else  
' SORGU = SORGU + " BOOKNAME LIKE'" + Trim(Text2) + "%' "  
' End If  
' End If  
' If Not Trim(Text3) = "" Then  
' If Trim(Text1) = "" And Trim(Text2) = "" Then  
' SORGU = SORGU + " AUTHOR LIKE'" + Trim(Text3) + "%' "  
' Else  
' SORGU = SORGU + " AND AUTHOR LIKE'" + Trim(Text3) + "%' "  
' End If

```

' End If
' If Not Text4 = "" Then
' If Trim(Text1) = "" Or Trim(Text2) = "" Or Trim(Text3) = "" Then
' SORGU = SORGU + "BOOKID like " + Trim(Text4) + "%'"
'Else
'SORGU = SORGU + "and BOOKID like " + Trim(Text4) + "%'"
'End If
'End If
' bulAç SORGU + " ORDER BY BOOKNAME"
' Set DataGrid1.DataSource = bul
'End If
'End If
skip:
End Sub

```

### 8.1.7 New Membwe Form

```

Private Sub Form_Load()
Text1.Text = "ID will be entered "
Text2.Text = "enter name"
Text3.Text = "enter surname"
Text4.Text = "enter address"
Text5.Text = "enter e_mail"
Text6.Text = "enter phonenumber"
Text7.Text = "enter age"

```

End Sub

```

Private Sub KDCButton1_Click()
Text1 = ""
Text2 = ""
Text3 = ""
Text4 = ""
Text5 = ""
Text6 = ""
Text7 = ""
Text2_GotFocus
End Sub

```

```

Private Sub KDCButton2_Click()
If Text1 <> "MEMBERID will be entered " And Text1 <> "" And Text2 <> "" And Text3 <> "" And Text4
<> "" And Text5 <> "" And Text6 <> "" Then
memberkayitAc "select * from MEMBERRECORDTABLE"

```

```

If memberkayit.RecordCount > 0 Then
memberkayitAc "select * from MEMBERRECORDTABLE where MEMBERID=" & Val(Text1)
If memberkayit.RecordCount > 0 Then
cevap = MsgBox("There is just record has such a MEMBERID.Do you want to change the record.",
vbYesNo + vbDefaultButton1, "WARNING")
If cevap = vbYes Then
memberkayit("MEMBERID") = Val(Text1)
memberkayit("NAME") = UCase(Trim(Text2))
memberkayit("SURNAME") = UCase(Trim(Text3))
memberkayit("ADDRESS") = UCase(Trim(Text4))
memberkayit("E_MAIL") = Trim(Text5)

```

```

memberkayit("TELEPHONE") = Trim(Text6)
memberkayit("AGE") = Trim(Text7)
memberkayit("GENDER") = UCase(Trim(Combo1.Text))
memberkayit.Update
End If
Else
memberkayit.AddNew
memberkayit("MEMBERID") = Val(Text1)
memberkayit("NAME") = UCase(Trim(Text2))
memberkayit("SURNAME") = UCase(Trim(Text3))
memberkayit("ADDRESS") = UCase(Trim(Text4))
memberkayit("E_MAIL") = Trim(Text5)
memberkayit("TELEPHONE") = Trim(Text6)
memberkayit("AGE") = Trim(Text7)
memberkayit("GENDER") = UCase(Trim(Combo1.Text))
memberkayit.Update
End If
Else
memberkayit.AddNew
memberkayit("MEMBERID") = UCase(Trim(Text1))
memberkayit("NAME") = UCase(Trim(Text2))
memberkayit("SURNAME") = UCase(Trim(Text3))
memberkayit("ADDRESS") = UCase(Trim(Text4))
memberkayit("E_MAIL") = Trim(Text5)
memberkayit("TELEPHONE") = Trim(Text6)
memberkayit("AGE") = Trim(Text7)
memberkayit("GENDER") = UCase(Trim(Combo1.Text))
memberkayit.Update
End If
Text1.Text = "ID will be entered "
Text2.Text = "enter name"
Text3.Text = "enter surname"
Text4.Text = "enter address"
Text5.Text = "enter e_mail"
Text6.Text = "enter phonenumber"
Text7.Text = "enter age"
Else
MsgBox ("You hve to fill into all information department")
End If
End Sub

Private Sub KDCButton3_Click()
Form9.Show
Form4.Visible = False
End Sub

Private Sub KDCButton4_Click()
Form4.Visible = False
End Sub

Private Sub Text2_GotFocus()
memberkayitAc "select * from MEMBERRECORDTABLE"

If memberkayit.RecordCount > 0 Then
memberkayit.MoveFirst
For i = 1 To memberkayit.RecordCount
If i = memberkayit("MEMBERID") Then

