

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

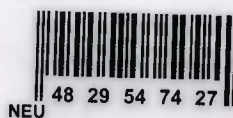
Car Gallery

**Graduation Project
COM- 400**

Student Name: Juma Al-khatib (20032650)

Supervisor: Assoc. Prof Dr. Adil Amirjanov

Nicosia - 2007





Acknowledgement.

Abstract.

Introduction.

Contents	Page No
Chapter 1: VISUAL BASIC	1
1.1 OVERVIEW	1
1.2 GRAFICAL CMPONENTS	2
1.3 COMMON COMMAND BUTTONS	4
1.4 FUNCTIONS	5
1.5 PRIVATE AND PUBLIC PROCEDURE	7
1.6 SHARING EVENT HANDLERS.....	8
1.7 EVENTS	9
1.8 OVERVIEW OF ARRAYS	10
 Chapter 2: DATA BASE.....	11
2.1 OVERVIEW	11
2.2 RELATIONAL MODEL	13
2.3 RELATIONAL OPERATIONS.....	14
2.4 OBJECT DATABASE MODELS	15
2.5 DATABASE TABLES	16
2.6 WHAT IS SQL	17
2.6.1 SQL KEYWORDS	17
2.6.2 DATA RETRIEVAL	17
2.6.3 DATA MANIPULATION	18

Chapter 3: ODBC.....	19
3.1 OPENING DATABASE.....	19
3.2 ADDING A RECORD TO A RECORD SET.....	19
3.3 EDITING A RECORD IN A RECORD SET.....	20
3.4 MOVING TO THE FIRST RECORD IN A RECORD SET	20
3.5 MOVING TO THE LAST RECORD IN A RECORD SET.....	20
3.6 DELETING A RECORD IN A RECORD SET.....	21
3.7 UPDATING A RECORD IN A RECORD SET.....	22
3.8 SORTING A RECORD SET	23
3.9 SEARCHING A RECORD SET.....	23
Chapter 4: CAR GALLERY	24
4.1 OVERVIEW	24
4.2 BLOCK DIAGRAM.....	25
4.3 SECURITY.....	26
4.4 MAIN FORM.....	27
4.5 FORM USER.....	28
4.6 FORM CAR.....	30
4.6.1 FORM ADD NEW CAR.....	30
4.6.2 SEARCH CAR FORM.....	32
4.6.3 SEARCH FOR A UNIQUE CAR.....	33
4.6.4 VIEW FORM	34
4.7 FORM EDIT A CAR.....	35
4.8 FORM SELL A CAR.....	36
4.9 CUSTOMER FORM.....	38
4.9.1 ADD CUSTOMER FORM.....	38
4.9.2 FORM SEARCH FOR A CUSTOMER.....	40
4.9.3 FORM EDIT FOR A CUSTOMER.....	42
CONCLUSION	43
REFERENCES.....	44

ACKNOLEDGMENT

First, I would like to thank Allah (God) for guiding me through my study and also a special thanks to my supervisor ASSOC. Prof. Dr. Adil Amirjanov Who has supervised this project and who has reviewed each chapter as it was produced, and he never deprived me from his helpful which without I will not have achieved this work. I would like also to thank my family that supports me any time that I needed. Especially to my father and mother and I wish from my god to give them along life. I would like also to thank all Drs. in the computer-engineering department for their incredible assistance. Finally, I would like to declare that is my responsibility if any mistake occurred in my document of either omissions or commissions.

Abstract

Car gallery is a business of buying and selling cars. The company that works in this business called car dealer. The dealer either buys brand new car import from the other countries or buys a second hand car from these clients. The project is to solve the dealer's main problems of keeping the record of all cars in the gallery wise and category wise. As well as at the same time we keep the record from whom the car was bought , when and for how much , as well as we do keep the record of those to whom we sell our car just in case we need to know where that car is now.

Day after day, the numbers of people that use cars are increasing rapidly, and the number of gallery companies is increasing too, the main role of these companies is SELLING AND BUYING CARS.

Buying cars is the process when a customer requests a certain cars, then the employee checks for available cars then the customer choose a car will be provided to him after payment.

Computers have become an essential part of every aspects of life, car gallery companies are using computers and computer systems to ease up the selling process.

INTRODUCTION

This project is talking about car gallery by using the visual basic program language and database access 2003.

It is explained in my chapters about how visual basic work and which data base it's used and how the connection is occurs between visual basic and Microsoft access 2003.

This project contains four chapters:

The first chapter describes the general information about visual basic programming language and variable scope (including object variables), overview about function in visual basic.

The second chapter talks about the data base how to construct it, and some explanations about my tables, SQL statement: insert, delete, update...etc.

The third chapter describes the connection between database and visual basic, how they are working together in the program.

The forth chapter represents my program, explained step by step, and first showed the whole project in a big block diagram. Then discussed how to insert, delete, update, sell and search for a car and customer in the gallery.

CHAPTER 1

VISUAL BASIC

1.1 OVERVIEW:

Visual Basic (VB) is an event driven programming language and associated development environment from Microsoft. VB has been replaced by Visual Basic .NET. The older version of VB was derived heavily from BASIC and enables the rapid application development (RAD) of graphical user interface (GUI) applications, access to databases using DAO, RDO, or ADO, and creation of ActiveX controls and objects. In many ways, Visual Basic is a lot like the language that you use every day. When you speak or write, you use different types of words, such as nouns or verbs, which define how they are used. Visual Basic also has different types of words known as programming elements that define how they are used to write programs.

Programming elements in Visual Basic include statements, declarations, methods, operators, and keywords.

A programmer can put together an application using the components provided with Visual Basic itself. Programs written in Visual Basic can also use the Windows API, but doing so requires external function declarations.

In business programming, Visual Basic has one of the largest user bases. According to some sources, as of 2003, 52% of software developers used Visual Basic, making it the most popular programming language at that time. Another point of view was provided by the research done by Evans Data that found that 43% of Visual Basic developers planned to move to other languages. Visual Basic currently competes with PHP and C++ as the third most popular programming language behind Java and C [1].

1.2 GRAFICAL COMPONENTS OF VISUAL BASIC

Projects can become quite advanced in Visual Basic, even containing subprojects of different types. From a programming point of view, however, standard Visual Basic projects usually contain just four types of items: global items, forms, modules, and variable scope.

- **Forms**

Forms are familiar to all Visual Basic programmers, of course they are the templates you base windows on. Besides standard forms, Visual Basic also supports Multiple Document Interface (MDI) forms, as well as a whole number of predefined forms.

- **Modules**

Modules are collections of code and data that function something like objects in object-oriented programming (OOP), but without defining OOP characteristics like inheritance, polymorphism, and so on. The point behind modules is to enclose procedures and data in a way that hides them from the rest of the program.

We'll discuss the importance of doing this later in this chapter when we cover Visual Basic programming techniques and style; breaking a large program into smaller, self-contained modules can be invaluable for creating and maintaining code.

You can think of well-designed modules conceptually as programming objects; for example, you might have a module that handles screen display that includes a dozen internal (unseen by the rest of the program) procedures and one or two procedures accessible to the rest of the program. In this way, the rest of the program only has to deal with one or two procedures, not a dozen.

- **Global Items**

Global items are accessible to all modules and forms in a project, and you declare them with the Public keyword. However, Microsoft recommends that you keep the number of global items to an absolute minimum and, in fact, suggests their use only when you need to communicate between forms.

One reason to avoid global variables is their accessibility from anywhere in the program; while you are working with a global variable in one part of a program, another part of the program might be busy changing that variable, giving you unpredictable results.

