NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

STOCK CONTROL MANAGEMENT

Graduation Project
COM- 400

Student: Cemal Şirin (970348)

Supervisor: Mr. Ümit İlhan

Nicosia-2002

ACKNOWLEDGEMENTS

First of all, I would like to say how grateful I am to my supervisor, Mr.Umit Ilhan, friends, parents and brother.

I would like to thank my supervisor Mr.Umit Ilhan. Under his guidance, I successfully overcome many difficulties and learn a lot about Delphi Programming language .I asked him many questions in Delphi, he explained my questions patiently.

I would like to express my gratitude to Vice-President Prof. Dr. Şenol Bektaş for him because he helped to me at each stage of my Undergraduate Education in Near East University.

I also wish to thank my advisor Mr.Tayseer Alshanableh at my Undergraduate Education for his invaluable advices, for his help and for his patience also for his support.

Finally, I want to thank my family, especially my parents without their endless support, I could never have prepared this thesis without the encouragement and support of my parents,Burcu Baturalp and Mehmet Celik.

ABSTRACT

Back in 1994 or so, Borland began working on a RAD tool that it code-named Delphi. When it was decided that the component model architecture was the best way to implement RAD, it was then necessary to settle on the programming language that would be the heart of the system.

Now a days computer system are everywhere.We are using them nearly for every purpose. For example we are using it for preparing assignments, works etc.

This project has firm definitions, stock cards, and entry and exit invoice and stock control.

TABLE OF CONTENTS

INTRODUCTION

Delphi is Borland's best-selling rapid application development (RAD) product for writing Windows applications. With Delphi, you can write Windows programs more quickly and more easily than was ever possible before. You can create Win32 console applications or Win32 graphical user interface (GUI) programs. When creating Win32 GUI applications with Delphi, you have all the power of a true compiled programming language (Object Pascal) wrapped up in a RAD environment.

The Thesis consists of the Introduction, three chapters, and conclusion.

The Chapter-1 presents What is the Delphi?, Graphic Display, Setting Component Properties, Types Of Databases and chapter concluded with a brief information about Modifying Data.

Chapter-2 presents the informations about Data Base, Copying from Paradox to dBASE tables, Create Table, stock control database tables.

Chapter-3 presents the stock control program of form`s codes.

CHAPTER 1. AN OVERVIEW OF BORLAND DELPHI

1.1.WHAT IS DELPHI?

Delphi is Borland's best-selling rapid application development (RAD) product for writing Windows applications. With Delphi, you can write Windows programs more quickly and more easily than was ever possible before. You can create Win32 console applications or Win32 graphical user interface (GUI) programs. When creating Win32 GUI applications with Delphi, you have all the power of a true compiled programming language (Object Pascal) wrapped up in a RAD environment. What this means is that you can create the user interface to a program (the *user interface* means the menus, dialog boxes, main window, and so on) using drag-and-drop techniques for true rapid application development. You can also drop ActiveX controls on forms to create specialized programs such as Web browsers in a matter of minutes. Delphi gives you all this, and at virtually no cost: You don't sacrifice program execution speed because Delphi generates fast compiled code.

1.1.1 Object Pascal and the VCL

Back in 1994 or so, Borland began working on a RAD tool that it code-named Delphi. When it was decided that the component model architecture was the best way to implement RAD, it was then necessary to settle on the programming language that would be the heart of the system.

Object Pascal, a set of object-oriented extensions to standard Pascal, is the language of Delphi. The Visual Component Library (VCL) is a hierarchy of classes—written in Object Pascal and tied to the Delphi IDE—that allows you to develop applications quickly. Using Delphi's Component palette and Object Inspector, you can place VCL components on forms and manipulate their properties without writing code. All VCL objects descend from *TObject*, an abstract class whose methods encapsulate fundamental behavior like construction, destruction, and message handling. *Tobject* is the immediate ancestor of many simple classes. *Components* in the VCL descend from the abstract class *TComponent*. Components are objects that you can manipulate on forms at design time. Visual components—that is, components like *TForm* and *TSpeedButton* that appear on the screen at runtime—are called *controls*, and they

descend from *TControl*. Despite its name, the VCL consists mostly of nonvisual objects. The Delphi IDE allows you to add many nonvisual components to your programs by dropping them onto forms. For example, if you were writing an application that connects to a database, you might place a *TDataSource* component on a form. Although *TDataSource* is nonvisual, it is represented on the form by an icon (which doesn't appear at runtime). You can manipulate the properties and events of *TDataSource* in the Object Inspector just as you would those of a visual control. When you write classes of your own in Object Pascal, they should descend from *TObject*. By deriving new classes from the VCL's base class (or one of its descendants), you provide your classes with essential functionality and ensure that they work with the VCL.

1.1.2 The Delphi Workspace

The main part of the Delphi IDE is the workspace. The workspace initially displays the Form Designer. It should come as no surprise that the Form Designer enables you to create forms. In Delphi, a *form* represents a window in your program. The form might be the program's main window, a dialog box, or any other type of window. You use the Form Designer to place, move, and size components as part of the form creation process.

Hiding behind the Form Designer is the Code Editor. The Code Editor is where you type code when writing your programs. The Object Inspector, Form Designer, Code Editor, and Component palette work interactively as you build applications.

1.1.3 What is an Object?

An object, or *class*, is a data type that encapsulates *data* and *operations on data* in a single unit. Before object-oriented programming, data and operations (functions) were treated as separate elements. You can begin to understand objects if you understand Object Pascal *records*. Records (analogous to *structures* in C) are made of up fields that contain data, where each field has its own type. Records make it easy to refer to a collection of varied data elements. Objects are also collections of data elements. But objects—unlike records—contain procedures and functions that operate on their data. These procedures and functions are called *methods*. An object's data elements are accessed through *properties*. The properties of Delphi objects have values that you can change at design time without writing code. If you want a property value to

change at runtime, you need to write only a small amount of code. The combination of data and functionality in a single unit is called *encapsulation*. In addition to encapsulation, object-oriented programming is characterized by *inheritance* and *polymorphism*. Inheritance means that objects derive functionality from other objects (called *ancestors*); objects can modify their inherited behavior. Polymorphism means that different objects derived from the same ancestor support the same method and property interfaces, which often can be called interchangeably.

1.1.4 Private, Protected, Public, and Published Declarations

When you declare a field, property, or method, the new member has a *visibility* indicated by one of the keywords private, protected, public, or published. The visibility of a member determines its accessibility to other objects and units.

- A private member is accessible only within the unit where it is declared. Private members are often used within a class to implement other (public or published) methods and properties.
- A protected member is accessible within the unit where its class is declared and within any descendant class, regardless of the descendant class's unit.
- A public member is accessible from wherever the object it belongs to is accessible—that is, from the unit where the class is declared and from any unit that uses that unit.
- A published member has the same visibility as a public member, but the compiler generates runtime type information for published members. Published properties appear in the Object Inspector at design time.

1.2.USING COMPONENTS

All components share features inherited from *TComponent*. By placing components on forms, you build the interface and functionality of your application. The standard components included with Delphi are sufficient for most application development, but you can extend the VCL by creating components of your own.

1.2.1 Delphi's standard components

The Component palette contains a selection of components that handle a wide variety of programming tasks. You can add, remove, and rearrange components on the

palette, and you can create component *templates* and *frames* that group several components. The components on the palette are arranged in pages according to their purpose and functionality. Which pages appear in the default configuration depends on the version of Delphi you are running.

Figure 1.1 Components of menu

Page name	**Contents**
Standard	Standard Windows controls, menus
Additional	Additional controls
Win32	Windows 9x/NT 4.0 common controls
System	Components and controls for system-level access, including timers, multimedia, and DDE
Internet	Components for internet communication protocols and Web applications
Data Access	Nonvisual components for accessing databases tables, queries, and reports
Data Controls	Visual, data-aware controls
Decision Cube	Controls that let you summarize information from databases and view it from a variety of perspectives
QReport Quick	Report components for creating embedded reports
Dialogs	Windows common dialog boxes
Win 3.1	Components for compatibility with Delphi 1.0 projects
Samples	Sample custom components
ActiveX	Sample ActiveX controls
Midas	Components used for creating multi-tiered database applications

1.2.2.Properties common to visual components

All visual components (descendants of *TControl*) share certain properties including

- Position and size properties
- Display properties

- Parent properties
- Navigation properties
- Drag-and-drop properties
- Drag-and-dock properties

While these properties are inherited from *TControl*, they are published—and hence appear in the Object Inspector—only for components to which they are applicable. For example, *TImage* does not publish the *Color* property, since its color is determined by the graphic it displays.

1.2.3.Position and Size Properties

Four properties define the position and size of a control on a form:

- *Height* sets the vertical size
- *Width* sets the horizontal size
- *Top* positions the top edge
- *Left* positions the left edge

These properties aren't accessible in nonvisual components, but Delphi does keep track of where you place the component icons on your forms. Most of the time you'll set and alter these properties by manipulating the control's image on the form or using the Alignment palette. You can, however, alter them at runtime.

1.2.4.Display Properties

Four properties govern the general appearance of a control:

- *BorderStyle* specifies whether a control has a border.
- *Color* changes the background color of a control.
- *Ctrl3D* specifies whether a control will have a 3-D look or a flat border.
- *Font* changes the color, type family, style, or size of text.

1.2.5.Parent Properties

To maintain a consistent appearance across your application, you can make any control look like its container—called its *parent*—by setting the *parent...* properties to *True*. For example, if you place a button on a form and set the button's *ParentFont* property to *True*, changes to the form's *Font* property will automatically propagate to

the button (and to the form's other children). Later, if you change the button's *Font* property, your font choice will take effect and the *ParentFont* property will revert to *False*.

1.2.6.Navigation Properties

Several properties determine how users navigate among the controls in a form:

- *Caption* contains the text string that labels a component. To underline a character in a string, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the control or menu item by pressing Alt while typing the underlined character.
- *TabOrder* indicates the position of the control in its parent's tab order, the order in which controls receive focus when the user presses the Tab key. Initially, tab order is the order in which the components are added to the form, but you can change this by changing *TabOrder*. *TabOrder* is meaningful only if *TabStop* is *True*.
- *TabStop* determines whether the user can tab to a control. If *TabStop* is *True*, the

control is in the tab order.

1.2.7Drag-and-Drop Properties

Two component properties affect drag-and-drop behavior:

- *DragMode* determines how dragging starts. By default, *DragMode* is *dmManual*, and the application must call the *BeginDrag* method to start dragging. When *DragMode* is *dmAutomatic*, dragging starts as soon as the mouse button goes down.
- *DragCursor* determines the shape of the mouse pointer when it is over a draggable component.
- *DockSite*
- *DragKind*
- *DragMode*
- *FloatingDockSiteClass*
- *AutoSize*

1.2.8.Text Controls

Many applications present text to the user or allow the user to enter text. The type of control used for this purpose depends on the size and format of the information.

Used components:

Edit	Edit a single line of text
Memo	Edit multiple lines of text
MaskEdit	Adhere to a particular format, such as a postal code or phone number
RichEdit	Edit multiple lines of text using rich text format

1.2.9.Properties Common to All Text Controls

All of the text controls have these properties in common:

- *Text* determines the text that appears in the edit box or memo control.
- *CharCase* forces the case of the text being entered to lowercase or uppercase.
- *ReadOnly* specifies whether the user is allowed to change the text.
- *MaxLength* limits the number of characters in the control.
- *PasswordChar* hides the text by displaying a single character (usually an asterisk).
- *HideSelection* specifies whether selected text remains highlighted when the control

does not have focus.

Properties shared by memo and rich text controls Memo and rich text controls, which handle multiple lines of text, have several properties in common:

- *Alignment* specifies how text is aligned (left, right, or center) in the component.
- The *Text* property contains the text in the control. Your application can tell if the text changes by checking the *Modified* property.
- *Lines* contains the text as a list of strings.
- *OEMConvert* determines whether the text is temporarily converted from ANSI to OEM as it is entered. This is useful for validating file names.
- *WordWrap* determines whether the text will wrap at the right margin.
- *WantReturns* determines whether the user can insert hard returns in the text.
- *WantTabs* determines whether the user can insert tabs in the text.

- *AutoSelect* determines whether the text is automatically selected (highlighted) when the control becomes active.
- *SelText* contains the currently selected (highlighted) part of the text.
- *SelStart* and *SelLength* indicate the position and length of the selected part of the text.

  At runtime, you can select all the text in the memo with the *SelectAll* method.

1.2.10.Rich Text Controls

The rich edit component is a memo control that supports rich text formatting, printing, searching, and drag-and-drop of text. It allows you to specify font properties, alignment, tabs, indentation, and numbering.

The following components provide additional ways of capturing input.

ScrollBar	Select values on a continuous range
TrackBar	Select values on a continuous range (more visually effective than scroll bar)
UpDown	Select a value from a spinner attached to an edit component
HotKey	Enter Ctrl/ Shift/ Alt keyboard sequences

1.2.11.Buttons and Similar Controls

A side from menus, buttons provide the most common way to invoke a command in an application. Delphi offers several button-like controls:

Button	Present command choices on buttons with text
BitBtn	Present command choices on buttons with text and glyphs
SpeedButton	Create grouped toolbar buttons
CheckBox	Present on/off options
RadioButton	Present a set of mutually exclusive choices
ToolBar	Arrange tool buttons and other controls in rows and automatically adjust their sizes and positions.
CoolBar	Display a collection of windowed controls within movable, resizable bands

1.2.12.Button controls

Users click button controls to initiate actions. Double-clicking a button at design time takes you to the button's *OnClick* event handler in the Code editor.

- Set *Cancel* to *True* if you want the button to trigger its *OnClick* event when the user

presses Esc.

- Set *Default* to *True* if you want the Enter key to trigger the button's *OnClick* event.

1.2.13.Bitmap buttons

A bitmap button *(BitBtn)* is a button control that presents a bitmap image on its face.

- To choose a bitmap for your button, set the *Glyph* property.
- Use *Kind* to automatically configure a button with a glyph and default behavior.
- By default, the glyph is to the left of any text. To move it, use the *Layout* property.
- The glyph and text are automatically centered in the button. To move their position, use the *Margin* property. *Margin* determines the number of pixels between the edge of the image and the edge of the button.
- By default, the image and the text are separated by 4 pixels. Use *Spacing* to increase or decrease the distance.
- Bitmap buttons can have 3 states: up, down, and held down. Set the *NumGlyphs* property to 3 to show a different bitmap for each state.

1.2.14.Speed Buttons

Speed buttons, which usually have images on their faces, can function in groups. They are commonly used with panels to create toolbars.

- To make speed buttons act as a group, give the *GroupIndex* property of all the buttons the same nonzero value.
- By default, speed buttons appear in an up (unselected) state. To initially display a speed button as selected, set the *Down* property to *True*.
- If *AllowAllUp* is *True*, all of the speed buttons in a group can be unselected. Set *AllowAllUp* to *False* if you want a group of buttons to act like a radio group.

1.2.15.Check Boxes

A check box is a toggle that presents the user with two, or sometimes three, choices.

- Set *Checked* to *True* to make the box appear checked by default.
- Set *AllowGrayed* to *True* to give the check box three possible states: checked, unchecked, and grayed.
- The *State* property indicates whether the check box is checked *(cbChecked)*, unchecked *(cbUnchecked)*, or grayed *(cbGrayed)*.

1.2.16.Radio Buttons

Radio buttons present a set of mutually exclusive choices. You can use individual radio buttons or the *radio group* component, which arranges groups of radio buttons automatically.

1.2.17.Toolbars

Toolbars provide an easy way to arrange and manage visual controls. You can create a toolbar out of a panel component and speed buttons, or you can use the *ToolBar* component, then right-click and choose New Button to add buttons to the toolbar. The *ToolBar* component has several advantages: Buttons on a toolbar automatically maintain uniform dimensions and spacing; other controls maintain their relative position and height; controls can automatically wrap around to start a new row when they do not fit horizontally; and the *ToolBar* offers display options like transparency, pop-up borders, and spaces and dividers to group controls.

1.2.18.Handling Lists

Lists present the user with a collection of items to select from. Several components display lists: Used the nonvisual *TStringList* and *TImageList* components to manage sets of strings and images.

Used components:

ListBox	A list of text strings
CheckListBox	A list with a check box in front of each item
ComboBox	An edit box with a scrollable drop-down list

TreeView	A hierarchical list
ListView	A list of (draggable) items with optional icons, columns, and headings
DateTimePicker	A list box for entering dates or times
MonthCalendar	A calendar for selecting dates

1.2.19.List Boxes and Check-list Boxes

List boxes and check-list boxes display lists from which users can select items.

- *Items* uses a *TStringList* object to fill the control with values.
- *ItemIndex* indicates which item in the list is selected.
- *MultiSelect* specifies whether a user can select more than one item at a time.
- *Sorted* determines whether the list is arranged alphabetically.
- *Columns* specifies the number of columns in the list control.
- *IntegralHeight* specifies whether the list box shows only entries that fit completely

in the vertical space.

- *ItemHeight* specifies the height of each item in pixels. The *Style* property can cause *ItemHeight* to be ignored.
- The *Style* property determines how a list control displays its items. By default, items are displayed as strings. By changing the value of *Style*, you can create *owner-draw* list boxes that display items graphically or in varying heights.

1.2.20.Combo Boxes

A combo box combines an edit box with a scrollable list. When users enter data into the control—by typing or selecting from the list—the value of the *Text* property changes. Use the *Style* property to select the type of combo box you need:

- Use *csDropdown* if you want an edit box with a drop-down list. Use *csDropDownList* to make the edit box read-only (forcing users to choose from the list). Set the *DropDownCount* property to change the number of items displayed in the list.
- Use *csSimple* to create a combo box with a fixed list that does not close. Be sure to

resize the combo box so that the list items are displayed.

- Use *csOwnerDrawFixed* or *csOwnerDrawVariable* to create *owner-draw* combo boxes

that display items graphically or in varying heights.

1.2.21.Grouping Components

A graphical interface is easier to use when related controls and information are presented in groups. Delphi provides several components for grouping components:

Used components:

GroupBox A standard group box with a title

RadioGroup A simple group of radio buttons

Panel A more visually flexible group of controls

ScrollBox A scrollable region containing controls

TabControl A set of mutually exclusive notebook-style tabs

PageControl A set of mutually exclusive notebook-style tabs with corresponding pages, each of which may contain other controls HeaderControl Resizable column headers.

1.2.22.Group Boxes and Radio Groups

A group box is a standard Windows component that arranges related controls on a form. The most commonly grouped controls are radio buttons. After placing a group box on a form, select components from the Component palette and place them in the group box. The *Caption* property contains text that labels the group box at runtime. The radio group component simplifies the task of assembling radio buttons and making them work together. To add radio buttons to a radio group, edit the *Items* property in the Object Inspector; each string in *Items* makes a radio button appear in the group box with the string as its caption. The value of the *ItemIndex* property determines which radio button is currently selected. Display the radio buttons in a single column or in multiple columns by setting the value of the *Columns* property. To respace the buttons, resize the radio group component.

1.2.23.Panels

The panel component provides a generic container for other controls. Panels can be aligned with the form to maintain the same relative position when the form is

resized. The *BorderWidth* property determines the width, in pixels, of the border around a panel.

1.2.24.Visual Feedback

There are many ways to provide users with information about the state of an application. For example, some components—including *TForm*—have a *Caption* property that can be set at runtime. You can also create dialog boxes to display messages. In addition, the following components are especially useful for providing visual feedback at runtime.

Used component:

Label and StaticText	Display non-editable text
StatusBar	Display a status region (usually at the bottom of a window)
ProgressBar	Show the amount of work completed for a particular task
Hint and *ShowHint*	Activate fly-by or "tool-tip" help
HelpContext and *HelpFile*	Link context-sensitive online Help

1.2.25.Labels and Static-Text Components

Labels display text and are usually placed next to other controls. The standard label component, *TLabel*, is a non windowed control, so it cannot receive focus; when you need a label with a window handle, use *TStaticText* instead. Label properties include the following:

- *Caption* contains the text string for the label.
- *FocusControl* links the label to another control on the form. If *Caption* includes an accelerator key, the control specified by *FocusControl* receives focus when the user presses the accelerator key.
- *ShowAccelChar* determines whether the label can display an underlined accelerator character. If *ShowAccelChar* is *True*, any character preceded by an ampersand (&) appears underlined and enables an accelerator key.
- *Transparent* determines whether items under the label (such as graphics) are visible.

Help and hint properties

Most visual controls can display context-sensitive Help as well as fly-by hints at runtime. The *HelpContext* and *HelpFile* properties establish a Help context number and Help file for the control. The *Hint* property contains the text string that appears when the user moves the mouse pointer over a control or menu item. To enable hints, set *ShowHint* to *True*; setting *ParentShowHint* to *True* causes the control's *ShowHint* property to have the same value as its parent's.

