



**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**Multimedia Program For Video Compact  
Disk(CD) Company With Borland Delphi 6.0**

**Graduation Project  
COM400**

**Student: Hakan KILIÇ(20020468)**

**Supervisor: Assoc.Prof.Dr. Rahib ABIYEV**

**Nicosia-2007**



## TABLE OF CONTENTS

Contents	I
Acknowledgments	IV
Abstract	V
Introduction	VI
<b>1.INTRODUCTION TO DELPHI 6.0</b>	<b>1</b>
1.1 What is Delphi	1
1.1.2 What kind of programming can you do with Delphi?	2
1.1.3 How do they differ?	3
1.2 The VCL to Applications Developers	5
1.2.1 The VCL to Component Writers	5
1.2.2 The VCL is made up of components	6
1.3.1 Component Types	6
1.3.2 Standard Components	7
1.3.3 Custom components	8
1.3.4 Graphical components	8
1.3.5 Non-visual components	8
1.3.6 Structure of a component	9
1.3.7 Component properties	9
1.3.8 Properties provide access to internal storage field	9
1.4 Property-access methods	10
1.5 Types of properties	11
1.6 Methods	12
1.7 Events	12
1.8 Containership	14
1.9 Ownership	15
1.10 Parenthood	16
<b>2.BRIEF ABOUT DATABASE</b>	<b>17</b>
2.1 De-merits of absence of database	17
2.2 Merits of database	18
2.3 Introduction to database design	18
2.4 Database Models	19

2.4.1 Flat Model	19
2.4.2 Network Model	20
2.4.3 Relational Model	20
2.4.3.1 Why we use a Relational Database Design	21
2.5 Relationships between Tables	22
2.5.2 One-To-One Relationships	22
2.5.3 One-To-Many Relationships	22
2.6 Data Modeling	22
2.6.1 Database Normalization	23
2.6.2 Primary Key	23
2.6.3 Foreign Key	24
2.6.4 Compound Key	25
2. 2.7 Visual Basic Editor	25
2.8 Structured Query Language	27
2.9 Description of SQL	27
2.10 SQL Keywords	28
2.10.1 Data Retrieval	28
2.10.2 Data Manipulation	29
2.10.3 Data Transaction	29
2.10.4 Data Definition	30
2.11. Microsoft Access Database System	30
2.11.1 A Few Terms	30
2.11.2 Introductory Microsoft Access	31
2.11.3 Introduction to Tables	32
2.11.4 Table's data types	33
<b>3. SOFTWARE ABOUT CD APPLICATION PROGRAM</b>	<b>38</b>
3.1. Main Form	38
3.2. Search Product Form	39
3.3. Poster Form	41
3.4. Product Price Form	42
3.5. Rent Product Form	43
3.6. Product Selling Form	44
3.7. Register Product Form	46
3.8. Rented Product Form	47

3.9. Edit Product Form	48
3.10. Customer Analysis Form	50
3.11. Top Ten Form	51
3.12. Register Customer Form	52
3.13. Exit Form	53
Conclusion	54
<b>4.APPENDIX</b>	<b>55</b>
4.1.THE CODES OF MAIN FORM	55
4.2.THE CODES OF SEARCH PRODUCT FORM	64
4.3.THE CODES OF SEARCH ACCORDING TO CAST FORM	78
4.4.THE CODES OF PRODUCT PRICE FORM	83
4.5.THE CODES OF RENT PRODUCT FORM	85
4.6.THE CODES OF CD SELLING FORM	90
4.7.THE CODES OF REGISTER PRODUCT FORM	105
4.8.THE CODES OF RENTED PRODUCT FORM	112
4.9.THE CODES OF EDIT PRODUCT FORM	119
4.10.THE CODES OF CUSTOMER ANALYSIS FORM	129

## ACKNOWLEDGEMENT

*First of all, I thanked to my supervisor Assoc.Prof.Dr. Rahib ABIYEV who informed me about software.*

*I am very thankful for the help and support of my valuable friends Alp Soydan, Kağan Uzun, Gerçek Sezgin, Ünal Kurt, Uğur Emrah Çakmak, Rıdvan Sav and Erkan Kalkancı.*

*And also, very big thanks to my boss while I was doing my training -Mr. Ufuk Uygun. He taught me computer/ machine languages and he helped me about anything when I need.*

*And, I am very thankful for my valuable teachers who are teaching me and I send them my regards.*

*Finally, I sent my special thanks to my father Şentürk Kılıç for his financial and psychological support. And, again, I want to send my regards to all the professors who have being taught me.*



## **ABSTRACT**

This project is about CD Pursuit program. Borland Delphi 6.0 and Microsoft Office Access 2003 database were used in order to achieve the project. This program can be used in the market places where CDs are sold or rented.

First of all, it should be explained briefly why Delphi is being used nowadays. There are lots of programs which were created by using Delphi. For instance, these kinds of programs were created in visual basic languages, but visual basic language has lost its validity.

The most important feature of Delphi, and also its difference from Visual Basic is that you can create Class. Nowadays, Delphi is still being used and it is developed day by day.

The value of the Delphi can be understood with this fact. Microsoft has bought Delphi from Borland Company. This project will be reference for my next projects that I'll work on.

## INTRODUCTION

I started to work on this project to develop myself at the same time I started out to work as trainee. I came up with this idea when I went to a CD shop and asked the owner of the shop where the films that I was looking for were. The owner of the shop told me to check the film catalogs. So, I came up with the idea of studying on this project.

The aim of this program is to find; on which shelves the wanted CDs are and which actors/actresses films can be found in the store.

Later on, I developed the project and designed it to be more flexible for the users. I, of course, used the references of some pocket programs which have been designed previously and can be purchased from the web. But, all the codes and designing belongs to me. In this program, customer pursuit can be done as well.

Which customer bought which CD last, how many CD they have rented so far; the user of the program can reach all of these information. The flexibility and usability of Delphi helped me work with ease

## **1.INTRODUCTION TO DELPHI 6.0**

In this project I will answer some basic questions about Delphi, to give a feel for where it came from, what it has to offer, and where it is going in the future. This is an essential part of any course. We feel it is important for those studying a new programming language to understand the ideology and intended use of the language. Too many programmers are tempted to use the language that they know, rather than learn a new one to cope with the specific demands of the project that they have. At the end of this lecture, you should have gained sufficient understanding of the Delphi ideology to decide if it is a suitable language for a specific project that you have.

### **1.1 What is Delphi?**

Delphi is an object oriented, component based, visual, rapid development environment for event driven Windows applications, based on the Pascal language. Unlike other popular competing Rapid Application Development (RAD) tools, Delphi compiles the code you write and produces really tight, natively executable code for the target platform. In fact the most recent versions of Delphi optimise the compiled code and the resulting executables are as efficient as those compiled with any other compiler currently on the market. The term "visual" describes Delphi very well. All of the user interface development is conducted in a What You See Is What You Get environment (WYSIWYG), which means you can create polished, user friendly interfaces in a very short time, or prototype whole applications in a few hours.

Delphi is, in effect, the latest in a long and distinguished line of Pascal compilers (the previous versions of which went by the name "Turbo Pascal") from the company formerly known as Borland, now known as Inprise. In common with the Turbo Pascal compilers that preceded it, Delphi is not just a compiler, but a complete development environment. Some of the facilities that are included in the "Integrated Development Environment" (IDE) are listed below:

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



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

The development environment itself is extensible, and there are a number of add ins available to perform functions such as memory leak detection and profiling. In short, Delphi includes just about everything you need to write applications that will run on an Intel platform under Windows, but if your target platform is a Silicon Graphics running IRIX, or a Sun Sparc running SOLARIS, or even a PC running LINUX, then you will need to look elsewhere for your development tool.

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

### **1.1.1 What kind of programming can you do with Delphi?**

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

You probably won't be using it to write embedded systems for washing machines, toasters or fuel injection systems, but for more or less anything else, it can be used.

Some projects to which Delphi is suited:

- Simple, single user database applications
- Intermediate multi-user database applications
- Large scale multi-tier, multi-user database applications
- Internet applications
- Graphics Applications

- Multimedia Applications
- Image processing/Image recognition
- Data analysis
- System tools
- Communications tools using the Internet, Telephone or LAN
- Web based applications

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

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

### **1.1.2 How do they differ?**

Borland (as they were then) has a long tradition in the creation of high speed compilers. One of their best known products was Turbo Pascal - a tool that many programmers cut their teeth on. With the rise in importance of the Windows environment, it was only a matter of time before development tools started to appear that were specific to this new environment. In the very beginning, Windows produced SDKs (software development kits) that were totally non-visual (user interface development was totally separated from the development of the actual application), and required great patience and some genius to get anything working with. Whilst these tools slowly improved, they still required a really good understanding of the inner

workings of Windows a great extent these criticisms were dispatched by the release of Microsoft's Visual Basic product, which attempted to bring Windows development to the masses. It achieved this to a great extent too, and remains a popular product today. However, it suffered from several drawbacks:

- 1) It wasn't as stable as it might have been
- 2) It was an interpreted language and hence was slow to run
- 3) It had as its underlying language BASIC, and most "real" programmers weren't so keen!

Into this environment arrived the eye opening Delphi I product, and in many ways the standard for visual development tools for Windows was set. This first version was a 16 bit compiler, and produced executable code that would run on Windows 3.1 and Windows 3.11. Of course, Microsoft have ensured (up to now) that their 32 bit operating systems (Win95, Win98, and Win NT) will all run 16 bit applications, however, many of the features that were introduced in these newer operating systems are not accessible to the 16 bit applications developed with Delphi I. Delphi 2 was released quite soon after Delphi I, and in fact included a full distribution of Delphi I on the same CD. Delphi 2, (and all subsequent versions) have been 32 bit compilers, producing code that runs exclusively on 32bit Windows platforms. (We ignore for simplicity the WIN32S DLLs which allow Win 3.1x to run some 32 bit applications).

Delphi is currently standing at Version 4.0, with a new release (version 5.0) expected shortly. In its latest version, Delphi has become somewhat feature loaded, and as a result, we would argue, less stable than the earlier versions. However, in its defence, Delphi (and Borland products in general) have always been more stable than their competitors products, and the majority of Delphi 4's glitches are minor and forgivable - just don't try and copy/paste a selection of your code, midway through a debugging session!

The reasons for the version progression include the addition of new components, improvements in the development environment, the inclusion of more internet related support and improvements in the documentation. Delphi at version 4 is



a very mature product, and Inprise has always been responsive in developing the product in the direction that the market requires it to go. Predominantly this means right now, the inclusion of more and more Internet, Web and CORBA related tools and components - a trend we are assured continues with the release of version 5.0

For each version of Delphi there are several sub-versions, varying in cost and features, from the most basic "Developer" version to the most complete (and expensive) "Client Server" version. The variation in price is substantial, and if you are contemplating a purchase, you should study the feature list carefully to ensure you are not paying for features you will never use. Even the most basic "Developer" version contains the vast majority of the features you are likely to need on a day to day basis. Don't assume that you will need Client Server, simply because you are intending to write a large database application - The developer edition is quite capable of this.

## **1.2. The VCL to Applications Developers**

Applications Developers create complete applications by interacting with the Delphi visual environment (as mentioned earlier, this is a concept nonexistent in many other frameworks). These people use the VCL to create their user-interface and the other elements of their application: database connectivity, data validation, business rules, etc..

Applications Developers should know which properties, events, and methods each component makes available. Additionally, by understanding the VCL architecture, Applications Developers will be able to easily identify where they can improve their applications by extending components or creating new ones. Then they can maximize the capabilities of these components, and create better applications.

### **1.2.1 The VCL to Component Writers**

Component Writers expand on the existing VCL, either by developing new components, or by increasing the functionality of existing ones. Many component writers make their components available for Applications Developers to use.

A Component Writer must take their knowledge of the VCL a step further than that of the Application Developer. For example, they must know whether to write a new component or to extend an existing one when the need for a certain characteristic arises. This requires a greater knowledge of the VCL's inner workings.

### **1.2.2 The VCL is made up of components**

Components are the building blocks that developers use to design the user-interface and to provide some non-visual capabilities to their applications. To an Application Developer, a component is an object most commonly dragged from the Component palette and placed onto a form. Once on the form, one can manipulate the component's properties and add code to the component's various events to give the component a specific behavior. To a Component Writer, components are objects in Object Pascal code. Some components encapsulate the behavior of elements provided by the system, such as the standard Windows 95 controls. Other objects introduce entirely new visual or non-visual elements, in which case the component's code makes up the entire behavior of the component.

The complexity of different components varies widely. Some might be simple while others might encapsulate an elaborate task. There is no limit to what a component can do or be made up of. You can have a very simple component like a TLabel, or a much more complex component which encapsulates the complete functionality of a spreadsheet.

### **1.3.1 Component Types**

As a component writer, there are four primary types of components that you will work with in Delphi: standard controls, custom controls, graphical controls, and non-visual components. Although these component types are primarily of interest to component writers, it's not a bad idea for applications developers to be familiar with them. They are the foundations on which applications are built.



### 1.3.2 Standard Components,

Some of the components provided by Delphi 2.0 encapsulate the behavior of the standard Windows controls: TButton, TListBox. As a component writer, there are four primary types of components that you will work with in Delphi: standard controls, custom controls, graphical controls, and non-visual components. Although these component types are primarily of interest to component writers, it's not a bad idea for applications developers to be familiar with them. They are the foundations on which applications are built.

For example, you will find these components on the Standard page of the Component Palette. These components are Windows' common controls with Object Pascal wrappers around them.

Each standard component looks and works like the Windows' common control which it encapsulates. The VCL wrapper simply makes the control available to you in the form of a Delphi component—it doesn't define the common control's appearance or functionality, but rather, surfaces the ability to modify a control's appearance/functionality in the form of methods and properties. If you have the VCL source code, you can examine how the VCL wraps these controls in the file `STDCTRLS.PAS`.

If you want to use these standard components unchanged, there is no need to understand how the VCL wraps them. If, however, you want to extend or change one of these components, then you must understand how the Windows' common control is wrapped by the VCL into a Delphi component.

For example, the Windows class `LISTBOX` can display the list box items in multiple columns. This capability, however, isn't surfaced by Delphi's `TListBox` component (which encapsulates the Windows `LISTBOX` class). (`TListBox` only displays items in a single column.) Surfacing this capability requires that you override the default creation of the `TListBox` component.

This example also serves to illustrate why it is important for Applications Developers to understand the VCL. Just knowing this tidbit of information helps you to identify where enhancements to the existing library of components can help make your life easier and more productive.

### **1.3.3 Custom components**

Unlike standard components, custom components are controls that don't already have a method for displaying themselves, nor do they have a defined behavior. The Component Writer must provide to code that tells the component how to draw itself and determines how the component behaves when the user interacts with it. Examples of existing custom components are the TPanel and TStringGrid components.

It should be mentioned here that both standard and custom components are *windowed* controls. A "windowed control" has a window associated with it and, therefore, has a window handle. Windowed controls have three characteristics: they can receive the input focus, they use system resources, and they can be parents to other controls. (Parents is related to containership, discussed later in this paper.) An example of a component which can be a container is the TPanel component.

### **1.3.4 Graphical components**

Graphical components are visual controls which cannot receive the input focus from the user. They are non-windowed controls. Graphical components allow you to display something to the user without using up any system resources; they have less "overhead" than standard or custom components. Graphical components don't require a window handle-thus, they cannot get focus. Some examples of graphical components are the TLabel and TShape components.

Graphical components cannot be containers of other components. This means that they cannot own other components which are placed on top of them.

### **1.3.5 Non-visual components**

Non-visual components are components that do not appear on the form as controls at run-time. These components allow you to encapsulate some functionality of an entity within an object. You can manipulate how the component will behave, at design-time, through the Object Inspector. Using the Object Inspector, you can modify

### **1.3.3 Custom components**

Unlike standard components, custom components are controls that don't already have a method for displaying themselves, nor do they have a defined behavior. The Component Writer must provide to code that tells the component how to draw itself and determines how the component behaves when the user interacts with it. Examples of existing custom components are the TPanel and TStringGrid components.

It should be mentioned here that both standard and custom components are *windowed* controls. A "windowed control" has a window associated with it and, therefore, has a window handle. Windowed controls have three characteristics: they can receive the input focus, they use system resources, and they can be parents to other controls. (Parents is related to containership, discussed later in this paper.) An example of a component which can be a container is the TPanel component.

### **1.3.4 Graphical components**

Graphical components are visual controls which cannot receive the input focus from the user. They are non-windowed controls. Graphical components allow you to display something to the user without using up any system resources; they have less "overhead" than standard or custom components. Graphical components don't require a window handle-thus, they cannot get focus. Some examples of graphical components are the TLabel and TShape components.

Graphical components cannot be containers of other components. This means that they cannot own other components which are placed on top of them.

### **1.3.5 Non-visual components**

Non-visual components are components that do not appear on the form as controls at run-time. These components allow you to encapsulate some functionality of an entity within an object. You can manipulate how the component will behave, at design-time, through the Object Inspector. Using the Object Inspector, you can modify



a non-visual component's properties and provide event handlers for its events. Examples of such components are the TOpenDialog, TTable, and TTimer components

### **1.3.6 Structure of a component**

All components share a similar structure. Each component consists of common elements that allow developers to manipulate its appearance and function via properties, methods and events. The following sections in this paper will discuss these common elements as well as talk about a few other characteristics of components which don't apply to all components

### **1.3.7 Component properties**

Properties provide an extension of an object's fields. Unlike fields, properties do not store data: they provide other capabilities. For example, properties may use methods to read or write data to an object field to which the user has no access. This adds a certain level of protection as to how a given field is assigned data. Properties also cause "side effects" to occur when the user makes a particular assignment to the property. Thus what appears as a simple field assignment to the component user could trigger a complex operation to occur behind the scenes.

### **1.3.8 Properties provide access to internal storage fields**

There are two ways that properties provide access to internal storage fields of components: directly or through access methods. Examine the code below which illustrates this process.

```
TCustomEdit = class(TWinControl)
private
    FMaxLength: Integer;
protected
    procedure SetMaxLength(Value: Integer);
```

```

...
published
  property MaxLength: Integer read
    FMaxLength write SetMaxLength default 0;
...
end;

```

The code above is snippet of the TCustomEdit component class. TCustomEdit is the base class for edit boxes and memo components such as TEdit, and TMemo.

TCustomEdit has an internal field FMaxLength of type Integer which specifies the maximum length of characters which the user can enter into the control. The user doesn't directly access the FMaxLength field to specify this value. Instead, a value is added to this field by making an assignment to the MaxLength property.

The property MaxLength provides the access to the storage field FMaxLength. The property definition is comprised of the property name, the property type, a read declaration, a write declaration and optional default value.

The read declaration specifies how the property is used to read the value of an internal storage field. For instance, the MaxLength property has direct read access to FMaxLength. The write declaration for MaxLength shows that assignments made to the MaxLength property result in a call to an *access method* which is responsible for assigning a value to the FMaxLength storage field. This access method is SetMaxLength.

## 1.4Property-access methods

Access methods take a single parameter of the same type as the property. One of the primary reasons for write access methods is to cause some side-effect to occur as a result of an assignment to a property. Write access methods also provide a method layer over assignments made to a component's fields. Instead of the component user making the assignment to the field directly, the property's write access method will assign the value to the storage field if the property refers to a particular storage field. For example, examine the implementation of the SetMaxLength method below.

```

procedure TCustomEdit.SetMaxLength(Value: Integer);
begin

```



```

if FMaxLength <> Value then
begin
    FMaxLength := Value;
    if HandleAllocated then
        SendMessage(Handle, EM_LIMITTEXT, Value, 0);
end;
end;

```

The code in the `SetMaxLength` method checks if the user is assigning the same value as that which the property already holds. This is done as a simple optimization. The method then assigns the new value to the internal storage field, `FMaxLength`. Additionally, the method then sends an `EM_LIMITTEXT` Windows message to the window which the `TCustomEdit` encapsulates. The `EM_LIMITTEXT` message places a limit on the amount of text that a user can enter into an edit control. This last step is what is referred to as a *side-effect* when assigning property values. Side effects are any additional actions that occur when assigning a value to a property and can be quite sophisticated.

Providing access to internal storage fields through property access methods offers the advantage that the Component Writer can modify the implementation of a class without modifying the interface. It is also possible to have access methods for the read access of a property. The read access method might, for example, return a type which is different than that of a property's storage field. For instance, it could return the string representation of an integer storage field.

Another fundamental reason for properties is that properties are accessible for modification at run-time through Delphi's Object Inspector. This occurs whenever the declaration of the property appears in the published section of a component's declaration.

## 1.5 Types of properties

Properties can be of the standard data types defined by the Object Pascal rules. Property types also determine how they are edited in Delphi's Object Inspector. The

table below shows the different property types as they are defined in Delphi's online help.

## 1.6 Methods

Since components are really just objects, they can have methods. We will discuss some of the more commonly used methods later in this paper when we discuss the different levels of the VCL hierarchy.

## 1.7 Events

Events provide a means for a component to notify the user of some pre-defined occurrence within the component. Such an occurrence might be a button click or the pressing of a key on a keyboard.

Components contain special properties called events to which the component user assigns code. This code will be executed whenever a certain event occurs. For instance, if you look at the events page of a TEdit component, you'll see such events as OnChange, OnClick and OnDbClick. These events are nothing more than pointers to methods.