- **Variable Scope**

You declare variables in a number of ways. Most often, you use the Dim statement to declare a variable. If you do not specify the variable type when you use Dim, it creates a variant, which can operate as any variable type. You can specify the variable type using the as keyword like this:

Dim Integer Value as Integer

Besides Dim, you can also use ReDim to redimension space for dynamic arrays, Private to restrict it to a module or form, Public to make it global that is, accessible to all modules or forms or Static to make sure its value does not change between procedure calls.

There are three levels of variable scope in Visual Basic: at the procedure level, at the form or module level, and at the global level schematically [1].

1.3 COMMON COMMAND BUTTONS

a. Back Color:

Specifies the command button's background color. Click the Back Color's palette down arrow to see a list of colors and click Categorized to see a list of common Windows control colors. Before the command button displays the background color, you must change the Style property from 0-Standard to 1-Graphical.

b. Cancel:

Determines whether the command button gets a Click event if the user presses Esc.

c. Caption:

Holds the text that appears on the command button. Default determines if the command button responds to an Enter key press even if another control has the focus.

d. Enabled:

Determines whether the command button is active. Often, you'll change the Enabled property at runtime with code when a command button is no longer needed and you want to gray out the command button.

e. Font:

Produces a Font dialog box in which you can set the caption's font name, style, and size.

f. Height:

Holds the height of the command button.

g. Left:

Holds the number from the command button's left edge to the Form window's left edge.

h. Picture:

Holds the name of an icon graphic image that appears on the command button as long as the Style property is set to 1-Graphical.

i. Style:

Determines whether the command button appears as a standard Windows command button (if set to 0-Standard) or a command button with a color and possible picture (if set to 1-Graphical).

j. Tab Index:

Specifies the order of the command button in the focus order.

k. Visible:

Determines whether the command button appears or is hidden from the user. (Invisible controls cannot receive the focus until the running code changes the visible property to True.)

l. Width:

Holds the width of the command button [1].

1.4 FUNCTIONS

There are two types of procedures in Visual Basic: subroutines and functions. Subroutines can take arguments passed in parentheses but do not return a value.

Functions do the same but do return values (which can be discarded). A function is a block of code that you call and pass arguments to, and using a function helps break your code up into manageable parts. For reference's sake, here's how you declare a function:

```
[Private | Public] [Static] Function name [(arglist)] [As type]  
    [Statements]  
    ...  
    [Name = expression]  
    ...  
[Exit Function]  
    ...  
    [Statements]  
End Function
```

The Public keyword makes a procedure accessible to all other procedures in all modules and forms.

The Private keyword makes a procedure accessible only to other procedures in the Module or form in which it is declared.

The **Static** keyword specifies that the procedure local variables should be preserved between calls.

The *name* identifier is the name of the procedure.

The *arglist* identifier is a list of variables representing arguments that are passed to the procedure when it is called. You separate multiple variables with commas.

The *Statements* identifier is the group of statements to be executed within the procedure.

The *arglist* identifier has this following syntax:

```
[Optional] [ByVal | ByRef] [ParamArray] varname  
            [()] [As type]  
            [= defaultvalue]
```

In *arglist*

Optional means that an argument is not required.

ByVal means that the argument's passed by value.

ByRef means that the argument is passed by reference (**ByRef** is the default in Visual Basic).

ParamArray is used as the last argument in *arglist* to indicate that

The final argument is an array of Variant elements; *varname* is the name of the variable passed as an argument; *type* is the data type of the argument; and *defaultvalue* is any constant or constant expression, which is used as the argument's default value if you've used

The **Optional** keyword. The *type* identifier is the data type returned by the function.

The **Exit Function** keywords cause an immediate exit from a **Function** procedure. You call a **Function** procedure using the function name, followed by the argument list in Parentheses. You return a value from a function by assigning the value you want to return to the function's name like this:

Name = expression. Finally, End Function

Ends the procedure definition [1].

1.5 PRIVATE AND PUBLIC PROCEDURE

Using Call can be a time-saver and make your programs much more Maintainable because you put common code in a procedure and call that Procedure from anywhere in the program when you need the code to execute.

You might even write a routine in one application that you will want to use Elsewhere in another application. For example, perhaps you write a report title That includes your company's name and address, and you want to place that Title at the top of other reports generated in other applications.

If the procedure is located in the general section of a Form module, no other Application can use that procedure without that Form module. Therefore, you Can place that procedure inside a Code module.

Over time, you might fill a particular reporting Code module with several routines that you will use for reporting. Then, any application that produces reports can use those procedures without your having to rewrite them for each application. All you must do is right-click over the application's Project window and select Add, Module from the pop-up menu to bring your general procedure module into whatever application that can use the code.

Code inside a Form module can use the code inside an added Code module. All you need to do is call the procedure from the Form module code *with one Exception*: You can call public procedures from outside the current module, not Private Procedures. Consider the following procedure declaration statement:

Private Sub ReportIt ()

This procedure can only be called from the module in which it resides. If you Wrote the procedure as a public procedure, by defining it as follows, any Procedure from any module in that application can call the procedure:

Public Sub ReportIt ()

Therefore, the general-purpose procedures that you write should all be public If you want those procedures to be callable from other modules.

Therefore, you now can understand these rules:

- A procedure declared as Private can be used only within its own Module.
- A procedure declared as Public can be used by any procedure Within its application [1].

1.6 SHARING EVENT HANDLERS

The following example demonstrates sharing the Change event handler for a group of three command controls. In Visual Basic 2005, the Handles clause of the event handler specifies which control the event will handle. The event handler returns a generic Object, so it must be cast to the specific object type (in this case, command) that you want to handle using the Direct Cast method.

```
Private Sub Command1_Click (Index As Integer)
```

```
    Select Case Index
```

```
    Case 0
```

```
        MsgBox ("You clicked the first command button")
```

```
    Case 1
```

```
        MsgBox ("You clicked the second command button ")
```

Case 2

MsgBox ("You clicked the third command button ")

End Select

End Sub

1.7 EVENTS IN VISUAL BASIC

1.7.1 Initialize Event:

In Visual Basic 6.0, the **Initialize** event is used to execute code before a form is loaded.

```
Private Sub Form Initialize ()  
MsgBox ("The form is loading")  
End Sub
```

1.7.2 Terminate Event

In Visual Basic 6.0, the **Terminate** event is used to execute code after a form is unloaded

```
Private Sub Form Terminate ()  
MsgBox "The form was terminated"  
End Sub
```

1.7.3 Unload Event

In Visual Basic 6.0, the **Unload** event has a Cancel argument.

```
Private sub command_ click ( )  
Unload me  
End sub
```

1.8 OVERVIEW OF ARRAYS IN VISUAL BASIC

An *array* is a set of values that are logically related to each other, such as the number of students in each grade in a grammar school.

An array allows you to refer to these related values by the same name and to use a number, called an *index* or *subscript*, to tell them apart. The individual values are called the *elements* of the array. They are contiguous from index 0 through the highest index value [1].

CHAPTER 2

DATABASE

2.1 OVERVIEW

The term database originated within the computer industry. Although its meaning has been broadened by popular use, even to include non-electronic databases, this article takes a more technical perspective. A possible definition is that a database is a collection of records stored in a computer in a systematic way, so that a computer program can consult it to answer questions. The items retrieved in answer to queries become information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS). The properties and design of database systems are included in the study of information science.

The central concept of a database is that of a collection of records, or pieces of knowledge. Typically, for a given database, there is a structural description of the type of facts held in that database: this description is known as a schema. The schema describes the objects that are represented in the database, and the relationships among them. There are a number of different ways of organizing a schema, that is, of modeling the database structure: these are known as database models (or data models). The model in most common use today is the relational model, which in layman's terms represents all information in the form of multiple related tables each consisting of rows and columns (the true definition uses mathematical terminology). This model represents relationships by the use of values common to more than one table. Other models such as the hierarchical model and the network model use a more explicit representation of relationships.