```



```

GoTo git
ElseIf i < memberkayit("MEMBERID") Then
Exit For
End If
git: memberkayit.MoveNext
Next
Text1 = i
Else
Text1 = Str(1)
End If
End Sub

```

### **8.1.8 Main Book Form**

```

Private Sub KDCButton1_Click()
Form1.Visible = True
Form3.Visible = False
End Sub

```

```

Private Sub KDCButton2_Click()
Form5.Show
Form3.Visible = False
End Sub

```

```

Private Sub KDCButton3_Click()
Form8.Visible = True
Form3.Visible = False
End Sub

```

```

Private Sub KDCButton4_Click()
Form3.Visible = False
End Sub

```

### **8.1.9 Sifrelerformu form**

```

Private Sub Command1_Click()
Form11.Visible = False
End Sub

```

### **8.1.10 Raporformu Form**

```

Private Sub Combo2_Click()
RichTextBox1.SelFontSize = Val(Combo2.Text)
End Sub

```

```

Private Sub Combo3_Click()
Text1 = Combo3.Text
raporac "select * from REPORTABLOSU"
If rapor.RecordCount > 0 Then
rapor.MoveFirst

```

```

Do Until rapor.EOF
If Trim(Combo3.Text) = Trim(rapor("RAPORADI")) Then
If Not IsNull(rapor("RAPORADI")) Then
RichTextBox1.TextRTF = rapor("RAPORICERIGI")
RichTextBox1.BackColor = rapor("RAPORBACKGROUND")
Else
RichTextBox1.TextRTF = ""
End If
End If
rapor.MoveNext
Loop
End If
End Sub

Private Sub Combo1_Click()
If RichTextBox1.SelStart = 0 Then
If RichTextBox1.SelLength = Len(RichTextBox1.Text) Then
RichTextBox1.Font = Combo1.Text
Else
RichTextBox1.SelFontName = Combo1.Text
End If
End If
If RichTextBox1.SelStart <> 0 Then RichTextBox1.SelFontName = Combo1.Text
End Sub

Private Sub Combo3_DropDown()
Combo3.Clear
raporac "select * from REPORTABLOSU"
If rapor.RecordCount > 0 Then
rapor.MoveFirst
Do Until rapor.EOF
Combo3.AddItem rapor("RAPORADI")
rapor.MoveNext
Loop
End If
End Sub

Private Sub Form_Load()
Veriac
'empireofsky__
For i = 0 To Screen.FontCount - 1
Combo1.AddItem Screen.Fonts(i)
Next i
For i = 1 To 100
Combo2.AddItem i
Next i
Form_Resize
Combo2.Text = RichTextBox1.SelFontSize
Combo1.Text = RichTextBox1.SelFontName
raporac "select * from REPORTABLOSU"
If rapor.RecordCount > 0 Then
rapor.MoveFirst
Do Until rapor.EOF
Combo3.AddItem rapor("RAPORADI")
rapor.MoveNext