When the user of a component assigns code to one of those events, the user's code is referred to as an event handler. For example, by double clicking on the events page for a particular event causes Delphi to generate a method and places you in the Code Editor where you can add your code for that method. An example of this is shown in the code below, which is an OnClick event for a TButton component.

```
TButton component.  
TForm1 = class(TForm)  
  Button1: Tbutton;  
  procedure Button1Click(Sender: TObject);  
end;  
...  
procedure TForm1.Button1Click(Sender: TObject);  
begin
```

```
{ Event code goes here }  
end;
```

It becomes clearer that events are method pointers when you assign an event handler to an event programmatically. The above example was Delphi generated code. To link your own an event handler to a TButton's OnClick event at run time you must first create a method that you will assign to this event. Since this is a method, it must belong to an existing object. This object can be the form which owns the TButton component although it doesn't have to be. In fact, the event handlers which Delphi creates belong to the form on which the component resides. The code below illustrates how you would create an event handler method.

```
TForm1 = class(TForm)  
  Button1: TButton;  
  ...  
private  
  MyOnClickEvent(Sender: TObject); //  
  Your method declaration  
end;  
...  
{ Your method definition below }  
procedure TForm1.MyOnClickEvent(Sender: TObject);  
begin  
  { Your code goes here }  
end;
```

The MyOnClickEvent method becomes the event handler for Button1.OnClick when it is assigned to Button1.OnClick in code as shown below.

```
Button1.OnClick := MyOnClickEvent
```

This assignment can be made anytime at runtime, such as in the form's OnCreate event handler. This is essentially the same thing that happens when you create an event handler through Delphi's Object Inspector except that Delphi generates the method declaration.

When you define methods for event handlers, these methods must be defined as the same type as the event property and the field to which the event property refers. For instance, the OnClick event refers to an internal data field, FOnClick. Both the property OnClick, and field FOnClick are of the type TNotifyEvent. TNotifyEvent is a procedural type as shown below:

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

Therefore, if you are creating a method for an OnClick event, it must be defined with the same type and number of parameters as shown below.

```
TForm1 = class(TForm)
...
    procedure (Sender: TObject);
end;
```

Note the use of the of object specification. This tells the compiler that the procedure definition is actually a method and performs some additional logic like ensuring that an implicit Self parameter is also passed to this method when called. Self is just a pointer reference to the class to which a method belongs.

## **1.8 Containership**

Some components in the VCL can own other components as well as be parents to other components. These two concepts have a different meaning as will be discussed in the section to follow.



## 1.9 Ownership

All components may be owned by other components but not all components can own other components. A component's Owner property contains a reference to the component which owns it.

The basic responsibility of the owner is one of resource management. The owner is responsible for freeing those components which it owns whenever it is destroyed. Typically, the form owns all components which appear on it, even if those components are placed on another component such as a TPanel. At design-time, the form automatically becomes the owner for components which you place on it. At run-time, when you create a component, you pass the owner as a parameter to the component's constructor. For instance, the below shows how to create a TButton component at run-time and passes the form's implicit Self variable to the TButton's Create constructor. TButton.Create will then assign whatever is passed to it, in this case Self or rather the form, and assign it to the button's Owner property.

```
MyButton := TButton.Create(self);
```

When the form that now owns this TButton component gets freed, MyButton will also be freed.

You can create a component without an owner by passing nil to the component's Create constructor, however, you must ensure that the component is freed when it is no longer needed. The code below shows you how to do this for a TTable component.

```
MyTable := TTable.Create(nil)
try
  { Do stuff with MyTable }
finally
  MyTable.Free;
end;
```

As shown in the code above, it is best to use a try..finally block to ensure that the component gets freed even if an exception were to be raised.



The Components property of a component is an array property which contains a list of the components which it owns. For instance, the code below shows how to loop through a form's components and then shows their class name.

```
var
  I: integer;
begin
  for I := 0 to ComponentCount - 1 do
    ShowMessage(Components[I].ClassName);
  end;
```

## 1.10 Parenthood

Parenthood is a much different concept from ownership. It applies only to windowed components, which can be parents to other components. Later, when we discuss the VCL hierarchy, you will see the level in the hierarchy which introduces windowed controls.

Parent components are responsible for the display of other components. They call the appropriate methods internally that cause the children components to draw themselves. The Parent property of a component refers to the component which is its parent. Also, a component's parent does not have to be its owner. Although the parent component is mainly responsible for the display of components, it also frees children components when it is destroyed.

Windowed components are controls which are visible user interface elements such as edit controls, list boxes and memo controls. In order for a windowed component to be displayed, it must be assigned a parent on which to display itself.

## **2.BRIEF ABOUT DATABASE**

Every thing around us has a particular identity. To identify anything system, actor or person in words we need a data or information. So this information is valuable and in this advanced era we can store it in database and access this data by the blink of eye.

For an instant if we go through the definitions of database we may find following definitions.

A database is a collection of related information.

A database is an organized body of related information.

### **2.1 DEMERITS OF ABSENCE OF DATABASE**

A glance on the past will may help us to reveal the drawbacks in case of absence of database.

In the past when there wasn't proper system of database, Much paper work was need to do and to handle great deal of written paper documentation was giant among the problems itself.

In the huge networks to deal with equally bulky data, more workers are needed which affidavit cost much labor expanses.

The old criteria for saving data and making identification was much time consuming such as if we want to search the particular data of a person.

Before the Development of Computer database it was a great problem to search for some thing. Efforts to avoid the headache of search often results in new establishments of data.

Before the development of database it seemed very unsafe to keep the worthy information. In Some situation some big organization had to employe the special persons in order to secure the data.

Before the implementation of database any firm had to face the plenty of difficulties in order to maintain their Management. To hold the check on the expenses of the firm, the manager faced difficulties.

## **2.2 MERITS OF DATABASE**

The modern era is known as the golden age computer sciences and technology. In a simple phrase we can express that the modern age is built on the foundation of database.

If we carefully watch our daily life we can examine that some how our daily life is being connected with database.

There are several benefits of database developments.

Now with the help of computerized database we can access data in a second.

By the development of the database we can make data more secure.

By the development of database we can reduce the cost.

## **2.3 INTRODUCTION TO DATABASE DESIGN**

The design of a database has to do with the way data is stored and how that data is related. The design process is performed after you determine exactly what information needs to be stored and how it is to be retrieved.

A collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMS ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

Computerized library systems

Automated teller machines

Flight reservation systems

Computerized parts inventory systems



From a technical standpoint, DBMS can differ widely. The terms relational, network, flat, and hierarchical all refer to the way a DBMS organizes information internally. The internal organization can affect how quickly and flexibly you can extract information.

Requests for information from a database are made in the form of a query.

Database design is a complex subject. A properly designed database is a model of a business, Country Database or some other in the real world. Like their physical model counterparts, data models enable you to get answers about the facts that make up the objects being modeled. It's the questions that need answers that determine which facts need to be stored in the data model.

In the relational model, data is organized in tables that have the following characteristics: every record has the same number of facts, every field contains the same type of facts (Data) in each record, and there is only one entry for each fact. No two records are exactly the same.

The more carefully you design, the better the physical database meets users' needs. In the process of designing a complete system, you must consider user needs from a variety of viewpoints.

## **2.4 DATABASE MODELS**

Various techniques are used to model data structures. Certain models are more easily implemented by some types of database management systems than others. For any one logical model various physical implementation may be possible. An example of this is the relational model: in larger systems the physical implementation often has indexes which point to the data; this is similar to some aspects of common implementations of the network model. But in small relational database the data is often stored in a set of files, one per table, in a flat, un-indexed structure. There is some confusion below and elsewhere in this article as to logical data model vs. its physical implementation.

### **2.4.1 Flat Model**

The flat (or table) model consists of a single, two dimensional array of data elements, where all members of a given column are assumed to be similar values, and all

members of a row are assumed to be related to one another. For instance, columns for name and password might be used as a part of a system security database. Each row would have the specific password associated with a specific user. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers. This model is the basis of the spreadsheet.

#### **2.4.2 Network Model**

The network model allows multiple datasets to be used together through the use of pointers (or references). Some columns contain pointers to different tables instead of data. Thus, the tables are related by references, which can be viewed as a network structure. A particular subset of the network model, the hierarchical model, limits the relationships to a tree structure, instead of the more general directed graph structure implied by the full network model.

#### **2.4.3 Relational Model**

The relational data model was introduced in an academic paper by E.F. Cod in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model, defined in terms of predicate logic and set theory.

Although the basic idea of a relational database has been very popular, relatively few people understand the mathematical definition and only a few obscure DBMSs implement it completely and without extension. Oracle, for example, can be used in a purely relational way, but it also allow tables to be defined that allow duplicate rows an extension (or violation) of the relational model. In common English usage, a DBMS is called relational if it supports relational operational operations, regardless of whether it enforces strict adherence to the relational model. The following is an informal, not-technical explanation of how “relational” database management systems commonly work.

A relational database contains multiple tables, each similar to the one in the “flat” database model. However, unlike network databases, the tables are not linked by



pointers. Instead, keys are used to match up rows of data in different tables. A key is just one or more columns in one table that correspond to columns in other tables. Any column can be a key, or multiple columns can be grouped together into a single key. Unlike pointers, it's not necessary to define all the keys in advance; a column can be used as a key even if it wasn't originally intended to be one.

A key that can be used to uniquely identify a row in a table is called a unique key. Typically one of the unique keys is the preferred way to refer to row; this is defined as the table's primary key.

When a key consists of data that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number), it's called a "natural" key. If no nature key is suitable, an arbitrary key can be assigned (such as by given employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that can't break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed).

#### **2.4.3.1 Why we use a Relational Database Design**

Maintaining a simple, so-called flat database consisting of a single table doesn't require much knowledge of database theory. On the other hand, most database worth maintaining are quite a bit more complicated than that. Real life databases often have hundreds of thousands or even millions of records, with data that are very intricately related. This is where using a full-fledged relational database program becomes essential. Consider, for example, the Library of Congress, which has over 16 million books in its collection. For reasons that will become apparent soon, a single table simply will not do for this database.

### **2.5 RELATIONSHIPS BETWEEN TABLES**

When you create tables for an application, you should also consider the relationships between them. These relationships give a relational database much of its power. There

are three types of relationships between tables: one-to-one, one-to-many and many-to-many relationships.

### **2.5.2 One-To-One Relationships**

In a one-to-one relationship, each record in one table corresponds to a single record in a second table. This relationship is not very common, but it can offer several benefits. First, you can put the fields from both tables into a single, combined table. One reason for using two tables is that each field is a property of a separate entity, such as owner operators and their trucks. Each operator can operate just one truck at a time, but the fields for the operator and truck tables refer to different entities.

A one-to-one relationship can also reduce the time needed to open a large table by placing some of the table's columns in a second, separate table. This approach makes particular sense when a table has some fields that are used infrequently. Finally, a one-to-one relationship can support in a table requires security, placing them in a separate table lets your application restrict to certain fields. Your application can link the restricted table back to the main table via a one-to-one relationship so that people with proper permissions can edit, delete, and add new records to these fields.

### **2.5.3 One-To-Many Relationships**

A one-to-many relationship, in which a row from one table corresponds to one or more rows from a second table, is more common. This kind of relationship can form the basis for a Many-To-Many relationship as well.

## **2.6 DATA MODELING**

In information system design, data modeling is the analysis and design of the information in the system, concentrating on the logical entities and the logical dependencies between these entities. Data modeling is an abstraction activity in that the details of the values of individual data observations are ignored in favor of the structure,

relationships, names and formats of the data of interest, although a list of valid values is frequently recorded. It is by the data model that definitions of what the data means is related to the data structures.

While a common term for this activity is "Data Analysis" the activity actually has more in common with the ideas and methods of synthesis (putting things together), than it does in the original meaning of the term analysis (taking things apart). This is because the activity strives to bring the data structures of interest together in a cohesive, inseparable, whole by eliminating unnecessary data redundancies and relating data structures by relationships.

### **2.6.1 Database Normalization**

Database normalization is a series of steps followed to obtain a database design that allows for consistent storage and efficient access of data in a relational database. These steps reduce data redundancy and the risk of data becoming inconsistent.

However, many relational DBMS lack sufficient separation between the logical database design and the physical implementation of the data store, such that queries against a fully normalized database often perform poorly. In this case de-normalizations are sometimes used to improve performance, at the cost of reduced consistency.

### **2.6.2 Primary Key**

In database design, a primary key is a value that can be used to identify a particular row in a table. Attributes are associated with it. Examples are names in a telephone book (to look up telephone numbers), words in a dictionary (to look up definitions) and Dewey Decimal Numbers (to look up books in a library).

In the relational model of data, a primary key is a candidate key chosen as the main method of uniquely identifying a relation. Practical telephone books, dictionaries and libraries can not use names, words or Dewey Decimal System Numbers as candidate keys because they do not uniquely identify telephone numbers, word definitions or books. In some design situations it is impossible to find a natural key that uniquely



identifies a relation. A surrogate key can be used as the primary key. In other situations there may be more than one candidate key for a relation, and no candidate key is obviously preferred. A surrogate key may be used as the primary key to avoid giving one candidate key artificial primacy over the others. In addition to the requirement that the primary key be a candidate key, there are several other factors which may make a particular choice of key better than others for a given relation.

The primary key should generally be short to minimize the amount of data that needs to be stored by other relations that reference it. A compound key is usually not appropriate. (However, this is a design consideration, and some database management systems may be better than others in this regard.)

The primary key should be immutable, meaning its value should not be changed during the course of normal operations of the database. (Recall that a primary key is the means of uniquely identifying a tuple, and that identity by definition, never changes.) This avoids the problem of dangling references or orphan records created by other relations referring to a tuple whose primary key has changed. If the primary key is immutable, this can never happen.

### **2.6.3 Foreign Key**

A foreign key (FK) is a field in a database record under one primary key that points to a key field of another database record in another table where the foreign key of one table refers to the primary key of the other table. This way references can be made to link information together and it is an essential part of database normalization.

For example, a person sending an e-mail needs not to include the entire text of a book in the e-mail. Instead, they can include the ISBN of the book, and interested persons can then use the number to get information about the book, or even the book itself. The ISBN is the primary key of the book, and it is used as a foreign key in the e-mail.

Note that using a foreign key often assumes its existence as a primary key somewhere else. Improper foreign key/primary key relationships are the source of many database problems. •



### 2.6.4 Compound Key

In database design, a compound key (also called a composite key) is a key that consists on 2 or more attributes.

No restriction is applied to the attribute regarding their (initial) ownership within the data model. This means that any one, none or all, of the multiple attributes within the compound key can be foreign keys. Indeed, a foreign key may, itself, be a compound key.

Compound keys almost always originate from attributive or associative entities (tables) within the model, but this is not an absolute value.

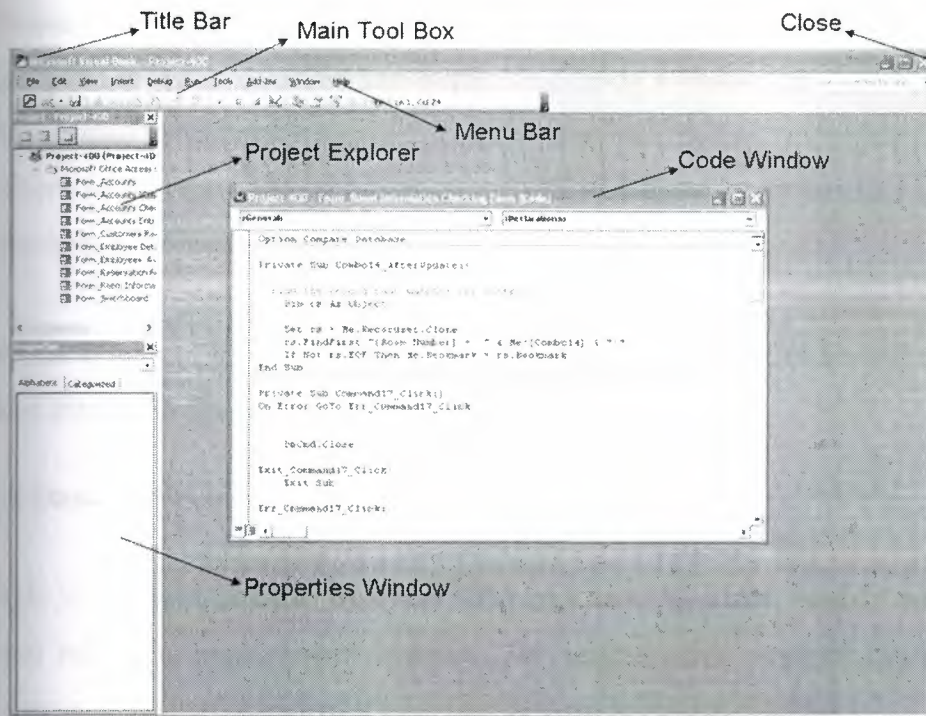
### 2.7 Visual Basic Editor

The visual basic editor can be used in MS Access with different tools e.g. command button, Text box, combo box etc. Now let we see how we can open the Visual Basic editor in MS Access.

- Open MS Access
- Create a new data base
- Enter a form in the data base and open the form in design view.
- Go to the Tools from main menu.
- Select Macro option.
- Select Visual Basic Editor

Important: The Visual Basic Editor can not be opened in the Form view.

After selecting the Visual Basic Editor this window will be displayed.



MS Access Visual Basic Editor

Let us see how to work with these windows.

Main Tool Box: It contains different main options e.g. save, open, run etc.

Project Explorer: This is a very important window if we are using a multi form project it helps us to explore the forms and the links which are related to these forms.

Properties Windows: The properties windows show all the properties of the selected object the most common properties are as fallows.

Caption: This property can be found with every object in the Visual Basic Editor. It is the label presented on the objects when they run.

Name: This is a very important property because it is used in the coding of the program in our program we can access the objects with the name which is given to it in the property windows.

There are several more common properties which can be assigning to the objects by the help of property window.

Code Window: In the code window we write different codes for our objects used in our forms. All the coding is done in the code window.

## **2.8 Structured Query Language**

SQL is the most popular computer language used to create, modify and retrieve data from relational management systems. The language has evolved beyond its original purpose to support object-relational database management systems.

During the 1970s, a group at IBM's San Jose research center developed a database system "System R" based upon Codd's model. Structured English Query Language ("SEQUEL") was designed to manipulate and retrieve data stored in System R. The acronym sequel was later condensed to SQL due to a trademark dispute (the word "sequel" was held as a trade mark by the Hawker-Siddeley aircraft company of the UK).

In 1979, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL and, soon many vendors developed dialects of it.

SQL was adopted as a standard by the ANSI (American National Standard Institute) in 1986 and IOS (International Organization for Standardization) in 1987. ANSI has declared that the official pronunciation for SQL is "es queue el", although many English-speaking database professionals still pronounce it as SEQUEL.

## **2.9 Description of SQL**

SQL allows the specification of queries in a high level, declarative manner. For example, to select rows from a database, the user need only specify the criteria that want



to search by; the details of performing the search operation efficiently is left up to the database system, and is invisible to the user.

Compared to general purpose programming languages, this structure allows the user/programmer to be less familiar with the information contained in the data. This blurs the line between user and programmer, appealing to individuals who fall more into the 'business' or 'research' area and less in the 'information technology' area. The original vision for SQL was to allow non-technical users to write their own database queries. While this has been realized to some extent, the complexity of querying an advanced database system using SQL can still require a significant learning curve.

SQL contrasts with the more powerful database-oriented fourth-generation programming languages such as focus, however, in its relative functional simplicity and simpler command set. This greatly reduces the degree of difficulty involved in maintaining the worst SQL source code, but it also makes programming such questions as 'Who had the top ten scores?' more difficult, leading to the development of procedural extensions, discussed above. However, it also makes it possible for SQL source code to be produced (and optimized) by software, leading to the development of a number of natural language database query languages, as well as 'drag and drop' database programming packages with 'object oriented' interfaces. Often these allow the resultant SQL source code to be examined, for educational purposes, further enhancement, or to be used in a different environment.

## **2.10 SQL Keywords**

SQL keywords fall into several groups, like:

### **2.10.1 Data Retrieval**

The most frequently used operation in transactional databases is the data retrieval operation.

- SELECT is used to retrieve zero or more rows from one or more tables in a database. In most applications, SELECT is the most commonly used DML command. In specifying a select query, the user specifies a description of the desired result set, but they do not specify what physical operations must be



executed to produce that result set. Translating the query into an optimal query plan is to leave the database system, more specifically to the query optimizer.

- Commonly available keywords related to SELECT includes:
  - FROM is used to indicate which tables the data is to be taken from, as well as how the tables join to each other.
  - WHERE is used to identify which rows to be retrieved, or applied to GROUP BY.
  - GROUP BY is used to combine rows with related values into elements of a smaller set of rows.

### **2.10.2 Data Manipulation**

First there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data.

- INSERT is used to add the zero or more rows (formally tuples) to an existing table.
- UPDATE is used to modify the values of a set of existing table rows.
- DELETE removes zero or more existing rows from a table.

### **2.10.3 Data Transaction**

Transaction, if available, can be used to wrap around the DML operations. BEGIN WORK (or START TRANSACTION, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all

COMMIT causes all data changes in a transaction to be made permanent.

ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is “Rolled Back” to the way it was prior to those changes being requested.

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a BEGIN WORK or similar statement, the semantics of SQL are implementation-dependent.

#### **2.10.4 Data Definition**

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system.