1.3.GRIDS

Grids display information in rows and columns. If you're writing a database application, use the *TDBGrid* or *TDBCtrlGrid*. Otherwise, use a standard draw grid or string grid.

1.4.GRAPHIC DISPLAY

The following components make it easy to incorporate graphics into an application.

Used components to display:

Image	Graphics files
Shape	Geometric shapes
Bevel	3D lines and frames
PaintBox	Graphics drawn by your program at runtime
Animate	AVI files

1.4.1.Images

The image component displays a graphical image, like a bitmap, icon, or metafile. The *Picture* property determines the graphic to be displayed. Use *Center*, *AutoSize*, *Stretch*, and *Transparent* to set display options.

1.4.2.Shapes

The shape component displays a geometric shape. It is a non windowed control and cannot receive user input. The *Shape* property determines which shape the control assumes. To change the shape's color or add a pattern, use the *Brush* property, which holds a *TBrush* object. How the shape is painted depends on the *Color* and *Style* properties of *TBrush*.

1.5 Setting Component Properties

Published properties can be set at design time in the Object Inspector and, in some cases, with special property editors. To set properties at runtime, assign them new values in your application source code

1.5.1.Using the Object Inspector

When you select a component on a form, the Object Inspector displays its published properties and (when appropriate) allows you to edit them. Use the Tab key to toggle between the Value column and the Property column. When the cursor is in the Property column, you can navigate to any property by typing the first letters of its name. For properties of Boolean or enumerated types, you can choose values from a drop-down list or toggle their settings by double-clicking in Value column. If a plus (+) symbol appears next to a property name, clicking the plus symbol displays a list of sub values for the property. By default, properties in the Legacy category are not shown; to change the display filters, right-click in the Object Inspector and choose View. When more than one component is selected, the Object Inspector displays all properties—except *Name*—that are shared by the selected components. If the value for a shared property differs among the selected components, the Object Inspector displays either the default value or the value from the first component selected. When you change a shared property, the change applies to all selected components.

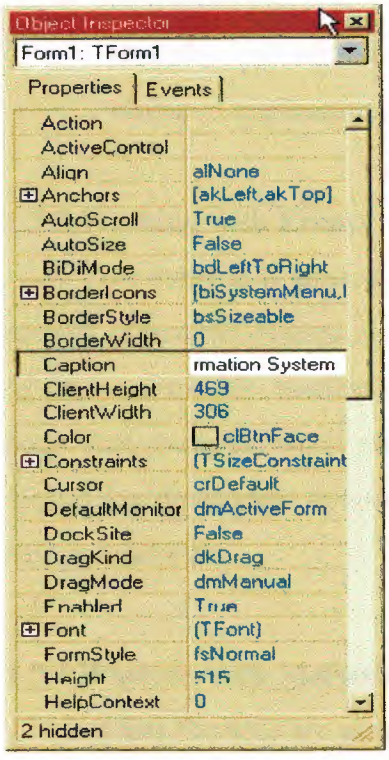

Figure 2.1 Object Inspector

1.5.2.Using Property Editors

Properties give the application developer the illusion of setting or reading the value of a variable, while allowing the component writer to hide the underlying data

structure or to implement special processing when the value is accessed. There are several advantages to using properties:

- Properties are available at design time. The application developer can set or change initial values of properties without having to write code.
- Properties can check values or formats as the application developer assigns them. Validating input at design time prevents errors.

- The component can construct appropriate values on demand. Perhaps the most common type of error programmers make is to reference a variable that has not been initialized. By representing data with a property, you can ensure that a value is always available on demand.
- Properties allow you to hide data under a simple, consistent interface. You can alter the way information is structured in a property without making the change visible to application developers.

1.5.3. Working with Events

An event is a special property that invokes code in response to input or other activity at runtime. Events give the application developer a way to attach specific blocks of code to specific runtime occurrences, such as mouse actions and keystrokes. The code that executes when an event occurs is called an *event handler*. Events allow application developers to specify responses to different kinds of input without defining new components.

1.5.4. Working with Methods

Class methods are procedures and functions that operate on a class rather than on specific instances of the class. For example, every component's constructor method *(Create)* is a class method. Component methods are procedures and functions that operate on the component instances themselves. Application developers use methods to direct a component to perform a specific action or return a value not contained by any property. Because they require execution of code, methods can be called only at runtime. Methods are useful for several reasons:

- Methods encapsulate the functionality of a component in the same object where

the data resides.

- Methods can hide complicated procedures under a simple, consistent interface. An

application developer can call a component's *AlignControls* method without knowing how the method works or how it differs from the *AlignControls* method in another component.

- Methods allow updating of several properties with a single call.

Table 1.1. Data types used in Delphi 5 (32-bit programs).

Data Type	*Size in Bytes*	*Possible Range of Values*
ShortInt	1	-128 to 127
Byte	1	0 to 255
Word	2	0 to 65,535
LongInt	4	-2,147,483,648 to 2,147,483,647
Integer	4	Same as LongInt
Real	8	$5.0 ¥ 10^{-324}$ to $1.7 ¥ 10^{308}$ (same as Double)
Currency	8	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Boolean	1	True or False
Variant	16	Varies

Table 1.2. String manipulation functions and procedures.

Name	*Description*
Copy	Returns a sub-string within a string.
Delete	Deletes part of a string.
Format	Formats and returns a string based on the format string and arguments passed.

Insert	Inserts text into a string.
IntToStr	Converts an integer value to a string.
Length	Returns the length of a string.
LowerCase	Converts a string to lowercase.
StrToInt	Converts a string to an integer. If the string cannot be converted, an exception is thrown.
StrToXXX	Additional conversion functions that convert a string to a floating point, Currency, Date, or Time value.
UpperCase	Converts a string to uppercase.

1.6.TYPES OF DATABASES

You can connect to different types of databases, depending on what drivers you have installed with the BDE or ADO. These drivers may connect your application to local databases such as Paradox, Access, and dBASE or remote database servers like Microsoft SQL Server, Oracle, and Informix. Similarly, the InterBase Express components can access either a local or remote version of InterBase..

1.6.1.Local Databases

Local databases reside on your local drive or on a local area network. They have proprietary APIs for accessing the data. Often, they are dedicated to a single system. When they are shared by several users, they use file-based locking mechanisms. Because of this, they are sometimes called file-based databases. Local databases can be faster than remote database servers because they often reside on the same system as the database application. Because they are file-based, local databases are more limited than remote database servers in the amount of data they can store. Therefore, in deciding whether to use a local database, you must consider how much data the tables are expected to hold. Applications that use local databases are called single-tiered applications because the application and the database share a single file system. Examples of local databases include Paradox, dBASE, FoxPro, and Access.

1.6.2.Connecting to Databases

The Borland Database Engine includes drivers to connect to different databases. The Standard version of Delphi includes only the drivers for local databases: Paradox, dBASE, FoxPro, and Access. With the Professional version, you also get an ODBC adapter that allows the BDE to use ODBC drivers. By supplying an ODBC driver, your application can use any ODBC-compliant database. Some versions also include drivers for remote database servers. Use the drivers installed with SQL Links to communicate with remote database servers such as InterBase, Oracle, Sybase, Informix, Microsoft SQL server, and DB2.

1.6.3.Understanding Datasets

In Delphi, the fundamental unit for accessing data is the dataset family of objects. Your application uses datasets for all database access. Generally, a dataset object represents a specific table belonging to a database, or it represents a query or stored procedure that accesses a database. All dataset objects that you will use in your database applications descend from the virtualized dataset object, *TDataSet*, and they inherit data fields, properties, events, and methods from *TDataSet*. This chapter describes the functionality of *TDataSet* that is inherited by the dataset objects you will use in your database applications. You need to understand this shared functionality to use any dataset object. Figure 1.1 illustrates the hierarchical relationship of all the dataset components:

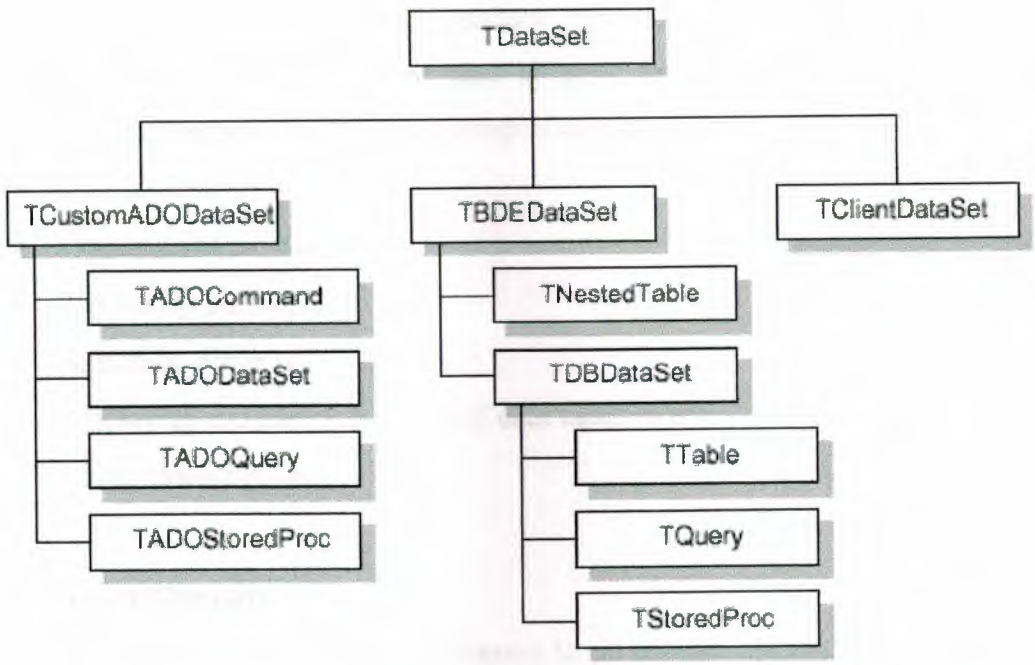

Figure 1.3 Delphi Dataset hierarchies

1.6.4.What is TDataSet?

TDataSet is the ancestor for all dataset objects you use in your applications. It defines a set of data fields, properties, events, and methods shared by all dataset objects. *TDataSet* is a virtualized dataset, meaning that many of its properties and methods are virtual or abstract. A *virtual method* is a function or procedure declaration where the implementation of that method can be (and usually is) overridden in descendant objects. An *abstract method* is a function or procedure declaration without an actual implementation. The declaration is a prototype that describes the method (and its parameters and return type, if any) that must be implemented in all descendant dataset objects, but that might be implemented differently by each of them. Because *TDataSet* contains abstract methods, you cannot use it directly in an application without generating a runtime error. Instead, you either create instances of *TDataSet*'s descendants, such as *TTable*, *TQuery*, *TStoredProc*, and *TClientDataSet*, and use them in your application, or you derive your own dataset object from *TDataSet* or its descendants and write implementations for all its abstract methods. Nevertheless, *TDataSet* defines much that is common to all dataset objects. For example, *TDataSet* defines the basic structure of all datasets: an array of *Tfield* components that correspond

to actual columns in one or more database tables, lookup fields provided by your application, or calculated fields provided by your application. For more information about *TField* components, the following topics are discussed:

- Types of datasets
- Opening and closing datasets
- Navigating datasets
- Searching datasets
- Displaying and editing a subset of data using filters
- Using dataset events

1.6.5.Types of Datasets

To understand the concepts common to all dataset objects, and to prepare for developing your own custom dataset objects that do not rely on either the Borland Database Engine (BDE) or ActiveX Data Objects (ADO). To develop traditional, two-tier client/server database applications using the Borland Database Engine (BDE),. That section introduces *TBDEDataSet* and *TDBDataSet*, and focuses on the shared features of *TQuery*, *TStoredProc*, and *TTable*, the dataset components used most commonly in all database applications. With some versions of Delphi, you can develop multi-tier database applications using distributed datasets.

1.6.6.Opening and Closing Datasets

To read or write data in a table or through a query, an application must first open a dataset. You can open a dataset in two ways,

- Set the *Active* property of the dataset to *True*, either at design time in the Object Inspector
- Call the *Open* method for the dataset at runtime,

You can close a dataset in two ways,

- Set the *Active* property of the dataset to *False*, either at design time in the Object Inspector,
- Call the *Close* method for the dataset at runtime.

1.6.7.Browsing a Dataset

When an application opens a dataset, the dataset automatically enters *dsBrowse* state. Browsing enables you to view records in a dataset, but you cannot edit records or insert new records. You mainly use *dsBrowse* to scroll from record to record in a dataset. From *dsBrowse* all other dataset states can be set. For example, calling the *Insert* or *Append* methods for a dataset changes its state from *dsBrowse* to *dsInsert* (note that other factors and dataset properties, such as *CanModify*, may prevent this change). Calling *SetKey* to search for records puts a dataset in *dsSetKey* mode. Two methods associated with all datasets can return a dataset to *dsBrowse* state. *Cancel* ends the current edit, insert, or search task, and always returns a dataset to *dsBrowse* state. *Post* attempts to write changes to the database, and if successful, also returns a dataset to *dsBrowse* state. If *Post* fails, the current state remains unchanged. The following diagram illustrates the relationship of *dsBrowse* both to the other dataset modes you can set in your applications, and the methods that set those modes.

Figure 1.2 Relationship of Browse to other dataset states

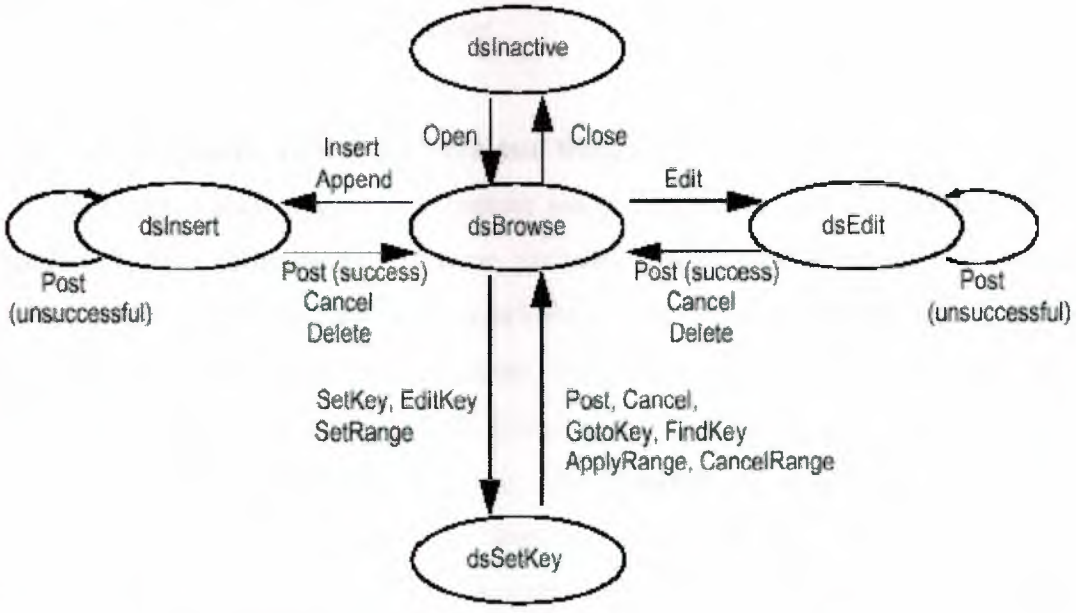

Figure 1.4 Relationship of Browse to other dataset states

1.6.8.Enabling Dataset Editing

A dataset must be in *dsEdit* mode before an application can modify records. In your code you can use the *Edit* method to put a dataset into *dsEdit* mode if the read-only *CanModify* property for the dataset is *True*. *CanModify* is *True* if the database

underlying a dataset permits read and write privileges. On forms in your application, some data-aware controls can automatically put a dataset into *dsEdit* state if:

- The control's *ReadOnly* property is *False* (the default),
- The *AutoEdit* property of the data source for the control is *True*, and
- *CanModify* is *True* for the dataset.

1.6.9.Enabling Insertion of New Records

A dataset must be in *dsInsert* mode before an application can add new records. In your code you can use the *Insert* or *Append* methods to put a dataset into *dsInsert* mode if the read-only *CanModify* property for the dataset is *True*. *CanModify* is *True* if the database underlying a dataset permits read and write privileges. On forms in your application, the data-aware grid and navigator controls can put a dataset into *dsInsert* state if

- The control's *ReadOnly* property is *False* (the default),
- The *AutoEdit* property of the data source for the control is *True*, and
- *CanModify* is *True* for the dataset.

1.6.10.Enabling Index-Based Searches and Ranges on Tables

You can search against any dataset using the *Locate* and *Lookup* methods of *TDataSet*. *TTable* components, however, provide an additional family of *GotoKey* and *FindKey* methods that enable you to search for records based on an index for the table. To use these methods on table components, the component must be in *dsSetKey* mode. *dsSetKey* mode applies only to *TTable* components. You put a dataset into *dsSetKey* mode with the *SetKey* method at runtime. The *GotoKey*, *GotoNearest*, *FindKey*, and *FindNearest* methods, which carry out searches, returns the dataset to *dsBrowse* state upon completion of the search. You can temporarily view and edit a subset of data for any dataset by using filters.

1.6.11.Filtering Records

Helphi puts a dataset into *dsFilter* mode whenever an application calls the dataset's *OnFilterRecord* event handler. This state prevents modifications or additions to the records in a dataset during the filtering process so that the filter request is not

invalidated. When the *OnFilterRecord* handler finishes, the dataset is returned to *dsBrowse* state.

1.6.12.Updating Records

When performing cached update operations, Delphi may put the dataset into *dsNewValue*, *dsOldValue*, or *dsCurValue* states temporarily. These states indicate that the corresponding properties of a field component *(NewValue*, *OldValue*, and *CurValue*, respectively) are being accessed, usually in an *OnUpdateError* event handler. Your applications cannot see or set these states.

1.6.13.Using the Eof and Bof Properties

Two read-only, runtime properties, *Eof* (End-of-file) and *Bof* (Beginning-of-file), are useful for controlling dataset navigation, particularly when you want to iterate through all records in a dataset.

1.6.14.End of File (Eof)

When *Eof* is *True*, it indicates that the cursor is unequivocally at the last row in a dataset. *Eof* is set to *True* when an application

- Opens an empty dataset.
- Calls a dataset's *Last* method.
- Calls a dataset's *Next* method, and the method fails (because the cursor is currently at the last row in the dataset.
- Calls *SetRange* on an empty range or dataset.

Eof is set to *False* in all other cases; you should assume *Eof* is *False* unless one of the conditions above is met *and* you test the property directly. *Eof* is commonly tested in a loop condition to control iterative processing of all records in a dataset. If you open a dataset containing records (or you call *First*) *Eof* is *False*. To iterate through the dataset a record at a time, create a loop that terminates when *Eof* is *True*. Inside the loop, call *Next* for each record in the dataset. *Eof* remains *False* until you call *Next* when the cursor is already on the last record.

1.6.15.Beginning of File (Bof)

When *Bof* is *True*, it indicates that the cursor is unequivocally at the first row in a dataset. *Bof* is set to *True* when an application

Opens a dataset.

- Calls a dataset's *First* method.
- Calls a dataset's *Prior* method, and the method fails (because the cursor is currently at the first row in the dataset.
- Calls *SetRange* on an empty range or dataset.

Bof is set to *False* in all other cases; you should assume *Bof* is *False* unless one of the conditions above is met *and* you test the property directly. Like *Eof*, *Bof* can be in a loop condition to control iterative processing of records in a dataset.

1.6.16.Using Locate

Locate moves the cursor to the first row matching a specified set of search criteria. In its simplest form, you pass *Locate* the name of a column to search, a field value to match, and an options flag specifying whether the search is case-insensitive or if it can use partial-key matching.

1.7.MODIFYING DATA

You can use the following dataset methods to insert, update, and delete data:

1.7.1.Editing Records

A dataset must be in *dsEdit* mode before an application can modify records. In your code you can use the *Edit* method to put a dataset into *dsEdit* mode if the read-only *CanModify* property for the dataset is *True*. *CanModify* is *True* if the table(s) underlying a dataset permits read and write privileges. On forms in your application, some data-aware controls can automatically put a dataset into *dsEdit* state if

- The control's *ReadOnly* property is *False* (the default),
- The *AutoEdit* property of the data source for the control is *True*, and
- *CanModify* is *True* for the dataset.

1.7.2.DataSet Actions

The standard dataset actions are designed to be used with a dataset component target. *TDataSetAction* is the base class for descendants that each override the *ExecuteTarget* and *UpdateTarget* methods to implement navigation and editing of the target. The *TDataSetAction* introduces a *DataSource* property which ensures actions are performed on that dataset. If *DataSource* is nil, the currently focused data-aware control is used.