```

```

Loop
End If
Form10.Top = 0
Form10.Left = (Screen.Width - (Form10.Width + 350))
Form10.Width = 10700
Form10.Height = 8880
Text1.Enabled = False
End Sub

Private Sub Form_Unload(Cancel As Integer)
Close
End Sub

Private Sub RichTextBox1_SelChange()

If IsNull(RichTextBox1.SelBullet) Then
Form10.ToolBar1.Buttons("Kopyala").Enabled = False
Else
Form10.ToolBar1.Buttons("Kopyala").Value = Abs(RichTextBox1.SelBullet)
Form10.ToolBar1.Buttons("Kopyala").Enabled = True
End If

If IsNull(RichTextBox1.SelBold) Then
Form10.ToolBar1.Buttons("Kalin").MixedState = True
Else
Form10.ToolBar1.Buttons("Kalin").Value = Abs(RichTextBox1.SelBold)
Form10.ToolBar1.Buttons("Kalin").MixedState = False
End If

If IsNull(RichTextBox1.SelItalic) Then
Form10.ToolBar1.Buttons("Egik").MixedState = True
Else
Form10.ToolBar1.Buttons("Egik").Value = Abs(RichTextBox1.SelItalic)
Form10.ToolBar1.Buttons("Egik").MixedState = False
End If

If IsNull(RichTextBox1.SelUnderline) Then
Form10.ToolBar1.Buttons("AltÇ").MixedState = True
Else
Form10.ToolBar1.Buttons("AltÇ").Value = Abs(RichTextBox1.SelUnderline)
Form10.ToolBar1.Buttons("AltÇ").MixedState = False
End If

If IsNull(RichTextBox1.SelFontSize) Then
Combo2.Text = ""
Else
Combo2.Text = RichTextBox1.SelFontSize
End If
If IsNull(RichTextBox1.SelFontName) Then
Combo1.Text = ""
Else
Combo1.Text = RichTextBox1.SelFontName
End If

End Sub

```



```

Private Sub ToolBar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Dim bul As Integer

raporac "select * from REPORTABLOSU"
On Error Resume Next
Select Case Button.Key

Case "Yeni":
Combo3.ListIndex = -1
RichTextBox1.TextRTF = ""

Case "Kaydet":
rapadı = InputBox("Enter report name that will be recorded", "REPORT NAME")
If Not rapadı = "" Then
raporac "select * from REPORTABLOSU where raporadı=" + UCase(Trim(rapadı)) + ""
Text1.Enabled = False
Text1 = rapadı
If rapor.RecordCount > 0 Then
cevap = MsgBox("just there is a report which name is " + UCase(Trim(rapadı)) + " " + Chr(10) +
Chr(13) + " Do you want to change content of this Report", vbYesNo + vbApplicationModal,
"CHANGING CONTENT OF REPORT")
If cevap = vbYes Then
rapor("RAPORADI") = UCase(Trim(Text1))
If RichTextBox1.TextRTF = "" Then
MsgBox ("You have not entered any content about report that will be recorded")
'Timer1.Enabled = True
'Timer1.Interval = 4000
Text1 = ""
'Frame1.Visible = True
'Label3 = "You have not entered any content about report that will be recorded"
GoTo skip
Else
rapor("RAPORICERIGI") = RichTextBox1.TextRTF
End If
rapor("RAPORBACKROUND") = RichTextBox1.BackColor
rapor.Update
MsgBox "Content of Report has been changed..", vbInformation, "CHANGING CONTENT OF REPORT"
Else
Text1 = ""
End If
Else
rapor.AddNew
rapor("RAPORADI") = UCase(Trim(Text1))
If RichTextBox1.Text = "" Then
MsgBox ("You have not entered any content about report that will be recorded")
'Timer1.Enabled = True
'Timer1.Interval = 4000
Text1 = ""
'Frame1.Visible = True
'Label3 = "You have not entered any content about report that will be recorded"
GoTo skip
Else
rapor("RAPORICERIGI") = RichTextBox1.TextRTF
End If