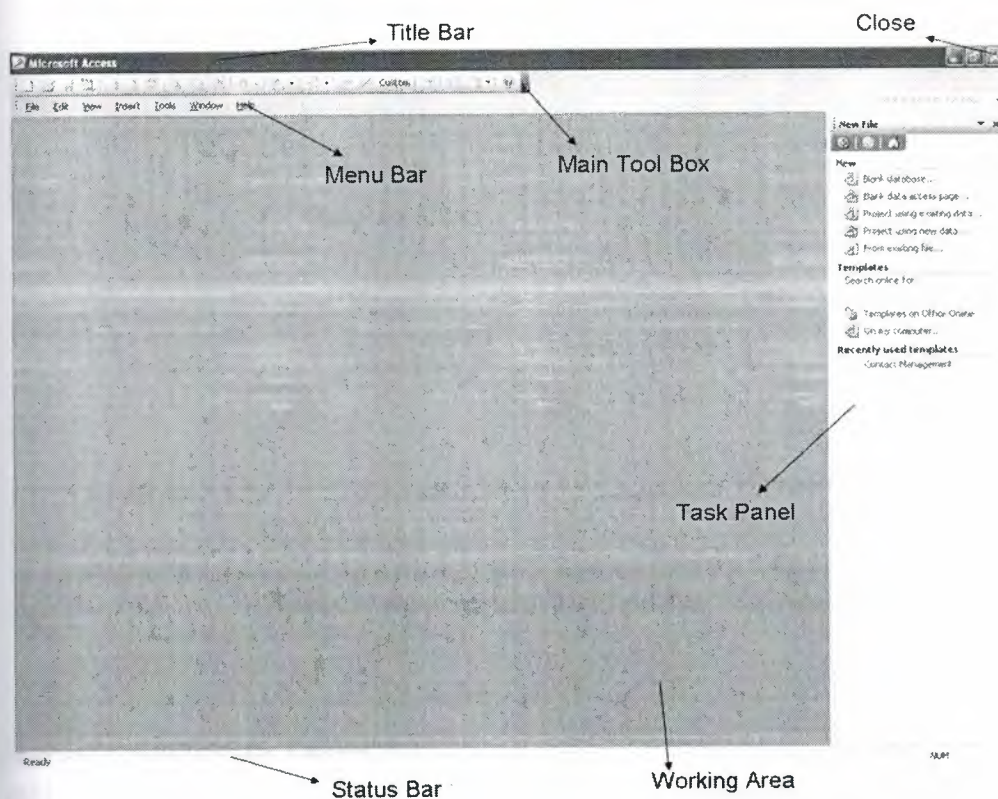
The most basic items of DDL are the CREATE and DROP commands. CREATE causes an object (a table, for example) to be created within the database. DROP causes an existing object within the database to be deleted, usually irretrievably.

### **2.11. MICROSOFT ACCESS DATABASE SYSTEM**

Some terms are used often in access. Some of them are stated below.

#### **2.11.1 A Few Terms**

- (1) An object is a competition in the database such as a table, query, form, or macro.
- (2) A table is a grouping of related data organized in fields (columns) and records (rows) on a datasheet. By using a common field in two tables, the data can be combined. Many tables can be stored in a single database.
- (3) A field is a column in database and defines a data type for a set of values in a table
- (4) All rows except first are records.
- (5) Design View provides the tools for creating fields in a table.
- (6) Datasheet View allows you to update, edit, and delete in formation from a table.

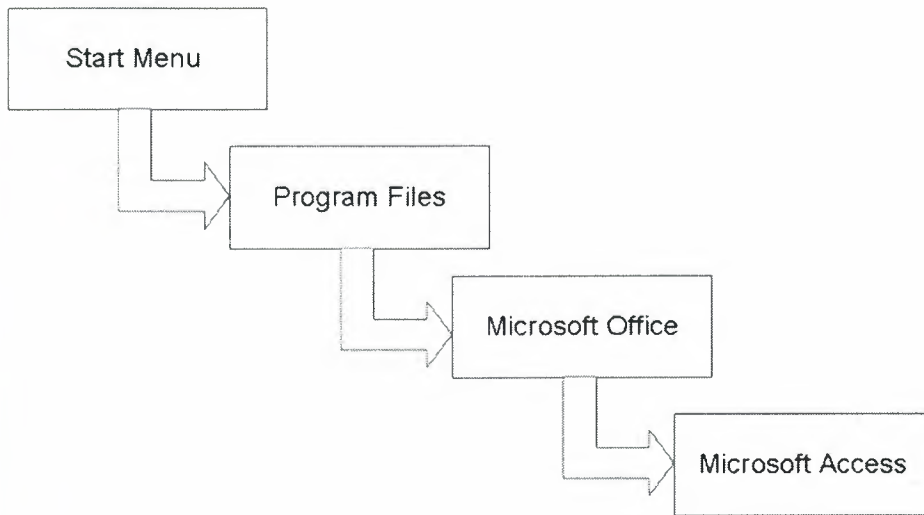


Main window of Microsoft Access

### 2.11.2 Introductory Microsoft Access

- Go to start menu
- Opening Access from Microsoft office
- Creating a Database
- Click File, New or click the new icon on the standard toolbar
- Select Blank Database from the Task Pane menu
- Type a name for the Database in the File Name window
- Click Create and wait for some time
- Closing a database click File, Close
- Opening a database click File, Open or click the open icon on the standard toolbar
- Browse, where to save the database
- Click the name of the database
- Click Open





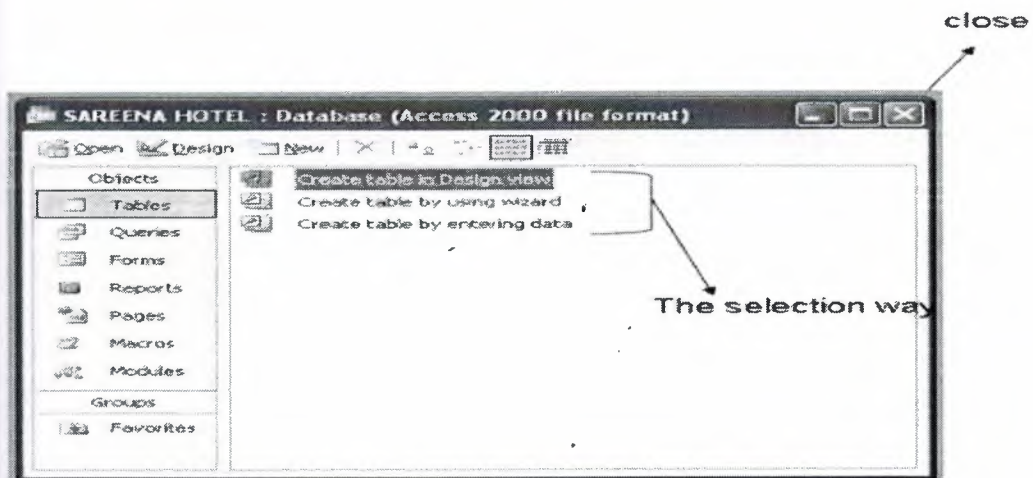
Pictorial representation of opening MS Access

### 2.11.3 Introduction to Tables

A table is a grouping of related data organized in fields (columns) and records (rows) on a datasheet. A database may have one or more tables.

Tables are grids that store information in a database

Access provides three ways to create a table for which there are icons in the Database Window. Double-click on the icons (how u want) to create a table.



Creation of tabels in three ways



1. Create table in Design view will allow you to create the fields of the table. This is the most common way of creating a table and is explained in detail below.
2. Create table using wizard will step you through the creation of a table.
3. Create table by entering data will give you a blank datasheet with unlabelled columns that looks much like an Excel worksheet. Enter data into the cells and click the Save button. You will be prompted to add a primary key field. After the table is saved, the empty cells of the datasheet are trimmed. The fields are given generic names such as "Field1", "Field2", etc. To rename them with more descriptive titles that reflect the content of the fields, select Format Rename Column from the menu bar or highlight the column, right-click on it with the mouse, and select Rename Column from the shortcut menu.

#### 2.11.4 Table's data types

First we introduce the following terms related to table design.

- Field Name - This is the name of the field and should represent the contents of the field such as "Name", "Address", etc. The name can not exceed 64 characters in length and may include spaces.

Data Type is the type of value that will be entered into the fields.

- Text - The default type, text type allows any combination of letters and numbers up to a maximum of 255 characters per field record.
- Memo - A text type that stores up to 64,000 characters.
- Number - includes, number, signs: + and -, decimal point, it can be used in calculations
- Date/Time - A date, time, or combination of both.
- Currency - Monetary values that can be set up to automatically include a dollar sign (\$) and correct decimal and comma positions.

➤ AutoNumber - When a new record is created, Access will automatically assign a unique integer to the record in this field. Since every record in a datasheet must include at least one field that distinguishes it from all others, this is a useful data type to use if the existing data will not produce such values, it can be a primary key.

➤ Yes/No - Use this option for True/False, Yes/No, On/Off, or other values that must be only one of two.

➤ OLE Object - An OLE (Object Linking and Embedding) object is a sound, picture, or other object such as a Word document or Excel spreadsheet that is created in another program. Use this data type to embed an OLE object or link to the object in the database.

➤ Hyperlink - A hyperlink will link to an Internet or Intranet site, or another location in the database (URL).

- Description (optional) - Enter a brief description of what the contents of the field are.

Field Properties - Select any pertinent properties for the field from the bottom pane of Design View window

- Field Size is used to set the number of characters needed in a text or number field. The default field size for the text type is 50 characters. Maximum size is 255.

- Byte - Positive integers between 1 and 255
- Integer - Positive and negative integers between -32,768 and 32,768
- Long Integer (default) - Larger positive and negative integers between -2 billion and 2 billion.
- Single - Single-precision floating-point number
- Double - Double-precision floating-point number
- Decimal - Allows for Precision and Scale property control

- Format changes the way a field is displayed or printed, but does not affect the stored value.

- Number format. Select one of the preset options from the drop down menu or construct a custom format using symbols explained below:

- Date format. In the table below, the value "1/1/01" is entered into the datasheet, and the following values are displayed as a result of the different assigned formats.

- Yes/No. fields are displayed as check boxes by default on the datasheet. To change the formatting of these fields, first click the Lookup tab and change the Display Control to a text box. Go back to the General tab choices to make formatting changes. The formatting is designated in three sections separated by semicolons. The first section does not contain anything but the semicolon must be included. The second section specifies formatting for Yes values and the third for No values.

- Default Value. There may be cases where the value of a field will usually be the same for all records. In this case, a changeable default value can be set to prevent typing the same thing numerous times. Set the Default Value property.

- Primary Key. Every record in a table must have a primary key that differentiates it from every other record in the table. A social security number is an example of a record whose values will only appear once in a database table.

Designate the primary key field by right-clicking on the record and selection Primary Key from the shortcut menu or select Edit Primary Key from the menu bar. The primary



key field will be noted with a key image to the left. To remove a primary key, repeat one of these steps.

If none of the existing fields in the table will produce unique values for every record, a separate field must be added. Access will prompt you to create this type of field at the beginning of the table the first time you save the table and a primary key field has not been assigned. The field is named "ID" and the data type is "auto number". Since this extra field serves no purpose to you as the user, the auto number type automatically updates whenever a record is added so there is no extra work on your part. You may also choose to hide this column in the datasheet as explained on a later page in this tutorial.

- **Indexes.** Creating indexes allows Access to query and sort records faster. To set an indexed field, select a field that is commonly searched and change the Indexed property to Yes (Duplicates OK), if multiple entries of the same data value are allowed or Yes (No Duplicates) to prevent duplicates.
- **Field Validation Rules.** Validation Rules specify requirements (change word) for the data entered in the worksheet. A customized message can be displayed to the user when data that violates the rule setting is entered. Click the expression builder ("...") button at the end of the Validation Rule box to write the validation rule. Examples of field validation rules include  $< 0$  to not allow zero values in the record, and  $> 3$  To only all data strings three characters in length.
- **Input Masks.** An input mask controls the value of a record and sets it in a specific format. They are similar to the Format property, but instead display the format on the datasheet before the data is entered. For example, a telephone number field can formatted with an input mask to accept ten digits that are automatically formatted as "(555) 123-4567". The blank field would look like ( ) - . An input mask to a field by following these steps:

1. In design view, place the cursor in the field that the input mask will be applied to.
2. Click in the white space following Input Mask under the General tab.

## 3. SOFTWARE ABOUT CD APPLICATION PROGRAM

### 3.1. Main Form



Fig.3.1 Main Form

In main form I used statusbar, toolbar and mainmenu. I explain all form step by step. Main menu which is shown in the below figure the important and useful buttons are at the top of the page. What you want to select you can just click it.



## 3.2. Search Product Form

Search Movie

Movie Name:  Movie Type:

Director:  CD Type:

Search a movie for Actor or Actress name, please [CLICK HERE](#)

Movie Name	Director	Cast	Movie Type	Self Code	CD Type
Babama Ve Dügüm	Çağan Irmak	Fikret Kuşkan	Drama	00001	DVD
Barda	Serdar Akar	Nejat İşler	Drama	00009	VCD
Buz Devri	Carloz Saldanha	Ray Romano	Animation	00007	DVD
Duvara Karşı	Fatih Akın	Bürol Ünel	Drama	00006	DIVX
Gladyatör	Ridley Scott	Russell Crowe	Action	00004	VCD
Kill Bill Volume 1	Quentin Tarantino	Uma Thurman	Action	00002	DVD
Kill Bill Volume 2	Quentin Tarantino	Uma Thurman	Action	00003	DVD
Matrix	Andy Wachowski	Keanu Reeves	Science-Specul	00008	DVD
Nemo:Kayıp Balık	Andrew Stanton	Albert Brooks	Animation	00009	DVD

Webpage:  Movie Time:

Scenario Writer:  Self Code:

Vision Director:  CD Number:

Language:  Music:

Production:  Colour:

Copyright ©2007 Company All Rights Reserved 28.05.2007 ADMIN Designed by Hakan KILIÇ

Fig 3.2 Search Product Form

When we opened search form, all movie name, director name, self code, cast and CD type appear in table. If we click the click here then open the actor form and we can search movie according to actor/actress name. We can't fill the boxes under the table because for example we wrote any movie name and when we doubleclick the table then webpage information appear the that boxes. Just we can fill the boxes above the table. Now I explain searching method.

Product Operations Product Database Search Product

Search Movie

Search Movie

Movie Name:  Movie Type:

Director:  CD Type:

Search a movie for Actor or Actress name, please [CLICK HERE](#)

Movie Name	Director	Cast	Movie Type	Sell Code	CD Type
Babam Ve Oğlum	Çağan Irmak	Fikret Kuşkan	Drama	00001	DVD
Barda	Serdar Akar	Nejat İşler	Drama	00009	VCD


Webpage:  Movie Time:

Scenario Writer:  Sell Code:

Vision Director:  CD Number:

Language:  Music:

Production:  Colour:



©2007 Company All Rights Reserved 28.05.2007 ADMIN Designed by Hakan KILIÇ

Fig 3.3 Search Product Form

Now I explain searching method. For example I go to any CD Market and I want to drama movie. Program user choose drama in Movie Type then all drama movies show the table. Also I want to movie type is drama and movie name looks like 'Ba'. Then Program user write 'Ba' in Movie Name boxes. All starts with 'Ba' Movie names show on the screen then I choose the my prefer (Barda), program user doubleclicks on that movie column in table then all information show below the table. Also we can write Director Name, and choose CD Type. Self Code means 'where is the movie in the Market?'.



### 3.3. Poster Form



Fig 3.4 Poster Form

Poster Form,if we click the picture which is poster on search product form,we can see it more bigger.



### 3.4. Product Price Form

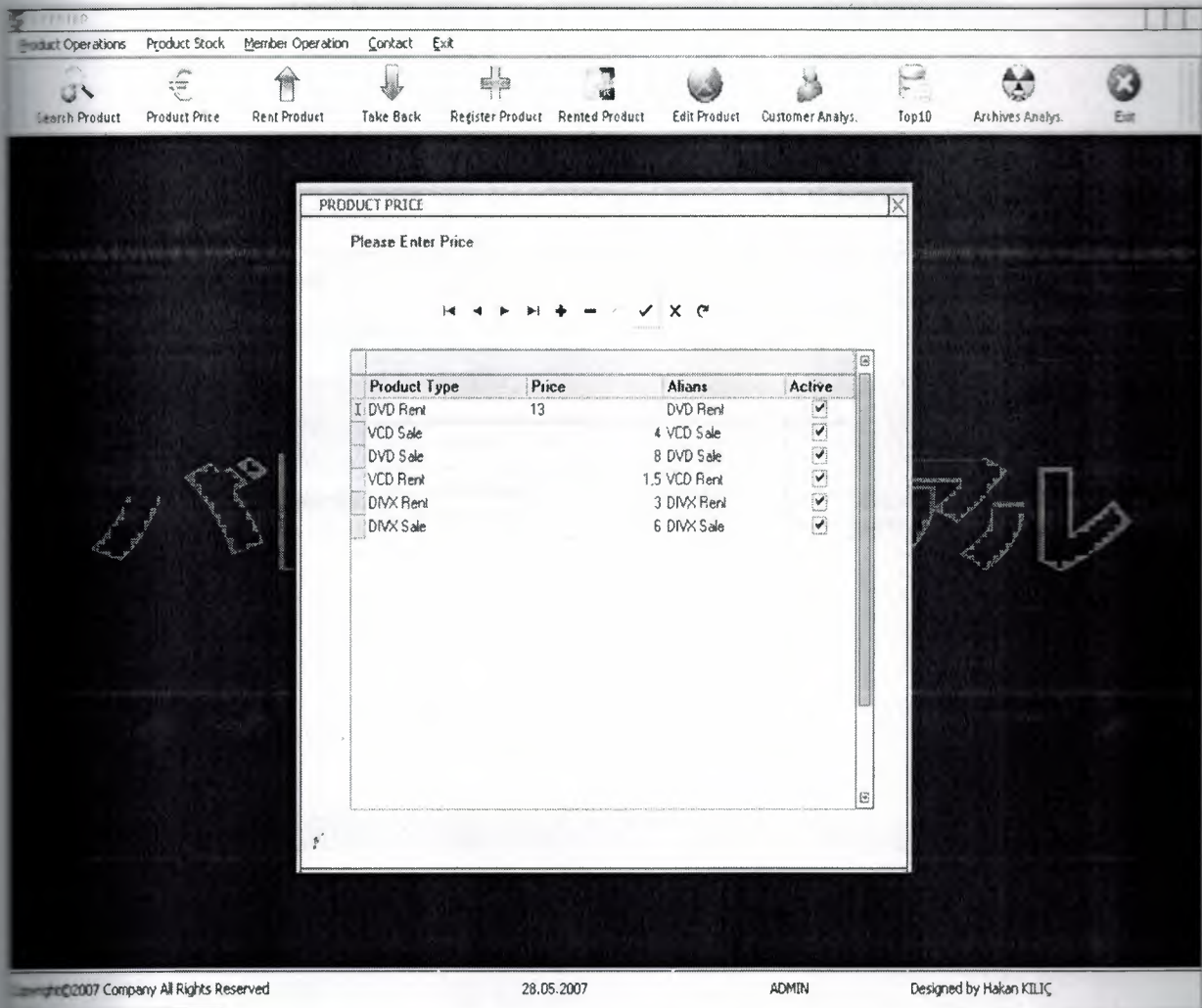


Fig 3.5 Product Price Form

In Product Price Form, we change a product price. Also we insert a new product price or delete recorded product price.

### 3.5. Rent Product Form

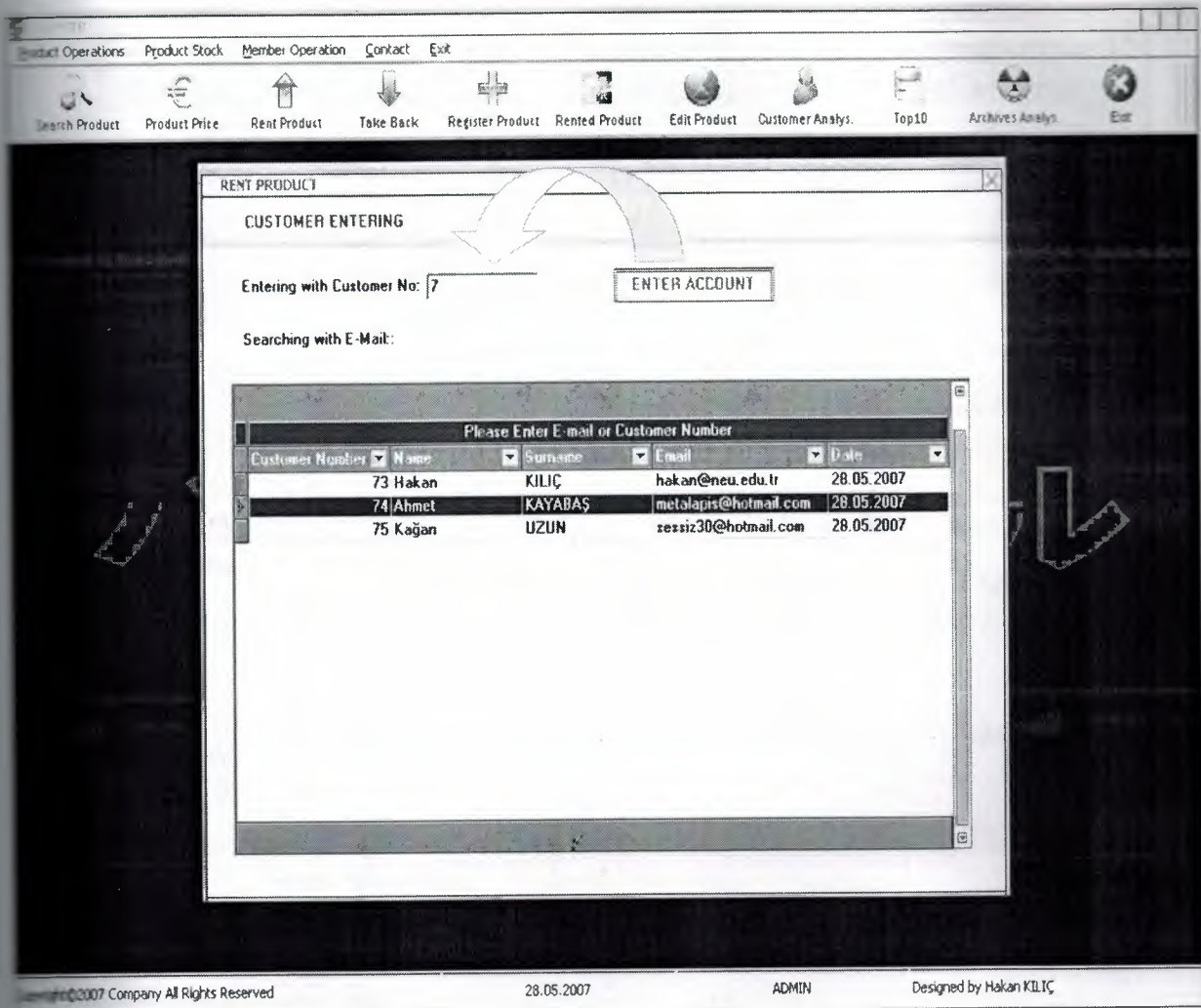


Fig.3.6 Rent Product Form

We use this form for when we enter customer number or customer e-mail, and we click enter account button or doubleclick coloumn table next page opens.