Strictly speaking, the term database refers to the collection of related records, and the software should be referred to as the database management system or DBMS.

integrity and quality, if it allows shared access by a community of users, if it has a schema, or if it supports a query language. However, there is no agreed definition of these properties.

Database management systems are usually categorized according to the data model that they support: relational, object-relational, network, and so on.

The data model will tend to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products

A database is a collection of information that's related to a particular subject or purpose, such as tracking customer orders or maintaining a music collection. If your database isn't stored on a computer, or only parts of it are, you may be tracking information from a variety of sources that you have to coordinate and organize yourself.

Good database design ensures that your database is easy to maintain. You store data in tables and each table contains data about only one subject, such as customers. Therefore, you update a particular piece of data, such as an address, in just one place and that change automatically appears throughout the database [2].

2.2 Relational model

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

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Code. The data structures in these products are tables, rather than relations: the main differences being that tables can contain duplicate rows, and that the rows (and columns) can be treated as being ordered. The same criticism applies to the SQL language which is the primary interface to these products. There has been considerable controversy, mainly due to Code himself, as to whether it is correct to describe SQL implementations as "relational": but the fact is that the world does so, and the following description uses the term in its popular sense.

A relational database contains multiple tables, each similar to the one in the "flat" database model. Relationships between tables are not defined explicitly; instead, keys are used to match up rows of data in different tables. A key is a collection of one or more columns in one table whose values match corresponding columns in other tables: for example, an Employee table may contain a column named Location which contains a value that matches the key of a Location table. Any column can be a key, or multiple columns can be grouped together into a single key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

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

A key that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number) is sometimes called a "natural" key. If no natural key is suitable (think of the many people named Brown), an arbitrary key can be assigned (such as by giving employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that cannot break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two

independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed[2].

2.3 Relational operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many web sites, such as Wikipedia, perform SQL queries when generating pages.

In response to a query, the database returns a result set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted.

Often, data from multiple tables are combined into one, by doing a join. Conceptually, this is done by taking all possible combinations of rows (the Cartesian product), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques.

There are a number of relational operations in addition to join. These include project (the process of eliminating some of the columns), restrict (the process of eliminating some of the rows), union (a way of combining two tables with similar structures), difference (which lists the rows in one table that are not found in the other), intersect (which lists the rows found in both tables), and product (mentioned above, which combines each row of one table with each row of the other). Depending on which other sources you consult, there are a number of other operators - many of which can be defined in terms of those listed above. These include semi-join, outer operators such as outer join and outer union, and various forms of division. Then there are operators to rename columns, and summarizing or aggregating operators, and if you permit relation values as attributes (RVA - relation-valued attribute), then operators such as group and ungroup. The SELECT statement in SQL serves to handle all of these except for the group and ungroup operators.

The flexibility of relational databases allows programmers to write queries that were not anticipated by the database designers. As a result, relational databases can be used by multiple applications in ways the original designers did not foresee, which is especially important for databases that might be used for decades. This has made the idea and implementation of relational databases very popular with businesses [2].

2.4 Object database models

In recent years, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the impedance mismatch) of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

Object databases suffered because of a lack of standardization: although standards were defined by ODMG, they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing. However, object database ideas were picked up by the

Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

Object databases suffered because of a lack of standardization: although standards were defined by ODMG, they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing. However, object database ideas were picked up by the relational vendors and influenced extensions made to these products and indeed to the SQL language [2].

2.5 DATABASE TABLES

CARS	CUSTOMERS	EMPLOYEES	USERS
ID	ID	ID	ID
MODEL	FNAME	FNAME	USERNAME
TYPE	LNAME	LNAME	PASSWORD
COLOR	ADDRESS	MOBILE	REST
CLASS	MOBILE	ADDRESS	
MOTORSIZE	SELLINGDATE	SALARY	
KMPASSED		DOFBIRTH	
POWERST			
ABSBRKE			
EXTRAS			
ORIGINALPRICE			
SOLDPRICE			
PROFIT			
CUSTID			
USERID			
MOTORNUM			
LCSPLT			

Figure 2.1: Database Tables

As we see it is used four tables in this project as follows:

- Cars table: indicate the car objectives include model,type,motorsize,color,class Kmpassed.
- Customer table: indicate the information about the client including first name, last name, address, mobile and selling date.
- Employee table: indicate the employee (salesman) in the project which include first name, last name, mobile, salary and date of birth.
- User table: indicate the user which allowed to enter the system which include user name, password.

SQL STATEMENTS THAT USED

2.6 WHAT IS SQL

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete" can be used to accomplish almost everything that one needs to do with a database. This tutorial will provide you with the instruction on the basics of each of these commands as well as allow you to put them to practice using the SQL Interpreter [3].

2.6.1 SQL keywords

SQL keywords fall into several groups.

2.6.2 Data retrieval

The most frequently used operation in transactional databases is the data retrieval operation. When restricted to data retrieval commands, SQL acts as a declarative language.

SELECT is used to retrieve zero or more rows from one or more tables in a database. In most applications, SELECT is the most commonly used Data Manipulation Language command. In specifying a SELECT query, the user specifies a description of the desired result set, but they do not specify what physical operations must be executed to produce that result set. Translating the query into an efficient query plan is left to the database system, more specifically to the query optimizer.

Commonly available keywords related to SELECT include:

FROM is used to indicate from which tables the data is to be taken, as well as how the tables JOIN to each other.

WHERE is used to identify which rows to be retrieved, or applied to GROUP BY. WHERE is evaluated before the GROUP BY.

GROUP BY is used to combine rows with related values into elements of a smaller set of rows.

HAVING is used to identify which of the "combined rows" (combined rows are produced when the query has a GROUP BY keyword or when the SELECT part contains aggregates), are to be retrieved. HAVING acts much like a WHERE, but it operates on the results of the GROUP BY and hence can use aggregate functions.

ORDER BY is used to identify which columns are used to sort the resulting data.

Data retrieval is very often combined with data projection; usually it isn't the data stored in primitive's data types that a user is looking for or a query is written to serve. Often the data needs to be expressed differently from how it's stored. SQL allows a wide variety of formulas included in the select list to project data. A common example would be:

```
SELECT Unit Cost * Quantity As Total Cost FROM Orders [3]
```

2.6.3 Data manipulation

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

INSERT is used to add zero or more rows to an existing table.

UPDATE is used to modify the values of a set of existing table rows.

MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements. It is defined in the SQL: 2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called an "upsert".

TRUNCATE deletes all data from a table (non-standard, but common SQL command).

DELETE removes zero or more existing rows from a table.

Example:

```
INSERT INTO my table (field1, field2, field3) VALUES ('test', 'N', NULL);
```

```
UPDATE my table SET field1 = 'updated value' WHERE field2 = 'N';
```

```
DELETE FROM my table WHERE field2 = 'N';
```

EXAMPLE FOR (insert, update, delete):

```
INSERT INTO CUSTOMERS (ID, FNAME, LNAME, ADDRESS, MOBILE)
VALUES ("& txtCUSID.Text &","& txtCUSFNAME.Text &","&
txtCUSLNANE.Text&","&txtCUSADD.Text&","& txtCUSMOB.Text &")
```

```
UPDATE CUSTOMER SET FNAME = "& txtCUSID.Text &", LNAME="&
txtCUSLNANE.Text"&,"& ADDRESS ="& txtCUSADD.Text"&
```

```
WHERE ID ="& txtCUSID.Text
```

CHAPTER 3

Open Database Connectivity ODBC

3.1 OPENING DATABASE:

To open an existing DAO Database, I use the DAO Open Database method, passing it the name of the Database to open, and this is the syntax:

Set Database=Open Database ("db path\db name.mdb")

- Db name the name of an existing Database file.
- Db path the location of my database.