- *TDataSetAction* ensures that the target is a *TDataSource* class and has an associated data set.
- *TDataSetCancel* cancels the edits to the current record, restores the record display to its condition prior to editing, and turns off Insert and Edit states if they are active.
- *TdataSet.Delete* deletes the current record and makes the next record the current record.
- *TdataSet.Edit* puts the dataset into *Edit* state so that the current record can be modified.
- *TdataSet.First* sets the current record to the first record in the dataset.
- *TdataSet.Insert* inserts a new record before the current record, and sets the dataset into Insert and Edit states.
- *TdataSet.Last* sets the current record to the last record in the dataset.
- *TdataSet.Next* sets the current record to the next record.
- *TdataSet.Post* writes changes in the current record to the dataset.
- *TdataSet.Prior* sets the current record to the previous record.
- *TdataSet.Refresh* refreshes the buffered data in the associated dataset.
- *TDataSet.Append* adds a new, empty record to the end of the dataset.

1.7.3.Adding New Records

A dataset must be in *dsInsert* mode before an application can add new records. In code, you can use the *Insert* or *Append* methods to put a dataset into *dsInsert* mode if the read-only *CanModify* property for the dataset is *True*. *CanModify* is *True* if the database underlying a dataset permits read and write privileges. On forms in your

application, the data-aware grid and navigator controls can put a dataset into *dsInsert* state if

- The control's *ReadOnly* property is *False* (the default), and
- *CanModify* is *True* for the dataset.

1.7.4. Inserting Records

Insert opens a new, empty record before the current record, and makes the empty record the current record so that field values for the record can be entered either by a user or by your application code. When an application calls *Post* (or *ApplyUpdates* when cached updating is enabled), a newly inserted record is written to a database in one of three ways:

- For indexed Paradox and dBASE tables, the record is inserted into the dataset in a

position based on its index.

- For unindexed tables, the record is inserted into the dataset at its current position.
- For SQL databases, the physical location of the insertion is implementation-specific.

If the table is indexed, the index is updated with the new record information.

1.7.5.Appending Records

Append opens a new, empty record at the end of the dataset, and makes the empty record the current one so that field values for the record can be entered either by a user or by your application code. When an application calls *Post* (or *ApplyUpdates* when cached updating is enabled), a newly appended record is written to a database in one of three ways:

- For indexed Paradox and dBASE tables, the record is inserted into the dataset in a

position based on its index.

- For unindexed tables, the record is added to the end of the dataset.

- For SQL databases, the physical location of the append is implementation-specific.

If the table is indexed, the index is updated with the new record information.

1.7.6.Deleting Records

A dataset must be active before an application can delete records. *Delete* deletes the current record from a dataset and puts the dataset in *dsBrowse* mode. The record that followed the deleted record becomes the current record. If cached updates are enabled for a dataset, a deleted record is only removed from the temporary cache buffer until you call *ApplyUpdate*s. If you provide a navigator component on your forms, users can delete the current record by clicking the navigator's Delete button. In code, you must call *Delete* explicitly to remove the current record.

1.7.7.Canceling Changes

An application can undo changes made to the current record at any time, if it has not yet directly or indirectly called *Pos*t. For example, if a dataset is in *dsEdit* mode, and a user has changed the data in one or more fields, the application can return the record back to its original values by calling the *Cancel* method for the dataset. A call to *Cancel* always returns a dataset to *dsBrowse* state. On forms, you can allow users to cancel edit, insert, or append operations by including the Cancel button on a navigator component associated with the dataset, or you can provide code for your own Cancel button on the form.

1.7.8.Using Dataset Events

Datasets have a number of events that enable an application to perform validation, compute totals, and perform other tasks. The events are listed in the following table. For more information about events for the *TDataSet* component, see the online *VCL Reference*.

1.7.9.Event Description

BeforeOpen, AfterOpen Called before/after a dataset is opened.

BeforeClose, AfterClose Called before/after a dataset is closed.

BeforeInsert, AfterInsert Called before/after a dataset enters Insert state.

BeforeEdit, AfterEdit Called before/after a dataset enters Edit state.

BeforePost, AfterPost Called before/after changes to a table are posted.

BeforeCancel, AfterCancel Called before/after the previous state is canceled.

BeforeDelete, AfterDelete Called before/after a record is deleted.

OnNewRecord Called when a new record is created; used to set default values.

OnCalcFields Called when calculated fields are calculated.

CHAPTER 2. DATA BASE

2.1Copying from Paradox to dBASE tables

For general information on copying from a Paradox table to a dBASE table, see Copying to a different table type.

Database Desktop automatically changes field types when you change table types. The following table shows what to expect when you copy from a Paradox table to a dBASE table.

From Paradox type	To dBASE type	Side effects
Alpha	Character	
Number	Number	Assigns size (20) and dec. (4)
Money	Number	Assigns size (20) and dec. (4)
Short	Number	Assigns size (6) and dec. (0)
Long Integer	Number	Assigns size (11) and dec.(0)
BCD	Number	Assigns size (20) and dec. (4)
Date	Date	
Time	Character	Assigns size (8)
Timestamp	Character	Assigns size (30)
Memo Memo		
Formatted memo	Memo	Formatting is lost
Graphic	Binary	
OLE	OLE	
Logical	Logical	
Autoincrement	Number	Assigns size (11) and dec. (0)
Binary	Memo	Data cannot be displayed

Bytes	Memo	Data	cannot	be displayed

If the new dBASE table contains no production index (.MDX file), no float number field type, and no memo field type, Database Desktop creates a dBASE III+ table. If the dBASE table contains an OLE or binary field, Database Desktop creates a dBASE for Windows table. Otherwise, Database Desktop creates a dBASE IV table.

If in the BDE Configuration Utility, the Level parameter for the dBASE driver is set to 4 instead of 5, graphic and OLE Paradox fields fields convert to dBASE memo fields. Also, bytes fields cannot be converted.

2.2CREATE TABLE

CREATE TABLE is supported with the following limitations:

Column definitions based on domains are not supported.

Constraints are limited to PRIMARY KEY for Paradox tables. Constraints are unsupported in dBASE tables.

For example, the following statement creates a Paradox table with a PRIMARY KEY constraint on the LAST_NAME and FIRST_NAME columns:

```
CREATE TABLE "employee.db"
(
LAST_NAME CHAR(20),
FIRST_NAME CHAR(15),
SALARY NUMERIC(10,2),
DEPT_NO SMALLINT,
PRIMARY KEY(LAST_NAME, FIRST_NAME)
)
```

The same statement for a dBASE table should omit the PRIMARY KEY definition:

```
CREATE TABLE "employee.dbf"
(
LAST_NAME CHAR(20),
FIRST_NAME CHAR(15),
SALARY NUMERIC(10,2),
DEPT_NO SMALLINT
```

Creating Paradox and dBASE tables

You create a Paradox or dBASE table using Local SQL by specifying the file extension when naming the table:

".DB" for Paradox tables

".DBF" for dBASE tables

If you omit the file extension for a local table name, the table created is the table type specified in the Default Driver setting in the System page of the BDE Configuration Utility.

Data type mappings for CREATE TABLE

The following table lists SQL syntax for data types used with CREATE TABLE, and describes how those types are mapped to Paradox and dBASE types by the BDE:

SQL Syntax	***BDE Logical***	***Paradox***	***dBASE***
SMALLINT	fldINT16	Short	Number (6,10)
INTEGER	fldINT32	Long Integer	Number (20,4)
DECIMAL(x,y)	fldBCD	BCD	N/A
NUMERIC(x,y)	fldFLOAT	Number	Number (x,y)
FLOAT(x,y)	fldFLOAT	Number	Float (x,y)
CHARACTER(n)	fldZSTRING	Alpha	Character
VARCHAR(n)	fldZSTRING	Alpha	Character
DATE	fldDATE	Date	Date
BOOLEAN	fldBOOL	Logical	Logical
BLOB(n,1)	fldstMEMO	Memo	Memo
BLOB(n,2)	fldstBINARY	Binary	Binary
BLOB(n,3)	fldstFMTMEMO	Formatted memo	N/A
BLOB(n,4)	fldstOLEOBJ	OLE	OLE
BLOB(n,5)	fldstGRAPHIC	Graphic	N/A
TIME	fldTIME	Time	N/A
TIMESTAMP	fldTIMESTAMP	Timestamp	N/A
MONEY	fldFLOAT, fldstMONEY	Money	Number (20,4)
AUTOINC	fldINT32, fldstAUTOINC	Autoincrement	N/A

BYTES(n)	fldBYTES(n)	Bytes	N/A

x = precision (default: specific to driver)

y = scale (default: 0)

n = length in bytes (default: 0)

1-5 = BLOB subtype (default: 1)

2.3 The Stock Control Database Tables

Table1.fatcıkıs.DB

Fatcıkıs	Invoice No	Current No	Current Name	Address	Tax office	T.O. NO	Pre. Date
1	**01**	**A02**	**Egemen**	**Lefkosa**	**Lefkosa**	**444444**	**12.02.2002**
2	**02**	**A03**	**Cemal**	**Luleburgaz**	**Istanbul**	**012**	**17.03.2001**

Sending Date	Total	Tax Total	General Total	Term	Exchange Type	Rate of Exchange	Exchange Amount
13.02.2002	0,00	0,00	0,00	0,00	USD	0,00	0,00
17.03.2001	0,00	0,00	0,00	0,00	USD	0,00	0,00

In this table it is shown that when you make out an invoice for selling a product it provides us to reach the information from the database easily.

Table 2.fatgiris.DB

Fatgiris	Invoice No	Current No	Current Name	Preparation Date	Delivering Date	Total	Tax Total
4	4	A04	**Cemal**	**12.06.2002**	**13.06.2002**	**1.000,00TL**	**150,00TL**

Genel Total	Term	Exchange Type	Rate Of Exchange	Exchange Amount	Term Difference	Address
1.150,00TL	**11**	**USD**	**100,00TL**	**11,50**	**0,00**	**Luleburgaz**

This table shows that the new products stock information that we buy is kept and it provides us to reach the information easily.

Table 3.fcıkısic.DB

Fcikisic	Product Code	Product Name	Invoice No	Amount	Unit	Unit Price	Tax	Total	Profit
3	**5**	**Telefon**	**03**	**10,00**	**20**	**220**	**15,00**	**2.200TL**	**120**

This table shows only the informations about the products that makes exit.

Table 4.fgirisic

Fgirisic	Product Code	Product Name	Invoice No	Amount	Unit	Unit Price	Tax	Total
4	**5**	**Telefon**	**4**	**10,00**	**20**	**100,00**	**15,00**	**1.000,00**

This table is nearly same as the previous table.The only difference is that the entry products information is given.

Table5.fiilcık.DB

Fiilcık	Produt Code	Exit Date	Product Name	Amount	Unit	Unit Price	Total	Profit

This table shows the products that we sell but not giving an invoice.And it deducts from the stock.

Table 6.fiilgir.DB

Fiilgir	Produt Code	Arrival Date	Product Name	Amount	Unit	Unit Price	Total

This table shows the products that we buy without an invoice and it keeps the information about the product.

Table 7.firma.DB

Firma	Current No	Current Name	Tax Offie	Tax Offie No	Related Person	Address/Tel
1	**A01**	**Cemal**	**Luleburgaz**	**39**	**Sirin**	**Luleburgaz/12345**

The new firms information is kept in this table.

Table 8.stockbak.DB

Stockbak	Product Code	Product Name	Amount	Unit
1	**4**	**Cakmak**	**10,00**	**10**

All the information in the stock is kept in this table.

Table 9.stockhrkt.DB

stockhrkt	Product Code	Product Name	GC	Date	Document No	Amount	Unit	Unit price	Total
6	5	Telefon	G	13.06.2002	4	10,00	20	100,00	1.000,00

You can reach extra information of any products in this table.

Table 10.stockkart.DB

Stock kart	Product Code	Product Name	Seller firm	Unit	Unit price	Exchange Type	Rate Of exchange	Unit Price($)	explanation
5	5	Telefon	Mehmet	20	100,00	USD	100,00	1,00	********

This table keeps the information in the stock card form that we gave it before.

Table 11.yedekc.DB

yedekc	Product Code	Product Name	Invoice No	Amount	Unit	Unit price	Tax	Total	Profit
1	3	Defter	02	0,00	12	12.000.000TL	15,00TL	0,00TL	20

When you make an exit process in fatcıkıs form,you decide not to enter any information it tows this information and automatically it returns to the previous record in this table.

Table 12.yedekg.DB

yedekg	Product Code	Product Name	Invoice No	Amount	Unit	Unit price	Tax	Total
1	5	Telefon	4	10,00	20	100,00	15,00	15,00

It makes the same process of the previous table.In here the entery processes is seen.

CHAPTER 3. STOCK CONTROL MANAGEMENT

3.1 Unit Abaut

In this form it is shown that what the program is, who prepared this program and on which day it is start to use.

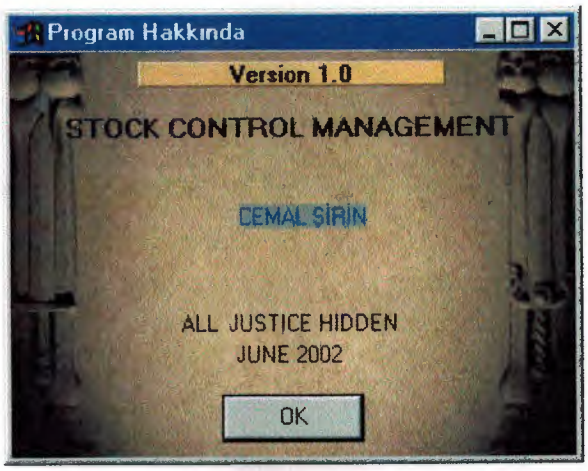

Figure 3.1. Stock Control Management

```
unit abaut;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;
type
  TAboutBox = class(TForm)
    OKButton: TButton;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
```

```
 end;
var
  AboutBox: TAboutBox;
implementation
{$R *.DFM}
end.
```

3.2 Unit Dialog

In ths form it is provded to get invoice and actual exit of the product in the stock.

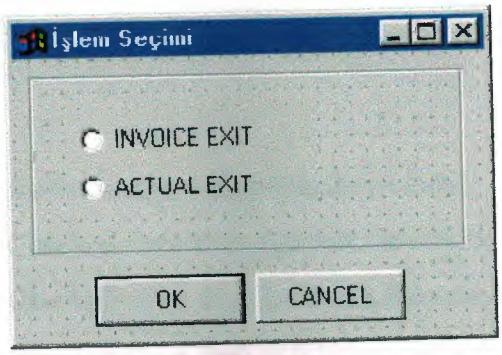

Figure 3.2. Operation Selection

```
unit dialog;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;
type
  TDlgfrm = class(TForm)
    OKBtn: TButton;
    CancelBtn: TButton;
    Bevel1: TBevel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    procedure OKBtnClick(Sender: TObject);
    procedure CancelBtnClick(Sender: TObject);
```

```
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 Dlgfrm: TDlgfrm;
implementation
uses fatgiris, figiris, fatcikis, ficikis;
{$R *.DFM}
procedure TDlgfrm.OKBtnClick(Sender: TObject);
begin
    Hide;
    if RadioButton1.Checked=true then fatgirfrm.showmodal
    else if RadioButton2.Checked=true then flgirfrm.showmodal
    else if RadioButton3.Checked=true then fatcikfrm.showmodal
    else if RadioButton4.Checked=true then flcikfrm.showmodal;
    close;
 end;
 procedure TDlgfrm.CancelBtnClick(Sender: TObject);
 begin
     Close;
 end;
 end.
```

3.3. Unit Dialog2

This form provides our choce of report.

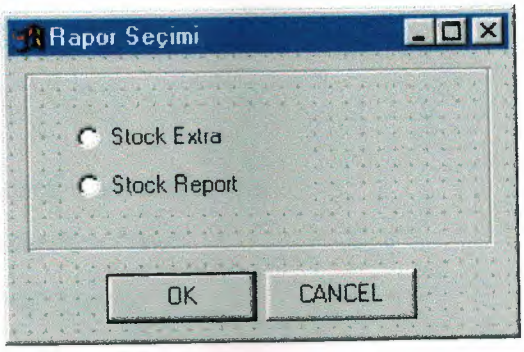

Figure 3.3. Report Selection

```
unit dialog2;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;
type
  TDlgfrm2 = class(TForm)
    OKBtn: TButton;
    CancelBtn: TButton;
    Bevel1: TBevel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    procedure OKBtnClick(Sender: TObject);
    procedure CancelBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Dlgfrm2: TDlgfrm2;
```

```
implementation
uses stokhar, stokbak;
{$R *.DFM}
procedure TDlgfrm2.OKBtnClick(Sender: TObject);
begin
    Hide;
    if RadioButton1.Checked=true then begin
      stokhfrm.Table1.Filtered:= false;
      stokhfrm.Table1.IndexName:= '';
      stokhfrm.QuickRep1.Preview;
    end
    else if RadioButton2.Checked=true then
        stokbfrm.QuickRep1.Preview;
    Close;
end;
procedure TDlgfrm2.CancelBtnClick(Sender: TObject);
begin
    Close;
end;
end.
```

3.4. Unit Ek

This form provides to reach to the product in the stock and it returns the information that we entered. It makes convenience to choose the product.

Figure 3.4. Product Selection

```
unit ek;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, Db, StdCtrls;
type
  Tekfrm = class(TForm)
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    procedure DBGrid1DblClick(Sender: TObject);
    procedure DBGrid1KeyPress(Sender: TObject; var Key: Char);
```

```
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 ekfrm: Tekfrm;
implementation
uses stokkart,fatgiris, figiris, fatcikis, ficikis;
{$R *.DFM}
procedure Tekfrm.DBGrid1DblClick(Sender: TObject);
begin
    if RadioButton1.Checked=true then
      fatgirfrm.Table2.Fields[0].AsString:= DataSource1.DataSet.Fields[0].AsString
    else if RadioButton2.Checked=true then
       flgirfrm.Table1.Fields[0].AsString:= DataSource1.DataSet.Fields[0].AsString
    else if RadioButton3.Checked=true then
       fatcikfrm.Table2.Fields[0].AsString:= DataSource1.DataSet.Fields[0].AsString
    else if RadioButton4.Checked=true then
       flcikfrm.Table1.Fields[0].AsString:= DataSource1.DataSet.Fields[0].AsString;
    Close;
 end;
 procedure Tekfrm.DBGrid1KeyPress(Sender: TObject; var Key: Char);
 begin
    if Key=#27 then Close;
 end;
 end.
```

3.5. Unit Ek2

We are using the firm in this form

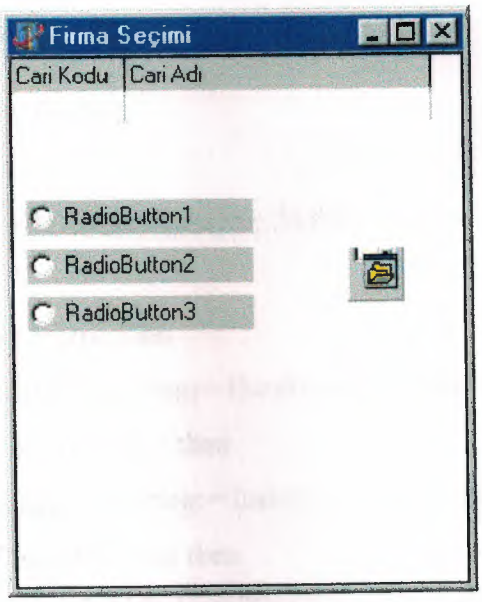

Figure 3.5. Firm Selection

```
unit ek2;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, Db, StdCtrls;
type
  Tek2frm = class(TForm)
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    procedure DBGrid1DblClick(Sender: TObject);
    procedure DBGrid1KeyPress(Sender: TObject; var Key: Char);
  private
```

```
  { Private declarations }
 public
  { Public declarations }
 end;
var
 ek2frm: Tek2frm;
implementation
uses firma,fatgiris, stokkart, fatcikis;
{$R *.DFM}
procedure Tek2frm.DBGrid1DblClick(Sender: TObject);
begin
    if RadioButton1.Checked=true then
      fatgirfrm.Table1.Fields[1].AsString:= DataSource1.DataSet.Fields[0].AsString
    else if RadioButton2.Checked=true then
       stokfrm.Table1.Fields[2].AsString:= DataSource1.DataSet.Fields[1].AsString
    else if RadioButton3.Checked= true then
       fatcikfrm.Table1.Fields[1].AsString:= DataSource1.DataSet.Fields[0].AsString;
    Close;
 end;
 procedure Tek2frm.DBGrid1KeyPress(Sender: TObject; var Key: Char);
 begin
    if Key=#27 then Close;
 end;
 end.
```

3.6. Unit Fatcikis

The invoice exit processes are kept in this form. The invoice that we make if out is deducted, from the stock. And it shows that the name of the firm that we sell the product, in which date and to howmuch TL or exchange that you sell it for?

