NEAR EAST UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

Name: Hakan Ürkmez
Number : 971303
Supervisor : Besime Erin
Project Title : Record Operation
Date : 2 June 2000

CONTENTS

AIMS OF MY PROJECT

I have learned visual basic program previous semester. It was the most exciting computer language product to hit market in quite a while .The press had rarely been excited by a product.so what was all thehype about?Exactly what is V isual Basic and what can it my do?Well ,it is an esay –to-use ,yet extraordinarily powerful tool for developing Windows applications .Before Visual Baisc was introduced ,developing Windows applications was much harder then developing Dos applications.Programmers had to worry about much,such as what the mouse was doing ,where the users was inside a menü,and whether he or she was clicking or double-clicking at a given place .Developnig a Windows application reguired expert C programmers and hundres of lines code for the simple task.even the experts had trouble.Visual Basic makes it easy to design the screen.you literally draw the user interface,almost as if you were using a point program.In addition,when you have finished drawing the interface ,the command buttons and other controls that you have placed in a blank window will automatically recognize user actions such as mouse movements and button clicks.Also comes with a menü design feature that makes creating both ordinary and pop-up a snap.

Because of reasons I want to developed my skills working on visual basic. My project is record operation. Visual basic have a data manager I used database management which Microsoft Access 7.0.More sophisticated databases,like the ones you can begin to build with the data manager(büild completely with Microsoft Access or the data access power of visula Baisc Professional) dont' fit indexed card problem.This makes it easy to avoid the update problem.They have many other advantages as well.There really is no convenient way to describe the underlying structure of the databases that you can build üsing the Access engine supplied with Visula Basic;that is what actually lies on the user's hard disk.

I have learned how is connect database management and visual basic projects. My program is similar to current account program. In program, users may make any operations for example new account, update account,delete account, search account etc. Generally commercial programs are write in visual programs for this reason basic project. My program is may use to commercial work.

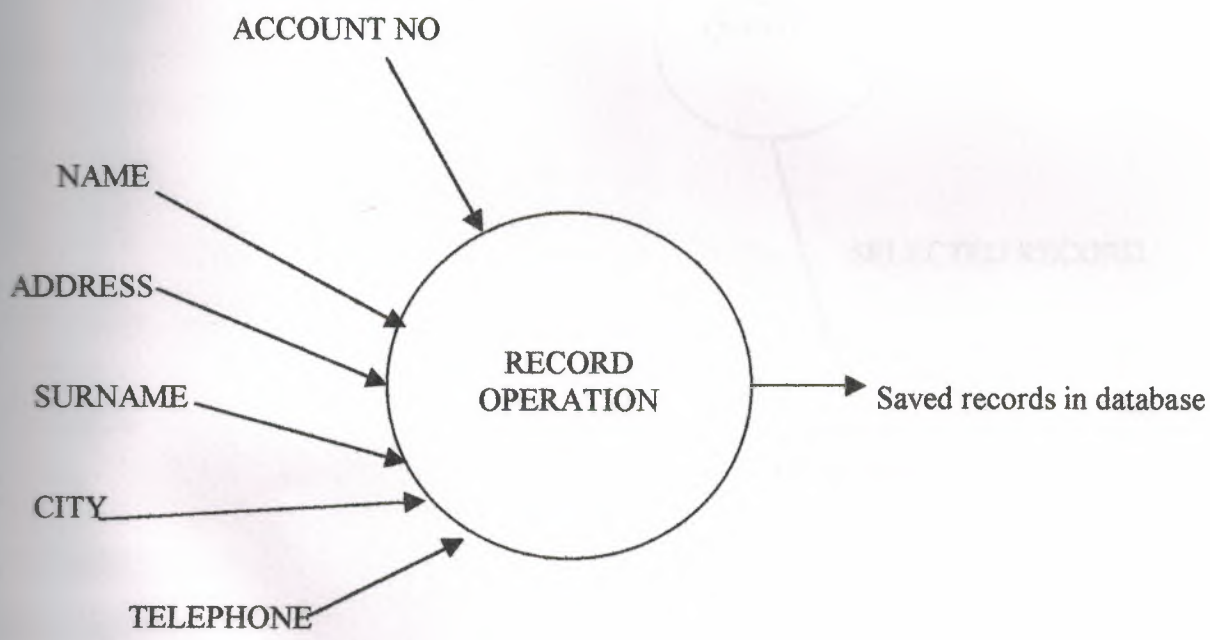
A lot of company want to have visual programmers.So After I graduate " gradution project" studies to get many advantages to me.

DATABASE

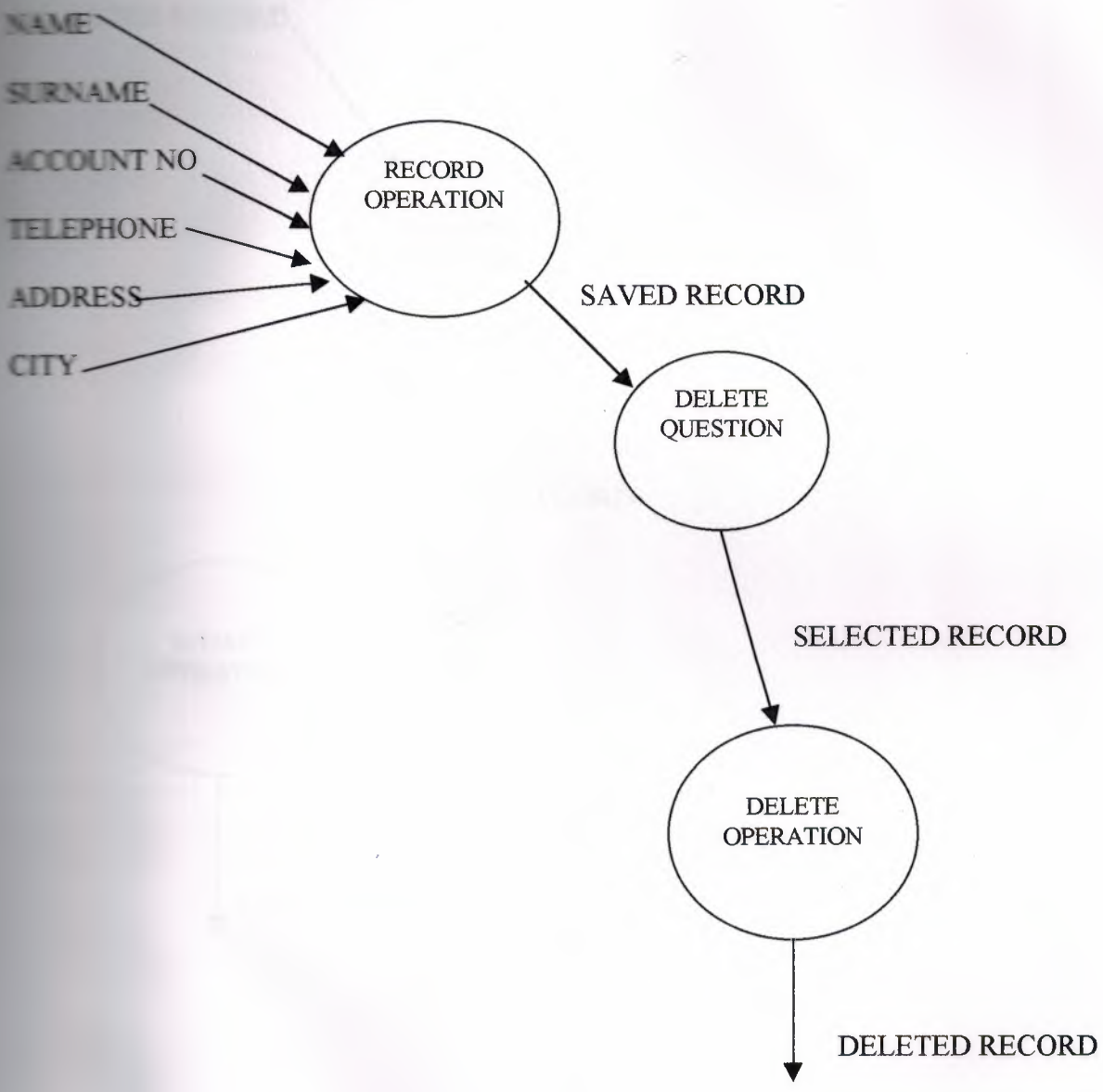
FIELD NAME	TYPE	SIZE
NAME	CHARACTER	15
SURNAME	CHARACTER	15
ACCOUNT NO	NUMERIC	7
TELEPHONE	CHARACTER	11
ADDRESS	CHARACTER	50
CITY	CHARACTER	15