3.2 ADDING A RECORD TO A RECORD SET:

To add a new record to a DAO record set, you use the AddNew method

```
Private Sub Command1_Click ()
```

```
Set db= open database ("path\cars.mdb")
```

```
Set tb= db.openrecordset ("cars")
```

```
dbrecordset.AddNew
```

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

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

```
Tb.close
```

```
Db.close
```

3.3 EDITING A RECORD IN A RECORD SET:

Besides adding new records to the record set, users might want to edit the existing records.

To do that, you use the Edit method like this in our DAO code.

```
Private Sub Command2_Click ()
```

```
    dbrecordset.Edit
```

```
End Sub
```

3.4 MOVING TO THE FIRST RECORD IN A RECORD SET:

To make the first record in a record set the current record, you use the MoveFirst method.

Here's how we move to the first record when the user clicks the appropriate button in our DAO code

```
Private Sub Command3_Click ()
```

```
    dbrecordset.MoveFirst
```

```
    ...
```

```
End Sub
```

3.5 MOVING TO THE LAST RECORD IN A RECORD SET:

To make the last record in a record set the current record, you use the MoveLast method. Here's how we move to the last record when the user clicks the appropriate button in our DAO code.

```
Private Sub Command4_Click()
```

```
    dbrecordset.MoveLast
```

```
    ...
```

```
End Sub
```


After moving to the last record, we display that record's fields in the two text boxes in the program,

Text1 and Text2:

```
Private Sub Command5_Click()  
    dbrecordset.MoveLast  
    Text1.Text = dbrecordset.fields(0)  
    Text2.Text = dbrecordset.fields(1)  
End Sub
```

3.6 DELETING A RECORD IN A RECORD SET:

To delete a record in a DAO record set, you use the Delete method.

```
Private Sub Command6_Click()  
    Text1.Text = ""  
    Text2.Text = ""  
    dbrecordset.Delete  
End Sub
```

3.7 UPDATING A RECORD IN A RECORD SET:

When the user changes the data in a record or adds a new record, we must update the Database to record that change, and you use the record set Update method to do that:

```
recordset.Update (type , force)
```

Here are the arguments in this function:

- Type—Constant indicating the type of update, as specified in Settings (ODBCDirect workspaces only).
- Force—Boolean value indicating whether or not to force the changes into the Database, regardless of whether the data has been changed by another user (ODBCDirect workspaces only).

```
Private Sub Command7_Click ()
```

```
    dbrecordset.fields(0) = Text1.Text
```

```
    dbrecordset.fields(1) = Text2.Text
```

```
End Sub
```

After loading the data into the current record's fields, we save that record to the Database using the Update method:

```
Private Sub Command6_Click()
```

```
    dbrecordset.fields(0) = Text1.Text
```

```
    dbrecordset.fields(1) = Text2.Text
```

```
    dbrecordset.Update
```

```
End Sub
```

3.8 SORTING A RECORD SET:

To sort a record set, you can install the index you want to sort with in the record set's Index property.

For example, we can sort the record set in our DAO code example, the DAO code project, with the index we've created this way:

```
Sub Sort_Click()  
    Set dbindex = td.Indexes(0)  
    dbrecordset.Index = dbindex.Name
```

After the record set is sorted, we display the first record in the two main text boxes, Text1 and Text2:

```
Sub Sort_Click()  
    Set dbindex = td.Indexes(0)  
    dbrecordset.Index = dbindex.Name  
    Text1.Text = dbrecordset.fields(0)  
    Text2.Text = dbrecordset.fields(1)  
End Sub
```

3.9 SEARCHING A RECORD SET:

You can search a record set with an index; we just set its Index property to the index we want to search and then set its Seek property to the string we want to search for. Let's see an example. When the user selects the Search menu item in our DAO.

Installed the index based on the first field in the record set and show the dialog box named Search.

```

Private Sub Search_Click()

    Set dbindex = td.Indexes(0)

    dbrecordset.Index = dbindex.Name

    SearchForm.Show

End Sub

```

After the user dismisses the Search... dialog box, we retrieve the text to search for from that dialog box's text box and place that text in the record set's Seek property, along with the command "=", which indicates we want to find exact matches to the search text:

```

Sub SearchTable()

    dbrecordset.Seek "=", SearchForm.Text1.Text

    ...

```

Besides =, you can also search using <, <=, >=, and >. When the search is complete, we display the found record in the project's main text boxes, Text1 and Text2:

```

Sub SearchTable()

    dbrecordset.Seek "=", SearchForm.Text1.Text

    Text1.Text = dbrecordset.fields(0)

    Text2.Text = dbrecordset.fields(1)

End Sub

```


CHAPTER 4

CAR GALLERY

4.1 OVERVIEW

This project is an efficient and flexible for using. And any one can use it without any effort.

My project contained many forms each of them have a task. The program has a security for accessing to the main program. This allowed the manager to access to main form and make all the operation that the system supports.

And allowed employee (salesman) to search about a car and sell it only.

Now I decided to create a block diagram for whole system to see the features and how the system work and what are forms contain.