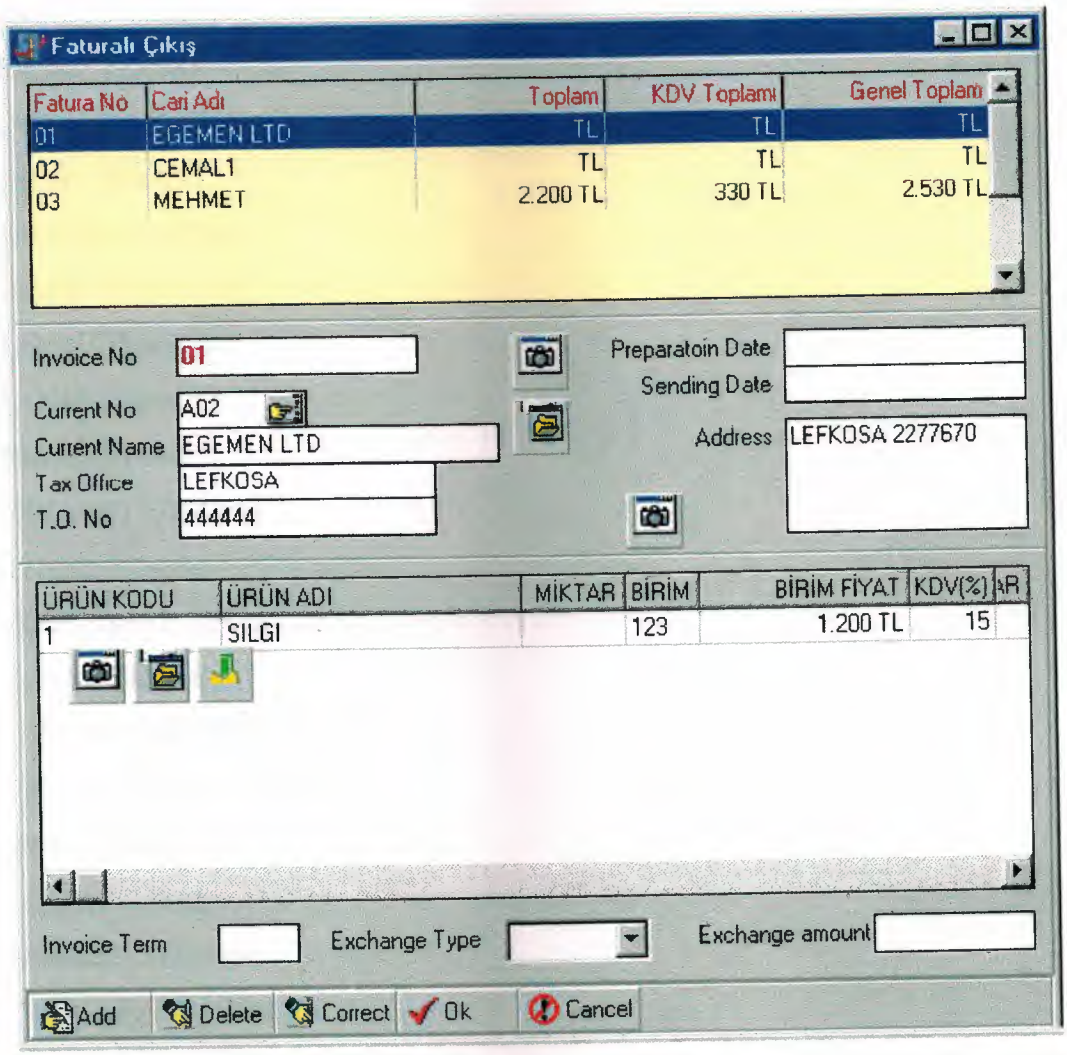

Figure 3.6. Invoice Exit

```
unit fatcikis;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, StdCtrls, Mask, DBCtrls, Db, DBTables, ExtCtrls, Buttons,
```

```
  ComCtrls, ToolWin, ToolEdit, RXDBCtrl, ImgList;
type
  Tfatcikfrm = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Table1: TTable;
    DataSource1: TDataSource;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit3: TDBEdit;
    Label3: TLabel;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    Label4: TLabel;
    Label5: TLabel;
    Panel2: TPanel;
    DBGrid1: TDBGrid;
    Table2: TTable;
    DataSource2: TDataSource;
    Label6: TLabel;
    DBEdit6: TDBEdit;
    Label7: TLabel;
    DBComboBox1: TDBComboBox;
    Label8: TLabel;
    DBEdit7: TDBEdit;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ImageList1: TImageList;
    Panel3: TPanel;
```

```
DBGrid2: TDBGrid;
ToolButton5: TToolButton;
Label9: TLabel;
DBEdit8: TDBEdit;
DBEdit9: TDBEdit;
Label10: TLabel;
DBMemo1: TDBMemo;
Label11: TLabel;
Table3: TTable;
Table1FATURA_NO: TStringField;
Table1CARI_KOD: TStringField;
Table1CARI_AD: TStringField;
Table1ADRES: TMemoField;
Table1VD: TStringField;
Table1VNO: TStringField;
Table1DUZ_TAR: TDateField;
Table1SEVK_TAR: TDateField;
Table1TOPLAM: TFloatField;
Table1KDV_TOPLAM: TFloatField;
Table1GENEL_TOPLAM: TFloatField;
Table1VADE: TFloatField;
Table1DOVIZ_CINS: TStringField;
Table1KUR: TFloatField;
Table1DOVIZ_TUTAR: TFloatField;
Table2URUN_KOD: TStringField;
Table2URUN_AD: TStringField;
Table2FATURA_NO: TStringField;
Table2MIKTAR: TFloatField;
Table2BIRIM: TStringField;
Table2BIRIM_FIYAT: TFloatField;
Table2KDV: TFloatField;
Table2TUTAR: TFloatField;
Table2KAR: TSmallintField;
```

```
RxDBComboEdit1: TRxDBComboEdit;
procedure DBEdit1KeyPress(Sender: TObject; var Key: Char);
procedure DBGrid1EditButtonClick(Sender: TObject);
procedure DBGrid1ColExit(Sender: TObject);
procedure DBGrid1Enter(Sender: TObject);
procedure Table2BeforePost(DataSet: TDataSet);
procedure Table1BeforePost(DataSet: TDataSet);
procedure Table2FilterRecord(DataSet: TDataSet; var Accept: Boolean);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton4Click(Sender: TObject);
procedure ToolButton1Click(Sender: TObject);
procedure ToolButton2Click(Sender: TObject);
procedure Table1BeforeDelete(DataSet: TDataSet);
procedure DataSource1StateChange(Sender: TObject);
procedure Table1BeforeCancel(DataSet: TDataSet);
procedure DBComboBox1Change(Sender: TObject);
procedure ToolButton5Click(Sender: TObject);
procedure DataSource1DataChange(Sender: TObject; Field: TField);
procedure FormActivate(Sender: TObject);
procedure DBGrid1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Table1BeforeEdit(DataSet: TDataSet);
procedure DataSource2DataChange(Sender: TObject; Field: TField);
procedure DBGrid1CellClick(Column: TColumn);
procedure DBGrid2DblClick(Sender: TObject);
procedure RxDBComboEdit1Exit(Sender: TObject);
procedure RxDBComboEdit1ButtonClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
 fatcikfrm: Tfatcikfrm;
implementation
uses stokkart,ek, ek2, firma, stokhar;
{$R *.DFM}
procedure Tfatcikfrm.DBEdit1KeyPress(Sender: TObject; var Key: Char);
begin
    if Key=#13 then begin
      key:= #0;
      Perform(WM_NEXTDLGCTL,0,0);
    end;
end;
procedure Tfatcikfrm.DBGrid1EditButtonClick(Sender: TObject);
var yedek:shortint;
begin
    if DBGrid1.SelectedField=Table2.Fields[0] then begin
      ekfrm.RadioButton3.Checked:= true;
      ekfrm.ShowModal;
    end
    else begin
        yedek:= Table2.Fields[8].AsInteger;
        if Table2.State=dsBrowse then Table2.Edit;
        Table2.Fields[8].AsInteger:=Table2.Fields[8].AsInteger+
StrToInt(InputBox('Birim Fiyat','Kâr Artışı','0'));
        Table2.Fields[5].AsFloat:=
(Table2.Fields[5].AsFloat/(1+yedek/100))*(1+Table2.Fields[8].AsInteger/100);
    end;
end;
procedure Tfatcikfrm.DBGrid1ColExit(Sender: TObject);
begin
    if (Table2.State=dsEdit) or (Table2.State=dsInsert) then begin
      if DBGrid1.SelectedField= Table2.Fields[0] then
        if stokfrm.Table1.FindKey([Table2.Fields[0].AsString]) then begin
```

```
        Table2.Fields[1].AsString:=
stokfrm.Table1.FieldByName('URUN_AD').AsString;
        Table2.Fields[4].AsString:= stokfrm.Table1.FieldByName('BIRIM').AsString;
        if Table2.State=dsInsert then
          Table2.Fields[5].AsFloat:=
stokfrm.Table1.FieldByName('BIRIM_FIYAT').AsFloat*1.20;
      end
      else begin
          ShowMessage('Bu ürün stok kartlarında tanımlı değil !');
          Table2.Cancel;
      end;
     if DBGrid1.SelectedField=Table2.Fields[5] then
       Table2.Fields[7].AsFloat:= Table2.Fields[3].AsFloat*Table2.Fields[5].AsFloat;
    end;
end;
procedure Tfatcikfrm.DBGrid1Enter(Sender: TObject);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
      stokfrm.Table1.IndexName:= '';
      Table2.Edit;
    end;
    DBGrid1.SelectedField:= Table2.Fields[0];
    DBGrid1.SetFocus;
end;
procedure Tfatcikfrm.Table2BeforePost(DataSet: TDataSet);
begin
    Table2.Fields[2].AsString:= Table1.Fields[0].AsString;
    Table2.Fields[7].AsFloat:= Table2.Fields[3].AsFloat*Table2.Fields[5].AsFloat;
end;
procedure Tfatcikfrm.Table1BeforePost(DataSet: TDataSet);
var i:byte;
begin
    Table1.FieldByName('TOPLAM').AsFloat:= 0;
```

```
   Table1.FieldByName('KDV_TOPLAM').AsFloat:= 0;
   Table2.Filtered:= true;
   Table2.First;
   While not Table2.EOF do begin
    Table1.FieldByName('TOPLAM').AsFloat:=
Table1.FieldByName('TOPLAM').AsFloat+Table2.FieldByName('TUTAR').AsFloat;
    Table1.FieldByName('KDV_TOPLAM').AsFloat:=
Table1.FieldByName('KDV_TOPLAM').AsFloat+Table2.Fields[7].AsFloat/100*Table
2.Fields[6].AsFloat;
     Table2.Next;
   end;
   Table1.FieldByName('GENEL_TOPLAM').AsFloat:=
Table1.FieldByName('TOPLAM').AsFloat+Table1.FieldByName('KDV_TOPLAM').A
sFloat;
   if Table1.FieldByName('KUR').AsFloat>0 then
     Table1.FieldByName('DOVIZ_TUTAR').AsFloat:=
Table1.FieldByName('GENEL_TOPLAM').AsFloat/Table1.FieldByName('KUR').AsFl
oat;
   if Table1.State=dsInsert then begin
     Table2.First;
     While not Table2.EOF do begin
      stokhfrm.Table1.Append;
      for i:= 0 to 1 do stokhfrm.Table1.Fields[i].AsString:= Table2.Fields[i].AsString;
      stokhfrm.Table1.Fields[2].AsString:= 'C';
      stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[6].AsDateTime;
      stokhfrm.Table1.Fields[4].AsString:= Table1.Fields[0].AsString;
      stokhfrm.Table1.Fields[5].AsFloat:= Table2.Fields[3].AsFloat;
      stokhfrm.Table1.Fields[6].AsString:= Table2.Fields[4].AsString;
      stokhfrm.Table1.Fields[7].AsFloat:= Table2.Fields[5].AsFloat;
      stokhfrm.Table1.Fields[8].AsFloat:= Table2.Fields[7].AsFloat;
      stokhfrm.Table1.Post;
      Table2.Next;
     end;
```

```
  end
  else if Table1.State=dsEdit then begin
      stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
      stokhfrm.RadioButton1.Checked:= false;
      with stokhfrm.Table1 do begin
         IndexName:='EVRAK';
         Filtered:= true;
         First;
         While not EOF do begin
           if Fields[2].AsString='G' then Next
           else Delete;
         end;
      end;
      Table2.First;
      While not Table2.EOF do begin
        stokhfrm.Table1.Append;
        for i:= 0 to 1 do stokhfrm.Table1.Fields[i].AsString:=
Table2.Fields[i].AsString;
        stokhfrm.Table1.Fields[2].AsString:= 'C';
        stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[6].AsDateTime;
        stokhfrm.Table1.Fields[4].AsString:= Table1.Fields[0].AsString;
        stokhfrm.Table1.Fields[5].AsFloat:= Table2.Fields[3].AsFloat;
        stokhfrm.Table1.Fields[6].AsString:= Table2.Fields[4].AsString;
        stokhfrm.Table1.Fields[7].AsFloat:= Table2.Fields[5].AsFloat;
        stokhfrm.Table1.Fields[8].AsFloat:= Table2.Fields[7].AsFloat;
        stokhfrm.Table1.Post;
        Table2.Next;
      end;
    end;
end;
procedure Tfatcikfrm.Table2FilterRecord(DataSet: TDataSet;
 var Accept: Boolean);
begin
```

```
  if Table2.Fields[2].AsString=Table1.Fields[0].AsString then
    Accept:= true
  else accept:= false;
end;
procedure Tfatcikfrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
    if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
      mtConfirmation,[mbYes,mbNo],0)=mrYes then begin
      Table1.Cancel;
      Table2.Filtered:= False;
    end
    else action:= caNone;
  end;
  if stokfrm.Table1.Filtered=true then
    stokfrm.Table1.Filtered:= false;
end;

procedure Tfatcikfrm.ToolButton3Click(Sender: TObject);
begin
  if (Table2.State=dsEdit) or (Table2.State=dsInsert) then
    Table2.Post;
  Table1.Post;
end;
procedure Tfatcikfrm.ToolButton4Click(Sender: TObject);
begin
  Table1.Cancel;
end;
procedure Tfatcikfrm.ToolButton1Click(Sender: TObject);
var
  yedek: string;
  KeyState : TKeyboardState;
begin
```

```
  GetKeyboardState(KeyState);
  if (KeyState[VK_CAPITAL] = 0) then
    KeyState[VK_CAPITAL] := 1;
  SetKeyboardState(KeyState);
  yedek:= InputBox('Yeni Kayıt','Fatura No','');
  if yedek<>'' then
  if Table1.FindKey([yedek])=false then begin
    Table1.Append;
    Table1.Fields[0].AsString:= yedek;
    RxDBComboEdit1.SetFocus;
  end
  else ShowMessage('Bu kayıt daha önce zaten girilmiş !');
end;
procedure Tfatcikfrm.ToolButton2Click(Sender: TObject);
begin
  if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz',
    mtConfirmation,[mbYes,mbNo],0)= mrYes then
    Table1.Delete;
end;
procedure Tfatcikfrm.Table1BeforeDelete(DataSet: TDataSet);
begin
  stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
  stokhfrm.RadioButton1.Checked:= false;
  with stokhfrm.Table1 do begin
    IndexName:='EVRAK';
    Filtered:= true;
    First;
    While not EOF do begin
      if Fields[2].AsString='G' then Last
      else Delete;
    end;
  end;
  Table2.Filtered:= true;
```

```
    Table2.First;
    Table2.Active:= false;
    Table2.ReadOnly:= false;
    Table2.Active:= true;
    While not Table2.EOF do Table2.Delete;
    DataSource1StateChange(Self);
end;
procedure Tfatcikfrm.DataSource1StateChange(Sender: TObject);
begin
    case Table1.State of
    dsEdit,dsInsert: begin
                ToolButton3.Enabled:= true;
                ToolButton4.Enabled:= true;
                RxDBComboEdit1.Button.Visible:= true;
                ToolButton1.Enabled:= false;
                ToolButton2.Enabled:= false;
                ToolButton5.Enabled:= false;
                if Table2.ReadOnly=true then begin
                  Table2.Active:= false;
                  Table2.ReadOnly:= false;
                  Table2.Active:= true;
                end;
                DBGrid1.Columns[0].ButtonStyle:= cbsEllipsis;
                DBGrid1.Columns[4].ButtonStyle:= cbsEllipsis;
                end;
     dsBrowse:      begin
                ToolButton1.Enabled:= true;
                ToolButton2.Enabled:= true;
                ToolButton5.Enabled:= true;
                RxDBComboEdit1.Button.Visible:= false;
                ToolButton3.Enabled:= false;
                ToolButton4.Enabled:= false;
                if Table2.ReadOnly=false then begin
```

```
                Table2.Active:= false;
                Table2.ReadOnly:= true;
                Table2.Active:= true;
              end;
              DBGrid1.Columns[0].ButtonStyle:= cbsAuto;
              DBGrid1.Columns[4].ButtonStyle:= cbsAuto;
              end;
  end;
end;
procedure Tfatcikfrm.Table1BeforeCancel(DataSet: TDataSet);
var i:byte;
begin
  if Table1.State=dsInsert then begin
    Table2.Filtered:= true;
    Table2.First;
    While not Table2.EOF do Table2.Delete;
  end
  else if Table1.State=dsEdit then begin
      Table2.First;
      While not Table2.EOF do Table2.Delete;
      Table3.First;
      While not Table3.EOF do begin
       Table2.Append;
       for i:= 0 to 2 do Table2.Fields[i].AsString:= Table3.Fields[i].AsString;
       Table2.Fields[3].AsFloat:= Table3.Fields[3].AsFloat;
       Table2.Fields[4].AsString:= Table3.Fields[4].AsString;
       Table2.Fields[5].AsFloat:= Table3.Fields[5].AsFloat;
       Table2.Fields[6].AsFloat:= Table3.Fields[6].AsFloat;
       Table2.Fields[8].AsInteger:= Table3.Fields[8].AsInteger;
       Table2.Post;
       Table3.Next;
      end;
   end;
```

```
end;
procedure Tfatcikfrm.DBComboBox1Change(Sender: TObject);
begin
   Table1.FieldByName('KUR').AsFloat:= StrToFloat(InputBox('Kur Girişi','Kur','0'));
   if Table1.FieldByName('KUR').AsFloat>0 then
     Table1.FieldByName('DOVIZ_TUTAR').AsFloat:=
Table1.FieldByName('GENEL_TOPLAM').AsFloat/Table1.FieldByName('KUR').AsFl
oat;
end;
procedure Tfatcikfrm.ToolButton5Click(Sender: TObject);
begin
   Table1.Edit;
   RxDBComboEdit1.SetFocus;
end;
procedure Tfatcikfrm.DataSource1DataChange(Sender: TObject; Field: TField);
begin
   DBEdit7.Hint:= 'Kur = '+FloatToStr(Table1.FieldByName('KUR').AsFloat);
end;
procedure Tfatcikfrm.FormActivate(Sender: TObject);
begin
   DBGrid2.SetFocus;
end;
procedure Tfatcikfrm.DBGrid1KeyDown(Sender: TObject; var Key: Word;
 Shift: TShiftState);
begin
   if Key=VK_RETURN then Key:= VK_TAB;
end;
procedure Tfatcikfrm.Table1BeforeEdit(DataSet: TDataSet);
var i:byte;
begin
    Table3.First;
    While not Table3.EOF do Table3.Delete;
    Table2.Filtered:= true;
```

```
  Table2.First;
  While not Table2.EOF do begin
      Table3.Append;
      for i:= 0 to 2 do Table3.Fields[i].AsString:= Table2.Fields[i].AsString;
      Table3.Fields[3].AsFloat:= Table2.Fields[3].AsFloat;
      Table3.Fields[4].AsString:= Table2.Fields[4].AsString;
      Table3.Fields[5].AsFloat:= Table2.Fields[5].AsFloat;
      Table3.Fields[6].AsFloat:= Table2.Fields[6].AsFloat;
      Table3.Fields[7].AsFloat:= Table2.Fields[7].AsFloat;
      Table3.Fields[8].AsInteger:= Table2.Fields[8].AsInteger;
      Table3.Post;
      Table2.Next;
   end;
end;
procedure Tfatcikfrm.DataSource2DataChange(Sender: TObject; Field: TField);
begin
   DBGrid1.Hint:= 'Kâr = %'+IntToStr(Table2.FieldByName('KAR').AsInteger);
end;
procedure Tfatcikfrm.DBGrid1CellClick(Column: TColumn);
begin
   if DBGrid1.SelectedField=Table2.Fields[7] then DBGrid1.ShowHint:= true
     else DBGrid1.ShowHint:= false;
end;
procedure Tfatcikfrm.DBGrid2DblClick(Sender: TObject);
begin
   Table1.IndexName:='';
   Table1.FindNearest([InputBox('Kayıt Arama','Fatura No','')]);
end;
procedure Tfatcikfrm.RxDBComboEdit1Exit(Sender: TObject);
begin
   if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
     firmafrm.Table1.IndexName:='';
     if firmafrm.Table1.FindKey([RxDBComboEdit1.Text]) then begin
```

```
        Table1.FieldByName('CARI_AD').AsString:=
firmafrm.Table1.Fields[1].AsString;
        Table1.FieldByName('VD').AsString:= firmafrm.Table1.Fields[2].AsString;
        Table1.FieldByName('VNO').AsString:= firmafrm.Table1.Fields[3].AsString;
        Table1.FieldByName('ADRES').AsString:= firmafrm.Table1.Fields[5].AsString;
      end
      else begin
         ShowMessage('"Firma Tanımları" nda bu kayıt bulunamadı');
         RxDBComboEdit1.SetFocus;
      end;
    end;
end;
procedure Tfatcikfrm.RxDBComboEdit1ButtonClick(Sender: TObject);
begin
    ek2frm.RadioButton3.Checked:= true;
    ek2frm.showModal;
end;
end.
```

3.7. Unit Fatgiris

In this form invoice entry is provided. It adds the product that we enter to the stock. And it shows that the name of the firm that we buy the product, in which date and to how much TL or exchange that you buy it for?