DATA FLOW ANALYSIS

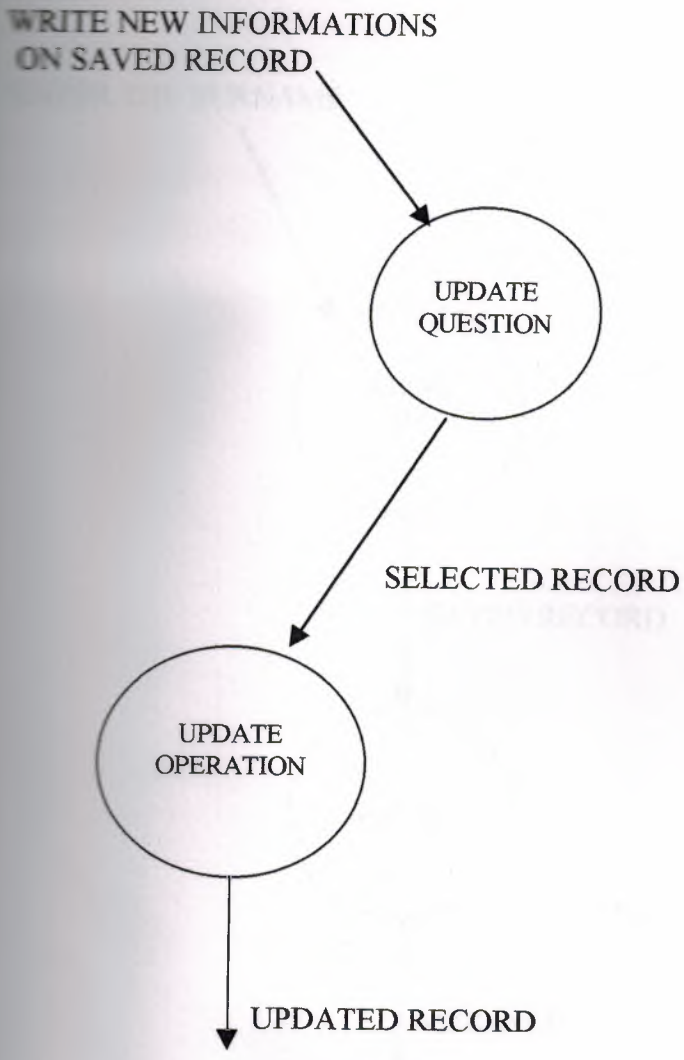
Add new record



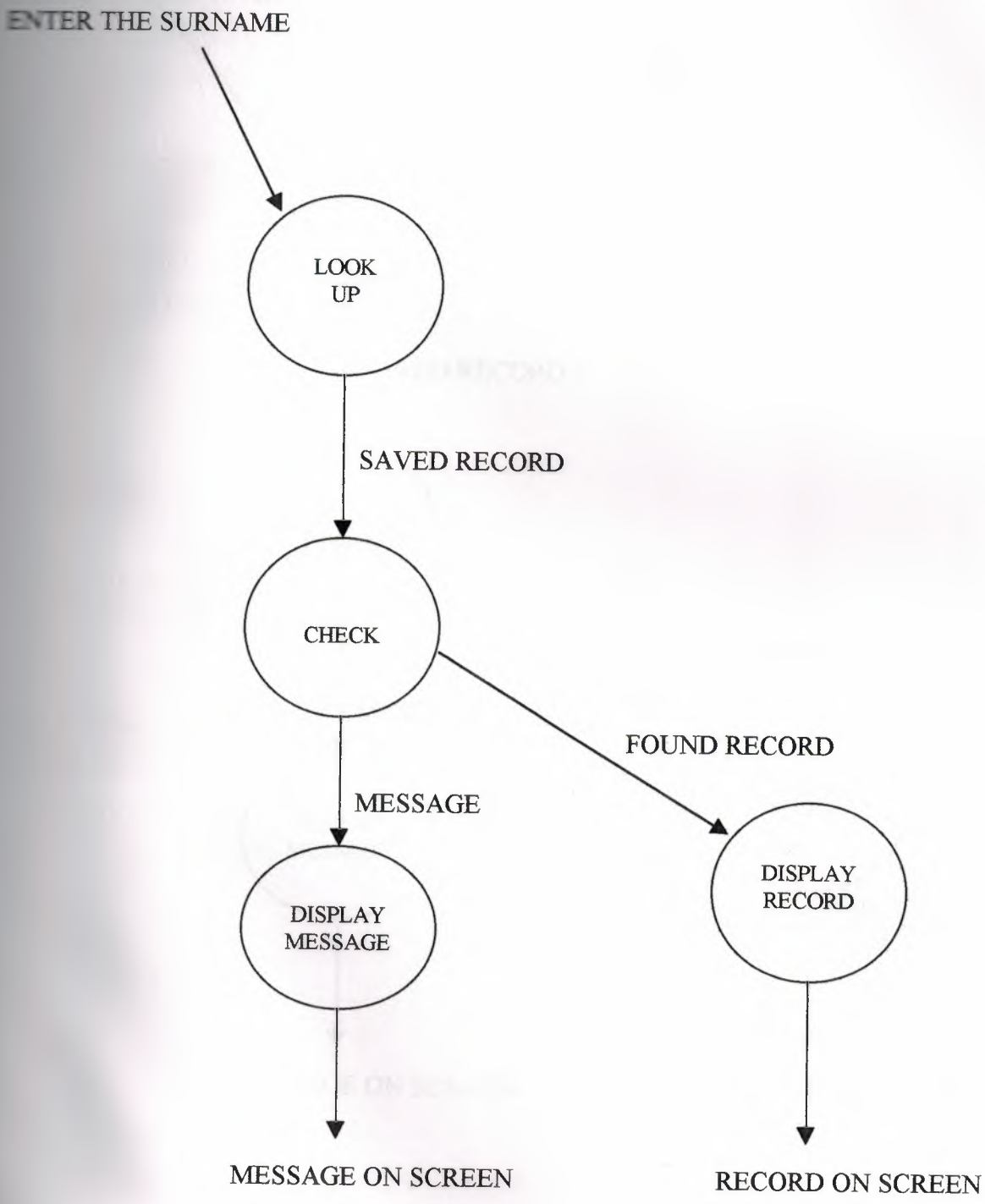
DELETE RECORD



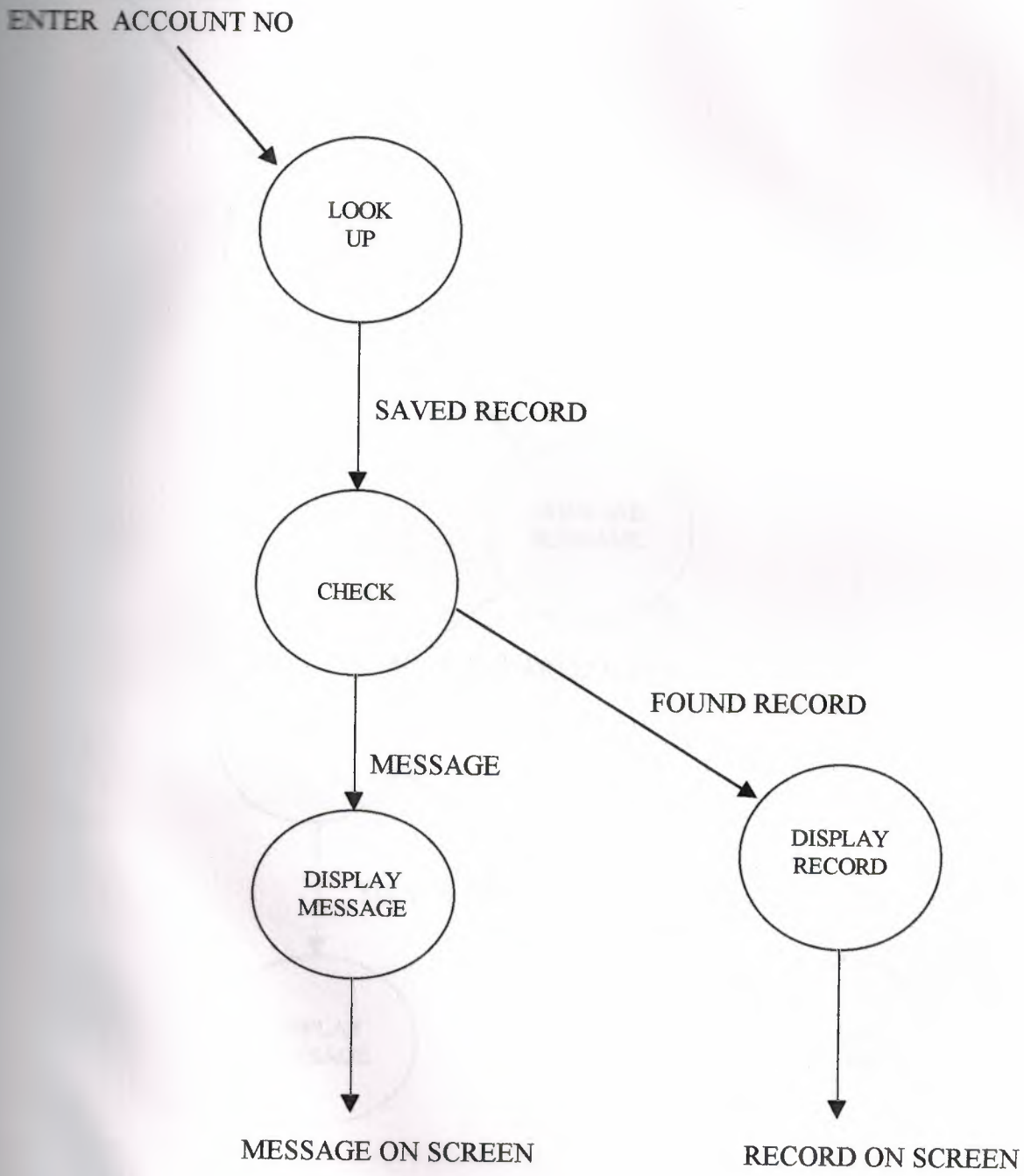
UPDATE RECORDS