### 3.6. Product Selling Form

**Movies**

Movie Name	Director	Actor/Actress	Movie Type	Scenario	Self Code	CD Type
Matrix	Andy Wachowski	Keanu Reeves	Science-Speculative	Andy Wachowski	00008	DVD
Baban Ve Oglum	Cağan Irmak	Fikret Kuşkan	Drama	Cağan Irmak	00001	DVD
Barda	Serdar Akar	Nejmi İşler	Drama	Serdar Akar	00009	VCD
Duvana Karşı	Fatih Akar	Reza Zarrabzadeh	Drama	Fatih Akar	00006	DVD

Total Record: 10

---

**Product Selling**

Movie Name:

Selling Type:

Selling Time:

Take Back Time:

Price Alternatives:

Movie Price:

Discount(%):

Last Price:

Him/His Payment:

Debt:  Credit:

Name: Ahmet  
GSM: 05338769924  
Surname: KAYABAŞ  
E-Mail: metalapis@hotmail.com

Bought Product Number: 1  
Rented Product Number: 3

Cur No	Movie Name	Rent	Sale	Selling Date	Back Time
74	Duvana Karşı	1	0	30.05.2007	31.05.2007
74	Duvana Karşı	1	0	28.05.2007	02.06.2007
74	Vizontele Tubaa	1	0	28.05.2007	30.05.2007
74	Barda	0	1	28.05.2007	

Fig 3.7 Product Selling Form

in that form, previous form we chose customer who rented products or bought products automatically listed by all them in table. Also customer name, surname, e-mail and gsm shown on that form. Total bought product number and total rented product number shown in boxes. In that form we make renting a new product or selling a new product to that customer. I show button on form with arrow when we click it all products list to that form then we choose a movie there doing doubleclick the column. Then we continue filling the boxes on the form.



Product Operations Product Stock Member Operation Contact Exit

Search Product Product Price Rent Product Take Back Register Product Rented Product Edit Product Customer Analsys. Top10 Archives Analsys Exit

CD SELLING

Product Selling

Movie Name Babam Ve Oğlum

Selling Type Rent

Selling Time 30.05.2007

Take Back Time 31.05.2007

Price Alternativex DVD Rent

Movie Price 7,00 TL

Discount(%) 10%

Last Price 6,30 TL

Him/His Payment 10

Debt 3,70 TL Credit 0,00 TL

CLEAR FINISH

Name Ahmet GSM 05338769924

Surname KAYABAŞ E-Mail metalapis@hotmail.com

Bought Product Number----> 1

Rented Product Number----> 3

Cuz.No	Movie Name	Rent	Sale	Selling Date	Back Time
74	Duvara Karşı		1	0 30.05.2007	31.05.2007
74	Duvara Karşı		1	0 28.05.2007	02.06.2007
74	Vizontele Tubaa		1	0 28.05.2007	30.05.2007
74	Barda		0	1 28.05.2007	

©2007 Company All Rights Reserved 30.05.2007 Designed by Hakan KILIÇ

Fig.3.8 Product Selling Form

After we choose the movie name we start filling other boxes now I explain its. We choose selling type rent or sale if we choose the rent we must fill the take back date. I choose dvd rent then automatically movie price shown in boxes. I explain before in fig 1.4 we can change movie price. Anyway then if we want to doing discount that customer, write percent quantity in discount. boxes. I wrote %10. When we exit the discount boxes, automatically program calculates new price and shown new price in last price boxes. Last one is him/his payment boxes. Customer gives money to us for product payment so we write it that boxes. Also when we exit that boxes, again automatically program calculates debt or credit then writes in their boxes

### 3.7. Register Product Form

The screenshot shows a web browser window with a menu bar containing 'Product Operations', 'Product Stock', 'Member Operation', 'Contact', and 'Exit'. Below the menu is a toolbar with icons for 'Search Product', 'Product Price', 'Rent Product', 'Take Back', 'Register Product', 'Rented Product', 'Edit Product', 'Customer Analyze', 'Top Of', 'Archives Analyze', and 'Exit'. The main content area displays the 'REGISTER MOVIE FORM' dialog box. The form has the following fields and controls:

- Movie Name:
- Director:
- Vision Director:
- Actor/Actress: 1.Actor/Actress, 2.Actor/Actress, 3.Actor/Actress, 4.Actor/Actress, 5.Actor/Actress (each with a text input)
- Production:
- Colour:
- Music:
- Language: Language Choose (dropdown menu)
- CD Type: CD Type (dropdown menu)
- CD Number: 0 (text input)
- Web Pages:
- Scenario:
- Self Code:
- Time: 0 (Minute) (text input)
- Buttons: Add Poster, Register, Exit, Clear

At the bottom of the browser window, the footer contains: Copyright©2007 Company All Rights Reserved, 28.05.2007, ADMIN, and Designed by Hakan KILIÇ.

Fig 3.9 Register Product Form

We use that form for record a new product. We write movie name, director, actor/actress name... add poster and click register button. If we click the clear button all boxes and picture are cleared.



### 3.8. Rented Product Form

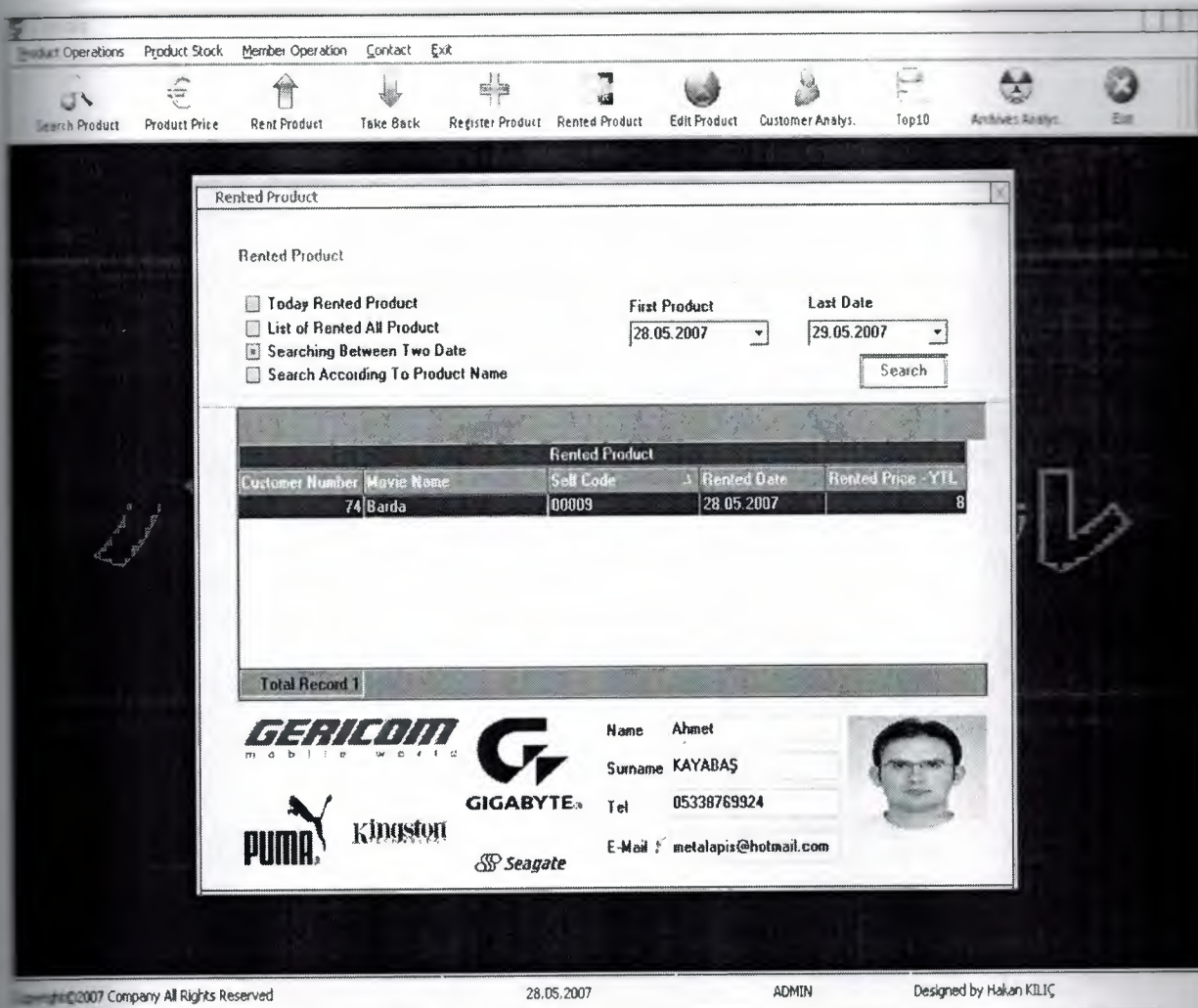


Fig 3.10 Rented Product Form

In that form, we can see rented product. We have four checkboxes and we can choose one of them to list all rented products in the table. We can list rented products in the table by 'today rented product', 'list of rented all product', 'searching between two dates', or 'search according to product name'. I show rented products with 'searching between two dates' in that form. I wrote the first date and last date, then the shown products are in the table. It is very important for me, who did rent that product? When you click a column in the table, you may see customer data below the table. Also, we can see his/her picture.



### 3.9. Edit Product Form

EDIT PRODUCT

Edit Product

Movie Name:  Vision Director:

Director: Çağan Irmak Production: Şükrü Avcı

Actor/Actress: Fikret Kuşkan  
Çetin Tekindor  
Ege Tanman

Colour: Coloured

Music

Language:

CD Type:

CD Number:

Movie Type:

Web Page: [www.babamveolum.com](http://www.babamveolum.com)

Scenario: Çağan Irmak

Self Code: 00001

Time:  [Min.]

©2007 Company All Rights Reserved 28.05.2007 ADMIN Designed by Hakan KILIÇ

Fig 3.11 Edit Product Form

First we have to write movie name for editing in that form. Most important thing is when writing movie name automatically coming the other data to boxes. Just I write a letter what is 'B'. It is not necessary to click anywhere just enough to writing movie name. Also we can click combobox and choose a movie name there. We change movie name, director name or other information of movie and we update poster to click change poster button in that form. Now I continue to writing movie name. We will see changing on form again I explain we don't click any button for coming data to boxes just we write movie name and all information of movie automatically coming boxes.

Product Operations Product Stock Member Operation Contact Exit

Search Product Product Price Archives Analyz Edit

**EDIT PRODUCT**

Edit Product

Movie Name:  Vision Director:

Director: Serdar Akar Production: Turkey 2006

Actor/Actress: Nejat İşler  
Serdar Orçin Colour: Coloured

Music


Movie Type:  Language:

Web Page: www.bardafilm.com CD Type:

Scenario: Serdar Akar CD Number:

Self Code: 00009

Time:  (Min.)



©2007 Company All Rights Reserved 28.05.2007 ADMIN Designed by Hakan KILIÇ

Fig 3.12 Edit Product Form

So I write 'Bar' in movie name boxes and change poster and other information automatically. We can delete the record that form when click the delete record button. If the movie name is empty, error message comes screen. We have to fill in the movie name boxes on the form we cant pass empty. It is very important.



### 3.10. Customer Analysis Form

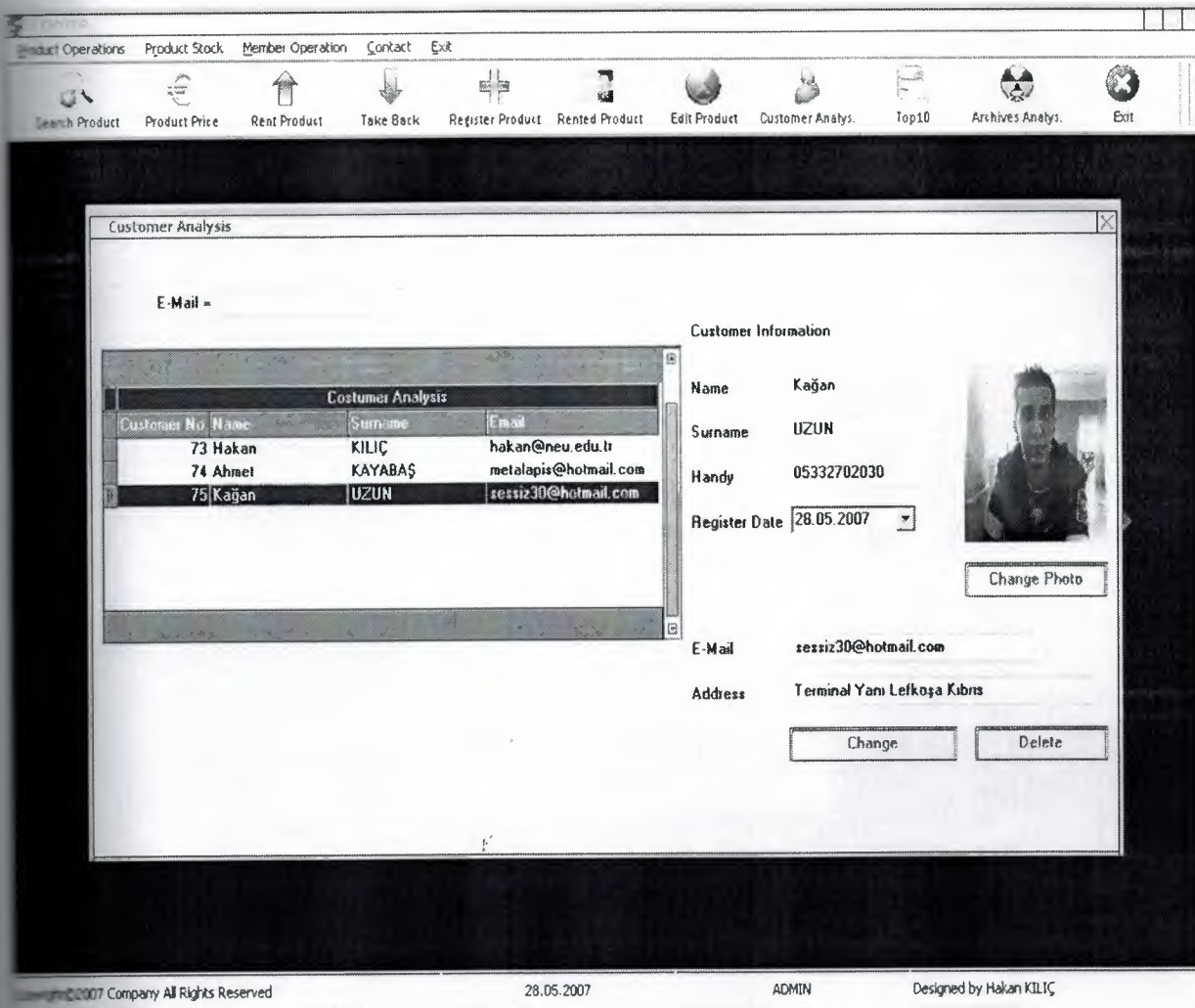


Fig 3.13 Customer Analysis Form

We have a customers and their datas. In that form we consider customers informations. We can find customer to entering your e-mail address boxes where is top of form. After we enter e-mail address shown in table him/her data. Doubleclick column and comes data the other boxes automatically. We can change customer name, customer surname, customer telephone, e-mail and address also customer photo. We can see customer register date but cant change it. But if we delete customer necessary the click the delete button. Program ask us 'do you want to delete customer?' before delete the customer. Generally user write customer e-mail address a lot of times in that program so customer e-mail address is very important for that program.



### 3.11. Top Ten Form

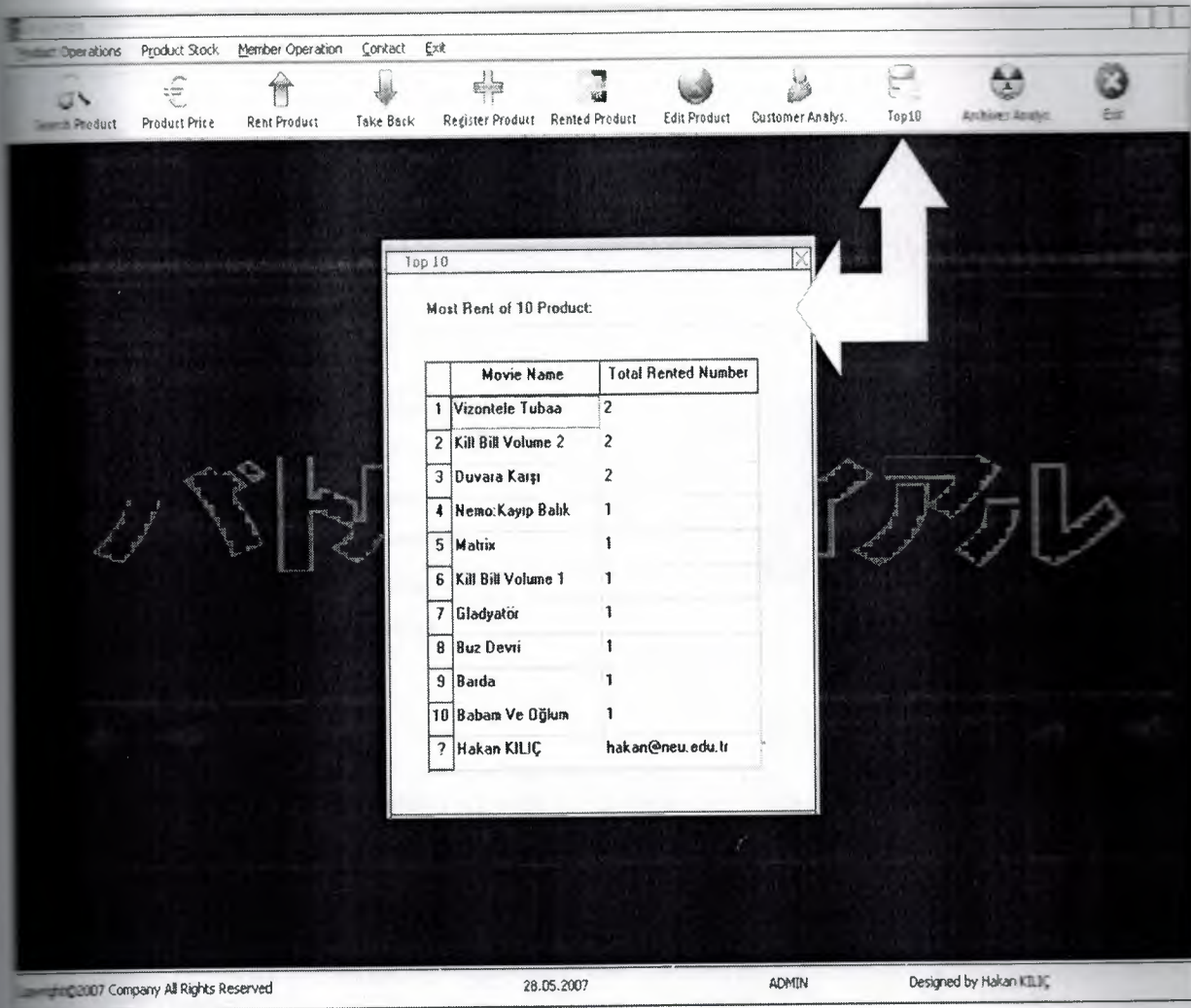


Fig 3.14 Top10 Form

In that form, most rented product of top ten. In that form reset of data every week automatically. User can see most popular film of means by that form. Also user reach program designer's e-mail on top10 form.

### 3.12. Register Customer Form



Fig 3.15 Register Customer Form

We use this form for register customer. We don't fill customer number program gives us automatically that number. Just we write customer name, customer surname, customer e-mail, customer telephone number and address. Then we choose register date, and click add photo button then choose customer picture finally we click register button. If customer name, surname or customer e-mail boxes is empty, program gives error message so it is necessary to fill that boxes.