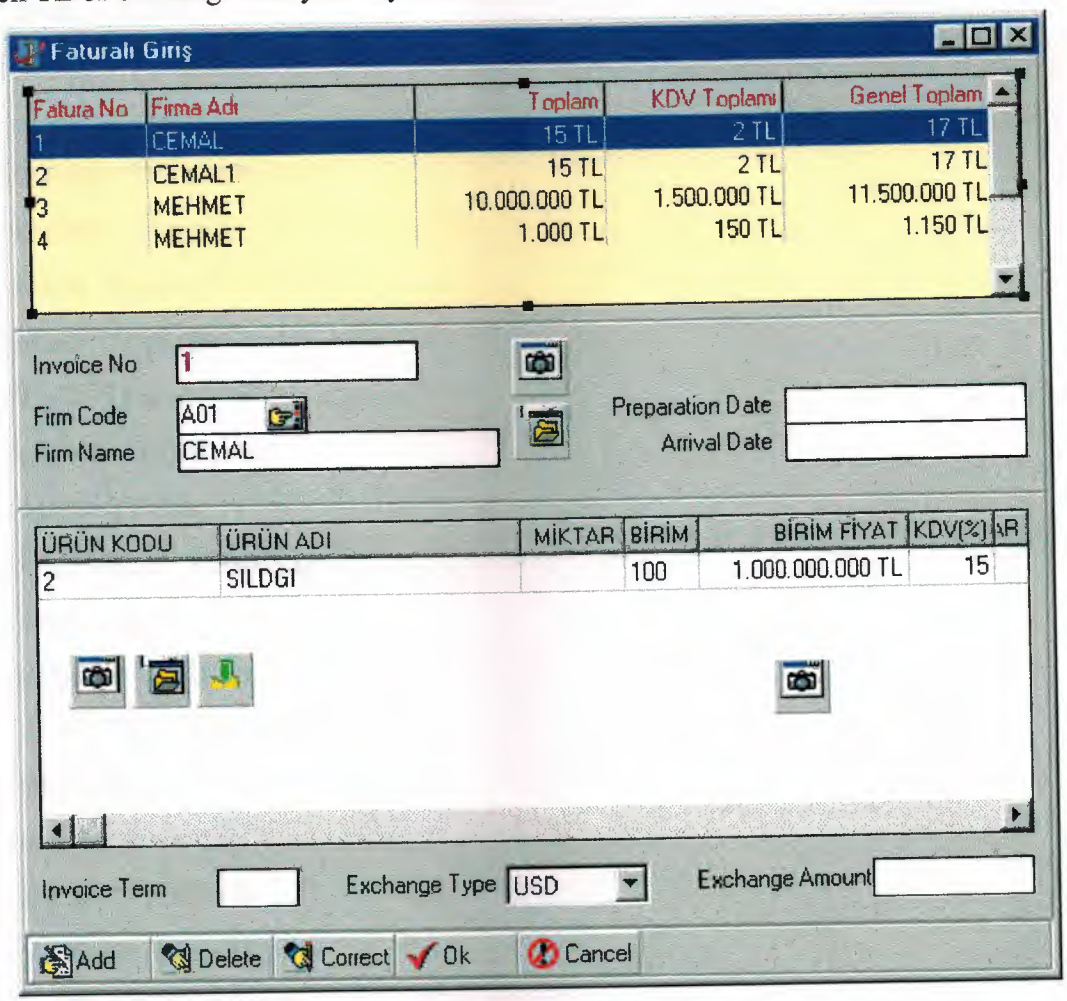

Figure 3.7. Invoice Entry

```
unit fatgiris;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, StdCtrls, Mask, DBCtrls, Db, DBTables, ExtCtrls, Buttons,
  ComCtrls, ToolWin, ToolEdit, RXDBCtrl, ImgList;
```

```
type
  Tfatgirfrm = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Table1: TTable;
    DataSource1: TDataSource;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit3: TDBEdit;
    Label3: TLabel;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    Label4: TLabel;
    Label5: TLabel;
    Panel2: TPanel;
    DBGrid1: TDBGrid;
    Table2: TTable;
    DataSource2: TDataSource;
    Label6: TLabel;
    DBEdit6: TDBEdit;
    Label7: TLabel;
    DBComboBox1: TDBComboBox;
    Label8: TLabel;
    DBEdit7: TDBEdit;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ImageList1: TImageList;
    Panel3: TPanel;
    DBGrid2: TDBGrid;
```

```
ToolButton5: TToolButton;
Table3: TTable;
Table1FATURA_NO: TStringField;
Table1CARI_KOD: TStringField;
Table1CARI_AD: TStringField;
Table1DUZ_TAR: TDateField;
Table1TES_TAR: TDateField;
Table1TOPLAM: TFloatField;
Table1KDV_TOPLAM: TFloatField;
Table1GENEL_TOP: TFloatField;
Table1VADE: TSmallintField;
Table1DOVIZ_CINS: TStringField;
Table1KUR: TFloatField;
Table1DOVIZ_TUTAR: TFloatField;
Table2URUN_KOD: TStringField;
Table2URUN_AD: TStringField;
Table2FATURA_NO: TStringField;
Table2MIKTAR: TFloatField;
Table2BIRIM: TStringField;
Table2BIRIM_FIYAT: TFloatField;
Table2KDV: TFloatField;
Table2TUTAR: TFloatField;
RxDBComboEdit1: TRxDBComboEdit;
procedure DBEdit1KeyPress(Sender: TObject; var Key: Char);
procedure DBGrid1EditButtonClick(Sender: TObject);
procedure DBGrid1ColExit(Sender: TObject);
procedure DBGrid1Enter(Sender: TObject);
procedure Table2BeforePost(DataSet: TDataSet);
procedure Table1BeforePost(DataSet: TDataSet);
procedure Table2FilterRecord(DataSet: TDataSet; var Accept: Boolean);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton4Click(Sender: TObject);
```

```
procedure ToolButton1Click(Sender: TObject);
procedure ToolButton2Click(Sender: TObject);
procedure Table1BeforeDelete(DataSet: TDataSet);
procedure DataSource1StateChange(Sender: TObject);
procedure Table1BeforeCancel(DataSet: TDataSet);
procedure DBComboBox1Change(Sender: TObject);
procedure ToolButton5Click(Sender: TObject);
procedure DataSource1DataChange(Sender: TObject; Field: TField);
procedure FormActivate(Sender: TObject);
procedure DBGrid1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure Table1BeforeEdit(DataSet: TDataSet);
procedure DBGrid2DblClick(Sender: TObject);
procedure RxDBComboEdit1Exit(Sender: TObject);
procedure RxDBComboEdit1ButtonClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
 fatgirfrm: Tfatgirfrm;
implementation
uses stokkart,ek, ek2, firma, stokhar;
{$R *.DFM}
procedure Tfatgirfrm.DBEdit1KeyPress(Sender: TObject; var Key: Char);
begin
    if Key=#13 then begin
      key:= #0;
      Perform(WM_NEXTDLGCTL,0,0);
    end;
end;
procedure Tfatgirfrm.DBGrid1EditButtonClick(Sender: TObject);
```

```
begin
    ekfrm.RadioButton1.Checked:= true;
    ekfrm.ShowModal;
end;
procedure Tfatgirfrm.DBGrid1ColExit(Sender: TObject);
begin
    if (Table2.State=dsEdit) or (Table2.State=dsInsert) then begin
     if DBGrid1.SelectedField= Table2.Fields[0] then
      if stokfrm.Table1.FindKey([Table2.Fields[0].AsString]) then begin
        Table2.Fields[1].AsString:=
stokfrm.Table1.FieldByName('URUN_AD').AsString;
        Table2.Fields[4].AsString:= stokfrm.Table1.FieldByName('BIRIM').AsString;
        Table2.Fields[5].AsFloat:=
stokfrm.Table1.FieldByName('BIRIM_FIYAT').AsFloat;
       end
       else begin
        ShowMessage('Bu kayıt stok kartlarında bulunmuyor ya da firma tanımı yanlış !');
          Table2.Cancel;
       end;
      if DBGrid1.SelectedField= Table2.Fields[3] then
        Table2.Fields[7].AsFloat:= Table2.Fields[3].AsFloat*Table2.Fields[5].AsFloat;
     end;
 end;
 procedure Tfatgirfrm.DBGrid1Enter(Sender: TObject);
 begin
     if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
       stokfrm.Edit2.Text:= Table1.FieldByName('CARI_AD').AsString;
       stokfrm.Table1.IndexName:= 'FIRMA';
       ekfrm.DataSource1.DataSet.Filtered:= true;
       stokfrm.Table1.IndexName:= '';
       Table2.Edit;
      end;
      DBGrid1.SelectedField:= Table2.Fields[0];
```

```
    DBGrid1.SetFocus;
end;
procedure Tfatgirfrm.Table2BeforePost(DataSet: TDataSet);
begin
    Table2.Fields[2].AsString:= Table1.Fields[0].AsString;
end;
procedure Tfatgirfrm.Table1BeforePost(DataSet: TDataSet);
var i:byte;
begin
    Table1.FieldByName('TOPLAM').AsFloat:= 0;
    Table1.FieldByName('KDV_TOPLAM').AsFloat:= 0;
    Table2.Filtered:= true;
    Table2.First;
    While not Table2.EOF do begin
     Table1.FieldByName('TOPLAM').AsFloat:=
Table1.FieldByName('TOPLAM').AsFloat+Table2.FieldByName('TUTAR').AsFloat;
     Table1.FieldByName('KDV_TOPLAM').AsFloat:=
Table1.FieldByName('KDV_TOPLAM').AsFloat+Table2.Fields[7].AsFloat/100*Table
2.Fields[6].AsFloat;
     Table2.Next;
    end;
    Table1.FieldByName('GENEL_TOP').AsFloat:=
Table1.FieldByName('TOPLAM').AsFloat+Table1.FieldByName('KDV_TOPLAM').A
sFloat;
    if Table1.FieldByName('KUR').AsFloat>0 then
      Table1.FieldByName('DOVIZ_TUTAR').AsFloat:=
Table1.FieldByName('GENEL_TOP').AsFloat/Table1.FieldByName('KUR').AsFloat;
    if Table1.State=dsInsert then begin
      Table2.First;
      While not Table2.EOF do begin
       stokhfrm.Table1.Append;
       for i:= 0 to 1 do stokhfrm.Table1.Fields[i].AsString:= Table2.Fields[i].AsString;
       stokhfrm.Table1.Fields[2].AsString:= 'G';
```

```
    stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[4].AsDateTime;
    stokhfrm.Table1.Fields[4].AsString:= Table1.Fields[0].AsString;
    stokhfrm.Table1.Fields[5].AsFloat:= Table2.Fields[3].AsFloat;
    stokhfrm.Table1.Fields[6].AsString:= Table2.Fields[4].AsString;
    stokhfrm.Table1.Fields[7].AsFloat:= Table2.Fields[5].AsFloat;
    stokhfrm.Table1.Fields[8].AsFloat:= Table2.Fields[7].AsFloat;
    stokhfrm.Table1.Post;
    Table2.Next;
   end;
 end
 else if Table1.State=dsEdit then begin
     stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
     stokhfrm.RadioButton1.Checked:= false;
     with stokhfrm.Table1 do begin
        IndexName:='EVRAK';
        Filtered:= true;
        First;
        While not EOF do begin
          if Fields[2].AsString='C' then Next
          else Delete;
        end;
      end;
      Table2.First;
      While not Table2.EOF do begin
        stokhfrm.Table1.Append;
        for i:= 0 to 1 do stokhfrm.Table1.Fields[i].AsString:=
Table2.Fields[i].AsString;
        stokhfrm.Table1.Fields[2].AsString:= 'G';
        stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[4].AsDateTime;
        stokhfrm.Table1.Fields[4].AsString:= Table1.Fields[0].AsString;
        stokhfrm.Table1.Fields[5].AsFloat:= Table2.Fields[3].AsFloat;
        stokhfrm.Table1.Fields[6].AsString:= Table2.Fields[4].AsString;
        stokhfrm.Table1.Fields[7].AsFloat:= Table2.Fields[5].AsFloat;
```

```
          stokhfrm.Table1.Fields[8].AsFloat:= Table2.Fields[7].AsFloat;
          stokhfrm.Table1.Post;
          Table2.Next;
        end;
      end;
end;
procedure Tfatgirfrm.Table2FilterRecord(DataSet: TDataSet; var Accept: Boolean);
begin
    if Table2.Fields[2].AsString=Table1.Fields[0].AsString then
      Accept:= true
    else accept:= false;
end;
procedure Tfatgirfrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
      if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
        mtConfirmation,[mbYes,mbNo],0)=mrYes then begin
        Table1.Cancel;
        Table2.Filtered:= False;
      end
      else action:= caNone;
    end;
    if stokfrm.Table1.Filtered=true then
      stokfrm.Table1.Filtered:= false;
end;
procedure Tfatgirfrm.ToolButton3Click(Sender: TObject);
begin
    if (Table2.State=dsEdit) or (Table2.State=dsInsert) then
      Table2.Post;
    Table1.Post;
end;
procedure Tfatgirfrm.ToolButton4Click(Sender: TObject);
begin
```

```
    Table1.Cancel;
end;
procedure Tfatgirfrm.ToolButton1Click(Sender: TObject);
var
  yedek:string;
  KeyState : TKeyboardState;
begin
    GetKeyboardState(KeyState);
    if (KeyState[VK_CAPITAL] = 0) then
      KeyState[VK_CAPITAL] := 1;
    SetKeyboardState(KeyState);
    yedek:= InputBox('Yeni Kayıt','Fatura No','');
    if yedek<>'' then
    if Table1.FindKey([(yedek)])=false then begin
      Table1.Append;
      Table1.Fields[0].AsString:= yedek;
      RxDBComboEdit1.SetFocus;
    end
    else ShowMessage('Bu kayıt daha önce zaten girilmiş !');
end;
procedure Tfatgirfrm.ToolButton2Click(Sender: TObject);
begin
    if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz',
      mtConfirmation,[mbYes,mbNo],0)= mrYes then
      Table1.Delete;
end;
procedure Tfatgirfrm.Table1BeforeDelete(DataSet: TDataSet);
begin
    stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
    stokhfrm.RadioButton1.Checked:= false;
    with stokhfrm.Table1 do begin
      IndexName:='EVRAK';
      Filtered:= true;
```

```
      First;
      While not EOF do begin
          if Fields[2].AsString='C' then Next
          else Delete;
      end;
   end;
   Table2.Filtered:= true;
   Table2.First;
   Table2.Active:= false;
   Table2.ReadOnly:= false;
   Table2.Active:= true;
   While not Table2.EOF do Table2.Delete;
   DataSource1StateChange(Self);
end;
procedure Tfatgirfrm.DataSource1StateChange(Sender: TObject);
begin
   case Table1.State of
   dsEdit,dsInsert: begin
                ToolButton3.Enabled:= true;
                ToolButton4.Enabled:= true;
                RxDBComboEdit1.Button.Visible:= true;
                ToolButton1.Enabled:= false;
                ToolButton2.Enabled:= false;
                ToolButton5.Enabled:= false;
                if Table2.ReadOnly=true then begin
                  Table2.Active:= false;
                  Table2.ReadOnly:= false;
                  Table2.Active:= true;
                end;
                DBGrid1.Columns[0].ButtonStyle:= cbsEllipsis;
                end;
    dsBrowse:      begin
                ToolButton1.Enabled:= true;
```

```
            ToolButton2.Enabled:= true;
            ToolButton5.Enabled:= true;
            RxDBComboEdit1.Button.Visible:= false;
            ToolButton3.Enabled:= false;
            ToolButton4.Enabled:= false;
            if Table2.ReadOnly=false then begin
              Table2.Active:= false;
              Table2.ReadOnly:= true;
              Table2.Active:= true;
            end;
            DBGrid1.Columns[0].ButtonStyle:= cbsAuto;
            end;
  end;
end;
procedure Tfatgirfrm.Table1BeforeCancel(DataSet: TDataSet);
var i:byte;
begin
  if Table1.State=dsInsert then begin
    Table2.Filtered:= true;
    Table2.First;
    While not Table2.EOF do Table2.Delete;
  end
  else if Table1.State=dsEdit then begin
      Table2.First;
      While not Table2.EOF do Table2.Delete;
      Table3.First;
      While not Table3.EOF do begin
        Table2.Append;
        for i:= 0 to 2 do Table2.Fields[i].AsString:= Table3.Fields[i].AsString;
        Table2.Fields[3].AsFloat:= Table3.Fields[3].AsFloat;
        Table2.Fields[4].AsString:= Table3.Fields[4].AsString;
        for i:= 5 to 7 do Table2.Fields[i].AsFloat:= Table3.Fields[i].AsFloat;
        Table2.Post;
```

```
        Table3.Next;
      end;
    end;
end;
procedure Tfatgirfrm.DBComboBox1Change(Sender: TObject);
begin
    Table1.FieldByName('KUR').AsFloat:= StrToFloat(InputBox('Kur Girişi','Kur','0'));
    if Table1.FieldByName('KUR').AsFloat>0 then
      Table1.FieldByName('DOVIZ_TUTAR').AsFloat:=
Table1.FieldByName('GENEL_TOP').AsFloat/Table1.FieldByName('KUR').AsFloat;
end;
procedure Tfatgirfrm.ToolButton5Click(Sender: TObject);
begin
    Table1.Edit;
    RxDBComboEdit1.SetFocus;
end;
procedure Tfatgirfrm.DataSource1DataChange(Sender: TObject; Field: TField);
begin
    DBEdit7.Hint:= 'Kur = '+FloatToStr(Table1.FieldByName('KUR').AsInteger);
    DBEdit7.ShowHint:=true;
end;
procedure Tfatgirfrm.FormActivate(Sender: TObject);
begin
    DBGrid2.SetFocus;
end;
procedure Tfatgirfrm.DBGrid1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if Key=VK_RETURN then Key:= VK_TAB;
end;
procedure Tfatgirfrm.Table1BeforeEdit(DataSet: TDataSet);
var i:byte;
begin
```

```
  Table3.First;
  While not Table3.EOF do Table3.Delete;
  Table2.Filtered:= true;
  Table2.First;
  While not Table2.EOF do begin
      Table3.Append;
      for i:= 0 to 2 do Table3.Fields[i].AsString:= Table2.Fields[i].AsString;
      Table3.Fields[3].AsFloat:= Table2.Fields[3].AsFloat;
      Table3.Fields[4].AsString:= Table2.Fields[4].AsString;
      for i:= 5 to 7 do Table3.Fields[i].AsFloat:= Table2.Fields[i].AsFloat;
      Table3.Post;
      Table2.Next;
   end;
end;
procedure Tfatgirfrm.DBGrid2DblClick(Sender: TObject);
begin
   Table1.IndexName:='';
   Table1.FindNearest([InputBox('Kayıt Arama','Fatura No','')]);
end;
procedure Tfatgirfrm.RxDBComboEdit1Exit(Sender: TObject);
begin
   if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
     firmafrm.Table1.IndexName:='';
     if firmafrm.Table1.FindKey([RxDBComboEdit1.Text]) then
      Table1.FieldByName('CARI_AD').AsString:=firmafrm.Table1.Fields[1].AsString
     else begin
        ShowMessage('"Firma Tanımları" nda bu kayıt bulunamadı');
        RxDBComboEdit1.SetFocus;
      end;
    end;
 end;
 procedure Tfatgirfrm.RxDBComboEdit1ButtonClick(Sender: TObject);
 begin
```

```
    ek2frm.RadioButton1.Checked:= true;
    ek2frm.showModal;
end;
end.
```

3.8. Unit Ficikis

This form provides us to get the actual exit. And it shows the date that we sell the product and howmuch money that we get. It deducts the product that we sell from the stock.