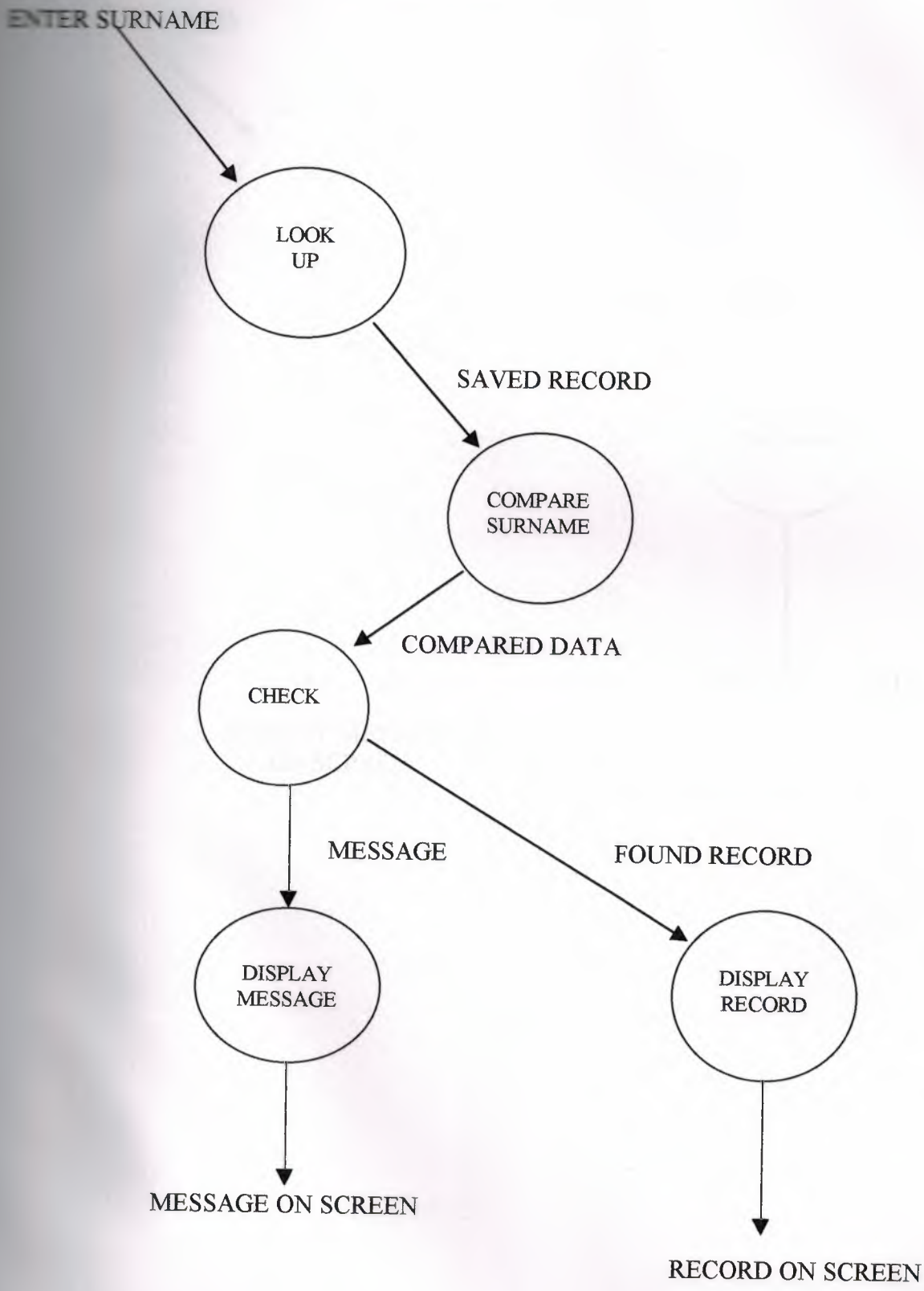
SEARCH RECORD BY SURNAME (IN PROJECT SEEK1)



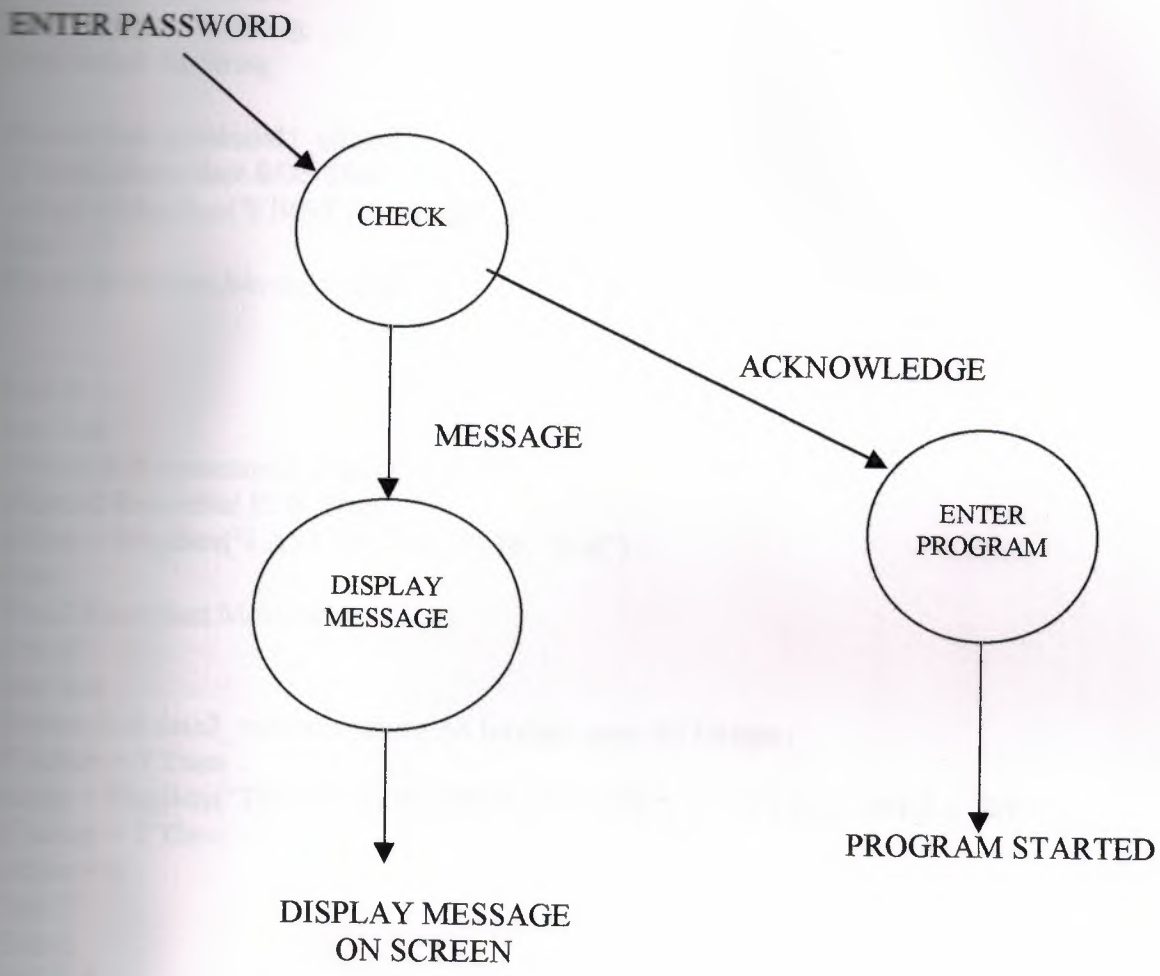
SEARCH RECORD BY ACCOUNT NO (IN PROJECT SEEK)



SEARCH BY SAME SURNAME(IN PROJECT SEEK2)