```

```

rapor("RAPORBACKGROUND") = RichTextBox1.BackColor
rapor.Update
MsgBox "Report has been saved..", vbInformation, "RECORDING"
skip: End If

Else
MsgBox ("You have to enter reportname...")
'Frame1.Visible = True
'Label3 = "You have to enter reportname..."
'Timer1.Interval = 4000
'Timer1.Enabled = True
End If
Case "Aç":
    sec = InputBox("select report name that will be indicated", "REPORT NAME")
    raporac "select * from REPORTABLOSU"
    If rapor.RecordCount > 0 Then
        i = 1
        rapor.MoveFirst
        Do Until rapor.EOF
            If Trim(sec) = Trim(rapor("RAPORADI")) Then
                bul = i
                End If
                i = i + 1
            Loop

        If bul > 0 Then Combo3.ListIndex = bul - 1
        End If

Case "yRenk":
    dlgCommonDialog.Flags = cdlCFEffects Or cdlCFBoth
    dlgCommonDialog.ShowFont

    RichTextBox1.SelFontName = dlgCommonDialog.FontName
    Combo1.Text = RichTextBox1.SelFontName

    RichTextBox1.SelFontSize = dlgCommonDialog.FontSize
    Combo2.Text = RichTextBox1.SelFontSize

    RichTextBox1.SelBold = dlgCommonDialog.FontBold
    RichTextBox1.SelItalic = dlgCommonDialog.FontItalic
    RichTextBox1.SelUnderline = dlgCommonDialog.FontUnderline
    RichTextBox1.SelColor = dlgCommonDialog.Color

Case "Renk":
    dlgCommonDialog.ShowColor
    RichTextBox1.BackColor = dlgCommonDialog.Color

Case "Yazdır":
    On Error Resume Next

    dlgCommonDialog.DialogTitle = "Print"
    dlgCommonDialog.CancelError = True
    dlgCommonDialog.Flags = cdlPDReturnDC + cdlPDNoPageNums
    If RichTextBox1.SelLength = 0 Then
        dlgCommonDialog.Flags = dlgCommonDialog.Flags + cdlPDAllPages
    
```

```

Else
    dlgCommonDialog.Flags = dlgCommonDialog.Flags + cdlPDSelection
End If
dlgCommonDialog.ShowPrinter
If Err <> MSComDlg.cdlCancel Then
    RichTextBox1.SelPrint dlgCommonDialog.hDC
End If

Case "Kes":
    On Error Resume Next
    Clipboard.SetText RichTextBox1.SelRTF
    RichTextBox1.SelText = vbNullString
Case "Kopyala":
    On Error Resume Next
    Clipboard.SetText RichTextBox1.SelRTF

Case "Yapıştır":
    On Error Resume Next
    RichTextBox1.SelRTF = Clipboard.GetText

Case "Kalın": RichTextBox1.SelBold = Not RichTextBox1.SelBold

Case "Eğik": RichTextBox1.SelItalic = Not RichTextBox1.SelItalic

Case "AltÇ": RichTextBox1.SelUnderline = Not RichTextBox1.SelUnderline

Case "Sol":
    RichTextBox1.SelAlignment = rtfLeft

Case "Ortaya":
    RichTextBox1.SelAlignment = rtfCenter

Case "Sağ":
    RichTextBox1.SelAlignment = rtfRight

Case "Sil":
    If Combo3.ListIndex < 0 Then
        MsgBox "Select report that will be deleted....!", vbCritical, "WARNING"
    End If
    If Combo3.ListIndex > -1 Then
        raporac "select * from REPORTABLOSU where RAPORADI=" + UCase(Trim(Combo3.Text)) +
        """"
        If rapor.RecordCount > 0 Then
            rapor.MoveFirst
            sıradaki: cevap = MsgBox("Now, Report which name is "" + rapor("RAPORADI") + "" will be delete",
            vbYesNoCancel + vbCritical, "DELETING")

            Select Case cevap

            Case vbYes: rapor.Delete
                Do Until rapor.EOF
                    rapor.MoveNext
                GoTo sıradaki
            Loop

            Case vbNo:

```



```

        rapor.MoveNext
    Do Until rapor.EOF
        GoTo siradaki
    Loop

    Case vbCancel:
        GoTo iptal

    End Select

End If

Reset
RichTextBox1.TextRTF = ""
Combo3.Clear

End If

End Select

iptal: raporac "select * from REPORTABLOSU"

If rapor.RecordCount > 0 Then
    rapor.MoveFirst
    Do Until rapor.EOF
        Combo3.AddItem rapor("RAPORADI")
        rapor.MoveNext
    Loop
End If

End Sub

Private Sub Form_Resize()
    On Error Resume Next
    RichTextBox1.RightMargin = RichTextBox1.Width - 400
    RichTextBox1.SelIndent = 300
End Sub

```

### 8.1.11 New Book Registration Form

```

Private Sub Form_Load()
    Text1.Text = "ID will be entered "
    Text2.Text = "enter Bookname"
    Text3.Text = "enter Author"
    Text4.Text = "enter Publisher"
    Text5.Text = "enter Title"
    Text6.Text = "enter Column no"
    Text7.Text = "enter Row no"

    Text1.ToolTipText = "Enter Book ID"
    Text2.ToolTipText = "Enter Book Name"
    Text3.ToolTipText = "Enter Author Name"
    Text4.ToolTipText = "Enter Publisher Name"
    Text5.ToolTipText = "Enter Content Of Book"
    Text6.ToolTipText = "Enter Colomn"