4.2 BLOCK DIAGRAM

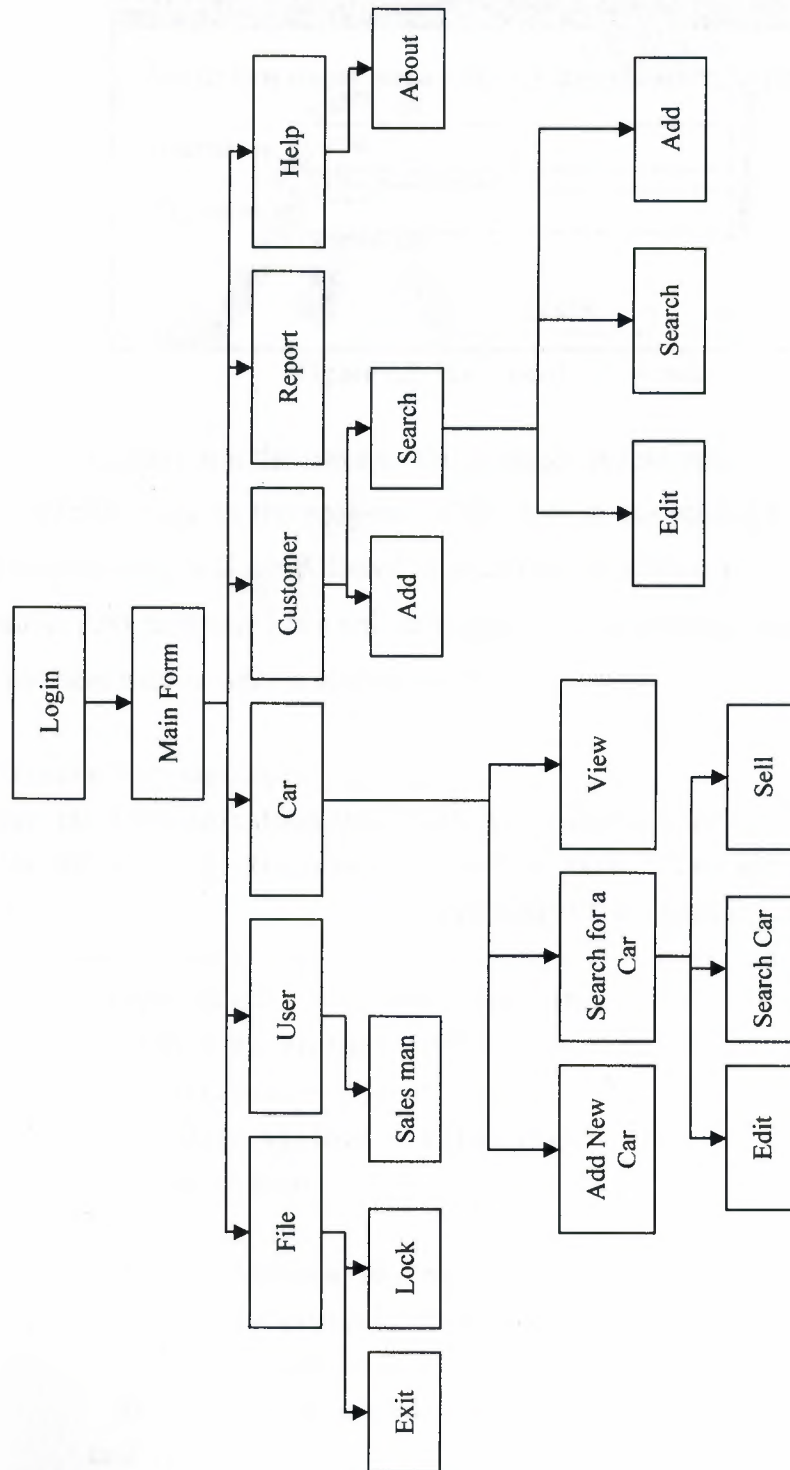


Figure4.1: block diagram

4.3 SECURITY

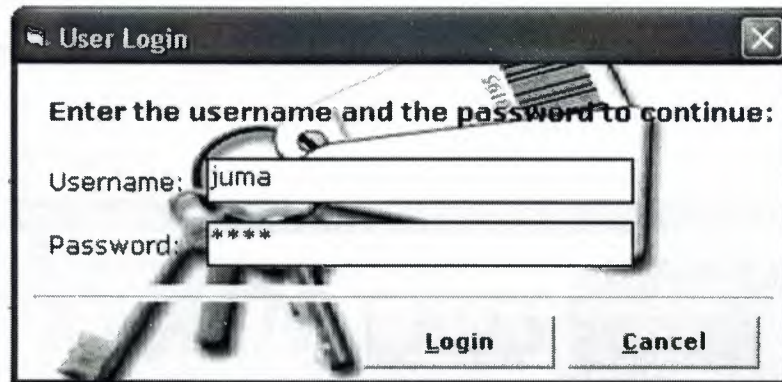


Figure 4.2: the security of system

In this form the users or the manager should enter there username and password to log in the program, if the user or the manager entered a wrong password they will not Allowed to access to the system visa versa for the user name. And each user has a unique password and username for entering.

And here what is written in login button:

```
Private Sub cmdLogin_Click()  
Set DB = OpenDatabase(App.Path & "\carsdb.mdb")  
Set TB = DB.OpenRecordset("SELECT * FROM USERS WHERE " & _  
                           "USERNAME='" & txtUser.Text & "'")  
If TB.EOF = False Then  
    If TB("PASSWORD") = txtPass.Text Then  
        myUserID = TB.Fields("ID")  
        If txtUser.Text = "juma" Then  
            If frmMain.Visible = False Then  
                frmMain.Show  
            Else  
                MsgBox "Access To The Main Program Is Forbidden!", _  
                    vbCritical, "Sales Manager"  
                frmSrchForCar.Show  
                frmSrchForCar.cmdEditCar.Enabled = False  
            End If  
            Unload Me  
        Else  
            MsgBox "Wrong Password!", vbCritical, "Error"  
        End If  
    End If  
End Sub
```

```
Else
    MsgBox "User Not Found!", vbCritical, "Error"
End If
End Sub
```

4.4 MAIN FORM



Figure 4.3: main form (home page)

This is the main form for this project that has the following menu file, cars, customer, report, and help.

4.5 FORM USER

In this form it is mentioned about the users (employee) that I have them in my gallery which is salesman.

ID	First Name	Last Name	Mobile	Salary	Address	Date Of Birth
1	fadi	alKhayb	533847906	Un Known	Clas lefkosa	8/5/1982
2	omar	kamal	8384010	Un Known	Clas lefkosa	3/12/1986

Close

Figure 4.4: Users

In the users form it is used msflexgrid components for construct a table, as follows:

```
Private Sub RefillGrid()  
    With flxMain  
        .Clear  
        .Rows = 1  
        .Cols = 7  
  
        .Row = 0  
        .Col = 0  
        .Text = "ID"  
  
        .Row = 0  
        .Col = 1  
        .Text = "First Name"  
  
        .Row = 0  
        .Col = 2  
        .Text = "Last Name"
```

```

.Row = 0
.Col = 3
.Text = "Mobile"

.Row = 0
.Col = 4
.Text = "Salary"

.Row = 0
.Col = 5
.Text = "Address"

.Row = 0
.Col = 6
.Text = "Date Of Birth"

.ColWidth(0) = 800
.ColWidth(1) = 2000
.ColWidth(2) = 2000
.ColWidth(3) = 1600
.ColWidth(4) = 1400
.ColWidth(5) = 2400
.ColWidth(6) = 1400

While Not TB.EOF
    .AddItem TB.Fields("ID") & Chr(9) & _
        TB.Fields("FNAME") & Chr(9) & _
        TB.Fields("LNAME") & Chr(9) & _
        TB.Fields("MOBILE") & Chr(9) & _
        GetCarClass(TB.Fields("CLASS")) & Chr(9) & _
        TB.Fields("ADDRESS") & Chr(9) & _
        TB.Fields("DOFBIRTH")

    TB.MoveNext
    DoEvents
Wend
End With
End Sub

```

4.6 FORM CAR

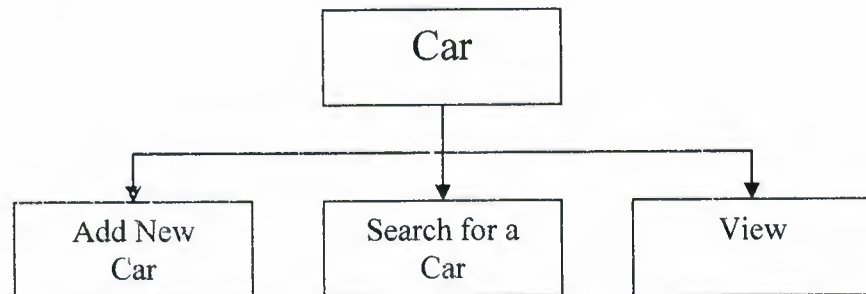


Figure 4.5: this diagram showed the car form

4.6.1 FORM ADD NEW CAR

Add A New Car

Please Enter Car Details

Car ID:

License Plate:

Model:

Type:

Color:

Class:

Motor Size:

Moto Number:

Power Steering:

KMs Passed:

App. makes:

extra Options:

Figure 4.6: this form for add anew car

This form have capable of adding (buying) anew car that the manager buying it from the free zone car. And having full information including the price and extra options and here we will see how add car button work by code:

```

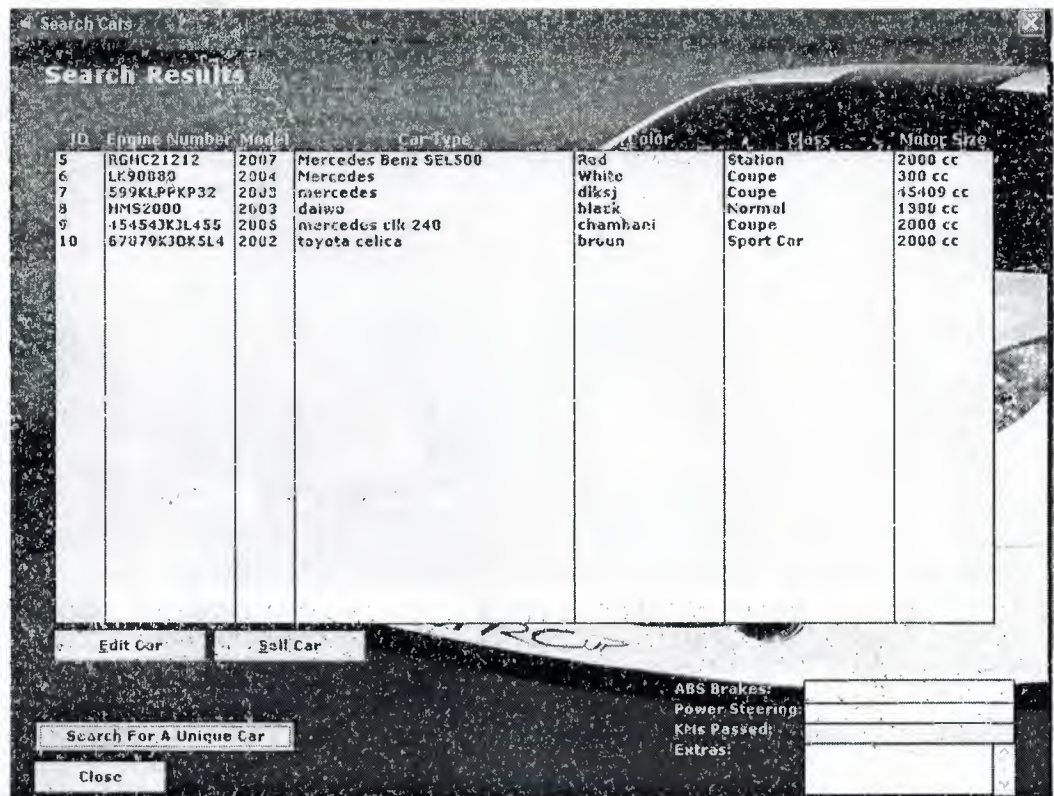
Private Sub cmdAddCar_Click()
If CheckTexts = False Then Exit Sub

DB.Execute ("INSERT INTO CARS(ID, LCSPLT, MODEL, TYPE, " & _
"COLOR, CLASS, MOTORSIZE, MOTORNUM, POWERST, KMPASSED, " & _
"ABSBRKE, ORIGINALPRICE, EXTRAS, SOLDPRICE) " & _
"VALUES(" & txtCarDet(0).Text & ", " & _
"" & txtCarDet(9).Text & ", " & _
txtCarDet(1).Text & ", " & _
"" & txtCarDet(2).Text & ", " & _
"" & txtCarDet(3).Text & ", " & _
comClass.ListIndex & ", " & _
txtCarDet(4).Text & ", " & _
"" & txtCarDet(5).Text & ", " & _
comSteering.ListIndex & ", " & _
txtCarDet(6).Text & ", " & _
IIf(comBrakes.ListIndex = 0, DB_TRUE, DB_FALSE) & ", " & _
txtCarDet(7).Text & ", " & _
"" & txtCarDet(8).Text & ", 0)")

If MsgBox("Car Was Added Successfully!" & vbNewLine & _
"Add Another?", _
vbYesNo + vbInformation, "Add A New Car") = vbYes Then
    Call ClearTexts
    Call MakeID
    txtCarDet(0).SetFocus
Else
    Unload Me
End If
End Sub

```


4.6.2 SEARCH CAR FORM



The screenshot shows a window titled "Search Cars" with a "Search Results" section. It contains a table with 7 columns: ID, Engine Number, Model, Car Type, Color, Class, and Motor Size. The table lists 10 cars. Below the table are buttons for "Edit Car" and "Sell Car". At the bottom left is a "Close" button. At the bottom right are input fields for "ABS Brakes:", "Power Steering:", "KMs Passed:", and "Extras:". A "Search For A Unique Car" button is also present.

ID	Engine Number	Model	Car Type	Color	Class	Motor Size
5	RGHC21212	2007	Mercedes Benz SEL500	Red	Station	2000 cc
6	LK900880	2004	Mercedes	White	Coupe	3000 cc
7	599KLPPKP32	2003	Mercedes	black	Coupe	45409 cc
8	HMS2000	2003	daiwo	black	Normal	1300 cc
9	45454JKL455	2005	Mercedes clk 240	chamhani	Coupe	2000 cc
10	67879K30KSL4	2002	toyota celica	brun	Sport Car	2000 cc

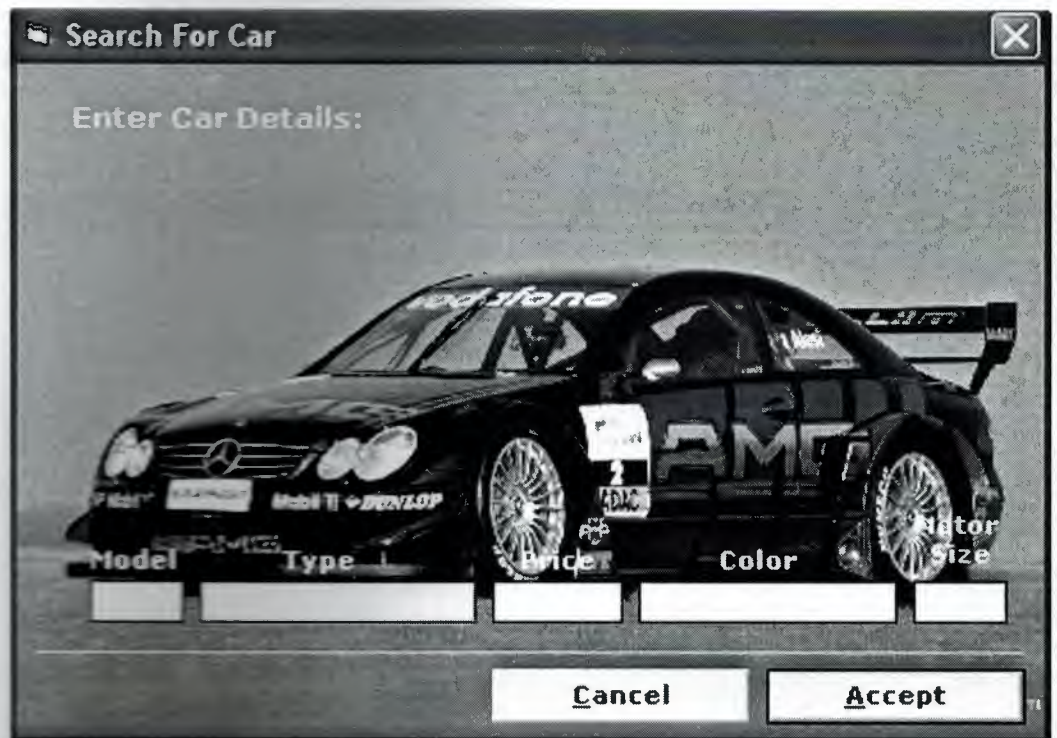
Buttons: Edit Car, Sell Car, Search For A Unique Car, Close

Form fields: ABS Brakes: _____, Power Steering: _____, KMs Passed: _____, Extras: _____

Figure 4.7: The form for search a unique car

This form allow the manager to edit car and sell it if it is need, and allow for the users to sell a car only if it is need.