PASSWORD


ENTER PASSWORD
CHECK
ACKNOWLEDGE
MESSAGE
ENTER
PROGRAM
DISPLAY
MESSAGE
PROGRAM STARTED
DISPLAY MESSAGE
ON SCREEN

Graduation form1(Code)

```
Dim status As Variant
Dim measure As String
Dim soyad As String

Private Sub command1_click()
If Data2.Recordset.BOF Then
dikkat = MsgBox("FİRST RECORD", 16, "first")
Else
Data2.Recordset.MovePrevious


End If
End Sub
Private Sub command2_click()
If Data2.Recordset.EOF Then
dikkat = MsgBox("LAST RECORD", 16, "first")
Else
Data2.Recordset.MoveNext
End If
End Sub
Private Sub data2_validate(action As Integer, save As Integer)
If action = 7 Then
cevap = MsgBox("THİS RECORD WİİL DELETE", 4, "DELETE OPERATİON")
If cevap = 7 Then
action = 0
End If
End If
End Sub
Private Sub command3_click()
Data2.Recordset.Delete
Data2.Recordset.MoveNext
End Sub
Private Sub command4_click()
seçim = MsgBox("YOU WİLL ENTER NEW RECORD", 68, "NEW")
 If seçim = 6 Then
Data2.Recordset.AddNew
 Else
 action = 0
 End If
End Sub
Private Sub command5_click()
seçim = MsgBox("ARE YOU WANT TO UPDATE", 36)
If seçim = 6 Then
Data2.ReadOnly = False
Data2.Refresh
Data2.Recordset.Edit
Data2.Recordset.Update
Else
```

```
action = 0
End If
End Sub
Private Sub command6_click()
Data2.ReadOnly = True
Data2.Refresh
End Sub
Private Sub command7_click()
MsgBox Data2.Recordset.RecordCount
End Sub
Private Sub command8_click()
Surname$ = InputBox("ENTER THE SURNAME :")
measure = "Surname='" & Surname & "'"
Data2.Recordset.FindFirst measure
   End Sub
Private Sub command13_click()
AccountNo$ = InputBox("ENTER THE ACCOUNTNO:")
measure = "AccountNo='" & AccountNo & "'"
Data2.Recordset.FindFirst measure
End Sub
Private Sub command9_click()
Data2.Recordset.FindNext measure
End Sub
Private Sub command10_click()
seçim = MsgBox("EXİT THE RECORD OPERATİON", 4)
If seçim = 6 Then
End
End If
End Sub
Private Sub command11_click()
 Data2.Recordset.Bookmark = status
End Sub
Private Sub command12_click()
status = Data2.Recordset.Bookmark
End Sub
Private Sub timer1_timer()
Text6.Text = Time
End Sub
Private Sub data2_reposition()
Recordno = Data2.Recordset.AbsolutePosition
Recordnumber = Data2.Recordset.RecordCount
Data2.Caption = Str(Recordno + 1) + "/" + Str(Recordnumber)
End Sub
Private Sub command14_click()
Dim pr
pr = Shell("C:\Program Files\Microsoft Visual Studio\VB98\Visdata")
End Sub
Private Sub form_click()
MsgBox "exe file name: " & App.EXEName
End Sub
```

```
Sub form_mousemove(button As Integer, shift As Integer, x As Single, y As Single)
If x > 1000 And y > 3000 Then
MousePointer = 0
Else
MousePointer = 5
End If
End Sub
```

Graduate form2(code)

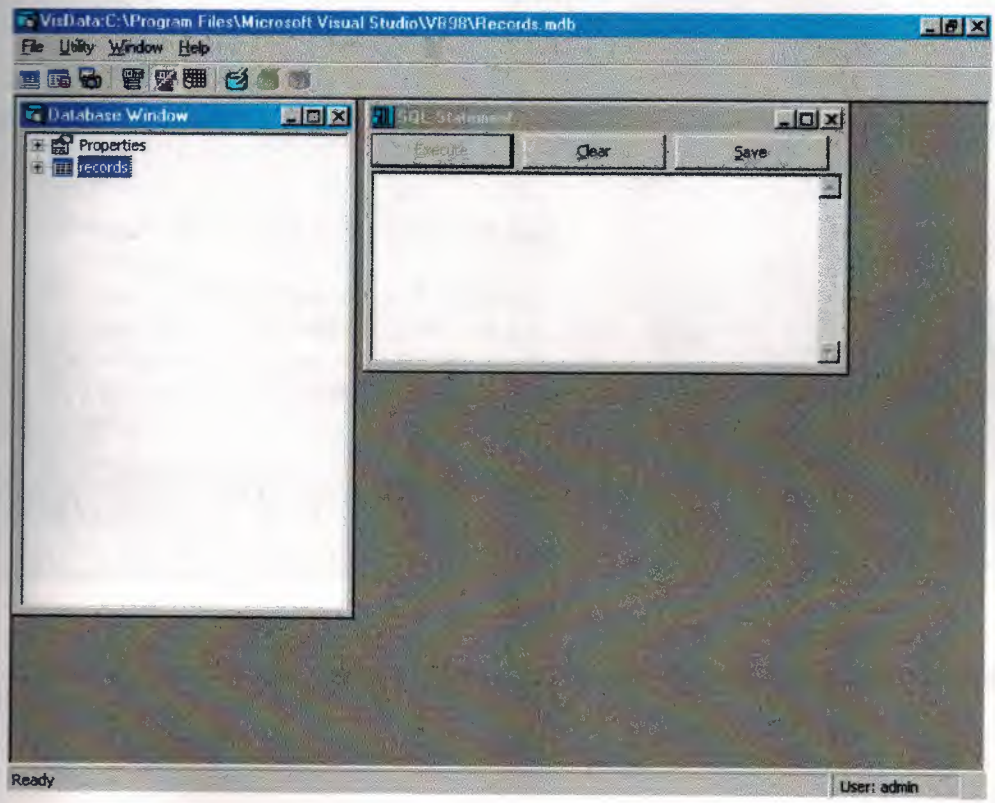
```
Private Sub Form_Load()
Text1.Text = ""
Text1.PasswordChar = "*"
End Sub
Private Sub text1_keypress(keyascii As Integer)
Static sifre As String
Static tur As Integer
sifre = sifre + Chr$(keyascii)
If Len(sifre) = 5 Then
If sifre <> "hakan" Then
seçim = MsgBox("şifreniz yanlış", 5, "mistake")
If seçim = 2 Then
End
Else
Text1.Text = ""
keyascii = 0
sifre = ""
End If
End If
If sifre = "hakan" Then
seçim = MsgBox("CORRECT CİPHER", 0, "correct")
If seçim = 1 Then
        Form1.Show
        End If
        End If
          End If
            End Sub
```

DATA MANAGER

In Visual Basic ,all database programs possible to examine .In visual basic to form we will form database files.Visual basic's and microsoft access database files features nearly same.

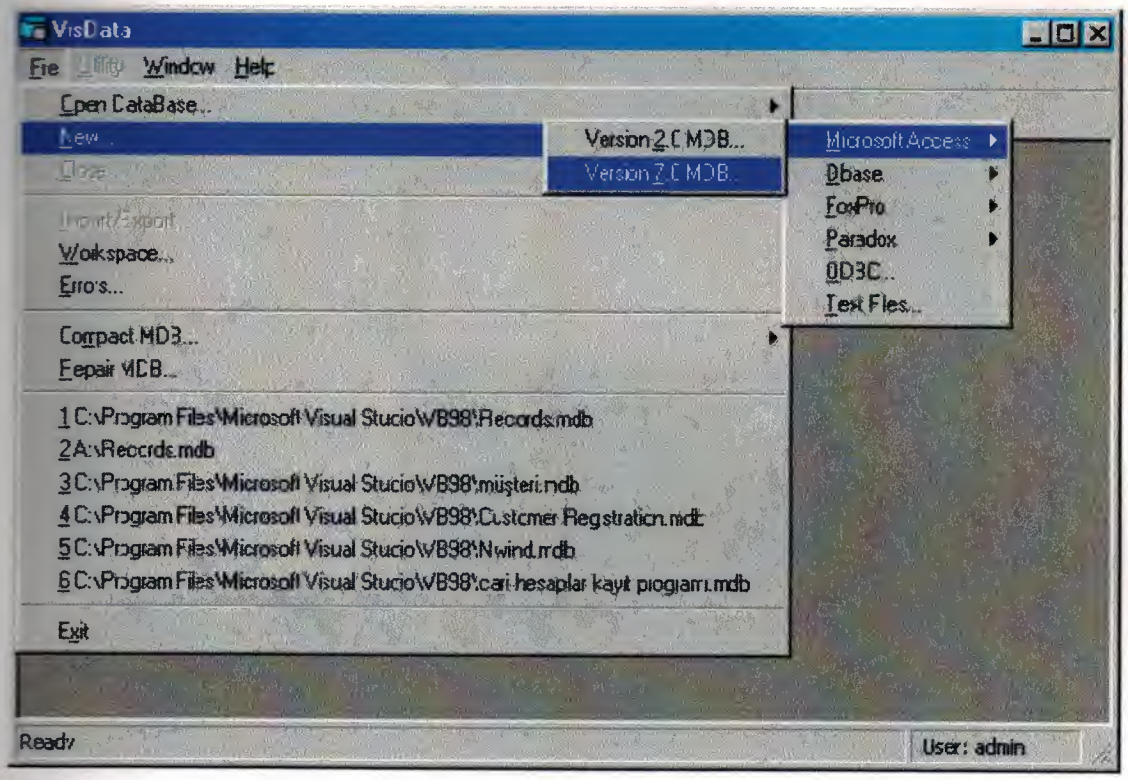
Working in visual basic professional edition may be formed extended part of .MDB database files. For the form of database files, we click add_ins menu and select the visual data maneger command also we don't need program code. Shown in figure 1.