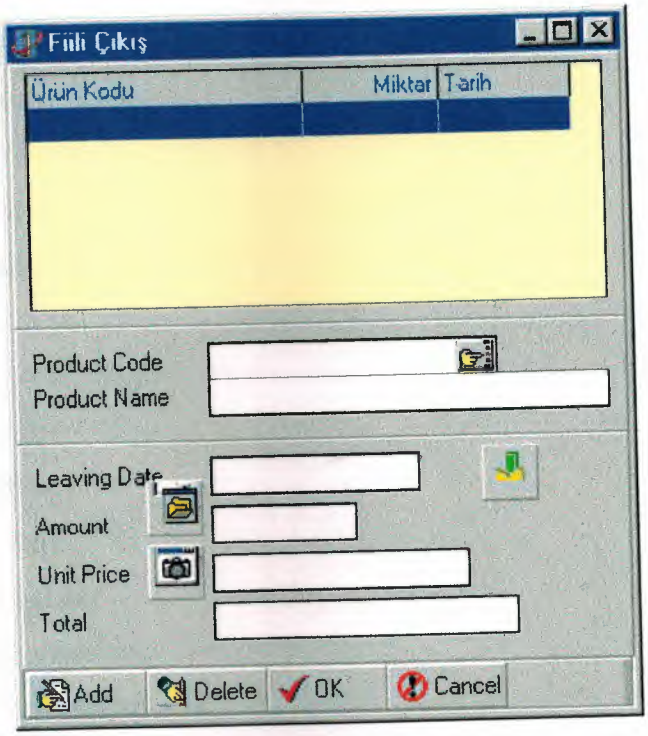

Figure 3.8. Actual Exit

```
unit ficikis;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, DBCtrls, StdCtrls, Mask, ExtCtrls, Db, DBTables,
  ComCtrls, ToolWin, Buttons, ToolEdit, RXDBCtrl, ImgList;
type
  Tficikfrm = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
```

```
Panel1: TPanel;
Label1: TLabel;
Label2: TLabel;
DBEdit2: TDBEdit;
Panel2: TPanel;
Label3: TLabel;
DBEdit3: TDBEdit;
Label4: TLabel;
DBEdit4: TDBEdit;
DBText1: TDBText;
Label5: TLabel;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
Label6: TLabel;
Panel3: TPanel;
DBGrid1: TDBGrid;
CoolBar1: TCoolBar;
ToolBar1: TToolBar;
ImageList1: TImageList;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
ToolButton3: TToolButton;
ToolButton4: TToolButton;
Table1URUN_KOD: TStringField;
Table1URUN_AD: TStringField;
Table1CIK_TAR: TDateField;
Table1MIKTAR: TFloatField;
Table1BIRIM: TStringField;
Table1BIRIM_FIYAT: TFloatField;
Table1TUTAR: TFloatField;
Table1KAR: TSmallintField;
RxDBComboEdit1: TRxDBComboEdit;
procedure ToolButton1Click(Sender: TObject);
```

```
procedure ToolButton2Click(Sender: TObject);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton4Click(Sender: TObject);
procedure Table1BeforePost(DataSet: TDataSet);
procedure DataSource1StateChange(Sender: TObject);
procedure Table1AfterInsert(DataSet: TDataSet);
procedure DBEdit5Enter(Sender: TObject);
procedure DataSource1DataChange(Sender: TObject; Field: TField);
procedure DBGrid1DblClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Table1BeforeEdit(DataSet: TDataSet);
procedure Table1BeforeDelete(DataSet: TDataSet);
procedure RxDBComboEdit1Exit(Sender: TObject);
procedure RxDBComboEdit1KeyPress(Sender: TObject; var Key: Char);
procedure RxDBComboEdit1ButtonClick(Sender: TObject);
private
  { Private declarations }
public
tarih:TDateTime;
  { Public declarations }
end;
var
 flcikfrm: Tflcikfrm;
implementation
uses stokkart, ek2, ek, stokhar;
{$R *.DFM}
procedure Tflcikfrm.ToolButton1Click(Sender: TObject);
begin
    Table1.Append;
end;
procedure Tflcikfrm.ToolButton2Click(Sender: TObject);
begin
    if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz ?',
```

```
    mtConfirmation,[mbYes,mbNo],0)=mrYes then
    Table1.Delete;
end;
procedure Tflcikfrm.ToolButton3Click(Sender: TObject);
begin
   Table1.Post;
end;
procedure Tflcikfrm.ToolButton4Click(Sender: TObject);
begin
   Table1.Cancel;
end;
procedure Tflcikfrm.Table1BeforePost(DataSet: TDataSet);
var kontrol,kont:boolean;
begin
   Table1.Fields[6].AsFloat:= Table1.Fields[3].AsFloat*Table1.Fields[5].AsFloat;
   if Table1.State=dsInsert then begin
      stokhfrm.Table1.Append;
      stokhfrm.Table1.Fields[0].AsString:= Table1.Fields[0].AsString;
      stokhfrm.Table1.Fields[1].AsString:= Table1.Fields[1].AsString;
      stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[2].AsDateTime;
      stokhfrm.Table1.Fields[2].AsString:= 'C';
      stokhfrm.Table1.Fields[5].AsFloat:= Table1.Fields[3].AsFloat;
      stokhfrm.Table1.Fields[6].AsString:= Table1.Fields[4].AsString;
      stokhfrm.Table1.Fields[7].AsFloat:= Table1.Fields[5].AsFloat;
      stokhfrm.Table1.Fields[8].AsFloat:= Table1.Fields[6].AsFloat;
      stokhfrm.Table1.Post;
    end
    else if Table1.State=dsEdit then begin
       stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
       stokhfrm.RadioButton1.Checked:= true;
       stokhfrm.Table1.IndexName:='';
       stokhfrm.Table1.Filtered:= true;
       stokhfrm.Table1.IndexName:='TARIHX';
```

```
      if tarih=Table1.Fields[2].AsDateTime then begin
        stokhfrm.Table1.FindKey([Table1.Fields[2].AsDateTime]);
        kontrol:= true;
      end
      else begin
          stokhfrm.Table1.FindKey([tarih]);
          kontrol:= false;
      end;
      kont:=false;
      repeat
          if stokhfrm.Table1.Fields[2].AsString='C' then begin
            if stokhfrm.Table1.Fields[4].AsString='' then
              kont:= true
            else stokhfrm.Table1.Next;
          end
          else stokhfrm.Table1.Next;
       until kont=true;
       stokhfrm.Table1.Edit;
       if kontrol=false then
         stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[2].AsDateTime;
       stokhfrm.Table1.Fields[5].AsFloat:= Table1.Fields[3].AsFloat;
       stokhfrm.Table1.Fields[6].AsString:= Table1.Fields[4].AsString;
       stokhfrm.Table1.Fields[7].AsFloat:= Table1.Fields[5].AsFloat;
       stokhfrm.Table1.Fields[8].AsFloat:= Table1.Fields[6].AsFloat;
       stokhfrm.Table1.Post;
    end;
end;
procedure Tflcikfrm.DataSource1StateChange(Sender: TObject);
begin
   case Table1.State of
   dsInsert:      begin
                  ToolButton1.Enabled:= False;
                  ToolButton2.Enabled:= False;
```

```
            ToolButton3.Enabled:= true;
            ToolButton4.Enabled:= true;
            RxDBComboEdit1.ReadOnly:= false;
            RxDBComboEdit1.Button.Visible:=true;
            end;
 dsEdit:      begin
            ToolButton1.Enabled:= False;
            ToolButton2.Enabled:= False;
            ToolButton3.Enabled:= true;
            ToolButton4.Enabled:= true;
            end;
  dsBrowse:     begin
            ToolButton3.Enabled:= False;
            ToolButton4.Enabled:= False;
            ToolButton1.Enabled:= true;
            ToolButton2.Enabled:= true;
            RxDBComboEdit1.Button.Visible:=False;
            RxDBComboEdit1.ReadOnly:= true;
            end;
   end;
end;
procedure Tflcikfrm.Table1AfterInsert(DataSet: TDataSet);
begin
   RxDBComboEdit1.SetFocus;
end;
procedure Tflcikfrm.DBEdit5Enter(Sender: TObject);
var yedek:shortint;
begin
    yedek:= Table1.Fields[7].AsInteger;
    if Table1.State=dsBrowse then Table1.Edit;
    Table1.Fields[7].AsInteger:= Table1.Fields[7].AsInteger + StrToInt(InputBox('Birim
Fiyat','Kâr Artışı','0'));
```

```
   Table1.Fields[5].AsFloat:=
(Table1.Fields[5].AsFloat/(1+yedek/100))*(1+Table1.Fields[7].AsInteger/100);
end;
procedure Tflcikfrm.DataSource1DataChange(Sender: TObject; Field: TField);
begin
   DBEdit5.Hint:= 'Kâr = %'+FloatToStr(Table1.Fields[7].AsInteger);
end;
procedure Tflcikfrm.DBGrid1DblClick(Sender: TObject);
begin
   Table1.IndexName:='';
   Table1.FindNearest([InputBox('Kayıt Arama','Ürün Kodu','')]);
end;
procedure Tflcikfrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
     if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
       mtConfirmation,[mbYes,mbNo],0)=mrYes then Table1.Cancel
     else action:= caNone;
   end;
end;
procedure Tflcikfrm.Table1BeforeEdit(DataSet: TDataSet);
begin
   tarih:= Table1.Fields[2].AsDateTime;
end;
procedure Tflcikfrm.Table1BeforeDelete(DataSet: TDataSet);
var kont:boolean;
begin
   stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
   stokhfrm.RadioButton1.Checked:= true;
   stokhfrm.Table1.IndexName:='';
   stokhfrm.Table1.Filtered:= true;
   stokhfrm.Table1.IndexName:='TARIHX';
   stokhfrm.Table1.FindKey([Table1.Fields[2].AsDateTime]);
```

```
  kont:=false;
  repeat
      if stokhfrm.Table1.Fields[2].AsString='C' then begin
        if stokhfrm.Table1.Fields[4].AsString='' then
          kont:= true
        else stokhfrm.Table1.Next;
      end
      else stokhfrm.Table1.Next;
   until kont=true;
   stokhfrm.Table1.Delete;
end;
procedure Tflcikfrm.RxDBComboEdit1Exit(Sender: TObject);
begin
   if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
     if stokfrm.Table1.FindKey([RxDBComboEdit1.Text])=false then begin
       ShowMessage('Bu ürün stok kartlarında tanımlı değil !');
       RxDBComboEdit1.SetFocus;
     end
     else begin
        Table1.Fields[1].AsString:= stokfrm.Table1.Fields[1].AsString;
        Table1.Fields[4].AsString:= stokfrm.Table1.FieldByName('BIRIM').AsString;
        Table1.Fields[5].AsFloat:=
stokfrm.Table1.FieldByName('BIRIM_FIYAT').AsFloat*1.20;
      end;
   end;
end;
procedure Tflcikfrm.RxDBComboEdit1KeyPress(Sender: TObject; var Key: Char);
begin
   if Key=#13 then begin
     Key:= #0;
     Perform(WM_NEXTDLGCTL,0,0);
   end;
end;
```

```
procedure Tflcikfrm.RxDBComboEdit1ButtonClick(Sender: TObject);
begin
   ekfrm.RadioButton4.Checked:= true;
   ekfrm.showModal;
end;
end.
```

3.9. Unit Figiris

This form provides us to get the actual entry. It shows the date that we get the product and howmoney product that we buy and howmuch money that we pay it for. It adds the a mount of product that we buy to the stock.

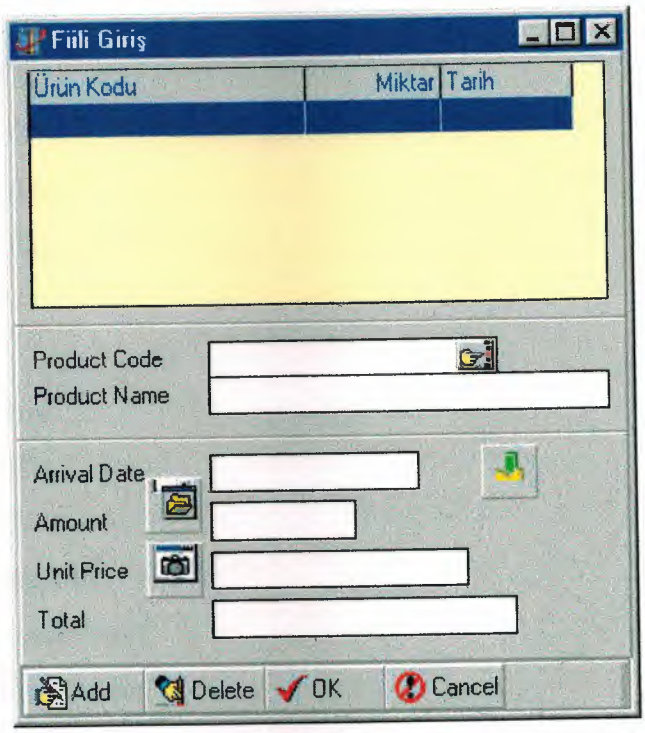

Figure 3.9. Actual Entry

```
unit figiris;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids, DBCtrls, StdCtrls, Mask, ExtCtrls, Db, DBTables,
  ComCtrls, ToolWin, Buttons, ToolEdit, RXDBCtrl, ImgList;
```

```
type
  Tflgirfrm = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Panel2: TPanel;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    DBText1: TDBText;
    Label5: TLabel;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    Label6: TLabel;
    Panel3: TPanel;
    DBGrid1: TDBGrid;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ImageList1: TImageList;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    Table1URUN_KOD: TStringField;
    Table1GEL_TAR: TDateField;
    Table1URUN_AD: TStringField;
    Table1MIKTAR: TFloatField;
    Table1BIRIM: TStringField;
    Table1BIRIM_FIYAT: TFloatField;
```

```
Table1TUTAR: TFloatField;
RxDBComboEdit1: TRxDBComboEdit;
procedure ToolButton1Click(Sender: TObject);
procedure ToolButton2Click(Sender: TObject);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton4Click(Sender: TObject);
procedure Table1BeforePost(DataSet: TDataSet);
procedure DataSource1StateChange(Sender: TObject);
procedure Table1AfterInsert(DataSet: TDataSet);
procedure DBGrid1DblClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Table1BeforeEdit(DataSet: TDataSet);
procedure Table1BeforeDelete(DataSet: TDataSet);
procedure RxDBComboEdit1Exit(Sender: TObject);
procedure RxDBComboEdit1ButtonClick(Sender: TObject);
procedure RxDBComboEdit1KeyPress(Sender: TObject; var Key: Char);
private
  { Private declarations }
public
tarih:TDateTime;
  { Public declarations }
end;
var
 flgirfrm: Tflgirfrm;
implementation
uses stokkart, ek2, ek, stokhar;
{$R *.DFM}
procedure Tflgirfrm.ToolButton1Click(Sender: TObject);
begin
    Table1.Append;
end;
procedure Tflgirfrm.ToolButton2Click(Sender: TObject);
begin
```

```
  if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz ?',
    mtConfirmation,[mbYes,mbNo],0)=mrYes then
    Table1.Delete;
end;
procedure Tflgirfrm.ToolButton3Click(Sender: TObject);
begin
  Table1.Post;
end;
procedure Tflgirfrm.ToolButton4Click(Sender: TObject);
begin
  Table1.Cancel;
end;
procedure Tflgirfrm.Table1BeforePost(DataSet: TDataSet);
var kontrol,kont:boolean;
begin
  Table1.Fields[6].AsFloat:= Table1.Fields[3].AsFloat*Table1.Fields[5].AsFloat;
  if Table1.State=dsInsert then begin
    stokhfrm.Table1.Append;
    stokhfrm.Table1.Fields[0].AsString:= Table1.Fields[0].AsString;
    stokhfrm.Table1.Fields[1].AsString:= Table1.Fields[2].AsString;
    stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[1].AsDateTime;
    stokhfrm.Table1.Fields[2].AsString:= 'G';
    stokhfrm.Table1.Fields[5].AsFloat:= Table1.Fields[3].AsFloat;
    stokhfrm.Table1.Fields[6].AsString:= Table1.Fields[4].AsString;
    stokhfrm.Table1.Fields[7].AsFloat:= Table1.Fields[5].AsFloat;
    stokhfrm.Table1.Fields[8].AsFloat:= Table1.Fields[6].AsFloat;
    stokhfrm.Table1.Post;
  end
  else if Table1.State=dsEdit then begin
    stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
    stokhfrm.RadioButton1.Checked:= true;
    stokhfrm.Table1.IndexName:='';
    stokhfrm.Table1.Filtered:= true;
```

```
        stokhfrm.Table1.IndexName:='TARIHX';
        if tarih=Table1.Fields[1].AsDateTime then begin
          stokhfrm.Table1.FindKey([Table1.Fields[1].AsDateTime]);
          kontrol:= true;
        end
        else begin
            stokhfrm.Table1.FindKey([tarih]);
            kontrol:= false;
        end;
        kont:=false;
        repeat
            if stokhfrm.Table1.Fields[2].AsString='G' then begin
              if stokhfrm.Table1.Fields[4].AsString='' then
                kont:= true
              else stokhfrm.Table1.Next;
            end
            else stokhfrm.Table1.Next;
        until kont=true;
        stokhfrm.Table1.Edit;
        if kontrol=false then
          stokhfrm.Table1.Fields[3].AsDateTime:= Table1.Fields[1].AsDateTime;
        stokhfrm.Table1.Fields[5].AsFloat:= Table1.Fields[3].AsFloat;
        stokhfrm.Table1.Fields[6].AsString:= Table1.Fields[4].AsString;
        stokhfrm.Table1.Fields[7].AsFloat:= Table1.Fields[5].AsFloat;
        stokhfrm.Table1.Fields[8].AsFloat:= Table1.Fields[6].AsFloat;
        stokhfrm.Table1.Post;
    end;
end;
procedure Tflgirfrm.DataSource1StateChange(Sender: TObject);
begin
    case Table1.State of
    dsInsert:       begin
                ToolButton1.Enabled:= False;
```

```
            ToolButton2.Enabled:= False;
            ToolButton3.Enabled:= true;
            ToolButton4.Enabled:= true;
            RxDBComboEdit1.ReadOnly:= false;
            RxDBComboEdit1.Button.Visible:=true;
            end;
    dsEdit:     begin
            ToolButton1.Enabled:= False;
            ToolButton2.Enabled:= False;
            ToolButton3.Enabled:= true;
            ToolButton4.Enabled:= true;
            end;
    dsBrowse:     begin
            ToolButton3.Enabled:= False;
            ToolButton4.Enabled:= False;
            ToolButton1.Enabled:= true;
            ToolButton2.Enabled:= true;
            RxDBComboEdit1.Button.Visible:=False;
            RxDBComboEdit1.ReadOnly:= true;
            end;
    end;
end;
procedure Tflgirfrm.Table1AfterInsert(DataSet: TDataSet);
begin
    RxDBComboEdit1.SetFocus;
end;
procedure Tflgirfrm.DBGrid1DblClick(Sender: TObject);
begin
    Table1.IndexName:='';
    Table1.FindNearest([InputBox('Kayıt Arama','Ürün Kodu','')]);
end;
procedure Tflgirfrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
```

```
  if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
    if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
      mtConfirmation,[mbYes,mbNo],0)=mrYes then Table1.Cancel
    else action:= caNone;
  end;
end;
procedure Tflgirfrm.Table1BeforeEdit(DataSet: TDataSet);
begin
  tarih:= Table1.Fields[1].AsDateTime;
end;
procedure Tflgirfrm.Table1BeforeDelete(DataSet: TDataSet);
var kont:boolean;
begin
  stokhfrm.Edit1.Text:= Table1.Fields[0].AsString;
  stokhfrm.RadioButton1.Checked:= true;
  stokhfrm.Table1.IndexName:='';
  stokhfrm.Table1.Filtered:= true;
  stokhfrm.Table1.IndexName:='TARIHX';
  stokhfrm.Table1.FindKey([Table1.Fields[1].AsDateTime]);
  kont:=false;
  repeat
    if stokhfrm.Table1.Fields[2].AsString='G' then begin
      if stokhfrm.Table1.Fields[4].AsString='' then
        kont:= true
      else stokhfrm.Table1.Next;
    end
    else stokhfrm.Table1.Next;
  until kont=true;
  stokhfrm.Table1.Delete;
end;
procedure Tflgirfrm.RxDBComboEdit1Exit(Sender: TObject);
begin
  if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
```

```
      if stokfrm.Table1.FindKey([RxDBComboEdit1.Text])=false then begin
        ShowMessage('Bu ürün stok kartlarında tanımlı değil !');
        RxDBComboEdit1.SetFocus;
      end
      else begin
          Table1.Fields[2].AsString:= stokfrm.Table1.Fields[1].AsString;
          Table1.Fields[4].AsString:= stokfrm.Table1.FieldByName('BIRIM').AsString;
          Table1.Fields[5].AsFloat:=
stokfrm.Table1.FieldByName('BIRIM_FIYAT').AsFloat;
       end;
    end;
end;
procedure Tflgirfrm.RxDBComboEdit1ButtonClick(Sender: TObject);
begin
    ekfrm.RadioButton2.Checked:= true;
    ekfrm.showModal;
end;
procedure Tflgirfrm.RxDBComboEdit1KeyPress(Sender: TObject; var Key: Char);
begin
    if Key=#13 then begin
      Key:= #0;
      Perform(WM_NEXTDLGCTL,0,0);
    end;
end;
end.
```

3.10. Firma

It provides the new firm information to record.