4.6.3 SEARCH FOR A UNIQUE CAR



Search For Car

Enter Car Details:

Model Type Price Color Motor Size

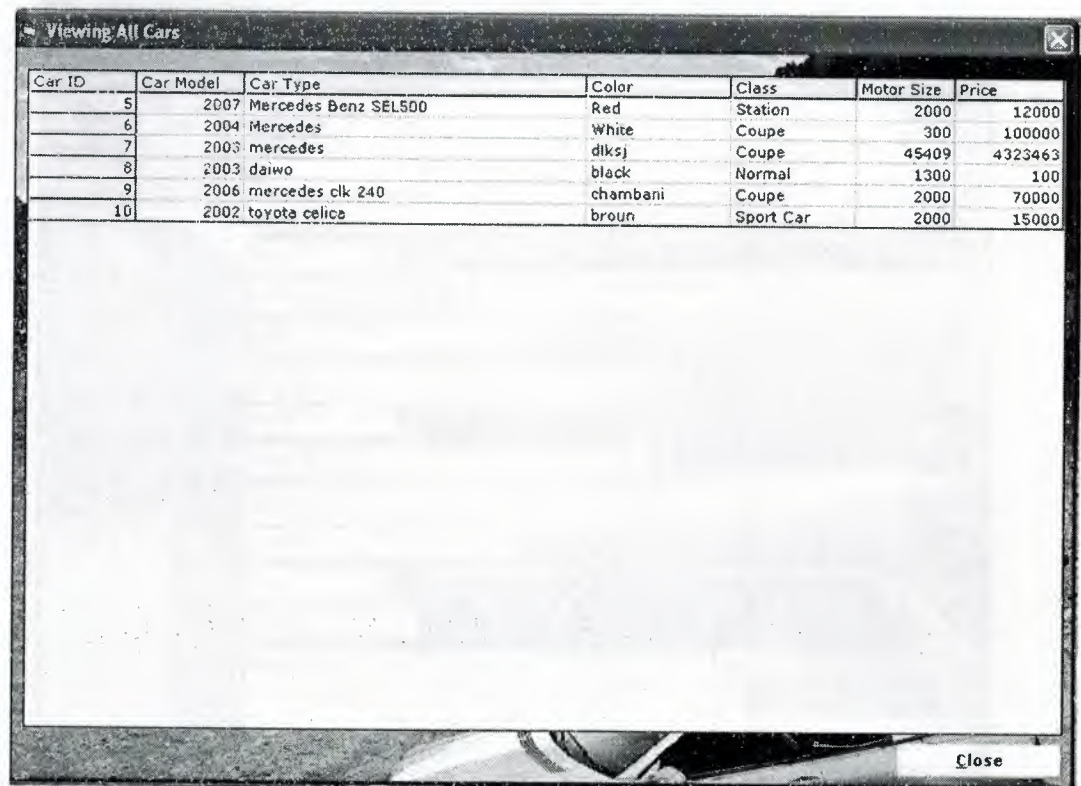
Cancel Accept

Figure 4.8: search for unique car

The purpose form this form is to find out the car which the customer has ordered. This form helps the customer to select a suitable car for him, which the gallery. In this form you don't need to fill all text. You have just filled one text.

Then the system automatically searches for you depend on what the field you interesting to search about.

4.6.4 VIEW FORM



The screenshot shows a web application window titled "Viewing All Cars". Inside the window, there is a table with the following data:

Car ID	Car Model	Car Type	Color	Class	Motor Size	Price
5	2007 Mercedes Benz SEL500		Red	Station	2000	12000
6	2004 Mercedes		White	Coupe	300	100000
7	2003 mercedes		diks j	Coupe	45409	4323463
8	2003 daiwo		black	Normal	1300	100
9	2006 mercedes clk 240		chambani	Coupe	2000	70000
10	2002 toyota celica		broun	Sport Car	2000	15000

Below the table is a large, empty rectangular area. In the bottom right corner of the window, there is a "Close" button.

Figure 4.9: this form for view all car

The purpose of this form is to view the entire car that the gallery includes

And containing some information about the car.

4.7 FORM EDIT A CAR

Edit Car

Please Enter Car Details

Car ID: 5

License Plate: AS321432

Model: 2007

Type: Mercedes Benz SEL500

Color: Red

Class: Station

Motor Size: 2000

Moto Number: RGHC21212

Power Steering: Normal

KM's Passed: 12200

App. Brakes: Yes

Car Price: 12000

Extra Options: none

Save Car

Cancel

Figure 4.10: edit a car

In this section the manager can update information about a special car after selected it and then he have to press to the save button to save the changes that he have made it to my database.

Here the code of save car button:

```
Private Sub cmdSave_Click()  
If CheckTexts = False Then Exit Sub  
Dim SQL$  
SQL = ("UPDATE CARS SET LCSPLT= '" & txtCarDet(9).Text & _  
      "'", MODEL= " & txtCarDet(1).Text & ", " & _  
      "TYPE='" & txtCarDet(2).Text & "', " & _  
      "COLOR='" & txtCarDet(3).Text & "', " & _  
      "CLASS=" & comClass.ListIndex & ", " & _  
      "MOTORSIZE=" & txtCarDet(4).Text & ", " & _
```



```

" MOTORNUM=" & txtCarDet(5).Text & ", " & _
" POWERST=" & comSteering.ListIndex & ", " & _
" KMPASSED=" & txtCarDet(6).Text & ", " & _
" ABSBRAKE=" & IIf(comBrakes.ListIndex = 0, DB_TRUE,
    DB_FALSE) & ", " & _
" ORIGINALPRICE=" & txtCarDet(7).Text & ", " & _
" EXTRAS=" & txtCarDet(8).Text & " " & _
" WHERE ID=" & txtCarDet(0).Text & ";"
DB.Execute SQL
MsgBox "Car Was Updated Successfully!", vbInformation, _
"Update Car"
frmSrchForCar.IsChanged = True
Unload Me
End Sub

```

4.8 FORM SELL A CAR

Figure 4.11: selling a car

After the customer takes decision for choosing his car finally we go to the last operation which is the selling part, after selling the car i made technique to save all cars that I have sold it, but the question now how we can

distinguish between the cars that have sold and the car that not sold, the answer is every car sold i remarked it by "sold " word in my database, After that we can calculate the profit.

And here the code of sell car button:

```
Private Sub cmdSellCar_Click()  
Dim SQL As String, curProfit As Currency  
If Val(txtCarDet(5).Text) <= 0 Then  
    MsgBox "Car Price Cannot Be Zero!", vbCritical, "Error"  
Else  
    curProfit = CCur(txtCarDet(5).Text) - OriginalPrice  
    SQL = "UPDATE CARS SET SOLDPRICE=" & txtCarDet(5).Text & _  
        ", PROFIT=" & curProfit & _  
        ", CUSTID=" & txtCustID.Text & _  
        ", USERID=" & myUserID & _  
        " WHERE ID=" & txtCarDet(0).Text  
  
    DB.Execute SQL  
    MsgBox "Car Was Sold Successfully!" & vbNewLine & _  
        "Congratulations For Both Dealer And Customer!", _  
        vbInformation, "Sold"  
  
    Unload Me  
End If  
EndSub
```


4.9 CUSTOMER FORM

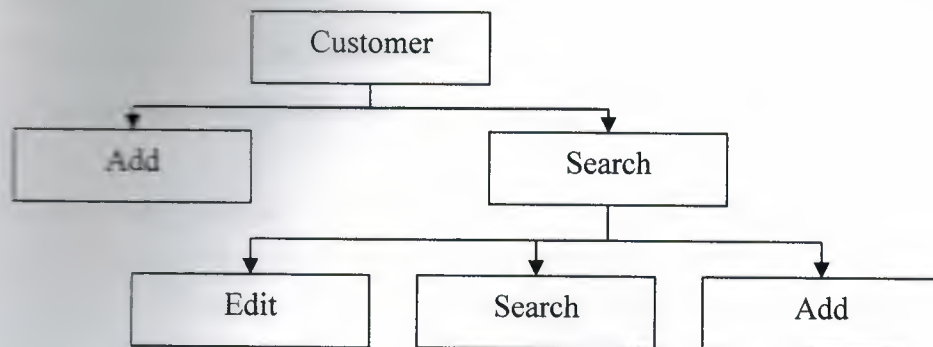


Figure 4.12: customer diagram

4.9.1 ADD CUSTOMER FORM

Add A Customer [X]

Please Enter Customer Information

Customer ID: 4

First Name: []

Last Name: []

Address: []

Mobile: []

Selling Date: []

[Cancel] [Add Customer]

Figure 4.13: add customer

In this form we can add customer whose have sold a car from our gallery

This form has all information about the customer.