Figure 1.



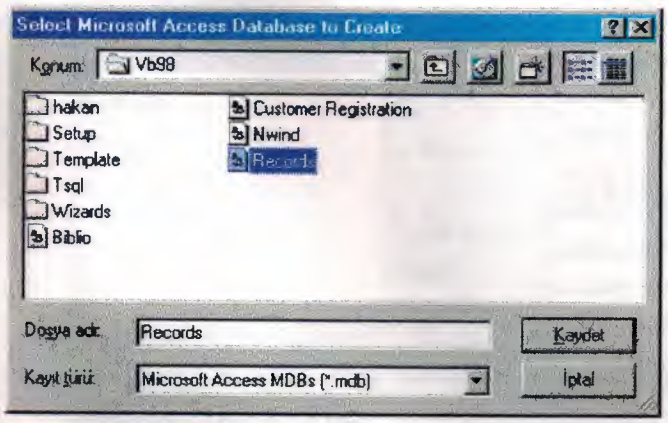
In visual basic with visual data manager we can prepare the format of access databases. This explanation showing in figure 2.

Figure 2.



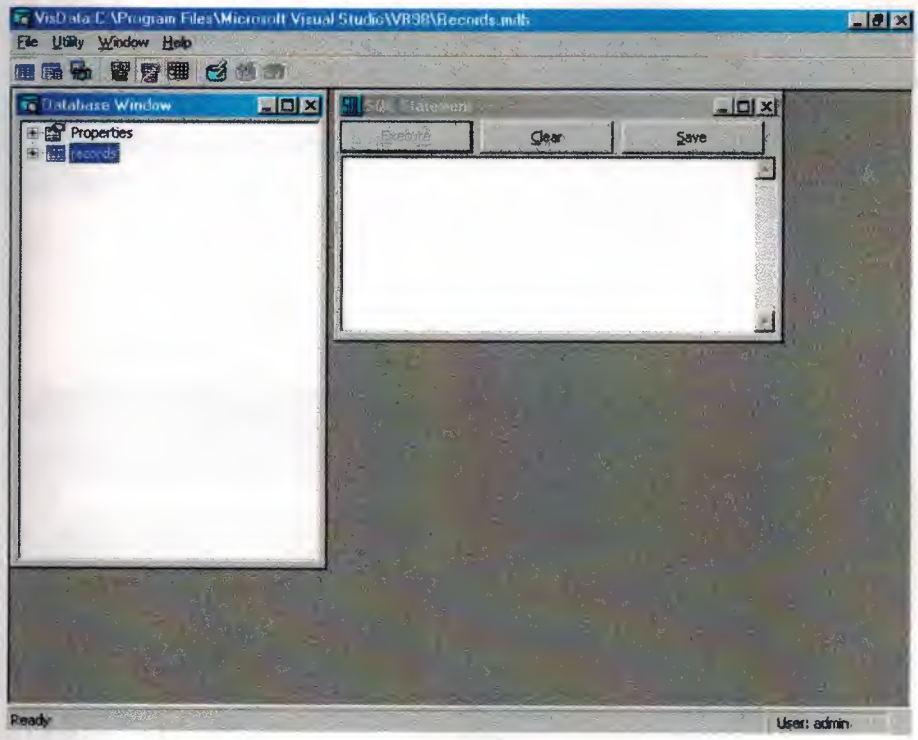
Write the dabase file names into **Select Microsoft Access Database to Create** dialog box. Shown in figure 3.

Figure 3.



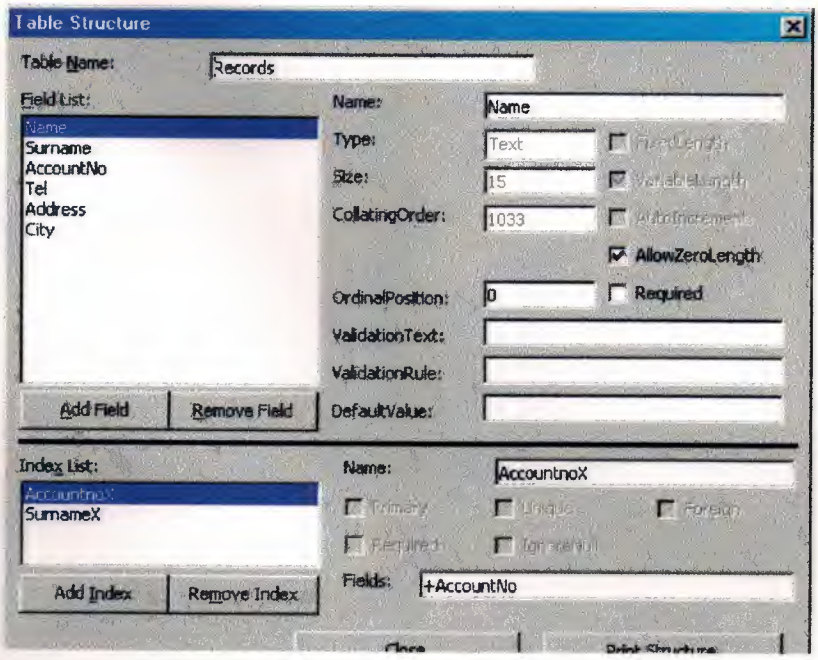
Then click Kaydet button and save the your database files.Then coming the following window.In figure 4.

Figure 4.



And then select the properties and click right button of mouse then select the newtable. Afterwards coming the **Table Structure** dialog box to screen. Shown in figure 5.

Figure 5.

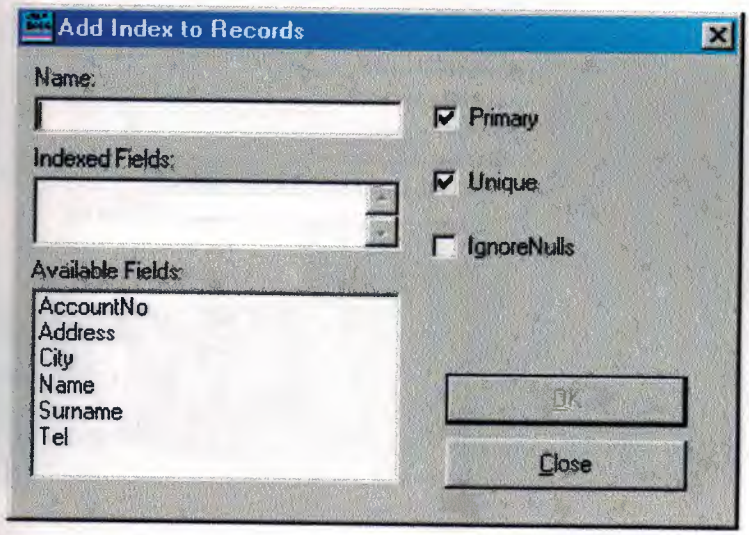


Click the Add Field button and you can enter field name,What you want to be formed in database.
If you want to change Table structure properties right click to records(From figure 4) and click the **design** then change, remove or add new fields.

Preparing To Index

In visual basic reading records ,to interrogate records from the tables for we use index. For add index ;we are goingto table structure dialog box then click **add index** button.We select any field according to key.If we want to use one primary key click the **primary** checkbox.If we dont want to same key in the table click the **unique** checkbox then click the close. Showing figure 6.

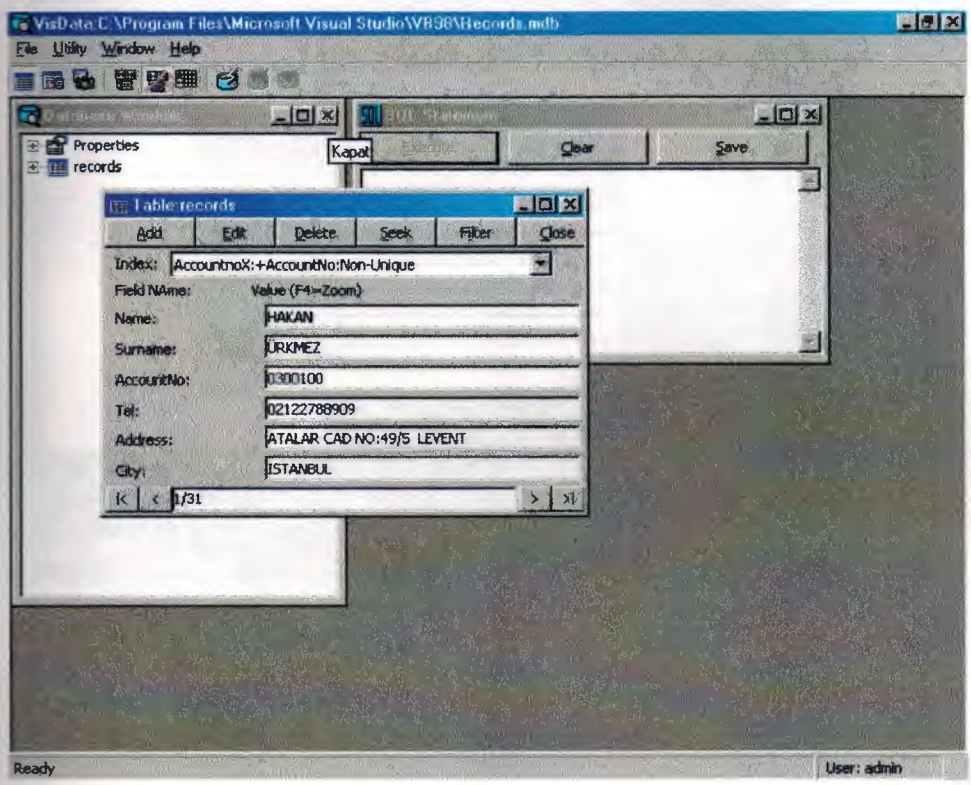
Figure 6.