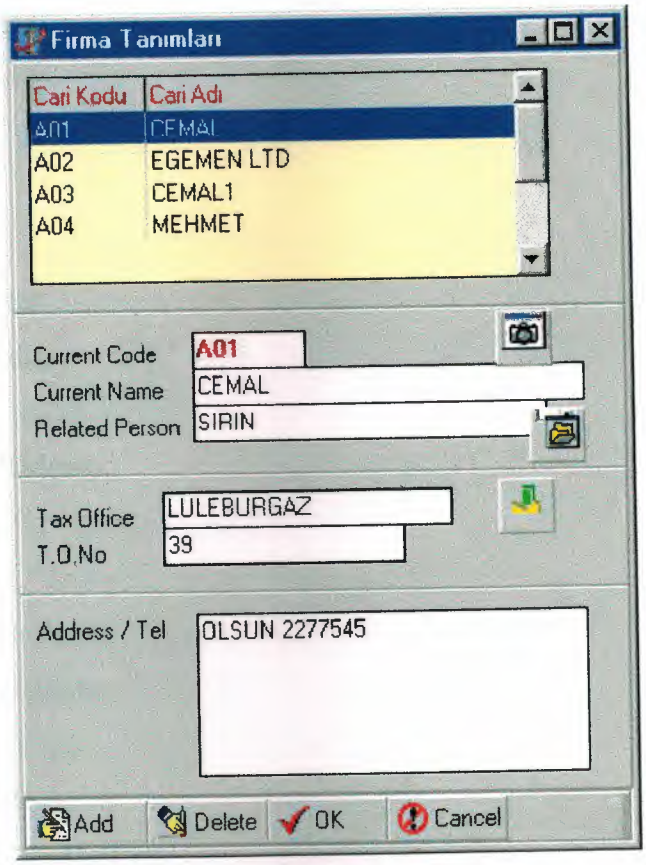

Figure 3.10. Firm Definitions

```
unit firma;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, ToolWin, Grids, DBGrids, StdCtrls, DBCtrls, Mask, Db, DBTables,
  ExtCtrls, ImgList;
type
  Tfirmafrm = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
    Table1: TTable;
    DataSource1: TDataSource;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    Panel2: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    Panel3: TPanel;
    DBMemo1: TDBMemo;
    Label6: TLabel;
    Panel4: TPanel;
    DBGrid1: TDBGrid;
    ImageList1: TImageList;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    procedure ToolButton1Click(Sender: TObject);
    procedure ToolButton2Click(Sender: TObject);
    procedure ToolButton3Click(Sender: TObject);
    procedure ToolButton4Click(Sender: TObject);
    procedure DBEdit1KeyPress(Sender: TObject; var Key: Char);
    procedure DataSource1StateChange(Sender: TObject);
    procedure Table1AfterInsert(DataSet: TDataSet);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
```

```
 public
   { Public declarations }
 end;
var
 firmafrm: Tfirmafrm;
implementation
{$R *.DFM}
procedure Tfirmafrm.ToolButton1Click(Sender: TObject);
var
  yedek:string;
  KeyState : TKeyboardState;
begin
    GetKeyboardState(KeyState);
    if (KeyState[VK_CAPITAL] = 0) then
      KeyState[VK_CAPITAL] := 1;
    SetKeyboardState(KeyState);
    yedek:= InputBox('Yeni Kayıt','Cari Kod','');
    if yedek<>'' then begin
      Table1.IndexName:= '';
      if Table1.FindKey([yedek])=false then begin
        Table1.Append;
        Table1.Fields[0].AsString:= yedek;
        DBEdit2.SetFocus;
      end
      else ShowMessage('Bu firma sistemde tanımlı !');
    end;
 end;
 procedure Tfirmafrm.ToolButton2Click(Sender: TObject);
 begin
    if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz ?',
      mtConfirmation,[mbYes,mbNo],0)=mrYes then
      Table1.Delete;
 end;
```

```
procedure Tfirmafrm.ToolButton3Click(Sender: TObject);
begin
   Table1.Post;
end;
procedure Tfirmafrm.ToolButton4Click(Sender: TObject);
begin
   Table1.Cancel;
end;
procedure Tfirmafrm.DBEdit1KeyPress(Sender: TObject; var Key: Char);
begin
   if Key=#13 then begin
     key:= #0;
     Perform(WM_NEXTDLGCTL,0,0);
   end;
end;
procedure Tfirmafrm.DataSource1StateChange(Sender: TObject);
begin
   case Table1.State of
   dsEdit,dsInsert: begin
                ToolButton3.Enabled:= true;
                ToolButton4.Enabled:= true;
                ToolButton1.Enabled:= false;
                ToolButton2.Enabled:= false;
                end;
   dsBrowse:       begin
                ToolButton1.Enabled:= true;
                ToolButton2.Enabled:= true;
                ToolButton3.Enabled:= false;
                ToolButton4.Enabled:= false;
                end;
   end;
end;
procedure Tfirmafrm.Table1AfterInsert(DataSet: TDataSet);
```

```
begin
   DBEdit1.SetFocus;
end;
procedure Tfirmafrm.DBGrid1DblClick(Sender: TObject);
begin
   Table1.IndexName:='';
   Table1.FindNearest([InputBox('Kayıt Arama','Cari Kod','')]);
end;
procedure Tfirmafrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
     if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
       mtConfirmation,[mbYes,mbNo],0)=mrYes then Table1.Cancel
     else action:= caNone;
    end;
end;
end.
```

3.11. Unit Giris

This form is same as about form. It shows who prepared this program.

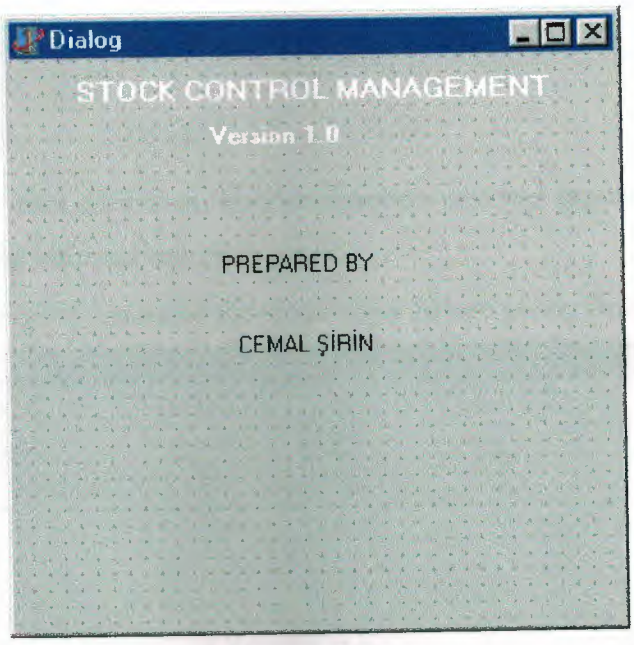

Figure 3.11. Dialog

```
unit giris;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;
type
  Tgirisfrm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  girisfrm: Tgirisfrm;
implementation
{$R *.DFM}
end.
```

3.12. Unit Main

In this stock entry, firm definition, invoice entrance-exit and report processes is made. We can follow all the informations of to stock.

Remark: We can not make the stock entry before giving firm definitions.

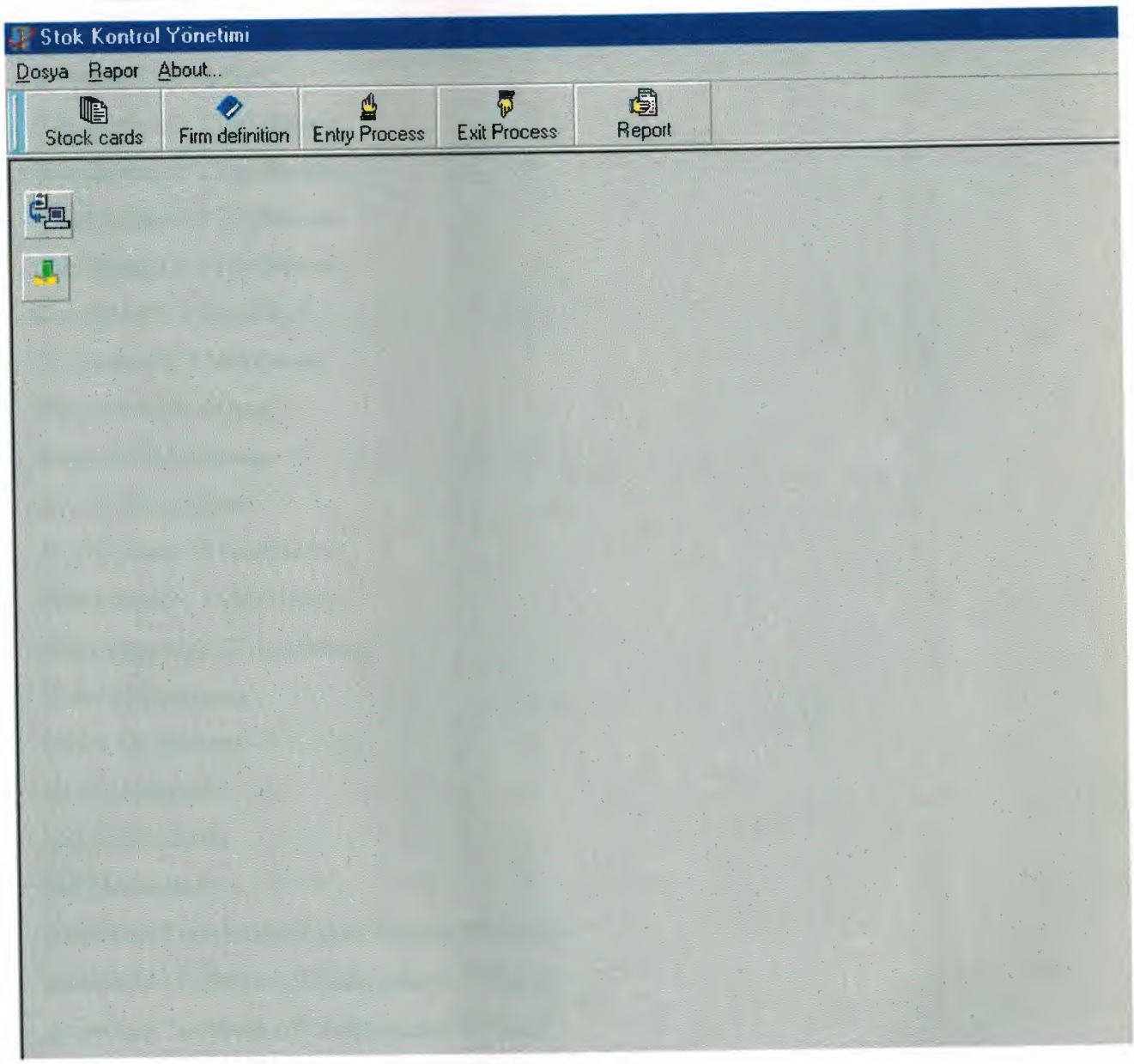

Figure 3.12. Unit Main

```
unit main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, ToolWin, Menus, ExtCtrls, ImgList;
type
  Tmainfrm = class(TForm)
    ImageList1: TImageList;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton3: TToolButton;
    ToolButton5: TToolButton;
    ToolButton7: TToolButton;
    ScrollBox1: TScrollBox;
    MainMenu1: TMainMenu;
    Dosya1: TMenuItem;
    Rapor1: TMenuItem;
    About: TMenuItem;
    ToolButton2: TToolButton;
    StokKartlar1: TMenuItem;
    FirmaTanmlar1: TMenuItem;
    Giris: TMenuItem;
    Cikis: TMenuItem;
    N1: TMenuItem;
    N3: TMenuItem;
    k1: TMenuItem;
    procedure ToolButton1Click(Sender: TObject);
    procedure ToolButton3Click(Sender: TObject);
    procedure ToolButton2Click(Sender: TObject);
    procedure ToolButton5Click(Sender: TObject);
    procedure StokKartlar1Click(Sender: TObject);
```

```
  procedure FirmaTanmlar1Click(Sender: TObject);
  procedure GirisClick(Sender: TObject);
  procedure CikisClick(Sender: TObject);
  procedure k1Click(Sender: TObject);
  procedure ToolButton7Click(Sender: TObject);
  procedure AboutClick(Sender: TObject);
  procedure Rapor1Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 mainfrm: Tmainfrm;
implementation
uses stokkart,fatgiris, firma, figiris, fatcikis, ficikis, stokhar, abaut,
 dialog, stokbak, dialog2;
{$R *.DFM}
procedure Tmainfrm.ToolButton1Click(Sender: TObject);
begin
   stokfrm.ShowModal;
end;
procedure Tmainfrm.ToolButton3Click(Sender: TObject);
begin
   dlgfrm.RadioButton1.Visible:= true;
   dlgfrm.RadioButton1.Checked:= true;
   dlgfrm.RadioButton2.Visible:= true;
   dlgfrm.RadioButton3.Visible:= false;
   dlgfrm.RadioButton4.Visible:= false;
   dlgfrm.ShowModal;
end;
procedure Tmainfrm.ToolButton2Click(Sender: TObject);
begin
```

```
    firmafrm.ShowModal;
end;
procedure Tmainfrm.ToolButton5Click(Sender: TObject);
begin
    dlgfrm.RadioButton3.Visible:= true;
    dlgfrm.RadioButton3.Checked:= true;
    dlgfrm.RadioButton4.Visible:= true;
    dlgfrm.RadioButton1.Visible:= false;
    dlgfrm.RadioButton2.Visible:= false;
    dlgfrm.ShowModal;
end;
procedure Tmainfrm.StokKartlar1Click(Sender: TObject);
begin
    ToolButton1Click(Self);
end;
procedure Tmainfrm.FirmaTanmlar1Click(Sender: TObject);
begin
    ToolButton2Click(self);
end;
procedure Tmainfrm.GirisClick(Sender: TObject);
begin
    ToolButton3Click(self);
end;
procedure Tmainfrm.CikisClick(Sender: TObject);
begin
    ToolButton5Click(self);
end;
procedure Tmainfrm.k1Click(Sender: TObject);
begin
    Close;
end;
procedure Tmainfrm.ToolButton7Click(Sender: TObject);
begin
```

```
    dlgfrm2.RadioButton1.Checked:=true;
    dlgfrm2.ShowModal;
end;
procedure Tmainfrm.AboutClick(Sender: TObject);
begin
    AboutBox.ShowModal;
end;
procedure Tmainfrm.Rapor1Click(Sender: TObject);
begin
    ToolButton7Click(self);
end;
end.
```

3.13. Unit Stokbak

We con follow all the product in the form.

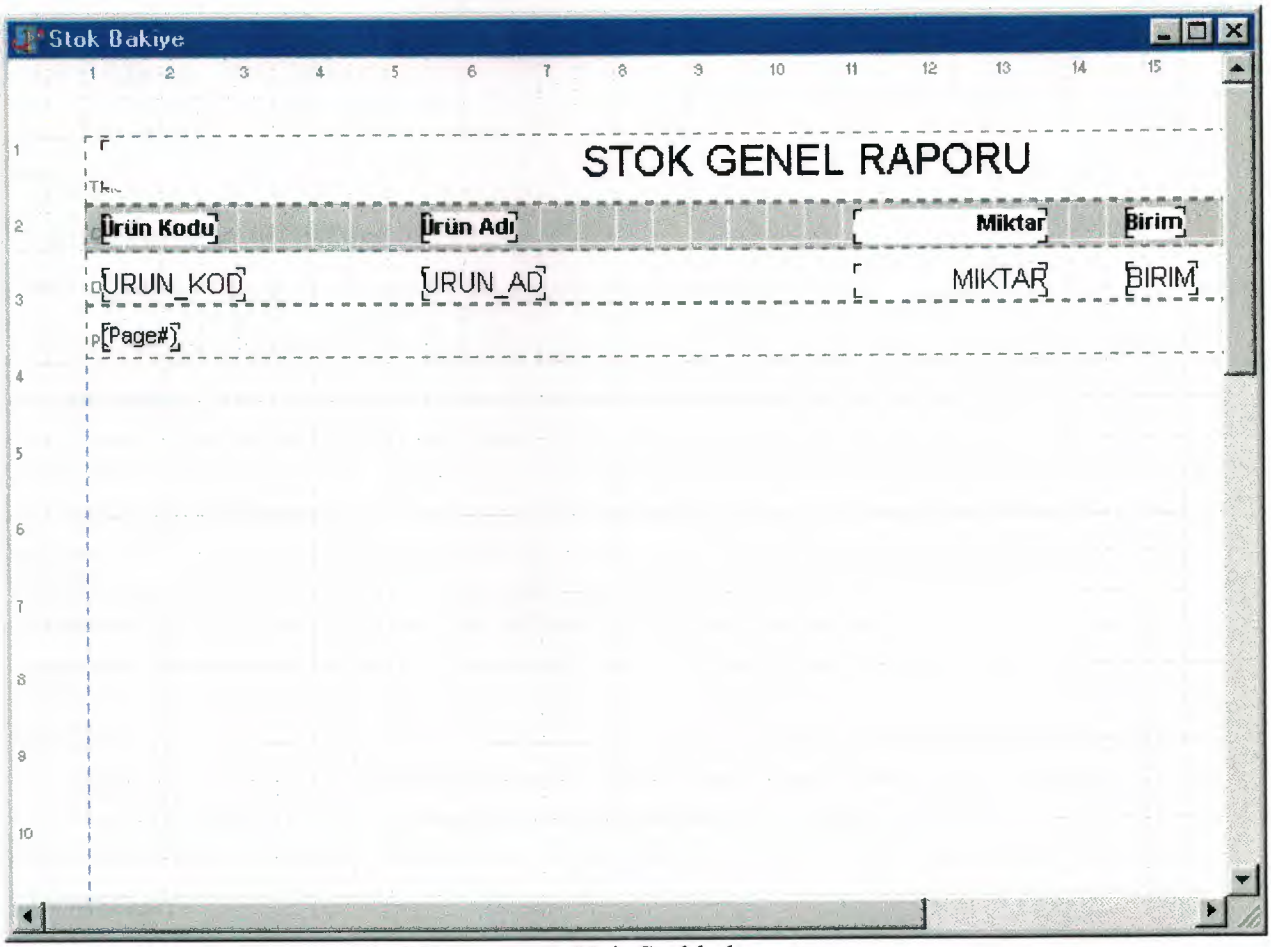

Figure 3.13. Unit Stokbak

```
unit stokbak;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Qrctrls, quickrpt, ExtCtrls, Db, DBTables;
type
  Tstokbfrm = class(TForm)
    QuickRep1: TQuickRep;
    ColumnHeaderBand1: TQRBand;
    DetailBand1: TQRBand;
    PageFooterBand1: TQRBand;
    TitleBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    QRLabel4: TQRLabel;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRDBText4: TQRDBText;
    QRLabel5: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData2: TQRSysData;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  stokbfrm: Tstokbfrm;
implementation
uses stokhar;
end.
```

3.14. Unit Stokhar

We can get extra information from any product in this form.