```

```
Text7.ToolTipText = "Enter Row"
```

```
End Sub
```

```
Private Sub KDCButton1_Click()
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Text3.Text = ""
```

```
Text4.Text = ""
```

```
Text5.Text = ""
```

```
Text6.Text = ""
```

```
Text7.Text = ""
```

```
bookkayitAc "select * from BOOKRECORDTABLE"
```

```
If bookkayit.RecordCount > 0 Then
```

```
bookkayit.MoveLast
```

```
Label9 = bookkayit("COLOMN")
```

```
Label11 = bookkayit("ROW")
```

```
End If
```

```
Text2_GotFocus
```

```
End Sub
```

```
Private Sub KDCButton2_Click()
```

```
If Text1 <> "BOOKID will be entered " And Text1 <> "" And Text2 <> "" And Text3 <> "" And Text4 <> "" And Text5 <> "" And Text6 <> "" And Text7 <> "" Then
```

```
bookkayitAc "select * from BOOKRECORDTABLE"
```

```
If bookkayit.RecordCount > 0 Then
```

```
bookkayitAc "select * from BOOKRECORDTABLE where BOOKID=" & Val(Text1)
```

```
booknumberAc "select * from BOOKNUMBER"
```

```
If bookkayit.RecordCount > 0 Then
```

```
booknumberAc "select * from BOOKNUMBER where BOOKNAME=" & bookkayit("BOOKNAME") + ""
```

```
cevap = MsgBox("There is just record has such a BOOKID.Do you want to change the record.", vbYesNo + vbCritical, "WARNING")
```

```
If cevap = vbYes Then
```

```
If bookkayit("BOOKNAME") <> UCase(Trim(Text2)) Then
```

```
If booknumber("NUMBER") > 1 Then
```

```
booknumber("NUMBER") = booknumber("NUMBER") - 1
```

```
booknumber.Update
```

```
ElseIf booknumber("NUMBER") = 1 Then
```

```
booknumber.Delete
```

```
End If
```

```
booknumberAc "select * from BOOKNUMBER where BOOKNAME=" & UCase(Trim(Text2)) + ""
```

```
If booknumber.RecordCount > 0 Then
```

```
booknumber("NUMBER") = booknumber("NUMBER") + 1
```

```
booknumber.Update
```

```
Else
```

```
booknumber.AddNew
```

```
booknumber("BOOKNAME") = UCase(Trim(Text2))
```

```
booknumber("AUTHOR") = UCase(Trim(Text3))
```

```
booknumber("PUBLISHER") = UCase(Trim(Text4))
```

```
booknumber("NUMBER") = 1
```

```
booknumber.Update
```

```

End If
End If
bookkayit("BOOKID") = UCase(Trim(Text1))
bookkayit("BOOKNAME") = UCase(Trim(Text2))
bookkayit("AUTHOR") = UCase(Trim(Text3))
bookkayit("PUBLISHER") = UCase(Trim(Text4))
bookkayit("TITLE") = UCase(Trim(Text5))
bookkayit("COLOMN") = UCase(Trim(Text6))
bookkayit("ROW") = UCase(Trim(Text7))
bookkayit("DURUM") = "icerde"
bookkayit.Update
End If

Else
If booknumber.RecordCount > 0 Then
booknumber("NUMBER") = booknumber("NUMBER") + 1
booknumber.Update
Else
booknumber.AddNew
booknumber("BOOKNAME") = UCase(Trim(Text2))
booknumber("AUTHOR") = UCase(Trim(Text3))
booknumber("PUBLISHER") = UCase(Trim(Text4))
booknumber("NUMBER") = 1
booknumber.Update
End If
bookkayit.AddNew
bookkayit("BOOKID") = UCase(Trim(Text1))
bookkayit("BOOKNAME") = UCase(Trim(Text2))
bookkayit("AUTHOR") = UCase(Trim(Text3))
bookkayit("PUBLISHER") = UCase(Trim(Text4))
bookkayit("TITLE") = UCase(Trim(Text5))
bookkayit("COLOMN") = UCase(Trim(Text6))
bookkayit("ROW") = UCase(Trim(Text7))
bookkayit("DURUM") = "icerde"
bookkayit.Update
End If
Else
If booknumber.RecordCount > 0 Then
booknumber("NUMBER") = booknumber("NUMBER") + 1
booknumber.Update
Else
booknumber.AddNew
booknumber("BOOKNAME") = UCase(Trim(Text2))
booknumber("AUTHOR") = UCase(Trim(Text3))
booknumber("PUBLISHER") = UCase(Trim(Text4))
booknumber("NUMBER") = 1
booknumber.Update
End If
bookkayit.AddNew
bookkayit("BOOKID") = UCase(Trim(Text1))
bookkayit("BOOKNAME") = UCase(Trim(Text2))
bookkayit("AUTHOR") = UCase(Trim(Text3))
bookkayit("PUBLISHER") = UCase(Trim(Text4))
bookkayit("TITLE") = UCase(Trim(Text5))
bookkayit("COLOMN") = UCase(Trim(Text6))
bookkayit("ROW") = UCase(Trim(Text7))
bookkayit("DURUM") = "icerde"

```



```

bookkayit.Update
End If
Text1.Text = "ID will be entered "
Text2.Text = "enter Bookname"
Text3.Text = "enter Author"
Text4.Text = "enter Publisher"
Text5.Text = "enter Title"
Text6.Text = "enter Column no"
Text7.Text = "enter Row no"
Else
MsgBox ("You hve to fill into all information department")
End If

```

```

End Sub

```

```

Private Sub KDCButton3_Click()
Form3.Visible = True
Form1.Visible = False
End Sub

```

```

Private Sub KDCButton4_Click()
Form1.Visible = False
End Sub

```

```

Private Sub Text2_GotFocus()

```

```

bookkayitAc "select * from BOOKRECORDTABLE"
If bookkayit.RecordCount > 0 Then
bookkayit.MoveFirst
For i = 1 To bookkayit.RecordCount
If i = bookkayit("BOOKID") Then
GoTo git
ElseIf i < bookkayit("BOOKID") Then
Exit For
End If
git: bookkayit.MoveNext
Next
Text1 = i
Else
Text1 = Str(1)
End If
End Sub