Opening the Database Files

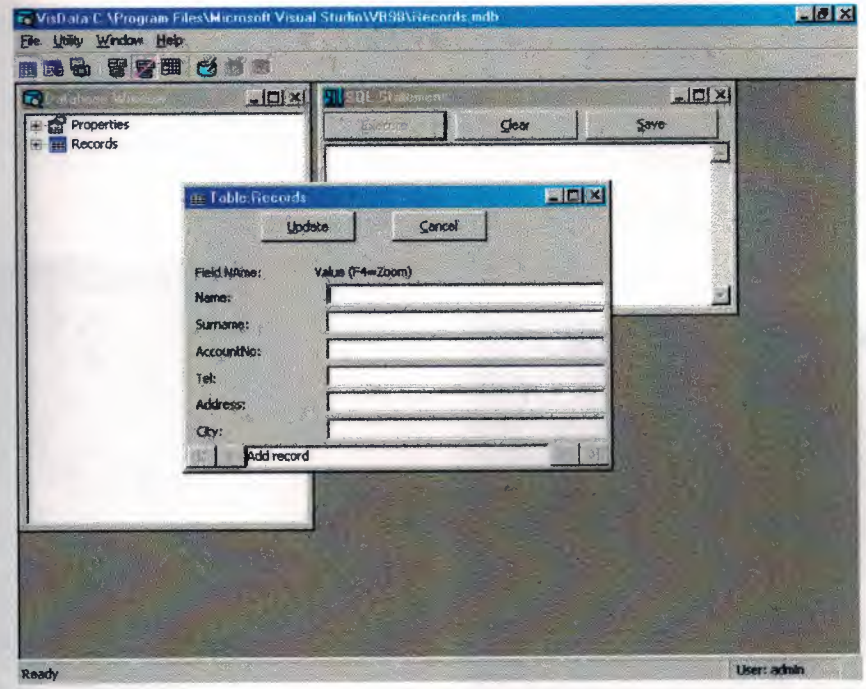
Click the right button on records (in figure 4) then select the open command .Then coming following figure 7.

Figure 7.



If we want to add new records click the add button then coming following figure 8. If we click the update button saving new records to table.

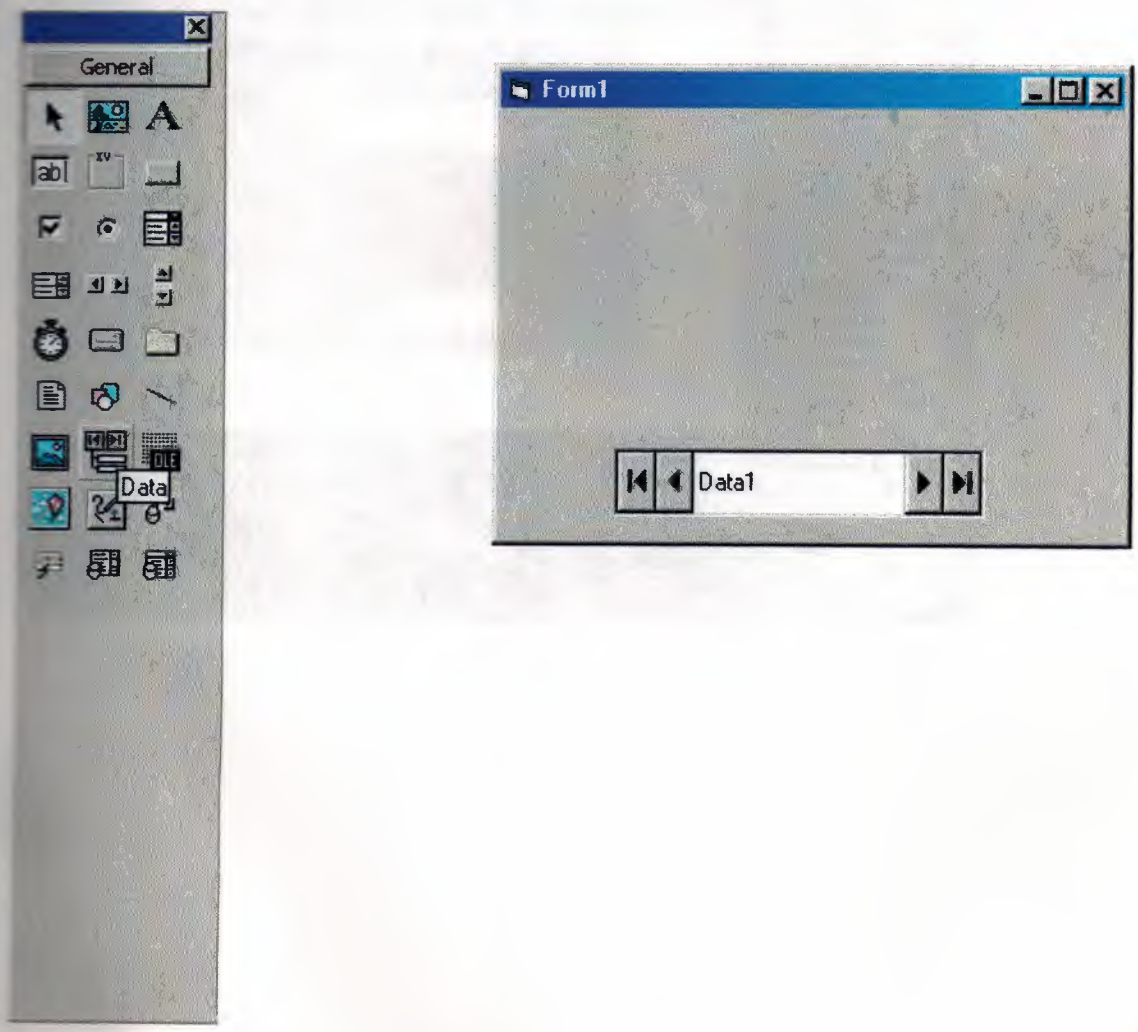
Figure 8.



DATA CONTROL

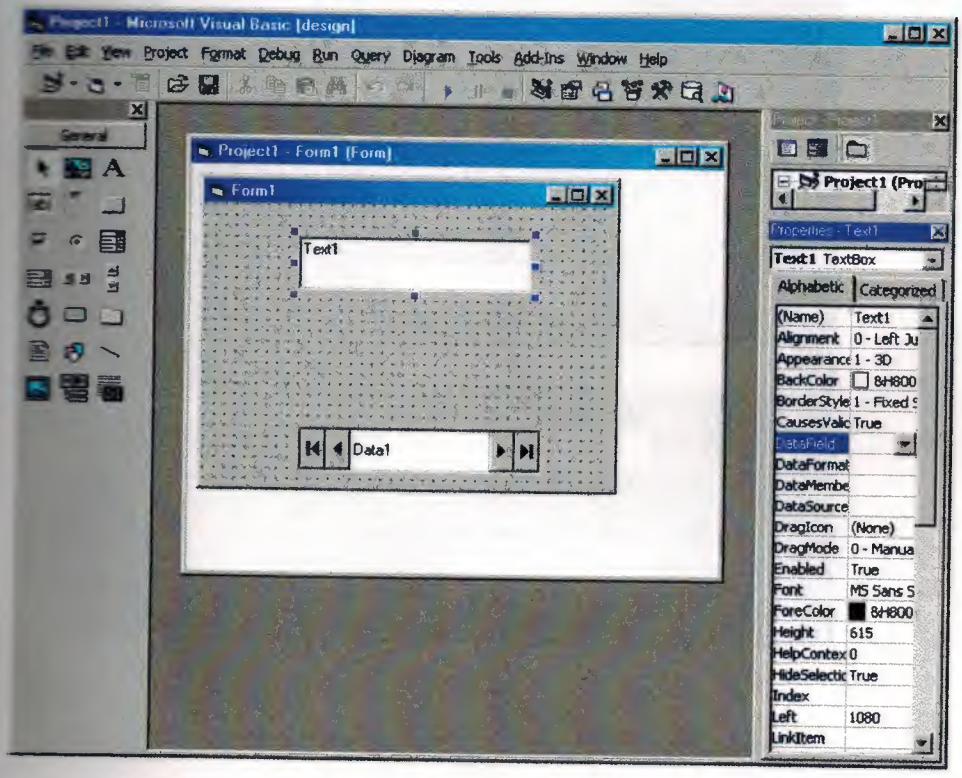
In visual basic we have a data object in processing database files .With **data** object database files easily put to form. In the following figure we can see data in toolbox.