The following code is indicated the add customer button (see figure13):

```
Private Sub cmdAddCust_Click()  
If IsNumeric(txtCusMob.Text) = False Then  
    MsgBox "Please enter a valid mobile number!", vbCritical, _  
        "Error"  
    txtCusMob.SetFocus  
    SendKeys "{home}+{end}"  
    Exit Sub  
End If  
  
If IsDate(txtCusSelDt.Text) = False Then  
    MsgBox "Please enter a valid date!", vbCritical, _  
        "Error"  
    txtCusSelDt.SetFocus  
    SendKeys "{home}+{end}"  
    Exit Sub  
End If  
  
If txtCusFName.Text = "" Or txtCusLName.Text = "" Or _  
    txtCusMob.Text = "" Or txtCusSelDt.Text = "" Then  
    MsgBox "Please enter all customer details!", vbCritical, _  
        "Error"  
Else  
    Dim SQL As String  
    SQL = "INSERT INTO CUSTOMERS (ID, FNAME, LNAME, ADDRESS, " & _  
        "MOBILE, SELLINGDATE) VALUES(" & _  
        txtCusID.Text & ", '" & txtCusFName.Text & "', '" & _  
        txtCusLName.Text & "', '" & _  
        txtCusAdd.Text & "', " & txtCusMob.Text & ", " & _  
        txtCusSelDt.Text & ");"  
  
    DB.Execute SQL  
    MsgBox "Customer Was Added Successfully.", vbInformation, _  
        "Successful"
```



```

If frmSellCar.Visible = True Then
    frmSellCar.txtCustFLName.Text = txtCusFName.Text & " " & _
    txtCusLName.Text
    frmSellCar.txtCustID.Text = txtCusID.Text
End If

Unload Me
End If
End Sub

```

4.9.2 FORM SEARCH FOR A CUSTOMER

The image shows two windows from a software application. The top window is a small dialog box titled 'Search For Customer' with a close button (X). It contains a text input field labeled 'Enter Customer Name:' and two buttons: 'OK' and 'Cancel'. The bottom window is a larger application window, also titled 'Search For Customer', which displays a table of customer data. The table has four columns: ID, Name, Mobile, and Address. It lists five customers. At the bottom of this window, there are four buttons: 'Search', 'Close', 'Edit', and 'Add New'.

ID	Name	Mobile	Address
1	Almaral Ghawanmeh	795558454	Amman, Tla3 al 3aly
2	qweqweqwe qweqweqwe	123123123	qweqweqweq
3	Fadi Aguel	323232323	Amman, Jordan
4	Emrah Isler	533888888	Gonyeli Belediyesi Arkasinda
5	Basem Isfash	533875665	gonialy

Figure 4.14: search for a customer

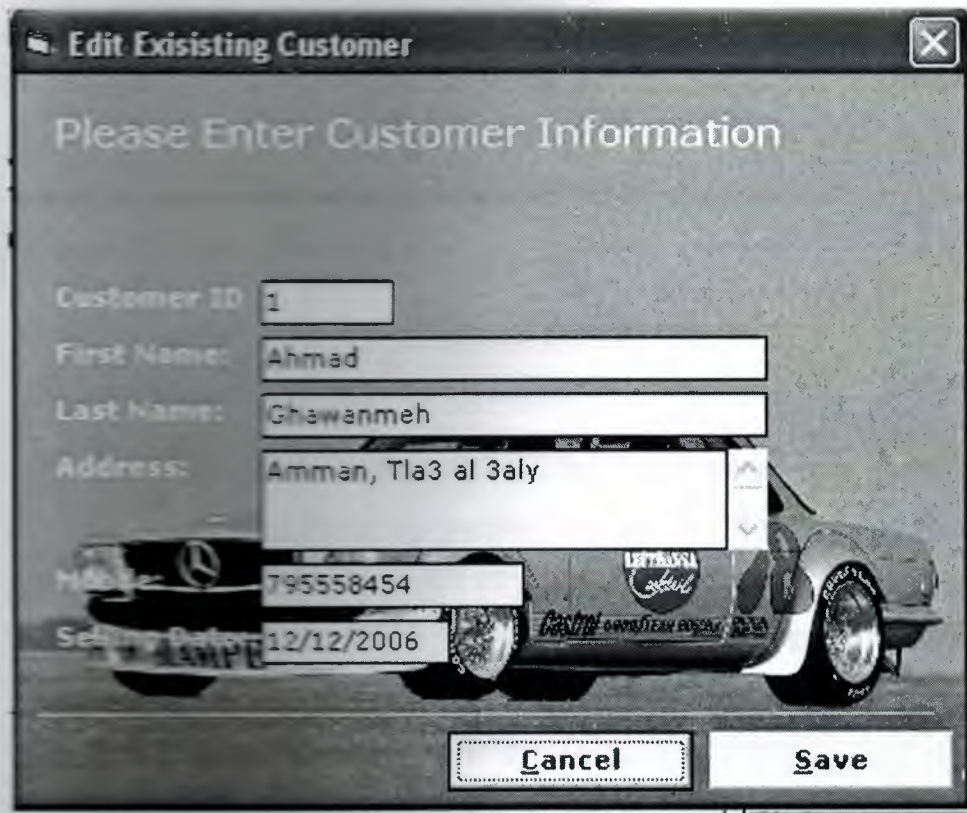
In this form we can search about any customer that we have saved them in my database and we want take some information about any customer that the manager wants.



Here the code of search button that indicates in figure14:

```
Private Sub cmdSearch_Click()  
Dim x$  
x = InputBox("Enter Customer Name:", "Search For Customer")  
If Not x = "" Then  
    Dim i As Integer  
    For i = 0 To lstID.ListCount  
        If LCase$(Left$(lstName.List(i), Len(x))) = LCase(x) Then  
            MsgBox "Selected Customer Was Found!", vbInformation, _  
                "Found"  
            lstID.ListIndex = i  
            Call lstID_Click  
            Exit Sub  
        End If  
    DoEvents  
    Next i  
Else  
    Exit Sub  
End If  
  
MsgBox "Selected Customer Was Not Found!", vbInformation, _  
    "Not Found"  
Call RefreshLists  
End Sub
```

4.9.3 FORM EDIT FOR A CUSTOMER



Edit Existing Customer

Please Enter Customer Information

Customer ID: 1

First Name: Ahmad

Last Name: Ghawanmeh

Address: Amman, Tla3 al 3aly

Mobile: 795558454

Select Date: 12/12/2006

Cancel **Save**

Figure 4.15: edit a customer

In this form after select the customer that we want to update some information about him and then we press the save button that will automatically changed in my database.

Here the code of Cancel Button:

```
Private Sub cmdCancel_Click()  
    Reload Me  
End Sub
```


Conclusion

The first intention was to build my car gallery system from the bottom up, and I did blessed by Allah. Started step by step, I used new programming languages, very efficient and easy to understand, learnt how to gather requirements, and now confidently used too many tools such as, Visual Studio 6, Microsoft office access2003.

The advantage of this project is:

- 1) Will make purchase a car so easy to the use without any much effort.
- 2) The important thing in my working that any one can use it and no need to professional person, so it's very easy and very clear.
- 3) Each user has a particular authority in using this program by defining his actions and that related to his name and password and it's considered to be security for the program

Reference

Books:

1. Visual basic 6 black book.
2. introduction to SQL
3. visual basic 6 deitel&detiel T.R.Nieto
4. the complete visual basic 6 by Evangelos Petros

Websites:

1. <http://www.montada.com>
2. <http://www.google>