```

### 8.1.12 VeriTabanıAc Module

```

Option Explicit

```

```

Public Dosyayeri As String
Public Baglan As Connection
Public bookkayit As Recordset
Public memberkayit As Recordset
Public bookarama As Recordset
Public booknumber As Recordset
Public bookverme As Recordset
Public memberarama As Recordset
Public outbookarama As Recordset
Public datekayit As Recordset

```

```

Public bookgeldi As Recordset
Public passwordkayit As Recordset
Public kitapicinrapor As Recordset
Public rapor As Recordset
Public counter As Recordset

```

```

Public Function Veriac() As Boolean
    Dosyayeri = "C:\Documents and Settings\Hakkinen\Desktop\CARLOS2\LIBRARY.MDB"
    Set Baglan = New Connection
    Baglan.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Dosyayeri & ";Persist Security
Info=False"

```

```

End Function

```

```

Public Function VeriKapat() As Boolean
    'raporac.Close
    'datekayit.Close
    'bookkayit.Close
    'memberkayit.Close
    'bookkarama.Close
    'bookverme.Close
    'bookgeldi.Close
    'outbookkarama.Close
    'memberarama.Close
    'passwordkayit.Close
    'kitapicinrapor.Close
    Baglan.Close
    Set Baglan = Nothing
    Set bookkayit = Nothing
    Set kitapicinrapor = Nothing
    Set passwordkayit = Nothing
    Set memberkayit = Nothing
    Set bookkarama = Nothing
    Set booknumber = Nothing
    Set bookverme = Nothing
    Set bookgeldi = Nothing
    Set memberarama = Nothing
    Set rapor = Nothing
    Set outbookkarama = Nothing
    Set counter = Nothing
    Set datekayit = Nothing
End Function

```

```

Public Sub bookkayitAc(sql As String)
    Set bookkayit = New Recordset
    bookkayit.CursorLocation = adUseClient
    bookkayit.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub

```

```

Public Sub memberkayitAc(sql As String)
    Set memberkayit = New Recordset
    memberkayit.CursorLocation = adUseClient
    memberkayit.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub

```

```

Public Sub bookkaramaAc(sql As String)
    Set bookkarama = New Recordset
    bookkarama.CursorLocation = adUseClient

```

```

    bookarama.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub memberaramaAc(sql As String)
    Set memberarama = New Recordset
    memberarama.CursorLocation = adUseClient
    memberarama.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub outbookaramaAc(sql As String)
    Set outbookarama = New Recordset
    outbookarama.CursorLocation = adUseClient
    outbookarama.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub datekayitAc(sql As String)
    Set datekayit = New Recordset
    datekayit.CursorLocation = adUseClient
    datekayit.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub bookvermeAc(sql As String)
    Set bookverme = New Recordset
    bookverme.CursorLocation = adUseClient
    bookverme.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub bookgeldiAc(sql As String)
    Set bookgeldi = New Recordset
    bookgeldi.CursorLocation = adUseClient
    bookgeldi.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub passwordkayitAc(sql As String)
    Set passwordkayit = New Recordset
    passwordkayit.CursorLocation = adUseClient
    passwordkayit.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub kitapicinraporac(sql As String)
    Set kitapicinrapor = New Recordset
    kitapicinrapor.CursorLocation = adUseClient
    kitapicinrapor.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub raporac(sql As String)
    Set rapor = New Recordset
    rapor.CursorLocation = adUseClient
    rapor.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub counterac(sql As String)
    Set counter = New Recordset
    counter.CursorLocation = adUseClient
    counter.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub
Public Sub booknumberAc(sql As String)
    Set booknumber = New Recordset
    booknumber.CursorLocation = adUseClient
    booknumber.Open sql, Baglan, adOpenKeyset, adLockOptimistic
End Sub

```



### 8.1.13 Globaller Module

Global datagridden As Integer  
Global data2gridden As Integer  
Global kitapekle As Integer  
Global datagriddenisim As String  
Global kitaprow As Integer  
Global kitapcolumn As Integer  
Global kitapid As Integer