### 3.13. Exit Form

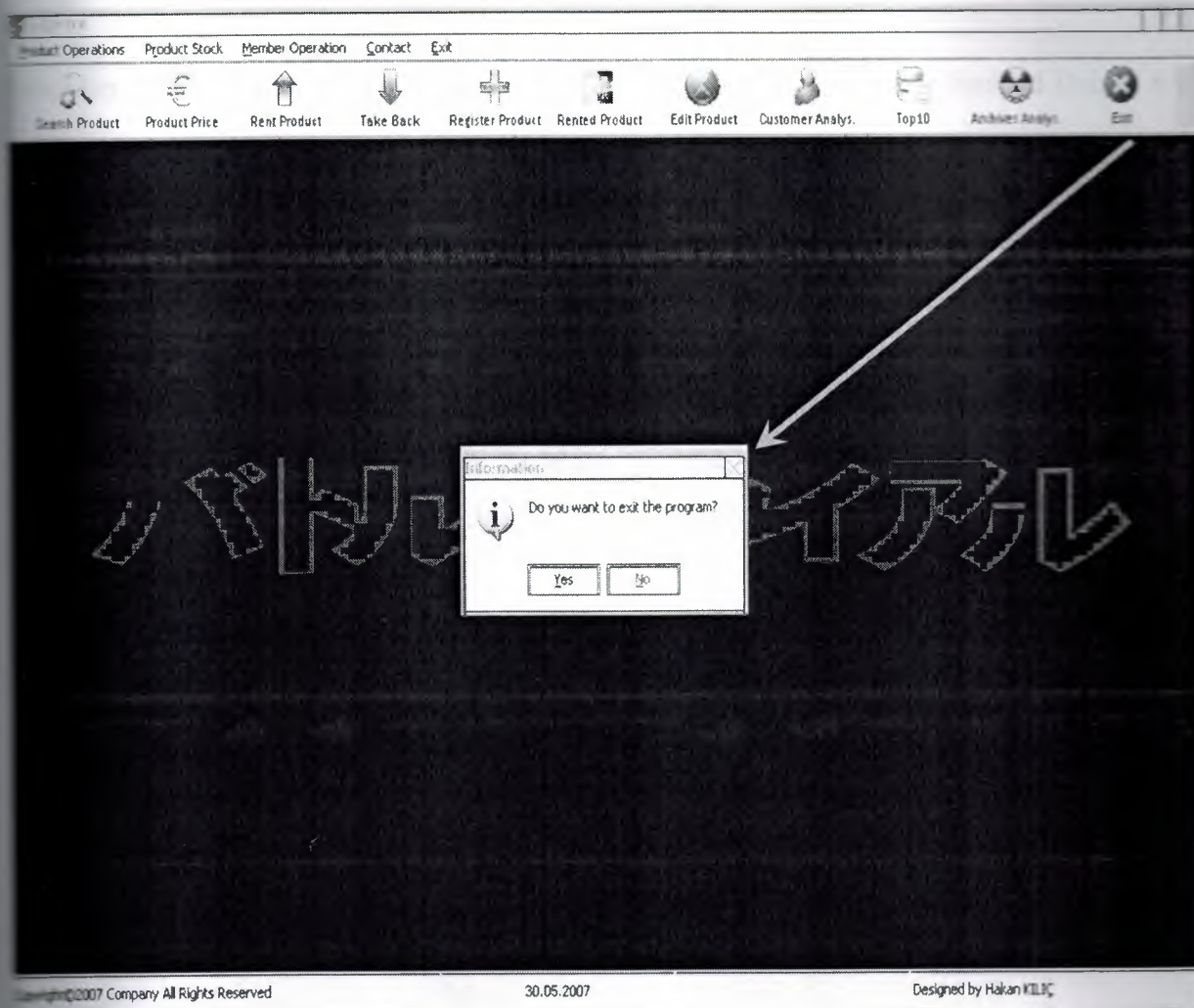


Fig 3.16 Exit Form

in that form when we click the exit button program ask us question if we enter yes button exit program else continue running program.



## CONCLUSION

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

In my Project, I have used important components of Delphi 6.0. So I learned these components very well. Now I can use these components of Borland Delphi 6.0 in an efficient manner. 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 form e that I have learned how to prepare an individual software project by using Borland Delphi 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 m e for the real life and I used the things in my Project that I learned from courses as theoretically.



## 4. APPENDIX

### 4.1. FORM OF MAIN FORM CODES

unit cdcenter;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ActnList, Menus, dxCntner, dxTL, dxCalc, ExtCtrls,  
ExtDlgs, DB, ADODB, dxEditor, dxExEdtr, dxEdLib, dxDBELib, ComCtrls, jpeg,  
WinSkinData, WinSkinStore, MPlayer, ToolWin, ImgList;

type

Tana = class(TForm)

i: TImage;

ToolBar1: TToolBar;

ToolButton1: TToolButton;

ToolButton2: TToolButton;

ToolButton3: TToolButton;

ToolButton4: TToolButton;

ToolButton5: TToolButton;

ToolButton7: TToolButton;

ToolButton8: TToolButton;

ToolButton9: TToolButton;

ToolButton10: TToolButton;

ToolButton11: TToolButton;

ToolButton12: TToolButton;

ToolButton15: TToolButton;

StatusBar1: TStatusBar;

MainMenu1: TMainMenu;

KAYIT1: TMenuItem;

FLMKAYIT1: TMenuItem;  
FilmArama1: TMenuItem;  
IKI1: TMenuItem;  
mAl1: TMenuItem;  
mKiralal: TMenuItem;  
mAnaliz1: TMenuItem;  
Kiradagirler1: TMenuItem;  
mFiyatlar1: TMenuItem;  
Stok1: TMenuItem;  
op101: TMenuItem;  
ArivAnaliz1: TMenuItem;  
KULLANICILEMLER1: TMenuItem;  
ifreDeitirme1: TMenuItem;  
yelikBilgileri1: TMenuItem;  
yeol1: TMenuItem;  
MteriAnaliz1: TMenuItem;  
IKI2: TMenuItem;  
ImageList1: TImageList;  
ToolButton6: TToolButton;  
ToolButton13: TToolButton;  
ToolButton14: TToolButton;  
ToolButton17: TToolButton;  
ToolButton16: TToolButton;  
Exit1: TMenuItem;  
Contact1: TMenuItem;  
EMail1: TMenuItem;  
procedure ifreDeitirme1Click(Sender: TObject);  
procedure FormShow(Sender: TObject);  
procedure yelikBilgileri1Click(Sender: TObject);  
procedure FilmArama1Click(Sender: TObject);  
procedure IKI1Click(Sender: TObject);  
procedure Giriik1Click(Sender: TObject);  
procedure FLMKAYIT1Click(Sender: TObject);  
procedure ToolButton1Click(Sender: TObject);



```
procedure yeol1Click(Sender: TObject);
procedure ToolButton2Click(Sender: TObject);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton12Click(Sender: TObject);
procedure mKiralal1Click(Sender: TObject);
procedure ToolButton5Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ToolButton10Click(Sender: TObject);
procedure ToolButton7Click(Sender: TObject);
procedure op101Click(Sender: TObject);
procedure ToolButton9Click(Sender: TObject);
procedure ToolButton8Click(Sender: TObject);
procedure ToolButton11Click(Sender: TObject);
procedure Kiradakimler1Click(Sender: TObject);
procedure mFiyatlar1Click(Sender: TObject);
procedure mAnaliz1Click(Sender: TObject);
procedure ArivAnaliz1Click(Sender: TObject);
procedure MteriAnaliz1Click(Sender: TObject);
procedure YardmveDestek1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var  
Tana;
```

```
implementation
```

```
SR *.dfm}
```

```
uses help,urunanaliz,arsivanaliz,musterianalizi,
```

kiradakiurun,duzelt,arama,filmlergenel,topon,frmkayit,  
Kullanici girisi, rm, User girisi, uyelik bilgiler, hesap, cdalim, stok, urun fiyat;

```
procedure Tana.ifreDeitirme1Click(Sender: TObject);
```

```
begin
```

```
Application.CreateForm(TForm3,Form3);
```

```
Form3.Showmodal;
```

```
Form3.Destroy;
```

```
end;
```

```
procedure Tana.FormShow(Sender: TObject);
```

```
begin
```

```
dm.SorguR('select * from hakan where kullanıcı= '"+StatusBar1.Panels[2].Text+"' ');
```

```
user.Hide;
```

```
statusbar1.Panels[1].Text:=datetostr(date);
```

```
end;
```

```
procedure Tana.yelikBilgileri1Click(Sender: TObject);
```

```
begin
```

```
application.CreateForm(TForm5,Form5);
```

```
form5.ShowModal;
```

```
form5.Destroy;
```

```
end;
```

```
procedure Tana.FilmArama1Click(Sender: TObject);
```

```
begin
```

```
application.CreateForm(Tform6,form6);
```

```
form6.ShowModal;
```

```
form6.Destroy;
```

```
end;
```

```
procedure Tana.IKI1Click(Sender: TObject);
```

```
begin
```

```
application.CreateForm(Tform8,form8);
```

```

form8.ShowModal;
form8.Destroy;
end;

procedure Tana.Giriik1Click(Sender: TObject);
begin
Application.CreateForm(Tform13, Form13);
Form13.ShowModal;
form13.Destroy;
end;

procedure Tana.FLMKAYIT1Click(Sender: TObject);
begin
Application.CreateForm(TForm2,Form2);
Form2.ShowModal;
Form2.Destroy;
end;

procedure Tana.ToolButton1Click(Sender: TObject);
begin
application.CreateForm(Tform6,form6);
form6.ShowModal;
form6.Destroy;
end;

procedure Tana.yeol1Click(Sender: TObject);
begin
application.CreateForm(Tform11,form11);
Form11.showmodal;
Form11.destroy;
end;

procedure Tana.ToolButton2Click(Sender: TObject);
begin

```



```

Application.CreateForm(TForm14, Form14);
Form14.showmodal;
Form14.destroy;
end;

procedure Tana.ToolButton3Click(Sender: TObject);
begin
application.CreateForm(Tform12,form12);
form12.ShowModal;
form12.Destroy
end;

procedure Tana.ToolButton12Click(Sender: TObject);
begin
if messagedlg('Do you want to exit the program?',mtinformation,[mbytes,mbno],0)=mrno then
exit;
Application.Terminate;
end;

procedure Tana.mKiralalClick(Sender: TObject);
begin
application.CreateForm(Tform12,form12);
form12.ShowModal;
form12.Destroy
end;

procedure Tana.ToolButton5Click(Sender: TObject);
begin
Application.CreateForm(TForm2,Form2);
Form2.ShowModal;
Form2.Destroy;
end;

procedure Tana.FormClose(Sender: TObject; var Action: TCloseAction);

```

```
begin
application.Terminate;
end;

procedure Tana.ToolButton10Click(Sender: TObject);
begin
application.CreateForm(Tform16,form16);
form16.showmodal;
form16.destroy;
end;

procedure Tana.ToolButton7Click(Sender: TObject);
begin
application.CreateForm(Tform17,form17);
form17.showmodal;
form17.destroy;
end;

procedure Tana.op101Click(Sender: TObject);
begin
application.CreateForm(Tform16,form16);
form16.ShowModal;
form16.Destroy;
end;

procedure Tana.ToolButton9Click(Sender: TObject);
begin
application.CreateForm(Tform18,form18);
form18.ShowModal;
form18.Destroy;
end;

procedure Tana.ToolButton8Click(Sender: TObject);
begin
```

```
application.CreateForm(Tform8,form8);
form8.ShowModal;
form8.Destroy;
end;

procedure Tana.ToolButton11Click(Sender: TObject);
begin
Application.CreateForm(Tform13, Form13);
Form13.ShowModal;
form13.Destroy;
end;

procedure Tana.Kiradakerimler1Click(Sender: TObject);
begin
application.CreateForm(Tform17,form17);
form17.showmodal;
form17.destroy;
end;

procedure Tana.mFiyatlar1Click(Sender: TObject);
begin
Application.CreateForm(TForm14, Form14);
Form14.showmodal;
Form14.destroy;
end;

procedure Tana.mAnaliz1Click(Sender: TObject);
begin
application.CreateForm(Tform19,form19);
form19.showmodal;
form19.destroy;
end;

procedure Tana.ArivAnaliz1Click(Sender: TObject);
```



```

begin
Application.CreateForm(Tform13, Form13);
Form13.ShowModal;
form13.Destroy;
end;

procedure Tana.MteriAnaliz1Click(Sender: TObject);
begin
application.CreateForm(Tform18,form18);
form18.ShowModal;
form18.Destroy;
end;

procedure Tana.YardmveDestek1Click(Sender: TObject);
begin
application.CreateForm(tfrmhelp,frmhelp);
frmhelp.Showmodal;
frmhelp.deStroy;
end;

procedure Tana.Exit1Click(Sender: TObject);
begin
if messagedlg('Do you want to exit the program?',mtinformation,[mbyes,mbno],0)=mrno then
exit;
Application.Terminate;
end;
end.

```

## 4.2. FORM OF SEARCH PRODUCT CODES

Label

unit arama;

Image

interface

Label

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, ADODB, dxDBCtrl, dxDBGrid, dxTL, dxCntner, ExtCtrls,  
StdCtrls, Grids, DBGrids, dxEditor, dxEdLib;

Label

type

TForm6 = class(TForm)

Image1: TImage;

Label7: TLabel;

Label1: TLabel;

Edit1: TEdit;

Label3: TLabel;

Edit3: TEdit;

Label4: TLabel;

ComboBox1: TComboBox;

Label5: TLabel;

ComboBox3: TComboBox;

Button2: TButton;

DataSource1: TDataSource;

ADOQuery1: TADOQuery;

Label6: TLabel;

Label8: TLabel;

Edit4: TEdit;

Edit5: TEdit;

Label9: TLabel;

Edit6: TEdit;

Edit12: TEdit;

Label12: TLabel;

Label16: TLabel;  
Label13: TLabel;  
Edit13: TEdit;  
Image2: TImage;  
Label11: TLabel;  
Edit9: TEdit;  
Label18: TLabel;  
Edit17: TEdit;  
Label15: TLabel;  
Edit15: TEdit;  
Label14: TLabel;  
Edit14: TEdit;  
Edit7: TEdit;  
Label2: TLabel;  
Label10: TLabel;  
dxDBGrid1: TdxDBGrid;  
dxDBGrid1Filminadi: TdxDBGridColumn;  
dxDBGrid1Yonetmen: TdxDBGridColumn;  
dxDBGrid1Oyuncular1: TdxDBGridColumn;  
dxDBGrid1Oyuncular2: TdxDBGridColumn;  
dxDBGrid1Oyuncular3: TdxDBGridColumn;  
dxDBGrid1Oyuncular4: TdxDBGridColumn;  
dxDBGrid1Oyuncular5: TdxDBGridColumn;  
dxDBGrid1Filmturu: TdxDBGridColumn;  
dxDBGrid1websitesi: TdxDBGridColumn;  
dxDBGrid1senaryo: TdxDBGridColumn;  
dxDBGrid1senaryo1: TdxDBGridColumn;  
dxDBGrid1senaryo2: TdxDBGridColumn;  
dxDBGrid1rafkodu: TdxDBGridColumn;  
dxDBGrid1suresi: TdxDBGridMaskColumn;  
dxDBGrid1goruntuyonetmen: TdxDBGridColumn;  
dxDBGrid1goruntuyonetmen1: TdxDBGridColumn;  
dxDBGrid1Yapm: TdxDBGridColumn;  
dxDBGrid1renk: TdxDBGridColumn;



```

dxDBGrid1muzik: TdxDBGridColumn;
dxDBGrid1dil: TdxDBGridColumn;
dxDBGrid1resim: TdxDBGridColumn;
dxDBGrid1Cdturu: TdxDBGridColumn;
dxDBGrid1cdadeti: TdxDBGridMaskColumn;
dxDBGrid1otomat: TdxDBGridMaskColumn;
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Edit1Change(Sender: TObject);
procedure DBGrid1DbClick(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure DataSource1DataChange(Sender: TObject; Field: TField);
procedure ComboBox3Change(Sender: TObject);
procedure Edit10KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Edit3KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Edit2KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure DBGrid1KeyPress(Sender: TObject; var Key: Char);
procedure Edit3Change(Sender: TObject);
procedure ComboBox1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure ComboBox1Change(Sender: TObject);
procedure Label10Click(Sender: TObject);
procedure Image2Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure dxDBGrid1DbClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

```

```

var
  Form6: TForm6;

implementation

uses rm,oyuncuyagorearama,resimgoster;
{$R *.dfm}

procedure TForm6.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var x:string;
begin
  x:=vartostr(key);
  case strtoint(x) of
    27:close;
  end;
end;

procedure TForm6.Edit1Change(Sender: TObject);
var tt:String;
var ss:string;
var sss:string;
var ssss:string;
var h:string;
begin
  tt:='';
  if edit3.Text<>'' then
  begin
    sss:='yonetmen like' +tt+ edit3.Text +'%'+tt + sss +' and+ ';
  end else begin
    sss:='';
  end;
end;

if combobox1.Text<>'' then
begin
  SS:='filmturu LIKE' + tt + combobox1.Text +'%'+ tt +ss+' and + ';
end else begin

```

```

ss:="";
end;
if combobox3.Text<>" then
begin
h:='cdturu LIKE ' + tt + combobox3.Text +%'+ tt +h+' and + ' ;
end else begin
h:="";
end;
sss:=ss+sss+h;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from cdcenter where '+sss+' filminadi LIKE ' + tt +
edit1.Text +%'+ tt;
ADOQuery1.open;
end;

```

```

procedure TForm6.DBGrid1DbClick(Sender: TObject);
begin
edit4.Text:=adoquery1.fieldbyname('websitesi').AsString;
edit5.Text:=adoquery1.fieldbyname('senaryo').AsString;
edit6.Text:=adoquery1.fieldbyname('suresi').asString;
edit12.Text:=adoquery1.fieldbyname('goruntuyonetmen').AsString;
edit7.Text:=adoquery1.fieldbyname('dil').asString;
edit13.Text:=adoquery1.fieldbyname('yapim').AsString;
edit6.Text:=adoquery1.fieldbyname('suresi').AsString;
combobox3.Text:=adoquery1.fieldbyname('cdturu').AsString;
combobox1.Text:=adoquery1.fieldbyname('filmturu').AsString;
edit1.Text:=adoquery1.fieldbyname('filminadi').AsString;
edit3.Text:=adoquery1.fieldbyname('yonetmen').AsString;
edit9.Text:=adoquery1.fieldbyname('rafkodu').AsString;
edit17.text:=adoquery1.fieldbyname('cdadeti').AsString;
edit15.Text:=adoquery1.fieldbyname('muzik').AsString;
edit14.Text:=adoquery1.fieldbyname('renk').AsString;

```



```

if adoquery1.fieldbyname('resim').AsString<>" then
image2.Picture.LoadFromFile(adoquery1.fieldbyname('resim').AsString);
end;

procedure TForm6.Button2Click(Sender: TObject);
begin
Edit1.Text:="";
edit3.Text:="";
combobox1.Text:="";
combobox3.Text:="";
edit4.Text:="";
edit5.Text:="";
edit12.Text:="";
edit7.Text:="";
edit13.Text:="";
edit6.Text:="";
image2.Picture:=nil;
edit9.Text:="";
edit17.Text:="";
edit15.Text:="";
edit14.Text:="";
edit1.SetFocus;
if edit1.Text="" then
begin
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from cdcenter';
ADOQuery1.Open;
exit;
end;
end;

procedure TForm6.DataSource1DataChange(Sender: TObject; Field: TField);
begin

```

```

dBGrid1DbIcIck(Sender);
end;

procedure TForm6.ComboBox3Change(Sender: TObject);
var tt:String;
var ss:string;
var sss:string;
var ssss:string;
var h:string;
var a:integer;
begin
tt="";
if edit3.Text<>" then
begin
sss:='yonetmen like' +tt+ edit3.Text +'%' +tt + sss + ' and+ ';
end else begin
sss="";
end;
if combobox1.Text<>" then
begin
SS:='filmturu LIKE ' + tt + combobox1.Text +'%' + tt +ss+' and + ';
end else begin
ss="";
end;
if edit1.Text<>" then
begin
h:='filminadi LIKE ' + tt + edit1.Text +'%' + tt +h+' and + ';
end else begin
h="";
end;
ssss:=ss+sss+h;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;

```

```

ADOQuery1.SQL.Text:='select * from cdcenter where '+ssss+' cdturu LIKE ' + tt
+combobox3.Text +'%' + tt ;
ADOQuery1.Open;
edit4.Text:="";
edit5.Text:="";
edit12.Text:="";
edit7.Text:="";
edit13.Text:="";
edit6.Text:="";
Image2.Picture:=nil;
edit9.Text:="";
edit17.Text:="";
edit15.Text:="";
edit14.Text:="";
end;

```

```

procedure TForm6.Edit10KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  edit4.Text:="";
  edit5.Text:="";
  edit12.Text:="";
  edit7.Text:="";
  edit13.Text:="";
  edit6.Text:="";
  Image2.Picture:=nil;
  edit9.Text:="";
  edit17.Text:="";
  edit15.Text:="";
  edit14.Text:="";
end;

```

```

procedure TForm6.Edit3KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);