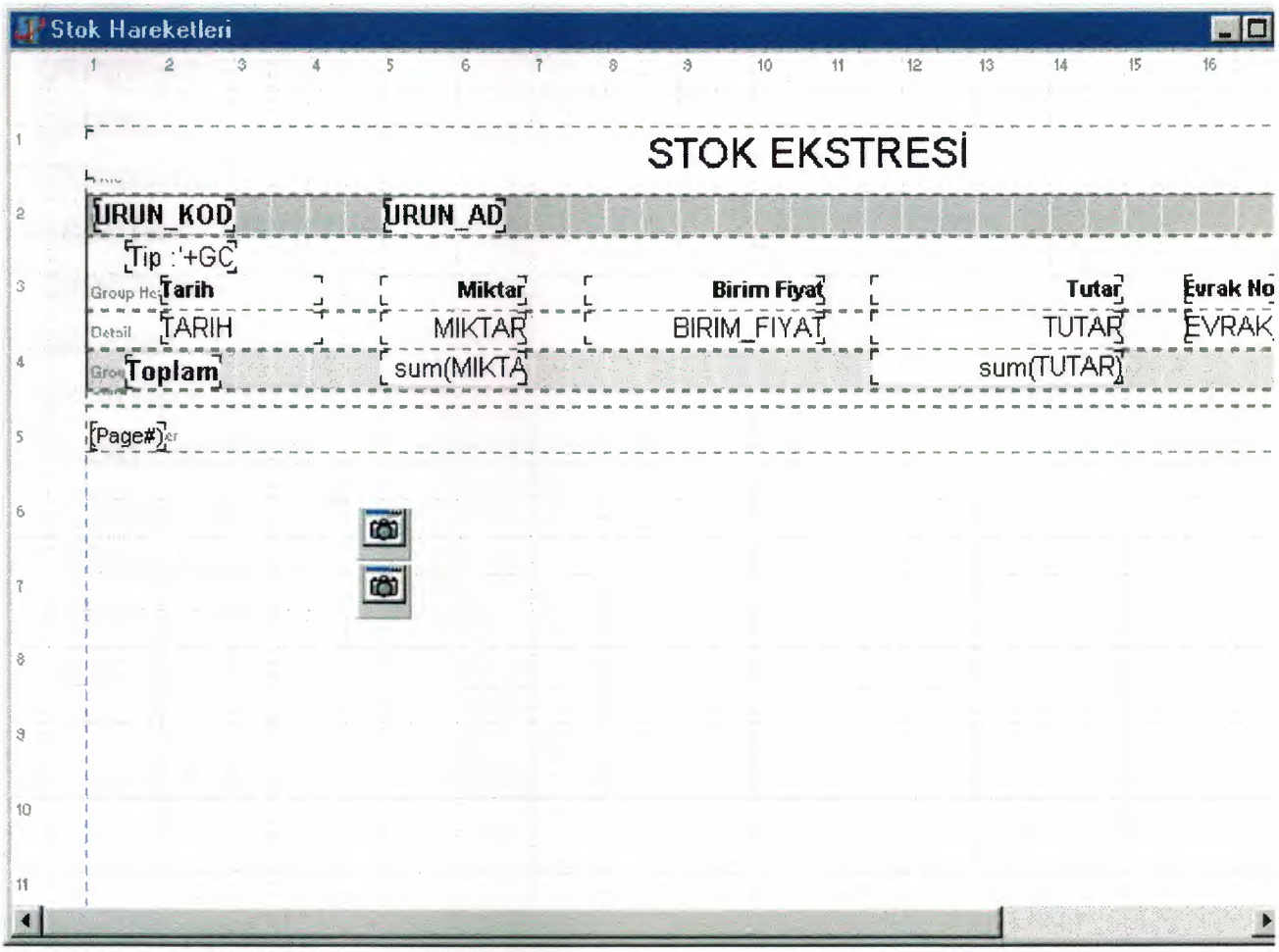

Figure 3.14. Unit Stokhar

```
unit stokhar;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Db, Grids, DBGrids, ExtCtrls, DBTables, StdCtrls, quickrpt, Qrctrls;
type
  Tstokhfrm = class(TForm)
    Table1: TTable;
    Table2: TTable;
    Table2URUN_KOD: TStringField;
    Table2URUN_AD: TStringField;
```

Table2MIKTAR: TFloatField;
Table2BIRIM: TStringField;
QuickRep1: TQuickRep;
DetailBand1: TQRBand;
QRDBText3: TQRDBText;
QRDBText4: TQRDBText;
QRDBText5: TQRDBText;
QRDBText2: TQRDBText;
QRDBText6: TQRDBText;
QRGroup1: TQRGroup;
QRDBText1: TQRDBText;
QRDBText7: TQRDBText;
QRGroup2: TQRGroup;
QRExpr2: TQRExpr;
QRLabel1: TQRLabel;
QRLabel2: TQRLabel;
QRLabel3: TQRLabel;
QRLabel5: TQRLabel;
QRLabel6: TQRLabel;
QRBand2: TQRBand;
QRExpr1: TQRExpr;
QRLabel4: TQRLabel;
QRExpr3: TQRExpr;
QRBand1: TQRBand;
QRChildBand1: TQRChildBand;
QRBand3: TQRBand;
QRSysData1: TQRSysData;
QRSysData2: TQRSysData;
TitleBand1: TQRBand;
QRLabel7: TQRLabel;
Edit1: TEdit;
RadioButton1: TRadioButton;
Table1URUN_KOD: TStringField;

```
    Table1URUN_AD: TStringField;
    Table1GC: TStringField;
    Table1TARIH: TDateField;
    Table1EVRAK_NO: TStringField;
    Table1MIKTAR: TFloatField;
    Table1BIRIM: TStringField;
    Table1BIRIM_FIYAT: TFloatField;
    Table1TUTAR: TFloatField;
    procedure Table1FilterRecord(DataSet: TDataSet; var Accept: Boolean);
    procedure Table1BeforePost(DataSet: TDataSet);
    procedure Table1BeforeEdit(DataSet: TDataSet);
    procedure Table1BeforeDelete(DataSet: TDataSet);
    procedure Table2AfterPost(DataSet: TDataSet);
  private
    { Private declarations }
  public
  yedek:double;
    { Public declarations }
  end;
var
  stokhfrm: Tstokhfrm;
implementation
{$R *.DFM}
procedure Tstokhfrm.Table1FilterRecord(DataSet: TDataSet;
  var Accept: Boolean);
begin
    if RadioButton1.Checked=true then begin
      if Table1.Fields[0].AsString=Edit1.Text then
        accept:= true
      else accept:= false;
    end
    else begin
       if Table1.Fields[4].AsString=Edit1.Text then
```

```
        accept:= true
      else accept:= false;
    end;
end;
procedure Tstokhfrm.Table1BeforePost(DataSet: TDataSet);
var tip:boolean;
begin
    if Table1.Fields[2].AsString='G' then tip:=true
    else tip:= false;
    if Table1.State=dsInsert then begin
      if Table2.FindKey([Table1.Fields[0].AsString])=false then
        Table2.Append
      else Table2.Edit;
      Table2.Fields[0].AsString:= Table1.Fields[0].AsString;
      Table2.Fields[1].AsString:= Table1.Fields[1].AsString;
      if tip=true then
        Table2.Fields[2].AsFloat:= Table2.Fields[2].AsFloat+Table1.Fields[5].AsFloat
      else Table2.Fields[2].AsFloat:=Table2.Fields[2].AsFloat-Table1.Fields[5].AsFloat;
      Table2.Fields[3].AsString:= Table1.Fields[6].AsString;
      Table2.Post;
    end
    else if Table1.State=dsEdit then begin
      if Table2.FindKey([Table1.Fields[0].AsString]) then Table2.Edit
      else begin
          Table2.Append;
          Table2.Fields[0].AsString:= Table1.Fields[0].AsString;
          Table2.Fields[1].AsString:= Table1.Fields[1].AsString;
          Table2.Fields[3].AsString:= Table1.Fields[6].AsString;
      end;
      if tip=true then
    Table2.Fields[2].AsFloat:=Table2.Fields[2].AsFloat-
yedek+Table1.Fields[5].AsFloat
```

```
      elseTable2.Fields[2].AsFloat:=Table2.Fields[2].AsFloat+yedek-
Table1.Fields[5].AsFloat;
      Table2.Post;
    end;
end;
procedure Tstokhfrm.Table1BeforeEdit(DataSet: TDataSet);
begin
    yedek:= Table1.Fields[5].AsFloat;
end;
procedure Tstokhfrm.Table1BeforeDelete(DataSet: TDataSet);
begin
    if Table2.FindKey([Table1.Fields[0].AsString]) then Table2.Edit
    else begin
        Table2.Append;
        Table2.Fields[0].AsString:= Table1.Fields[0].AsString;
        Table2.Fields[1].AsString:= Table1.Fields[1].AsString;
        Table2.Fields[3].AsString:= Table1.Fields[6].AsString;
    end;
    if Table1.Fields[2].AsString='G' then
      Table2.Fields[2].AsFloat:= Table2.Fields[2].AsFloat-Table1.Fields[5].AsFloat
    else Table2.Fields[2].AsFloat:= Table2.Fields[2].AsFloat+Table1.Fields[5].AsFloat;
    Table2.Post;
end;
procedure Tstokhfrm.Table2AfterPost(DataSet: TDataSet);
begin
    if Table2.Fields[2].AsFloat=0 then Table2.Delete;
end;
end.
```

3.15. Unit Stokkart

In this form stock entry is made. The information that we record it before is kept in this form. Whom the product is bought from, and we can calculate the cost price.

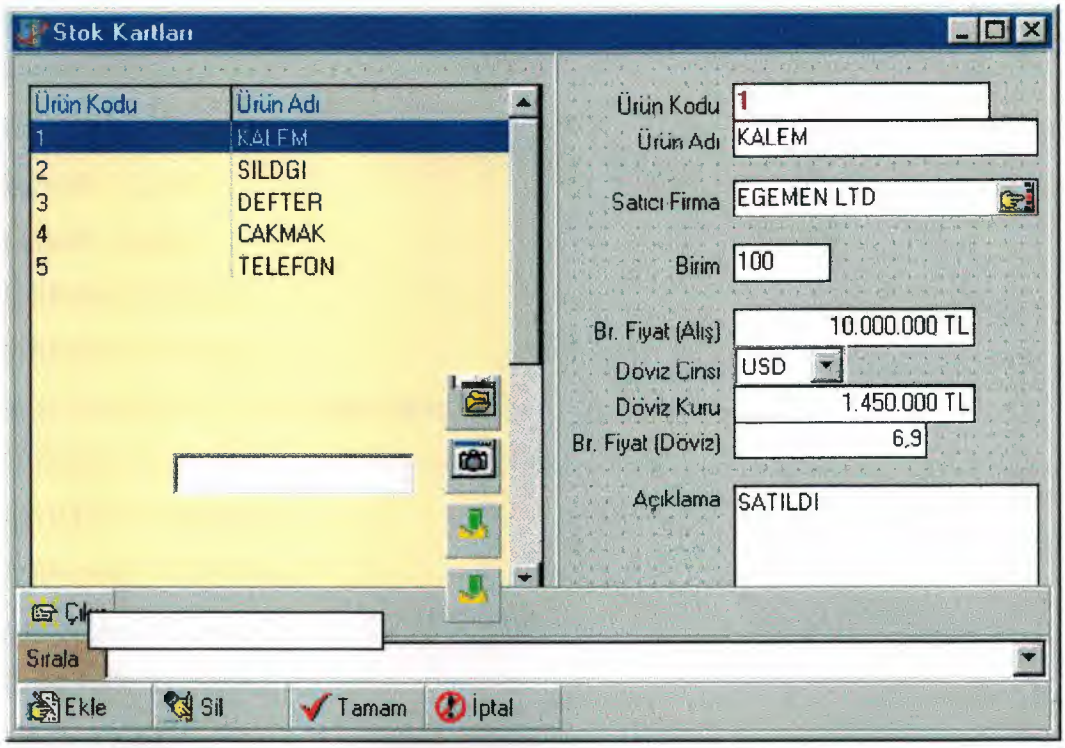

Figure 3.15. Stock Card

```
unit stokkart;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DBCtrls, Mask, Grids, DBGrids, Db, DBTables, ExtCtrls,
  ComCtrls, ToolWin, Buttons, ToolEdit, RXDBCtrl, ImgList;
type
  Tstokfrm = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
```

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBComboBox1: TDBComboBox;

DBEdit6: TDBEdit;

DBEdit7: TDBEdit;

DBMemo1: TDBMemo;

ImageList1: TImageList;

CoolBar1: TCoolBar;

Edit1: TEdit;

ToolBar1: TToolBar;

ToolButton1: TToolButton;

ToolButton2: TToolButton;

ToolButton3: TToolButton;

ToolButton4: TToolButton;

ToolBar2: TToolBar;

ToolButton5: TToolButton;

ComboBox1: TComboBox;

Label10: TLabel;

ImageList2: TImageList;

Edit2: TEdit;

Bevel1: TBevel;

Bevel2: TBevel;

Table1URUN_KOD: TStringField;

Table1URUN_AD: TStringField;

```
  Table1SATICI_FIRMA: TStringField;
  Table1BIRIM: TStringField;
  Table1BIRIM_FIYAT: TFloatField;
  Table1DOVIZ_CINS: TStringField;
  Table1KUR: TFloatField;
  Table1BIRIM_FIYAT2: TFloatField;
  Table1ACIKLAMA: TMemoField;
  RxDBComboEdit1: TRxDBComboEdit;
  procedure ToolButton1Click(Sender: TObject);
  procedure ToolButton2Click(Sender: TObject);
  procedure ToolButton3Click(Sender: TObject);
  procedure ToolButton4Click(Sender: TObject);
  procedure DataSource1StateChange(Sender: TObject);
  procedure Edit1Change(Sender: TObject);
  procedure DBEdit1KeyPress(Sender: TObject; var Key: Char);
  procedure DBEdit5Exit(Sender: TObject);
  procedure DBEdit7Exit(Sender: TObject);
  procedure DBEdit6Exit(Sender: TObject);
  procedure ComboBox1Change(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure ToolButton5Click(Sender: TObject);
  procedure Table1FilterRecord(DataSet: TDataSet; var Accept: Boolean);
  procedure Table1BeforeInsert(DataSet: TDataSet);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure RxDBComboEdit1Exit(Sender: TObject);
  procedure RxDBComboEdit1ButtonClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 stokfrm: Tstokfrm;
```

```
implementation
uses ek2, firma, fatgiris, ek, fatcikis;
{$R *.DFM}
procedure Tstokfrm.ToolButton1Click(Sender: TObject);
var
  kayit:string;
  KeyState : TKeyboardState;
begin
    GetKeyboardState(KeyState);
    if (KeyState[VK_CAPITAL] = 0) then
      KeyState[VK_CAPITAL] := 1;
    SetKeyboardState(KeyState);
    kayit:= InputBox('Yeni Kayıt','Stok Kodu :','');
    if kayit<>''then begin
      Table1.IndexName:='';
      ComboBox1.ItemIndex:=0;
      if Table1.FindKey([kayit])=false then begin
        Table1.Append;
        Table1.Fields[0].AsString:= kayit;
        DBEdit2.SetFocus;
      end
      else ShowMessage('Bu ürün zaten stok kartlarında tanımlı !');
    end;
end;
procedure Tstokfrm.ToolButton2Click(Sender: TObject);
begin
    if MessageDlg('Bu kaydı silmek istediğinizden emin misiniz ?',
      mtConfirmation,[mbYes,mbNo],0)=mrYes then
      Table1.Delete;
end;
procedure Tstokfrm.ToolButton3Click(Sender: TObject);
begin
    Table1.Post;
```

```
end;
procedure Tstokfrm.ToolButton4Click(Sender: TObject);
begin
  Table1.Cancel;
end;
procedure Tstokfrm.DataSource1StateChange(Sender: TObject);
begin
   case Table1.State of
   dsinsert,dsEdit: begin
                ToolButton1.Enabled:= False;
                ToolButton2.Enabled:= False;
                ToolButton3.Enabled:= True;
                ToolButton4.Enabled:= True;
                RxDBComboEdit1.Button.Visible:= true;
                end;
    dsBrowse:       begin
                ToolButton3.Enabled:= False;
                ToolButton4.Enabled:= False;
                RxDBComboEdit1.Button.Visible:= false;
                ToolButton1.Enabled:= True;
                ToolButton2.Enabled:= True;
                end;
   end;
end;
procedure Tstokfrm.Edit1Change(Sender: TObject);
begin
   Table1.FindNearest([Edit1.Text]);
end;
procedure Tstokfrm.DBEdit1KeyPress(Sender: TObject; var Key: Char);
begin
   if key=#13 then begin
     key:= #0;
     Perform(WM_NEXTDLGCTL,0,0);
```

```
    end;
end;
procedure Tstokfrm.DBEdit5Exit(Sender: TObject);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
    if (Table1.Fields[4].AsFloat<>0) and (Table1.Fields[6].AsFloat<>0) then
      Table1.Fields[7].AsFloat:=Table1.Fields[4].AsFloat/Table1.Fields[6].AsFloat;
    end;
end;
procedure Tstokfrm.DBEdit7Exit(Sender: TObject);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
    if (Table1.Fields[4].AsFloat=0) and (Table1.Fields[6].AsFloat=0) then begin
      ShowMessage('Lütfen döviz kurunu giriniz !');
      DBEdit6.SetFocus;
    end
    else if Table1.Fields[4].AsFloat=0 then
      Table1.Fields[4].AsFloat:= Table1.Fields[7].AsFloat*Table1.Fields[6].AsFloat;
    end;
end;
procedure Tstokfrm.DBEdit6Exit(Sender: TObject);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
    if (Table1.Fields[4].AsFloat<>0) and (Table1.Fields[6].AsFloat=0) then begin
      ShowMessage('Lütfen döviz kurunu giriniz !');
      DBEdit6.SetFocus;
    end
    else if (Table1.Fields[4].AsFloat<>0) and (Table1.Fields[6].AsFloat<>0) then
      Table1.Fields[7].AsFloat:=Table1.Fields[4].AsFloat/Table1.Fields[6].AsFloat;
    end;
end;
procedure Tstokfrm.ComboBox1Change(Sender: TObject);
begin
```

```
    case ComboBox1.ItemIndex of
    0: Table1.IndexName:= '';
    1: Table1.IndexName:= 'URUN_ADI';
    2: Table1.IndexName:= 'FIRMA';
    end;
end;
procedure Tstokfrm.FormCreate(Sender: TObject);
begin
    ComboBox1.ItemIndex:= 0;
end;
procedure Tstokfrm.ToolButton5Click(Sender: TObject);
begin
    stokfrm.Close;
end;
procedure Tstokfrm.Table1FilterRecord(DataSet: TDataSet;
  var Accept: Boolean);
begin
    if Table1.Fields[2].AsString=Edit2.Text then
       Accept:= true
    else Accept:= false;
end;
procedure Tstokfrm.Table1BeforeInsert(DataSet: TDataSet);
begin
    DBGrid1.SetFocus;
end;
procedure Tstokfrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
       if MessageDlg('Değişiklikleri kaydetmeden çıkmak istiyor musunuz?',
          mtConfirmation,[mbYes,mbNo],0)=mrYes then Table1.Cancel
       else action:= caNone;
    end;
 end;
```

```
procedure Tstokfrm.RxDBComboEdit1Exit(Sender: TObject);
begin
    if (Table1.State=dsEdit) or (Table1.State=dsInsert) then begin
      firmafrm.Table1.IndexName:= 'CARI_ADI';
      if firmafrm.Table1.FindKey([RxDBComboEdit1.Text])=false then begin
        ShowMessage('"Firma Tanımları" nda bu kayıt bulunamadı');
        RxDBComboEdit1.SetFocus;
      end;
    end;
end;
procedure Tstokfrm.RxDBComboEdit1ButtonClick(Sender: TObject);
begin
    ek2frm.RadioButton2.Checked:= true;
    ek2frm.ShowModal;
end;
end.
```

CONCLUSION

In my project I prepared a Stock Control Management by using Borland Delphi 5. We can record the necessary informations about firms, make stock cards and follow invoice and actual entry and exit. We can get any information about stock report.

There is several of places that we can use this program in real life, for example in factory, bookstore or anywhere that control the products entry and exit.

Delphi Programming Language provides when I started to write this program the things that I want to do. And I carried out this program and concluded successfully.

With this project I developed my information about Borland Delphi 5 and finding solution to some specific problems.

REFERENCES

[1] Borland Delphi 5 CD E:\Documentation\Developer's Guide

[2] Mastering Delphi 5 book

[3] http://www.borland.com/delphi