Figure 9.



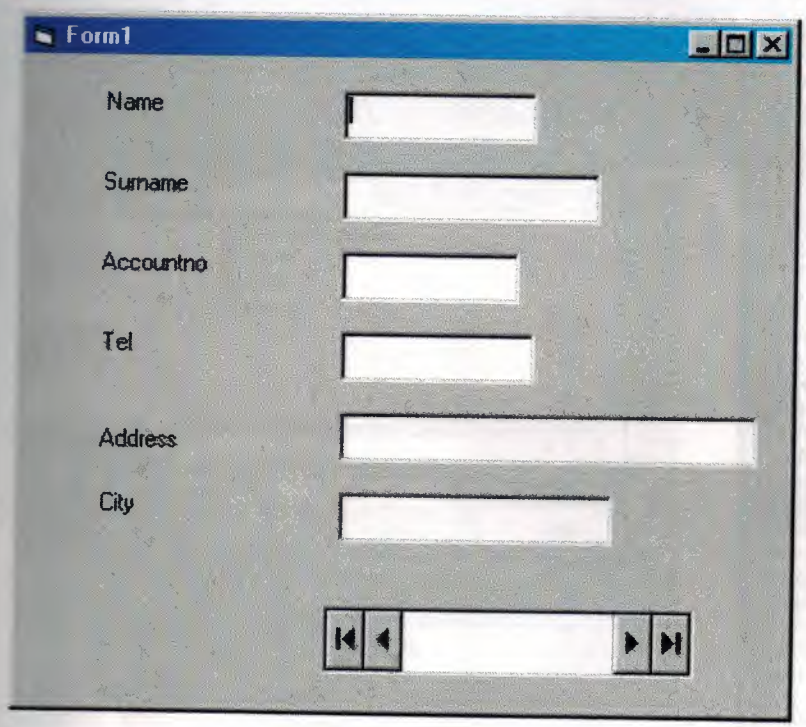
In database files records to write the form we use textbox.We use property **datafield** with connection between area of selected table(records) and textbox object. Showing following figure 10.

Figure 10.



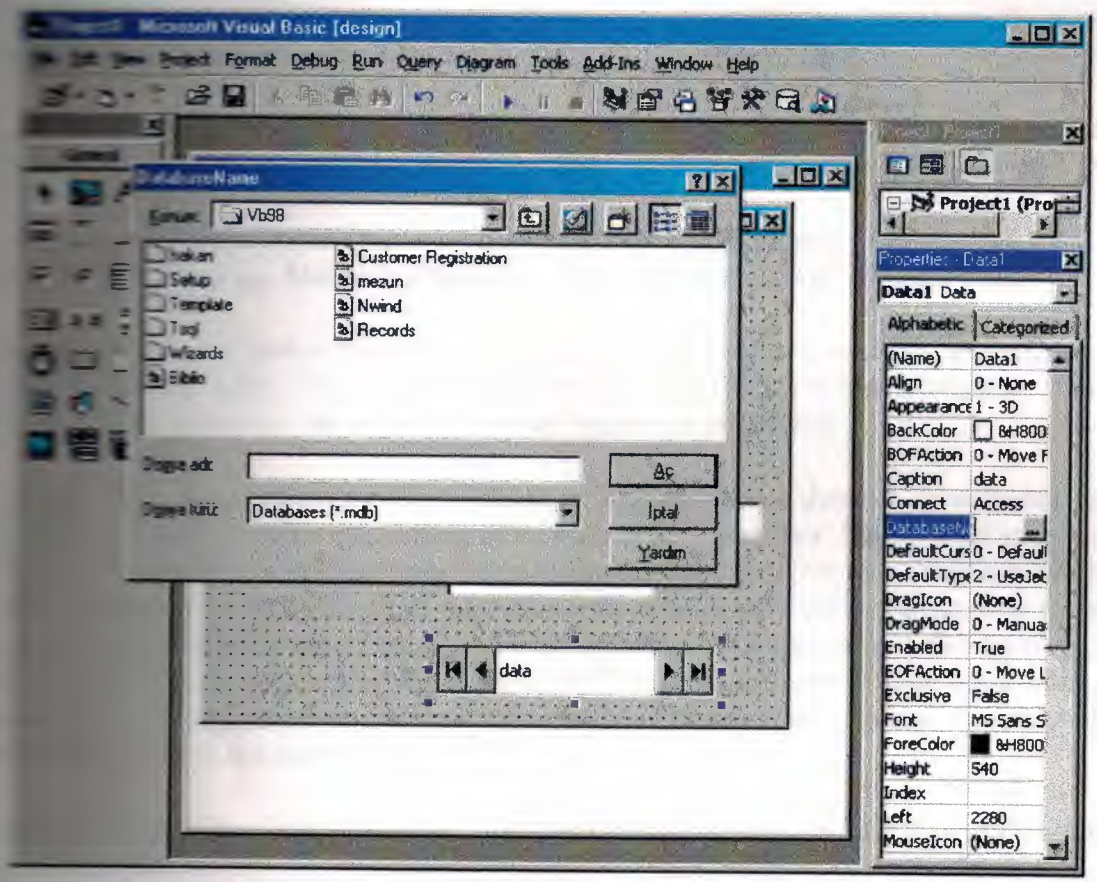
Property of toolboxes datafield,We dont use this property if there isn't **data** object on working project.Then connected textbox and records.mdb .And adding the **label** to textbox. Showing following figure 11.

Figure 11.



properties windows for data object's DatabaseName properties "records.mdb" and RecordSource properties name is given "records". Showing following figure 12.

Figure 12.



Examine the Records

To examine the records we use **MoveNext and MovePrevious** methods.

Move to the first, last, next, or previous record in a specified Recordset object and make that record the current record.

Syntax

recordset. MoveNext MovePrevious

The recordset placeholder is an object variable that represents an open Recordset object.

Use the Move methods to move from record to record without applying a condition.

Caution If you edit the current record, be sure you use the Update method to save the changes before you move to another record. If you move to another record without updating, your changes are lost without warning.

When you open a Recordset, the first record is current and the BOF property is False. If the Recordset contains no records, the BOF property is True, and there is no current record.

If the first or last record is already current when you use MoveFirst or MoveLast, the current record doesn't change.

If you use MovePrevious when the first record is current, the BOF property is True, and there is no current record. If you use MovePrevious again, an error occurs, and BOF remains True.

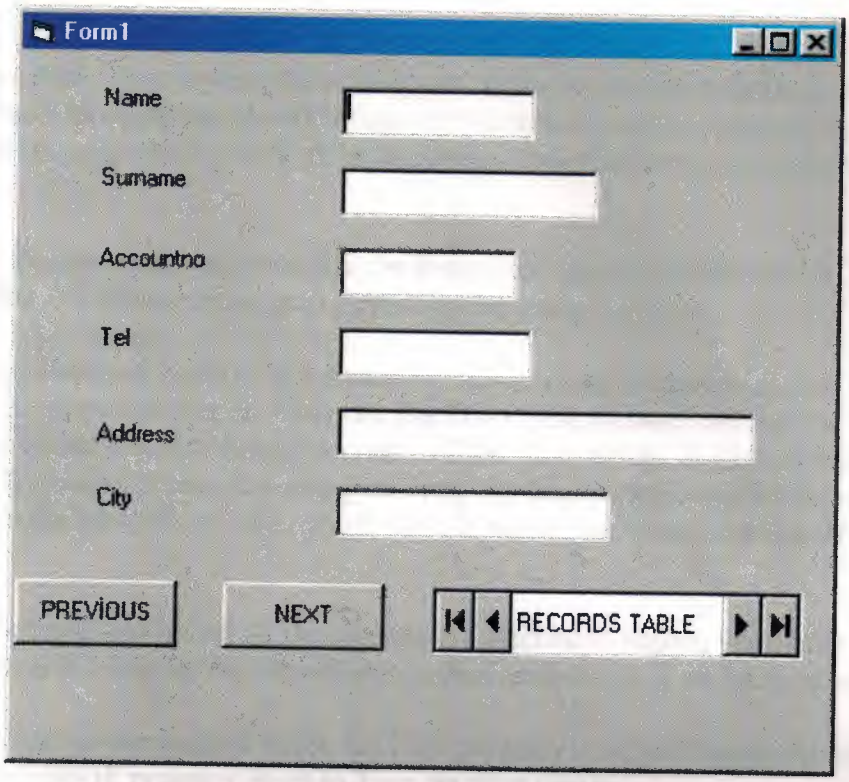
If you use MoveNext when the last record is current, the EOF property is True, and there is no current record. If you use MoveNext again, an error occurs, and EOF remains True.