```



```
begin
edit4.Text:="";
edit5.Text:="";
edit12.Text:="";
edit7.Text:="";
edit13.Text:="";
edit6.Text:="";
Image2.Picture:=nil;
edit9.Text:="";
edit17.Text:="";
edit15.Text:="";
edit14.Text:="";
end;
```

```
procedure TForm6.Edit2KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
begin
edit4.Text:="";
edit5.Text:="";
edit12.Text:="";
edit7.Text:="";
edit13.Text:="";
edit6.Text:="";
Image2.Picture:=nil;
edit9.Text:="";
edit17.Text:="";
edit15.Text:="";
edit14.Text:="";
end;
```

```
procedure TForm6.DBGrid1KeyPress(Sender: TObject; var Key: Char);
```

```
begin
edit4.Text:=adoquery1.fieldbyname('websitesi').AsString;
edit5.Text:=adoquery1.fieldbyname('senaryo').AsString;
```

```

edit6.Text:=adoquery1.fieldbyname('suresi').asString;
edit12.Text:=adoquery1.fieldbyname('goruntuyonetmeni').AsString;
edit7.Text:=adoquery1.fieldbyname('dil').asString;
edit13.Text:=adoquery1.fieldbyname('yapim').AsString;
edit6.Text:=adoquery1.fieldbyname('suresi').AsString;
combobox3.Text:=adoquery1.fieldbyname('cdturu').AsString;
combobox1.Text:=adoquery1.fieldbyname('filmturu').AsString;
edit1.Text:=adoquery1.fieldbyname('filminadi').AsString;
edit3.Text:=adoquery1.fieldbyname('yonetmen').AsString;
edit9.Text:=adoquery1.fieldbyname('rafkodu').AsString;
edit17.text:=adoquery1.fieldbyname('cdadeti').AsString;
edit15.Text:=adoquery1.fieldbyname('muzik').AsString;
edit14.Text:=adoquery1.fieldbyname('renk').AsString;
if adoquery1.fieldbyname('resim').AsString<>" then
image2.Picture.LoadFromFile(adoquery1.fieldbyname('resim').AsString);
end;

```

```

procedure TForm6.Edit3Change(Sender: TObject);
var tt:String;
var ss:string;
var sss:string;
var ssss:string;
var h:string;
begin
tt:="";
if edit1.Text<>" then
begin
sss:='filminadi like' +tt+ edit1.Text +'%' +tt + sss + ' and+ ' ;
end else begin
sss:="";
end;
if combobox1.Text<>" then
begin
ss:='filmturu LIKE ' + tt + combobox1.Text +'%' + tt +ss+' and + ' ;

```

```

end else begin
ss:="";
end;
if combobox3.Text<>" then
begin
h:='cdturu LIKE ' + tt + combobox3.Text +%'+ tt +h+' and + ' ;
end else begin
h:="";
end;
ssss:=ss+sss+h;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from cdcenter where '+ssss+' yonetmen LIKE ' + tt
+edit3.Text +%'+ tt ;
ADOQuery1.Open;
end;

```

```

procedure TForm6.ComboBox1KeyUp(Sender: TObject; var Key: Word;
Shift: TShiftState);

```

```

begin
edit4.Text:="";
edit5.Text:="";
edit12.Text:="";
edit7.Text:="";
edit13.Text:="";
edit6.Text:="";
Image2.Picture:=nil;
edit9.Text:="";
edit17.Text:="";
edit15.Text:="";
edit14.Text:="";
end;

```

```

procedure TForm6.ComboBox1Change(Sender: TObject);

```



```

var tt:String;
var ss:string;
var sss:string;
var ssss:string;
var h:string;
begin
tt:="";
if edit3.Text<>" then
begin
sss:='yonetmen like' +tt+ edit3.Text +'%'+tt + sss +' and+ ';
end else begin
sss:="";
end;
if edit1.Text<>" then
begin
SS:='filminadi LIKE ' + tt + edit1.Text +'%'+ tt +ss+' and + ';
end else begin
ss:="";
end;
if combobox3.Text<>" then
begin
h:='cdturu LIKE ' + tt + combobox3.Text +'%'+ tt +h+' and + ';
end else begin
h:="";
end;
ssss:=ss+sss+h;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from cdcenter where '+ssss+' filmturu LIKE ' + tt +
combobox1.Text +'%'+ tt ;
ADOQuery1.Open;
end;
procedure TForm6.Label10Click(Sender: TObject);

```

```

begin
edit1.Text:="";
edit3.Text:="";
combobox1.Text:="";
combobox3.Text:="";
application.CreateForm(Tform7,form7);
form7.ShowModal;
form7.Destroy;
end;

procedure TForm6.Image2Click(Sender: TObject);
begin
application.CreateForm(Tform9,form9);
form9.ShowModal;
form9.Destroy;
end;

procedure TForm6.FormShow(Sender: TObject);
begin
ADOQuery1.close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from cdcenter order by filminadi';
adoquery1.Open;
end;

procedure TForm6.dxDBGrid1DbClick(Sender: TObject);
begin
edit4.Text:=adoquery1.fieldbyname('websitesi').AsString;
edit5.Text:=adoquery1.fieldbyname('senaryo').AsString;
edit6.Text:=adoquery1.fieldbyname('suresi').AsString;
edit12.Text:=adoquery1.fieldbyname('goruntuyonetmen').AsString;
edit7.Text:=adoquery1.fieldbyname('dil').AsString;
edit13.Text:=adoquery1.fieldbyname('yapim').AsString;
edit9.Text:=adoquery1.fieldbyname('rafkodu').AsString;

```

```
edit17.Text:=adoquery1.fieldbyname('cdadeti').AsString;  
edit15.Text:=adoquery1.fieldbyname('muzik').AsString;  
edit14.Text:=adoquery1.fieldbyname('renk').AsString;  
if adoquery1.fieldbyname('resim').AsString<>" then  
image2.Picture.LoadFromFile(adoquery1.fieldbyname('resim').AsString);  
edit1.SetFocus;  
end;
```

```
end.
```



### 4.3.FORM OF SEARCH ACCORDING TO CAST CODES

```
unit oyuncuyagorearama;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, DB, ADODB, Grids, DBGrids, StdCtrls, ExtCtrls, dxDBCtrl,  
dxDBGrid, dxTL, dxCntner;
```

```
type
```

```
TForm7 = class(TForm)  
    Image1: TImage;  
    Label7: TLabel;  
    Label1: TLabel;  
    Edit1: TEdit;  
    ADOQuery1: TADOQuery;  
    DataSource1: TDataSource;  
    Label6: TLabel;  
    Label8: TLabel;  
    Label9: TLabel;  
    Label12: TLabel;  
    Label16: TLabel;  
    Label13: TLabel;  
    Image2: TImage;  
    Label11: TLabel;  
    Label18: TLabel;  
    Label15: TLabel;  
    Label14: TLabel;  
    Edit4: TEdit;  
    Edit5: TEdit;  
    Edit6: TEdit;
```

```
Edit12: TEdit;
Edit13: TEdit;
Edit9: TEdit;
Edit17: TEdit;
Edit15: TEdit;
Edit14: TEdit;
Edit7: TEdit;
dxDBGrid1: TdxDBGrid;
dxDBGrid1Filminadi: TdxDBGridColumn;
dxDBGrid1Yonetmen: TdxDBGridColumn;
dxDBGrid1Oyuncular1: TdxDBGridColumn;
dxDBGrid1Oyuncular2: TdxDBGridColumn;
dxDBGrid1Oyuncular3: TdxDBGridColumn;
dxDBGrid1Oyuncular4: TdxDBGridColumn;
dxDBGrid1Oyuncular5: TdxDBGridColumn;
dxDBGrid1Filmturu: TdxDBGridColumn;
dxDBGrid1websitesi: TdxDBGridColumn;
dxDBGrid1senaryo: TdxDBGridColumn;
dxDBGrid1senaryo1: TdxDBGridColumn;
dxDBGrid1senaryo2: TdxDBGridColumn;
dxDBGrid1rafkodu: TdxDBGridColumn;
dxDBGrid1suresi: TdxDBGridMaskColumn;
dxDBGrid1goruntuyonetmen: TdxDBGridColumn;
dxDBGrid1goruntuyonetmen1: TdxDBGridColumn;
dxDBGrid1Yapm: TdxDBGridColumn;
dxDBGrid1renk: TdxDBGridColumn;
dxDBGrid1muzik: TdxDBGridColumn;
dxDBGrid1dil: TdxDBGridColumn;
dxDBGrid1resim: TdxDBGridColumn;
dxDBGrid1Cdturu: TdxDBGridColumn;
dxDBGrid1cdadeti: TdxDBGridMaskColumn;
dxDBGrid1otomat: TdxDBGridMaskColumn;
procedure Edit1Change(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
```

```

    Shift: TShiftState);
procedure Image2Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure dxDBGrid1DbClick(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form7: TForm7;

implementation

{$R *.dfm}
uses rm, resimgosteriki;
procedure TForm7.Edit1Change(Sender: TObject);
var tt:string;
begin
    edit4.Text:="";
    edit13.Text:="";
    edit7.Text:="";
    edit5.Text:="";
    edit6.Text:="";
    edit12.Text:="";
    Image2.Picture:=nil;
    edit9.Text:="";
    edit17.Text:="";
    edit15.Text:="";
    edit14.Text:="";
    tt="";
    adoquery1.Close;
    adoquery1.SQL.Clear;

```



```
adoquery1.SQL.Text:='select *from cdcenter where oyuncular1 like'+tt+edit1.Text+'%'+tt+'or  
oyuncular2 like'+tt+edit1.Text+'%'+tt+'or oyuncular3 like'+tt+edit1.Text+'%'+tt+'or  
oyuncular4 like'+tt+edit1.Text+'%'+tt+'or oyuncular5 like'+tt+edit1.Text+'%'+tt;
```

```
adoquery1.Open;  
end;
```

```
procedure TForm7.FormKeyDown(Sender: TObject; var Key: Word;  
  Shift: TShiftState);  
  var a:string;  
  begin  
  a:=vartostr(key);  
  case strtoint(a) of  
  27:close;  
end;  
end;
```

```
procedure TForm7.Image2Click(Sender: TObject);  
begin  
application.CreateForm(Tform10,form10);  
form10.ShowModal;  
form10.Destroy;  
end;
```

```
procedure TForm7.FormShow(Sender: TObject);  
begin  
ADOQuery1.Close;  
adoquery1.SQL.Clear;  
adoquery1.SQL.Text:='select * from cdcenter order by filminadi';  
adoquery1.Open;  
end;
```

```
procedure TForm7.dxDBGrid1DbClick(Sender: TObject);  
begin
```

```
edit4.Text:=adoquery1.fieldbyname('websitesi').AsString;
edit5.Text:=adoquery1.fieldbyname('senaryo').AsString;
edit6.Text:=adoquery1.fieldbyname('suresi').asString;
edit12.Text:=adoquery1.fieldbyname('goruntuyonetmeni').AsString;
edit7.Text:=adoquery1.fieldbyname('dil').asString;
edit13.Text:=adoquery1.fieldbyname('yapim').AsString;
edit9.Text:=adoquery1.fieldbyname('rafkodu').AsString;
edit17.text:=adoquery1.fieldbyname('cdadeti').AsString;
edit15.Text:=adoquery1.fieldbyname('muzik').AsString;
edit14.Text:=adoquery1.fieldbyname('renk').AsString;
if adoquery1.fieldbyname('resim').AsString<>" then
image2.Picture.LoadFromFile(adoquery1.fieldbyname('resim').AsString);
edit1.SetFocus;
end;
end.
```

#### 4.4.FORM OF PRODUCT PRICE CODES

unit urunfiyat;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, dxDBTLCl, dxGrClms, dxDBGrid, dxTL, dxDBCtrl, ExtCtrls, DBCtrls,  
dxCntner, DB, ADODB, StdCtrls;

type

TForm14 = class(TForm)

  Image1: TImage;

  DataSource1: TDataSource;

  ADOQuery1: TADOQuery;

  dxDBGrid1: TdxDBGrid;

  dxDBGrid1ne: TdxDBGridPickColumn;

  dxDBGrid1fiyat: TdxDBGridMaskColumn;

  dxDBGrid1alians: TdxDBGridColumn;

  dxDBGrid1aktif: TdxDBGridCheckColumn;

  DBNavigator1: TDBNavigator;

  Label7: TLabel;

  procedure FormShow(Sender: TObject);

  procedure ADOQuery1PostError(DataSet: TDataSet; E: EDatabaseError;

    var Action: TDataAction);

  procedure FormKeyDown(Sender: TObject; var Key: Word;

    Shift: TShiftState);

private

  { Private declarations }

public

  { Public declarations }

end;

Form14: TForm14;



implementation

uses rm;

{SR \*.dfm}

procedure TForm14.FormShow(Sender: TObject);

begin

adoquery1.Open;

end;

procedure TForm14.ADOQuery1PostError(DataSet: TDataSet; E: EDatabaseError;

var Action: TDataAction);

begin

ADOQuery1.Cancel;

MessageDlg('There is any Problem to Record Operation Please Check it.',mterror,[mbok],0)

end;

procedure TForm14.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

var b:string;

begin

b:=vartostr(key);

case strtoint(b) of

27:close;

end;

end;

end.

#### 4.5.FORM OF RENT PRODUCT CODES

unit cdalim;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, dxCntner, dxEditor, dxExEdtr, dxEdLib, ExtCtrls,  
dxDBTLCl, dxGrClms, dxDBGrid, dxTL, dxDBCtrl, DB, ADODB;

type

TForm12 = class(TForm)

Image1: TImage;

Label1: TLabel;

Label2: TLabel;

Button1: TButton;

DataSource1: TDataSource;

ADOQuery1: TADOQuery;

Edit2: TEdit;

dxDBGrid1: TdxDBGrid;

Label7: TLabel;

dxDBGrid1Kimlik: TdxDBGridMaskColumn;

dxDBGrid1kullanici: TdxDBGridColumn;

dxDBGrid1password: TdxDBGridColumn;

dxDBGrid1dogumyeri: TdxDBGridColumn;

dxDBGrid1gizlisoru: TdxDBGridColumn;

dxDBGrid1cevap: TdxDBGridColumn;

dxDBGrid1sim: TdxDBGridColumn;

dxDBGrid1Soyisim: TdxDBGridColumn;

dxDBGrid1Email: TdxDBGridColumn;

dxDBGrid1Adres: TdxDBGridColumn;

dxDBGrid1Tarih: TdxDBGridDateColumn;

Edit3: TEdit;

Edit1: TdxCurrencyEdit;

procedure Edit1Change(Sender: TObject);

```

procedure Edit2Change(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure dxDBGrid1DbClick(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Edit1Click(Sender: TObject);
procedure Edit1Exit(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form12: TForm12;

implementation
  uses rm,hesap,arama, kiralama;
  {$R *.dfm}
procedure TForm12.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
  var x:string;
  begin
    x:=vartostr(key);
    case strtoint(x) of
      27:form12.close;
    end;
  end;
procedure TForm12.Edit1Change(Sender: TObject);
var tt:String;
var ss:string;
begin
  tt:="";

```



```

if (edit1.Text<>"") and (edit1.Text<>'0') then
begin
ss:='kimlik like' +tt+ edit1.Text +'%' +tt + ss +' and+ ';
end else begin
ss:="";
end;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from hakan where '+ss+' email LIKE ' + tt + edit2.Text
+'%' + tt;
adoquery1.Open;
end;

procedure TForm12.Edit2Change(Sender: TObject);
var tt:String;
var sss:string;
begin
tt:="";
if edit2.Text<>" then
begin
sss:='email like' +tt+ edit2.Text +'%' +tt + sss +' and+ ';
end else begin
sss:="";
end;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from hakan where '+sss+' kimlik LIKE ' + tt + edit1.Text
+'%' + tt;
adoquery1.Open;
end;

procedure TForm12.Button1Click(Sender: TObject);
begin
if (edit1.Text="") and (edit2.Text="") then

```

```

begin
messagedlg('Please Enter Customer Number or Email Address.',mtinformation,[mbok],0);
edit1.SetFocus;
exit;
end;
if edit1.Text="" then
begin
messagedlg('if you dont enter the Customer Number You can not
entering.',mterror,[mbok],0);
edit1.SetFocus;
exit;
end else begin
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from hakan where kimlik =' +edit1.Text+';
adoquery1.Open;
edit3.Text:=adoquery1.fieldbyname('Isim').AsString;
if adoquery1.RecordCount=0 then
begin
if messagedlg('There is no record.If you want to register please click the "yes"
button.',mtinformation,[mbyes,mbno],0)= mrno then
begin
edit1.Text:="";
edit2.Text:="";
exit;
end else begin
form11.ShowModal;
form11.Destroy;
end;
end else begin
application.CreateForm(Tform15,form15);
form15.ShowModal;
form15.Destroy;
end;
end;

```

end;

end;

```
procedure TForm12.FormShow(Sender: TObject);
begin
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from hakan order by isim';
adoquery1.Open;
end;
```

```
procedure TForm12.dxDBGrid1DbClick(Sender: TObject);
begin
edit1.Text:=adoquery1.fieldbyname('kimlik').AsString;
edit3.Text:=adoquery1.fieldbyname('İsim').AsString;
application.CreateForm(tform15,form15);
form15.ShowModal;
form15.Destroy;
end;
```

```
procedure TForm12.Edit1Click(Sender: TObject);
begin
if edit1.Text='0' then
begin
edit1.Text:="";
end;
end;
```

```
procedure TForm12.Edit1Exit(Sender: TObject);
begin
if edit1.Text='0' then edit1.Text:="";
end;
end;
```



#### 4.6.FORM OF CD SELLING CODES

```
unit kiralama;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, jpeg, ExtCtrls, DB, ADODB, dxCntner, dxTL, dxDBCtrl,  
dxDBGrid, dxEditor, dxExEdtr, dxEdLib, dxDBTLCl, dxGrClms, ComCtrls;
```

```
type
```

```
TForm15 = class(TForm)
```

```
Image1: TImage;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Label3: TLabel;
```

```
Label5: TLabel;
```

```
Label6: TLabel;
```

```
Label7: TLabel;
```

```
Label8: TLabel;
```

```
Label9: TLabel;
```

```
Label18: TLabel;
```

```
Label20: TLabel;
```

```
Label21: TLabel;
```

```
Label22: TLabel;
```

```
Label23: TLabel;
```

```
Label26: TLabel;
```

```
Label28: TLabel;
```

```
Edit1: TEdit;
```

```
Button1: TButton;
```

```
Edit2: TEdit;
```

```
Edit3: TEdit;
```

```
Edit4: TEdit;
```

```
Edit5: TEdit;
```

Edit7: TEdit;  
ComboBox1: TComboBox;  
Edit6: TEdit;  
Button2: TButton;  
Button3: TButton;  
dxDBGrid1: TdxDBGrid;  
Edit12: TEdit;  
Edit14: TEdit;  
Edit15: TEdit;  
dxCurrencyEdit2: TdxCurrencyEdit;  
DateTimePicker1: TdxDateEdit;  
DateTimePicker2: TdxDateEdit;  
ADOQuery1: TADOQuery;  
DataSource1: TDataSource;  
Label14: TLabel;  
Edit9: TEdit;  
Label16: TLabel;  
dxCurrencyEdit3: TdxCurrencyEdit;  
Edit11: TdxCurrencyEdit;  
dxDBGrid1kimlik: TdxDBGridMaskColumn;  
dxDBGrid1filminadi: TdxDBGridColumn;  
dxDBGrid1rafkodu: TdxDBGridColumn;  
dxDBGrid1cdkiralama: TdxDBGridMaskColumn;  
dxDBGrid1cdsatis: TdxDBGridMaskColumn;  
dxDBGrid1alistarih: TdxDBGridDateColumn;  
dxDBGrid1veristarih: TdxDBGridDateColumn;  
dxDBGrid1kiralananfiyat: TdxDBGridMaskColumn;  
dxDBGrid1veresiye: TdxDBGridMaskColumn;  
dxDBGrid1iskonto: TdxDBGridMaskColumn;  
dxDBGrid1verdigitutar: TdxDBGridMaskColumn;  
Label27: TLabel;  
dxCurrencyEdit4: TdxCurrencyEdit;  
ComboBox2: TComboBox;  
dxCurrencyEdit1: TdxCurrencyEdit;

```

Edit8: TdxCurrencyEdit;
procedure Button1Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Edit7Change(Sender: TObject);
procedure Edit9Change(Sender: TObject);
procedure dxCurrencyEdit2Enter(Sender: TObject);
procedure dxCurrencyEdit2Click(Sender: TObject);
procedure FormKeyPress(Sender: TObject; var Key: Char);
procedure Edit8Change(Sender: TObject);
procedure ComboBox2Enter(Sender: TObject);
procedure ComboBox2Click(Sender: TObject);
procedure dxCurrencyEdit2Change(Sender: TObject);
procedure dxCurrencyEdit5Change(Sender: TObject);
procedure ComboBox2Exit(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form15: TForm15;

implementation

{$R *.dfm}
uses filmlergenel,rm, hesap, cdalim;
procedure TForm15.Button1Click(Sender: TObject);
begin
  Application.CreateForm(TFilmler, Filmler);

```



```
Filmler.edit1.Text:='1';
```

```
Filmler.showmodal;
```

```
Filmler.destroy;
```

```
end;
```

```
procedure TForm15.FormShow(Sender: TObject);
```

```
begin
```

```
edit7.Text:=form12.Edit1.Text;
```

```
edit2.Text:=form12.Edit3.Text;
```

```
dm.adoquery1.Close;
```

```
dm.adoquery1.SQL.Clear;
```

```
dm.adoquery1.SQL.Text:='select * from hakan where kimlik='+edit7.Text+'';
```

```
dm.adoquery1.Open;
```

```
edit3.Text:=dm.adoquery1.fieldbyname('soyisim').AsString;
```

```
edit4.Text:=dm.adoquery1.fieldbyname('telefon').AsString;
```

```
edit5.Text:=dm.adoquery1.fieldbyname('email').AsString;
```

```
dm.ADOQuery1.Close;
```

```
form12.Edit3.Text:="";
```

```
form12.Edit1.Text:="";
```

```
DateTimePicker1.Date:=date;
```

```
dm.ADOQuery1.Close;
```

```
dm.ADOQuery1.SQL.Clear;
```

```
dm.ADOQuery1.SQL.Text:='select * from cdkiralama where kimlik='+edit7.Text+' order by  
alistarih desc';
```

```
dm.ADOQuery1.Open;
```

```
end;
```

```
procedure TForm15.Button3Click(Sender: TObject);
```

```
begin
```

```
edit12.Text:='1';
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select * from cdkiralama';
```

```
adoquery1.Open;
```

```

adoquery1.Insert;
if edit1.Text="" then
begin
messagedlg('Please Enter Product Name.',mtinformation,[mbok],0);
edit1.SetFocus;
exit;
end;
if combobox1.Text="" then
begin
messagedlg('You cant pass empty to Selling Type...',mtinformation,[mbok],0);
combobox1.SetFocus;
exit;
end;
if (combobox1.Text='Rent') and (datetimepicker2.Text=")then
begin
messagedlg('Kiralama seçeneğini seçtiniz ve müşterinin geri getirme tarihini boş bıraktınız,Lütfen dolurunuz',mtinformation,[mbok],0);
datetimepicker2.SetFocus;
exit;
end;
if combobox2.Text="" then
begin
messagedlg('Fiyat seçeneklerini boş geçemessiniz...',mtinformation,[mbok],0);
combobox2.SetFocus;
exit;
end;
if DateTimePicker1.Text="" then
begin
messagedlg('Müşterinin ürünü alış tarihini boş geçemessiniz...',mtinformation,[mbok],0);
DateTimePicker1.SetFocus;
exit;
end;

if dxCurrencyEdit2.Text="" then

```

```

begin
messagedlg('Müşterinin ürün için verecek olduğu fiyat yerini boş
geçemessiniz...',mtinformation,[mbok],0);
dxCurrencyEdit2.SetFocus;
exit;
end;
if (dxCurrencyEdit3.Text="") and (Edit11.Text="") then
begin
messagedlg('Please Enter Customer Price.',mtinformation,[mbok],0);
dxCurrencyEdit2.SetFocus;
exit;
end;
adoquery1.FieldName('kimlik').AsString:=edit7.Text;
adoquery1.FieldName('filminadi').AsString:=edit1.Text;
adoquery1.FieldName('rafkodu').AsString:=edit6.Text;
adoquery1.FieldName('alistarih').AsString:=DateTimePicker1.Text;
adoquery1.FieldName('kiralananfiyat').AsString:=dxCurrencyEdit1.Text;
if edit8.Text="" then edit8.Text:='0';
adoquery1.FieldName('iskonto').AsString:=Edit8.Text;
adoquery1.FieldName('indirimlifiyat').AsString:=dxCurrencyEdit4.Text;
adoquery1.FieldName('verdigitutar').AsString:=dxCurrencyEdit2.Text;
if dxCurrencyEdit3.Text<>"" then
adoquery1.FieldName('paraüstü').AsString:=dxCurrencyEdit3.Text;
if combobox1.Text='Rent' then //kiralama ise 1.if
begin
if (DateTimePicker2.Date<DateTimePicker1.Date) then //2. if
begin //geliş tarihi boşsa veya geliş tarihi alım tarihinden küçükse hata var...
messagedlg('Lütfen tarihlere dikkat ediniz',mtinformation,[mbok],0);
DateTimePicker2.SetFocus;
exit;
end else begin
adoquery1.FieldName('veristarih').AsString:=DateTimePicker2.Text;
end; //bu 2. if in end i
end else begin

```



```

DateTimePicker2.Text:="";
end;
if combobox1.Text='Rent' then
begin
adoquery1.FieldByName('cdkiralama').AsString:=edit12.Text;
end else begin;
adoquery1.FieldByName('cdsatis').AsString:=edit12.Text;
end;
if edit11.text<>" then adoquery1.FieldByName('veresiye').AsString:=Edit11.Text;
if edit8.text<>" then adoquery1.FieldByName('iskonto').AsString:=Edit8.Text;
adoquery1.Post;
adoquery1.Close;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.cdsatis) AS
Toplacdsatis '+
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
(((cdkiralama.kimlik)='+Edit7.Text+') '+
' AND ((Sum(cdkiralama.cdsatis)) Is Not Null And (Sum(cdkiralama.cdsatis))<>0))';
adoquery1.Open;
if adoquery1.RecordCount > 0 then
begin
edit15.text := adoquery1.Fields[1].AsString;
end else begin
edit15.Text := '0';
end;
adoquery1.close;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.veresiye) AS
Toplaveresiye '+
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
(((cdkiralama.kimlik)='+edit7.Text+') '+

```

```
' AND ((Sum(cdkiralama.veresiye)) Is Not Null And (Sum(cdkiralama.veresiye)<>0));
```

```
adoquery1.Open;
```

```
if adoquery1.RecordCount > 0 then
```

```
begin
```

```
end else begin
```

```
end;
```

```
adoquery1.close;
```

```
if edit7.Text = " Then exit;
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.cdkiralama) AS
```

```
Toplacdkiralama '+
```

```
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
```

```
((cdkiralama.kimlik)='+Edit7.Text+') '+
```

```
' AND ((Sum(cdkiralama.cdkiralama)) Is Not Null And
```

```
(Sum(cdkiralama.cdkiralama)<>0));
```

```
adoquery1.Open;
```

```
if adoquery1.RecordCount > 0 then
```

```
begin
```

```
edit14.text := adoquery1.Fields[1].AsString;
```

```
end else begin
```

```
edit14.text := '0';
```

```
end;
```

```
if DateTimePicker1.Text<>" then
```

```
begin
```

```
dxDBGrid1.ClearGroupColumns;
```

```
dm.ADOQuery1.Close;
```

```
dm.ADOQuery1.SQL.Clear;
```

```
dm.ADOQuery1.SQL.Text:='select * from cdkiralama where kimlik='+edit7.Text+' and
```

```
alistarih=#'+formatdatetime('mm dd yyyy',DateTimePicker1.Date)+'#';
```

```
dm.ADOQuery1.Open;
```

```
edit1.Text:=";
```

```
edit6.Text:=";
```

```
combobox1.Text:="";
combobox2.Text:="";
DateTimePicker1.Text:="";
DateTimePicker2.Text:="";
dxCurrencyEdit1.Text:="";
Edit8.Text:="";
end;
end;
```

```
procedure TForm15.Edit1Change(Sender: TObject);
begin
adoquery1.Close;
adoquery1.SQL.Text;
adoquery1.SQL.Text:='select * from cdcenter where filminadi="'+edit1.Text+'";
adoquery1.Open;
edit6.Text:=adoquery1.fieldbyname('rafkodu').AsString;
end;
```

```
procedure TForm15.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var a:string;
begin
a := varostr(key);
case strtoint(a) of
27:close;
end;
end;
```

```
procedure TForm15.Edit7Change(Sender: TObject);
begin
if edit7.Text = " Then exit;
```



```

adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.cdkiralama) AS
Toplacdkiralama '+
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
(((cdkiralama.kimlik)='+Edit7.Text+') '+
' AND ((Sum(cdkiralama.cdkiralama)) Is Not Null And
(Sum(cdkiralama.cdkiralama))<>0))';
adoquery1.Open;
if adoquery1.RecordCount > 0 then
begin
edit14.text := adoquery1.Fields[1].AsString;
end else begin
edit14.text := '0';
end;
adoquery1.close;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.cdsatis) AS
Toplacdsatis '+
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
(((cdkiralama.kimlik)='+Edit7.Text+') '+
' AND ((Sum(cdkiralama.cdsatis)) Is Not Null And (Sum(cdkiralama.cdsatis))<>0))';
adoquery1.Open;
if adoquery1.RecordCount > 0 then
begin
edit15.text := adoquery1.Fields[1].AsString;
end else begin
edit15.Text := '0';
end;
adoquery1.close;
adoquery1.Close;
adoquery1.SQL.Clear;

```

```

adoquery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.veresiye) AS
Toplaveresiye '+
' FROM cdkiralama GROUP BY cdkiralama.kimlik HAVING
(((cdkiralama.kimlik)='+edit7.Text+') '+
' AND ((Sum(cdkiralama.veresiye)) Is Not Null And (Sum(cdkiralama.veresiye))<>0)');
adoquery1.Open;
if adoquery1.RecordCount > 0 then
begin
end else begin
end;
adoquery1.close;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='SELECT cdkiralama.kimlik, Sum(cdkiralama.cdsatis) AS
Toplacdsatis '+
' FROM cdkiralama '+
' GROUP BY cdkiralama.kimlik '+
' HAVING (((cdkiralama.kimlik)='+edit7.Text+')');
ADOQuery1.Open;
if adoquery1.RecordCount>0 then
begin
edit15.Text:=adoquery1.Fields[1].AsString;
end else begin
edit15.Text:='0';
end;
adoquery1.Close;
end;

procedure TForm15.Edit9Change(Sender: TObject);
begin
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from fiyatlar where aliens="'+edit9.Text+'";
adoquery1.Open;

```

```
dxCurrencyEdit1.Text:=adoquery1.fieldbyname('fiyat').AsString;
dxCurrencyEdit4.Text:=adoquery1.fieldbyname('fiyat').AsString;
adoquery1.Close;
end;
```

```
procedure TForm15.dxCurrencyEdit2Enter(Sender: TObject);
begin
edit11.Text:="";
dxCurrencyEdit3.Text:="";
end;
```

```
procedure TForm15.dxCurrencyEdit2Click(Sender: TObject);
begin
if dxCurrencyEdit2.Text="" then dxCurrencyEdit2.Text:='0';//eğer verdiği miktar boşsa 0
yapıyor değeri..
end;
```

```
procedure TForm15.FormKeyPress(Sender: TObject; var Key: Char);
begin
if (key=#13) then button3click(sender); //enter tuşu ile tamamla butonuna gidiyor...
end;
```

```
procedure TForm15.Edit8Change(Sender: TObject);//iskonto
var i:real;
var j:real;
var k:real;
begin
if dxCurrencyEdit1.Text<>" then
begin
i:=strtofloat(dxCurrencyEdit1.Text);
j:=strtofloat(Edit8.Text);
```



```
k:=((i*(100-j))/100);  
dxCurrencyEdit4.Text:=floattostr(k);  
end;  
end;
```

```
procedure TForm15.ComboBox2Enter(Sender: TObject);  
var a:integer;  
begin  
dxCurrencyEdit1.Text:="";  
combobox2.Clear;  
adoquery1.Close;  
adoquery1.SQL.Clear;  
adoquery1.SQL.Text:='SELECT fiyatlar.ne, fiyatlar.aliens '+  
' FROM fiyatlar '+  
' GROUP BY fiyatlar.ne, fiyatlar.aliens '+  
' HAVING (((fiyatlar.aliens) Is Not Null))';  
adoquery1.Open;  
adoquery1.First;  
for a:=0 to adoquery1.RecordCount-1 do  
begin  
combobox2.Items.Add(adoquery1.fieldbyname('aliens').AsString);  
adoquery1.Next;  
end;  
end;
```

```
procedure TForm15.ComboBox2Click(Sender: TObject);  
begin  
adoquery1.Close;  
adoquery1.SQL.Clear;  
adoquery1.SQL.Text:='select *from fiyatlar where aliens="'+combobox2.Text+"'";  
adoquery1.Open;  
dxCurrencyEdit1.Text:=adoquery1.fieldbyname('fiyat').AsString;  
end;
```

```

procedure TForm15.dxCurrencyEdit2Change(Sender: TObject);
var a:real;
var b:real;
var c:real;
begin
if dxCurrencyEdit2.Text<>" then
if dxCurrencyEdit4.Text="" then
begin
exit;
end else begin
a:=strtofloat(dxCurrencyEdit2.Text);
b:=strtofloat(dxCurrencyEdit4.Text);
c:=(a-b);
if c>0 then
begin
dxCurrencyEdit3.Text:=floattostr(c);
Edit11.Text:="";
exit;
end else begin
edit11.Text:=floattostr(c);
dxCurrencyEdit3.Text:="";
exit;
end;
end;
end;
end;

```

```

procedure TForm15.dxCurrencyEdit5Change(Sender: TObject);
var i:real;
var j:real;
var k:real;
begin
if edit8.Text="" then exit;
i:=strtofloat(dxCurrencyEdit1.Text);
j:=strtofloat(Edit8.Text);

```

```
k:=((i*(100-j))/100);
```

```
dxCurrencyEdit4.Text:=floattostr(k);
```

```
end;
```

```
procedure TForm15.ComboBox2Exit(Sender: TObject);
```

```
begin
```

```
dxCurrencyEdit4.Text:='';
```

```
dxCurrencyEdit2.Text:='';
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select *from fiyatlar where aliens='''+combobox2.Text+'''';
```

```
adoquery1.Open;
```

```
dxCurrencyEdit4.Text:=adoquery1.fieldbyname('fiyat').AsString;
```

```
edit8.Text:='';
```

```
end;
```

```
end.
```



#### 4.7.FORM OF REGISTER PRODUCT CODES

unit frmkayit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, Menus, ExtDlgs, DB, ADODB, StdCtrls, ExtCtrls, jpeg, dxDBCtrl,  
dxDBGrid, dxTL, dxCntner, dxEditor, dxExEdtr, dxEdLib;

type

TForm2 = class(TForm)

OpenPictureDialog1: TOpenPictureDialog;

resimtemizle: TPopupMenu;

Resimtemizle1: TMenuItem;

hakan: TADOQuery;

Image1: TImage;

Label1: TLabel;

Label3: TLabel;

Label7: TLabel;

Label11: TLabel;

Label2: TLabel;

Label5: TLabel;

Label6: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Label16: TLabel;

Image2: TImage;

Label17: TLabel;

Label18: TLabel;

```
Label12: TLabel;  
Edit1: TEdit;  
edit11: TEdit;  
Edit3: TEdit;  
Edit4: TEdit;  
Edit5: TEdit;  
Edit6: TEdit;  
Edit7: TEdit;  
ComboBox1: TComboBox;  
Edit8: TEdit;  
Edit10: TEdit;  
Edit12: TEdit;  
Edit13: TEdit;  
Edit14: TEdit;  
Edit15: TEdit;  
Button3: TButton;  
Button4: TButton;  
ComboBox2: TComboBox;  
ComboBox3: TComboBox;  
Edit16: TEdit;  
Edit18: TEdit;  
Edit19: TEdit;  
Button1: TButton;  
Button2: TButton;  
Edit9: TEdit;  
Label4: TLabel;  
Edit17: TdxCurrencyEdit;  
Edit2: TdxCurrencyEdit;  
procedure Button2Click(Sender: TObject);  
procedure Button3Click(Sender: TObject);  
procedure Button1Click(Sender: TObject);  
procedure Button4Click(Sender: TObject);  
procedure Image1Click(Sender: TObject);  
procedure FormKeyDown(Sender: TObject; var Key: Word;
```

```

    Shift: TShiftState);
procedure FormShow(Sender: TObject);
procedure FormKeyPress(Sender: TObject; var Key: Char);
procedure Edit3Enter(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form2: TForm2;

implementation

{$R *.dfm}
uses rm, cdcenter, resimgoster, help;
procedure TForm2.Button2Click(Sender: TObject);
begin
    if edit1.Text="" then
        begin
            messagedlg(' Please Enter the Movie Name. ',mtwarning,[mbok],0);
            edit1.SetFocus;
            exit;
        end;
    if edit9.Text="" then
        begin
            messagedlg(' Please Enter the Self Number. ',mtwarning,[mbok],0);
            edit9.SetFocus;
            exit;
        end;
    hakan.Close;
    hakan.SQL.Clear;
    hakan.sql.Text:='select *from cdcenter';

```

```

hakan.Open;
hakan.insert;
hakan.FieldName('filminadi').AsString:=edit1.Text;
hakan.FieldName('yonetmen').AsString:=edit11.Text;
hakan.FieldName('oyuncular1').AsString:=edit3.Text;
hakan.FieldName('oyuncular2').AsString:=edit4.Text;
hakan.FieldName('oyuncular3').AsString:=edit5.Text;
hakan.FieldName('oyuncular4').AsString:=edit6.Text;
hakan.FieldName('oyuncular5').AsString:=edit7.Text;
hakan.FieldName('filmturu').AsString:=combobox1.Text;
hakan.FieldName('websitesi').AsString:=edit8.Text;
hakan.FieldName('senaryo').AsString:=edit10.Text;
hakan.FieldName('senaryo1').AsString:=edit16.Text;
hakan.FieldName('senaryo2').AsString:=edit18.Text;
hakan.FieldName('rafkodu').AsString:=edit9.Text;
hakan.FieldName('suresi').AsString:=edit2.Text;
hakan.FieldName('goruntuyonetmen').AsString:=edit12.Text;
hakan.FieldName('goruntuyonetmen1').AsString:=edit19.Text;
hakan.FieldName('yapim').AsString:=edit13.Text;
hakan.FieldName('renk').AsString:=edit14.Text;
hakan.FieldName('muzik').AsString:=edit15.Text;
hakan.FieldName('dil').AsString:=combobox2.Text;
hakan.FieldName('cdadeti').AsString:=edit17.Text;
if openpicturedialog1.FileName <> "" then
hakan.FieldName('resim').AsString:=OpenPictureDialog1.FileName;
hakan.FieldName('cdturu').AsString:=combobox3.Text;
hakan.FieldName('cdadeti').AsString:=edit17.Text;
hakan.post;
hakan.close;
edit1.Text:="";
edit2.Text:="";
edit3.Text:="";
edit4.Text:="";
edit5.Text:="";

```



```
edit6.Text:="";
edit7.Text:="";
edit8.Text:="";
edit9.Text:="";
edit10.Text:="";
edit11.Text:="";
edit12.Text:="";
edit13.Text:="";
edit14.Text:="";
edit15.Text:="";
edit16.Text:="";
edit18.Text:="";
edit19.Text:="";
combobox1.Text:="";
combobox2.Text:="";
combobox3.text:="";
OpenPictureDialog1.FileName:="";
Image2.Picture:=nil;
edit17.Text:="";
frmhelp.edit1.Text:='0';
label4.Caption:='Your informations have registered successful.'
end;
```

```
procedure TForm2.Button3Click(Sender: TObject);
```

```
begin
```

```
edit1.Text:="";
```

```
edit2.Text:="";
```

```
edit3.Text:="";
```

```
edit4.text:="";
```

```
edit5.Text:="";
```

```
edit6.Text:="";
```

```
edit7.Text:="";
```

```
edit8.Text:="";
```

```
edit9.Text:="";
```

```
edit10.Text:="";
edit11.Text:="";
edit12.Text:="";
edit13.Text:="";
edit14.Text:="";
edit15.Text:="";
edit16.Text:="";
edit18.Text:="";
edit19.Text:="";
combobox1.Text:="";
combobox2.Text:="";
combobox3.text:="";
OpenPictureDialog1.FileName:="";
edit17.Text:="";
Image2.Picture:=nil;
end;
```

```
procedure TForm2.Button1Click(Sender: TObject);
begin
OpenPictureDialog1.Execute;
if OpenPictureDialog1.FileName="" then exit;
Image2.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;
```

```
procedure TForm2.Button4Click(Sender: TObject);
begin
close;
end;
```

```
procedure TForm2.Image1Click(Sender: TObject);
begin
OpenPictureDialog1.FileName:="";
Image2.Picture:=nil;
end;
```

```
procedure TForm2.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var a:string;
begin
a:=vartostr(key);
case strtoint(a) of
27:close;
end;
end;
```

```
procedure TForm2.FormShow(Sender: TObject);
begin
dm.SorguR('select * from hakan where kullanici= '"+ana.StatusBar1.Panels[2].Text+"' ');
dm.ADOQuery1.Close;
end;
```

```
procedure TForm2.FormKeyPress(Sender: TObject; var Key: Char);
begin
if (key=#13) then button2click(sender);
end;
```

```
procedure TForm2.Edit3Enter(Sender: TObject);
begin
edit3.Text="";
edit4.Text="";
edit5.Text="";
edit6.Text="";
edit7.Text="";

end;
end.
```

#### 4.8.FORM OF RENTED PRODUCT CODES

```
unit kiradakiurun;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls, DB, ADODB, dxCntner, dxTL, dxDBCtrl,  
dxDBGrid, dxDBTLCl, dxGrClms, dxEditor, dxExEdtr, dxEdLib, jpeg;
```

```
type
```

```
TForm17 = class(TForm)
```

```
dxDBGrid1: TdxDBGrid;
```

```
DataSource1: TDataSource;
```

```
ADOQuery1: TADOQuery;
```

```
Panel1: TPanel;
```

```
RadioGroup1: TRadioGroup;
```

```
Label7: TLabel;
```

```
GroupBox1: TGroupBox;
```

```
Label1: TLabel;
```

```
dxDBGrid1kimlik: TdxDBGridMaskColumn;
```

```
dxDBGrid1filminadi: TdxDBGridColumn;
```

```
dxDBGrid1rafkodu: TdxDBGridColumn;
```

```
dxDBGrid1cdkiralama: TdxDBGridMaskColumn;
```

```
dxDBGrid1cdsatis: TdxDBGridMaskColumn;
```

```
dxDBGrid1alistarih: TdxDBGridDateColumn;
```

```
dxDBGrid1veristarih: TdxDBGridDateColumn;
```

```
dxDBGrid1kiralananfiyat: TdxDBGridMaskColumn;
```

```
dxDBGrid1veresiye: TdxDBGridMaskColumn;
```

```
dxDBGrid1iskonto: TdxDBGridMaskColumn;
```

```
dxDBGrid1verdigitutar: TdxDBGridMaskColumn;
```

```
dxDBGrid1indirimlifiyat: TdxDBGridMaskColumn;
```

```
dxDBGrid1parast: TdxDBGridMaskColumn;
```

```
dxDateEdit1: TdxDateEdit;
```



```
dxDateEdit2: TdxDateEdit;
Button1: TButton;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Edit1: TEdit;
Edit3: TEdit;
Label9: TLabel;
Edit6: TEdit;
ADOQuery2: TADOQuery;
Label5: TLabel;
Label6: TLabel;
Label8: TLabel;
Image2: TImage;
Edit2: TEdit;
Edit4: TEdit;
Edit5: TEdit;
Label10: TLabel;
Image3: TImage;
Image1: TImage;
Image4: TImage;
Image5: TImage;
Image6: TImage;
Label11: TLabel;
Edit7: TEdit;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure RadioButton2Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure RadioButton3Click(Sender: TObject);
```

```

procedure Button1Click(Sender: TObject);
procedure RadioButton4Click(Sender: TObject);
procedure dxDBGrid1DblClick(Sender: TObject);
procedure Edit6Change(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form17: TForm17;

implementation

{$R *.dfm}
uses rm,help;
procedure TForm17.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var a:string;
begin
a:=vartostr(key);
case strtoint(a) of
27:close;
end;
end;

procedure TForm17.RadioButton2Click(Sender: TObject);
begin
label2.Visible:=False;
edit1.Visible:=False;
label3.Visible:=False;

```

```
label4.Visible:=false;
dxdateedit1.Visible:=false;
dxdateedit2.Visible:=false;
button1.Visible:=False;
label1.Visible:=True;
label1.Caption:='List of All Rented Product';
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from cdkiralama order by alistarih';
adoquery1.Open;
end;
```

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

```
begin //form show
```

```
label1.Visible:=True;
```

```
Label1.Caption:='List of Rented Product in Today';
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select * from cdkiralama where alistarih=#'+formatdatettime('mm dd
yyyy',Date)+'#';
```

```
adoquery1.Open;
```

```
end;
```

```
procedure TForm17.RadioButton3Click(Sender: TObject);
```

```
begin
```

```
adoquery1.Close;
```

```
label1.Visible:=False;
```

```
label2.Visible:=True;
```

```
label2.Caption:='First Product';
```

```
label3.Visible:=True;
```

```
dxdateedit1.Visible:=True;
```

```
dxdateedit2.Visible:=True;
```

```
button1.Visible:=True;
```

```
edit1.Visible:=False;
```

```
Label4.Visible:=false;
```

```
end;
```

```
procedure TForm17.Button1Click(Sender: TObject);
```

```
begin //buton a basıldıgında arama yapar...
```

```
if (dxdateedit1.Text="" or (dxdateedit2.Text="")) then
```

```
begin
```

```
messaging('Please pay attention date boxes.You have to fill two boxes
```

```
!',mtinformation,[mbok],0);
```

```
exit;
```

```
end;
```

```
if dxdateedit1.Date>dxdateedit2.Date then
```

```
begin
```

```
messaging('Please pay attention last date.First date must be ex date to last
```

```
date.',mtinformation,[mbok],0);
```

```
dxdateedit2.SetFocus;
```

```
exit;
```

```
end;
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select * from cdkiralama where (alistarih between
```

```
#+formatdatetime('mm dd yyyy',dxDateEdit1.Date)+'# AND #'+formatdatetime('mm dd
```

```
yyyy',dxDateEdit2.Date)+'#)';
```

```
adoquery1.Open;
```

```
end;
```

```
procedure TForm17.RadioButton4Click(Sender: TObject);
```

```
begin
```

```
label1.Visible:=False;
```

```
label2.Visible:=false;
```

```
label3.Visible:=False;
```

```
Label4.Visible:=True;
```

```
dxdateedit1.Visible:=False;
```

```
dxdateedit2.Visible:=False;
```



```
button1.Visible:=False;
edit1.Visible:=True;
Label4.Caption:='Search Product = ';
end;
```

```
procedure TForm17.dxBGGrid1DbClick(Sender: TObject);
begin
edit6.Text:=adoquery1.fieldbyname('Kimlik').AsString;
end;
```

```
procedure TForm17.Edit6Change(Sender: TObject);
begin
adoquery2.Close;
adoquery2.SQL.Clear;
adoquery2.SQL.Text:='select * from hakan where kimlik='+edit6.Text+';
adoquery2.Open;
edit2.Text:=adoquery2.fieldbyname('İsim').AsString;
if adoquery2.fieldbyname('resim').AsString<>'' then
image2.Picture.LoadFromFile(adoquery2.fieldbyname('resim').AsString);
edit5.Text:=adoquery2.fieldbyname('soyisim').AsString;
edit4.Text:=adoquery2.fieldbyname('telefon').AsString;
edit7.Text:=adoquery2.fieldbyname('email').AsString;
adoquery2.Close;
end;
```

```
procedure TForm17.Edit1Change(Sender: TObject);
var tt:String;
begin
tt:='';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Text:='select * from cdkiralama where filminadi LIKE ' + tt + edit1.Text
+'%'+ tt;
```

```
ADOQuery1.open;
```

```
end;
```

```
procedure TForm17.RadioButton1Click(Sender: TObject);
```

```
begin
```

```
label1.Visible:=true;
```

```
Edit1.Visible:=False;
```

```
Label1.Caption:='List of Rented Product in Today';
```

```
label2.Visible:=False;
```

```
label3.Visible:=False;
```

```
label4.Visible:=False;
```

```
dxdateedit1.Visible:=False;
```

```
dxdateedit2.Visible:=False;
```

```
button1.Visible:=False;
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select * from cdkiralama where alistarih=#'+formatdatetime('mm dd  
yyyy',Date)+'#';
```

```
adoquery1.Open;
```

```
end;
```

```
end.
```

## 4.9.FORM OF EDIT PRODUCT CODES

unit duzelt;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, dxCntner, dxEditor, dxExEdtr, dxEdLib, StdCtrls, ExtCtrls, DB,  
ADODB, ExtDlgs, Menus;

type

TForm8 = class(TForm)

Image1: TImage;

Label1: TLabel;

Label3: TLabel;

Label7: TLabel;

Label11: TLabel;

Label2: TLabel;

Label5: TLabel;

Label6: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Label16: TLabel;

Image2: TImage;

Label17: TLabel;

Label18: TLabel;

Label12: TLabel;

Label4: TLabel;

edit11: TEdit;

Edit3: TEdit;

```
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Edit7: TEdit;
ComboBox1: TComboBox;
Edit8: TEdit;
Edit10: TEdit;
Edit12: TEdit;
Edit13: TEdit;
Edit14: TEdit;
Edit15: TEdit;
Button4: TButton;
ComboBox2: TComboBox;
ComboBox3: TComboBox;
Edit16: TEdit;
Edit18: TEdit;
Edit19: TEdit;
Edit2: TdxCurrencyEdit;
Edit9: TEdit;
Edit17: TdxCurrencyEdit;
ComboBox4: TComboBox;
Button1: TButton;
Button2: TButton;
ADOQuery1: TADOQuery;
Button5: TButton;
OpenPictureDialog2: TOpenPictureDialog;
Label19: TLabel;
Button3: TButton;
Edit1: TEdit;
procedure ComboBox4Change(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
```



```
procedure FormKeyDown(Sender: TObject; var Key: Word;
```

```
Shift: TShiftState);
```

```
procedure ComboBox4Enter(Sender: TObject);
```

```
procedure Image2DbClick(Sender: TObject);
```

```
procedure Button5Click(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
Form8: TForm8;
```

```
implementation
```

```
{ $R *.dfm }
```

```
uses rm,help;
```

```
procedure TForm8.ComboBox4Change(Sender: TObject);
```

```
begin
```

```
label19.Caption:="";
```

```
Image2.Picture:=nil;
```

```
adoquery1.Close;
```

```
adoquery1.SQL.Clear;
```

```
adoquery1.SQL.Text:='select *from cdcenter where filminadi="'+comboBox4.Text+"'";
```

```
adoquery1.Open;
```

```
edit11.Text:=adoquery1.fieldbyname('yonetmen').AsString;
```

```
edit3.Text:=adoquery1.fieldbyname('oyuncular1').AsString;
```

```
edit4.Text:=adoquery1.fieldbyname('oyuncular2').AsString;
```

```
edit5.Text:=adoquery1.fieldbyname('oyuncular3').AsString;
```

```
edit6.Text:=adoquery1.fieldbyname('oyuncular4').AsString;
```

```

edit7.Text:=adoquery1.fieldbyname('oyuncular5').AsString;
ComboBox1.Text:=adoquery1.fieldbyname('filmturu').AsString;
ComboBox2.Text:=adoquery1.fieldbyname('dil').AsString;
edit8.Text:=adoquery1.fieldbyname('websitesi').AsString;
edit10.Text:=adoquery1.fieldbyname('senaryo').AsString;
edit16.Text:=adoquery1.fieldbyname('senaryo1').AsString;
edit18.Text:=adoquery1.fieldbyname('senaryo2').AsString;
edit9.Text:=adoquery1.fieldbyname('rafkodu').AsString;
edit2.Text:=adoquery1.fieldbyname('suresi').AsString;
edit12.Text:=adoquery1.fieldbyname('goruntuyonetmenı').AsString;
edit19.Text:=adoquery1.fieldbyname('goruntuyonetmenı1').AsString;
edit13.Text:=adoquery1.fieldbyname('yapım').AsString;
edit14.Text:=adoquery1.fieldbyname('renk').AsString;
edit12.Text:=adoquery1.fieldbyname('muzik').AsString;
ComboBox3.Text:=adoquery1.fieldbyname('cdturu').AsString;
edit17.Text:=adoquery1.fieldbyname('cdadeti').AsString;
if adoquery1.fieldbyname('resim').AsString<>" then
image2.Picture.LoadFromFile(adoquery1.fieldbyname('resim').AsString);
end;

```

```

procedure TForm8.Button3Click(Sender: TObject);
begin
if messagedlg('Are you sure the clear the all
information?',mtinformation,[mbyes,mbno],0)=mryes then
begin
edit2.Text:="";
edit3.Text:="";
edit4.text:="";
edit5.Text:="";
edit6.Text:="";
edit7.Text:="";
edit8.Text:="";
edit9.Text:="";
edit10.Text:="";

```

```
edit11.Text:="";
edit12.Text:="";
edit13.Text:="";
edit14.Text:="";
edit15.Text:="";
edit16.Text:="";
edit18.Text:="";
edit19.Text:="";
combobox1.Text:="";
combobox2.Text:="";
combobox3.text:="";
edit17.Text:="";
Image2.Picture:=nil;
end else begin
exit;
end;
end;
```

```
procedure TForm8.Button2Click(Sender: TObject);
begin
if combobox4.Text="" then
begin
messagedlg('Please Choose Any Movie.',mtinformation,[mbok],0);
combobox4.SetFocus;
exit;
end;
adoquery1.Close;
adoquery1.SQL.clear;
adoquery1.sql.Text:='select * from cdcenter where filminadi="'+combobox4.Text+"'";
adoquery1.Open;
adoquery1.edit;
adoquery1.FieldByName('yonetmen').AsString:=edit11.Text;
adoquery1.fieldbyname('oyuncular1').AsString:=edit3.Text;
```

```

adoquery1.fieldbyname('oyuncular2').AsString:=edit4.Text;
adoquery1.fieldbyname('oyuncular3').AsString:=edit5.Text;
adoquery1.fieldbyname('oyuncular4').AsString:=edit6.Text;
adoquery1.fieldbyname('oyuncular5').AsString:=edit7.Text;
adoquery1.fieldbyname('filmturu').AsString:=ComboBox1.Text;
adoquery1.fieldbyname('dil').AsString:=ComboBox2.Text;
adoquery1.fieldbyname('websitesi').AsString:=edit8.Text;
adoquery1.fieldbyname('senaryo').AsString:=edit10.Text;
adoquery1.fieldbyname('senaryo1').AsString:=edit16.Text;
adoquery1.fieldbyname('senaryo2').AsString:=edit18.Text;
adoquery1.fieldbyname('rafkodu').AsString:=edit9.Text;
adoquery1.fieldbyname('suresi').AsString:=edit2.Text;
adoquery1.fieldbyname('goruntuyonetmen').AsString:=edit12.Text;
adoquery1.fieldbyname('goruntuyonetmen1').AsString:=edit19.Text;
adoquery1.fieldbyname('yapim').AsString:=edit13.Text;
adoquery1.fieldbyname('renk').AsString:=edit14.Text;
adoquery1.fieldbyname('muzik').AsString:=edit12.Text;
adoquery1.fieldbyname('cdturu').AsString:=ComboBox3.Text;
adoquery1.fieldbyname('cdadeti').AsString:=edit17.Text;
if openpicturedialog2.FileName<>" then
adoquery1.FieldByName('resim').AsString:=OpenPictureDialog2.FileName;
adoquery1.Post;
adoquery1.close;
frmhelp.Edit1.Text:='2';
label19.Caption:='Information has been Changed.'
end;

```

```

procedure TForm8.Button1Click(Sender: TObject);
begin
if combobox4.Text="" then
begin
messagedlg('Please Choose Any Movie For Changing Poster',mtinformation,[mbok],0);
combobox4.SetFocus;
exit;

```



```

end else begin
OpenPictureDialog2.Execute;
if OpenPictureDialog2.FileName="" then exit;//eger bu kodu yazmazsak bos oldugunda hata
verir
Image2.Picture.LoadFromFile(OpenPictureDialog2.FileName);
end;
end;

```

```

procedure TForm8.Button4Click(Sender: TObject);
begin
close;
end;

```

```

procedure TForm8.FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
var s:string;
begin
s:=vartostr(key);
case strtoint(s) of
27:close ;
end;
end;

```

```

procedure TForm8.ComboBox4Enter(Sender: TObject);
var a:integer; //Burada ne zaman combo ya girsek databasedeki bilgiyi getirir
begin
combobox4.Clear;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='SELECT cdcenter.filminadi FROM cdcenter '+
' GROUP BY cdcenter.filminadi HAVING (((cdcenter.filminadi) Is Not Null Or
(cdcenter.filminadi)<>"")) '+
' ORDER BY cdcenter.filminadi';
adoquery1.Open;

```

```

adoquery1.First;
for a:=0 to adoquery1.RecordCount-1 do
begin
combobox4.Items.Add(adoquery1.fieldbyname('filminadi').AsString);
adoquery1.Next;
end;
end;

```

```

procedure TForm8.Image2DblClick(Sender: TObject);
begin //dbclick resimsil
end;

```

```

procedure TForm8.Button5Click(Sender: TObject);
begin
if combobox4.Text="" then
begin
messagedlg('Please Choose the Movie',mtinformation,[mbok],0);
combobox4.SetFocus;
exit;
end else begin
if edit11.Text="" then
begin
messagedlg('There is no record for that name "' + combobox4.Text +
"',mtinformation,[mbok],0);
edit2.Text:="";
edit3.Text:="";
edit4.text:="";
edit5.Text:="";
edit6.Text:="";
edit7.Text:="";
edit8.Text:="";
edit9.Text:="";
edit10.Text:="";
edit11.Text:="";

```

```
edit12.Text:="";
edit13.Text:="";
edit14.Text:="";
edit15.Text:="";
edit16.Text:="";
edit18.Text:="";
edit19.Text:="";
combobox1.Text:="";
combobox2.Text:="";
combobox4.text:="";
edit17.Text:="";
Image2.Picture:=nil;
exit;
end else begin
if messagedlg('Do you want to delete
combobox4.Text+'?',mtinformation,[mbytes,mbno],0)=mryes then
begin
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='delete from cdcenter where Filminadi="'+combobox4.Text+'";
adoquery1.ExecSQL;
adoquery1.Close;
edit2.Text:="";
edit3.Text:="";
edit4.text:="";
edit5.Text:="";
edit6.Text:="";
edit7.Text:="";
edit8.Text:="";
edit9.Text:="";
edit10.Text:="";
edit11.Text:="";
edit12.Text:="";
edit13.Text:="";
edit14.Text:="";
```

```
edit15.Text:="";
edit16.Text:="";
edit18.Text:="";
edit19.Text:="";
combobox1.Text:="";
combobox2.Text:="";
combobox3.text:="";
combobox4.Text:="";
edit17.Text:="";
Image2.Picture:=nil;
end else begin
exit;
end;
end;
end;
end;
end.
```



#### 4.10.FORM OF CUSTOMER ANALYSIS CODES

unit musterianalizi;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, dxCntner, dxTL, dxDBCtrl, dxDBGrid, StdCtrls, ExtCtrls,  
dxEditor, dxExEdtr, dxEdLib, DB, ADODB, ExtDlgs;

type

```
TForm18 = class(TForm)
  dxDBGrid1: TdxDBGrid;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  Edit2: TEdit;
  Edit3: TEdit;
  Edit4: TEdit;
  Label6: TLabel;
  Edit5: TEdit;
  dxDateEdit1: TdxDateEdit;
  Label7: TLabel;
  Image1: TImage;
  Button2: TButton;
  ADOQuery1: TADOQuery;
  DataSource1: TDataSource;
  dxDBGrid1Kimlik: TdxDBGridMaskColumn;
  dxDBGrid1sim: TdxDBGridColumn;
  dxDBGrid1Soyisim: TdxDBGridColumn;
  dxDBGrid1Email: TdxDBGridColumn;
  Edit6: TEdit;
```

```

Label9: TLabel;
Edit8: TEdit;
Button4: TButton;
OpenPictureDialog1: TOpenPictureDialog;
Label1: TLabel;
Edit1: TEdit;
Button1: TButton;
ADOQuery2: TADOQuery;
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure FormShow(Sender: TObject);
procedure dxDBGrid1DbClick(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Image1DbClick(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form18: TForm18;

implementation

{$R *.dfm}
uses rm;
procedure TForm18.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var a:string;

```

```
begin
a:=vartostr(key);
case strtoint(a) of
27:close;
end;
end;
```

```
procedure TForm18.FormShow(Sender: TObject);
begin
dxDateEdit1.Date:=date;
end;
```

```
procedure TForm18.dxDBGrid1DbClick(Sender: TObject);
begin
adoquery1.Open;
edit6.Text:=adoquery1.fieldbyname('Kimlik').AsString;
edit2.Text:=adoquery1.fieldbyname('İsim').AsString;
edit3.Text:=adoquery1.fieldbyname('Soyisim').AsString;
edit4.Text:=adoquery1.fieldbyname('Telefon').asString;
edit8.text:=adoquery1.fieldbyname('Email').AsString;
edit5.Text:=adoquery1.fieldbyname('Adres').AsString;
dxDateEdit1.Text:=adoquery1.fieldbyname('Tarih').asString;
if adoquery1.fieldbyname('Resim').AsString<>"" then
begin
image1.Picture.LoadFromFile(adoquery1.fieldbyname('Resim').AsString);
end else begin
Image1.Picture:=nil;
end;
button4.Visible:=True;
button4.Caption:='Change Photo';
end;
```

```
procedure TForm18.Button2Click(Sender: TObject);
begin
```

```

adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from hakan where Kimlik='+edit6.Text+";
adoquery1.Open;
adoquery1.Edit;
adoquery1.FieldName('Isim').AsString:=edit2.Text;
adoquery1.FieldName('Soyisim').AsString:=edit3.Text;
adoquery1.FieldName('Telefon').AsString:=edit4.Text;
adoquery1.FieldName('Tarih').AsString:=dxDateEdit1.Text;
adoquery1.FieldName('Email').AsString:=edit8.Text;
adoquery1.FieldName('Adres').AsString:=edit5.Text;
adoquery1.FieldName('Resim').AsString:=OpenPictureDialog1.FileName;
adoquery1.Post;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from hakan';
adoquery1.Open;
edit1.Text:=";
end;

```

```

procedure TForm18.Image1DbClick(Sender: TObject);
begin
OpenPictureDialog1.FileName:=";
Image1.Picture:=nil;
Button4.Caption:='Add Photo';
end;

```

```

procedure TForm18.Button4Click(Sender: TObject);
begin
OpenPictureDialog1.Execute;
if OpenPictureDialog1.FileName="" then exit;
Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;

```



```

procedure TForm18.Edit1Change(Sender: TObject);
var tt:string;
begin
tt:="";
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select * from hakan where Email like'+tt+edit1.Text+'%'+tt;
adoquery1.Open; //Show Info to Grid According to E-Mail Address
end;

```

```

procedure TForm18.Button3Click(Sender: TObject);
begin
edit1.Text:="";
end;

```

```

procedure TForm18.Button1Click(Sender: TObject);
begin
if edit6.Text="" then
begin
messagedlg('Please Choose the Member Name',mtinformation,[mbok],0);
edit2.Text:="";
edit3.Text:="";
edit4.text:="";
edit5.Text:="";
edit8.Text:="";
edit6.SetFocus;
exit;
end else begin
if messagedlg('Do you want to delete '+edit2.Text+'
'+edit3.text+'?',mtinformation,[mbyes,mbno],0)=mryes then
begin
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='delete *from hakan where Kimlik='+edit6.Text+';
adoquery1.ExecSQL;

```

```
adoquery1.Close;
edit2.Text;
edit3.Text:="";
edit4.text:="";
edit5.Text:="";
edit8.Text:="";
Image1.Picture:=nil;
adoquery1.Close;
adoquery1.SQL.Clear;
adoquery1.SQL.Text:='select *from hakan';
adoquery1.Open;
end else begin
exit;
end;
end;
end;
end.
```