If recordset refers to a table-type Recordset (Microsoft Jet workspaces only), movement follows the current index. You can set the current index by using the Index property. If you don't set the current index, the order of returned records is undefined.

Important You can use the MoveLast method to fully populate a dynaset- or snapshot-type Recordset to provide the current number of records in the Recordset. However, if you use MoveLast in this way, you can slow down your application's performance. You should only use MoveLast to get a record count if it is absolutely necessary to obtain an accurate record count on a newly opened Recordset. If you use the dbRunAsync constant with MoveLast, the method call is asynchronous. You can use the StillExecuting property to determine when the Recordset is fully populated, and you can use the Cancel method to terminate execution of the asynchronous MoveLast method call.

You can't use the MoveFirst, MoveLast, and MovePrevious methods on a forward-only–type Recordset object.

To move the position of the current record in a Recordset object a specific number of records forward or backward, use the Move method.



If MoveNext or MovePrevious on the **First or Last** record.We use **BOF and EOF** action

- BOF returns a value that indicates whether the current record position is before the first record in a Recordset object.
- EOF returns a value that indicates whether the current record position is after the last record in a Recordset object.

Return Values

The return values for the BOF and EOF properties are Boolean values.

The BOF property returns True if the current record position is before the first record, and False if the current record position is on or after the first record.

The EOF property returns True if the current record position is after the last record, and False if the current record position is on or before the last record.

You can use the BOF and EOF properties to determine whether a Recordset object contains records or whether you've gone beyond the limits of a Recordset object when you move from record to record.

The location of the current record pointer determines the BOF and EOF return values.

If either the BOF or EOF property is True, there is no current record.

If you open a Recordset object containing no records, the BOF and EOF properties are set to True

, and the Recordset object's RecordCount property setting is 0. When you open a Recordset object that contains at least one record, the first record is the current record and the BOF and EOF

properties are False; they remain False until you move beyond the beginning or end of the Recordset object by using the MovePrevious or MoveNext method, respectively. When you move beyond the beginning or end of the Recordset, there is no current record or no record exists.

If you delete the last remaining record in the Recordset object, the BOF and EOF properties may remain False until you attempt to reposition the current record.

If you use the MoveLast method on a Recordset object containing records, the last record becomes the current record; if you then use the MoveNext method, the current record becomes invalid and the EOF property is set to True. Conversely, if you use the MoveFirst method on a Recordset object containing records, the first record becomes the current record; if you then use the MovePrevious method, there is no current record and the BOF property is set to True.

Typically, when you work with all the records in a Recordset object, your code will loop through the records by using the MoveNext method until the EOF property is set to True.

If you use the MoveNext method while the EOF property is set to True or the MovePrevious method while the BOF property is set to True, an error occurs.

This table shows which Move methods are allowed with different combinations of the BOF and EOF
properties.

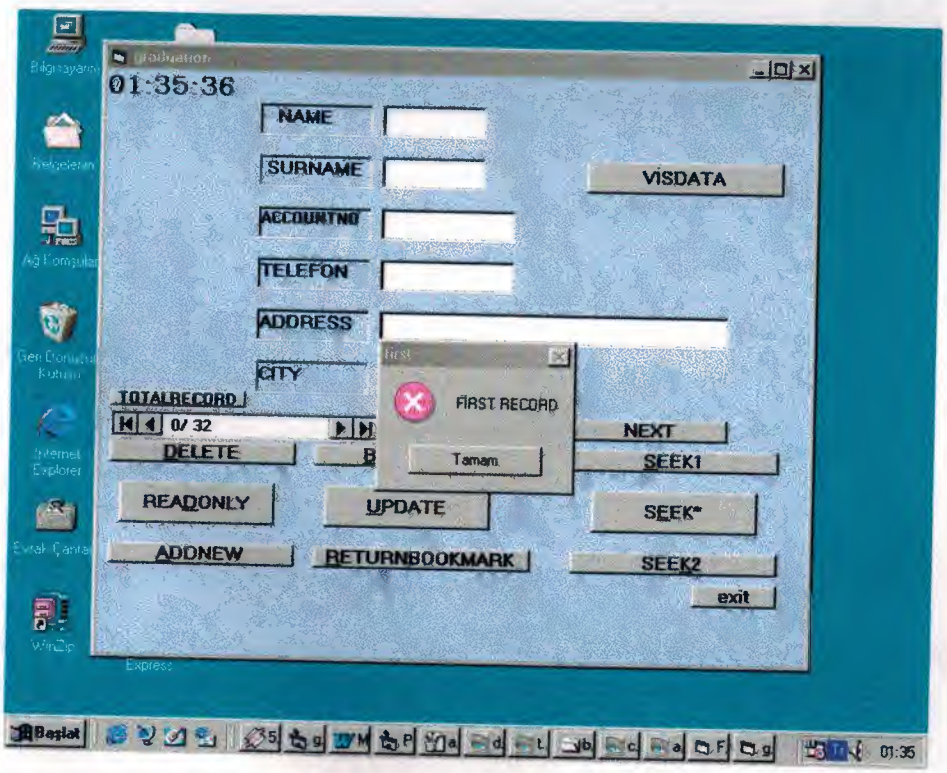
MoveFirst, MoveLast	MovePrevious, Move < 0	Move 0	MoveNext, Move > 0
BOF=True, EOF=False	Allowed	Error	Error Allowed
BOF=False, EOF=True	Allowed	Allowed	Error Error
Both True	Error Error	Error	Error
Both False	Allowed	Allowed	Allowed Allowed

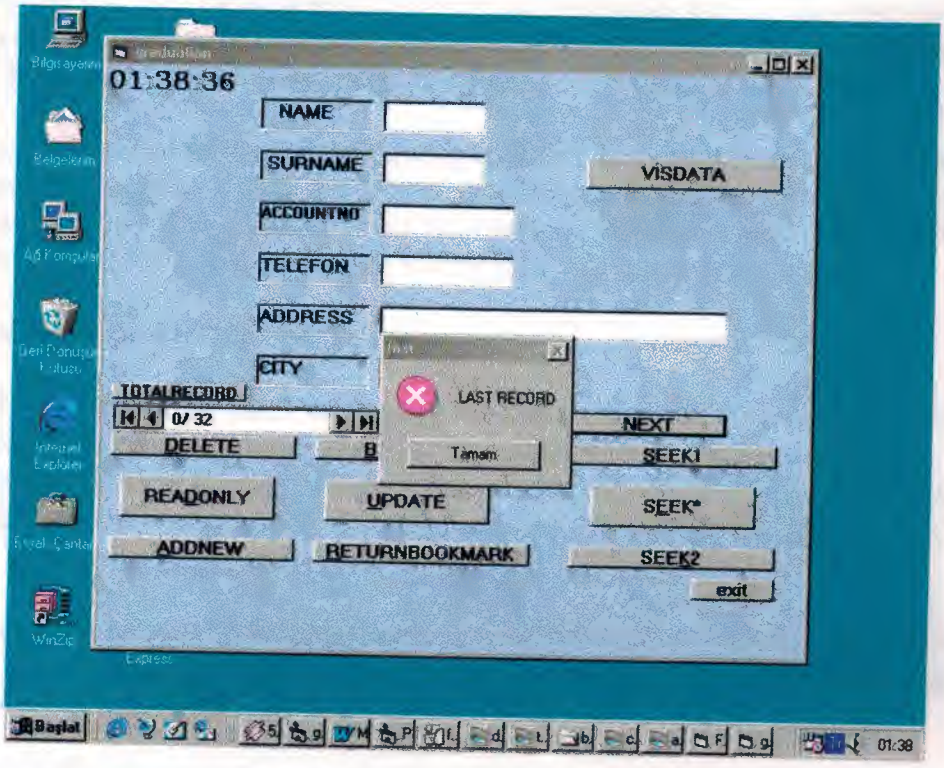
Allowing a Move method doesn't mean that the method will successfully locate a record. It merely indicates that an attempt to perform the specified Move method is allowed and won't generate an error. The state of the BOF and EOF properties may change as a result of the attempted Move.
An OpenRecordset method internally invokes a MoveFirst method. Therefore, using an OpenRecordset method on an empty set of records sets the BOF and EOF properties to True. (See the following table for the behavior of a failed MoveFirst method.)

All Move methods that successfully locate a record will set both BOF and EOF to False. In a Microsoft Jet workspace, if you add a record to an empty Recordset, BOF will become False, but EOF will remain True, indicating that the current position is at the end of Recordset. In an ODBCDirect workspace, both BOF and EOF will become False, indicating that the current position is on the new record.

Any Delete method, even if it removes the only remaining record from a Recordset, won't change the setting of the BOF or EOF property.
The following table shows how Move methods that don't locate a record affect the BOF and EOF property settings.

	BOF	EOF
MoveFirst, MoveLast	True	True
Move 0	No change	No change
MovePrevious, Move < 0	True	No change
MoveNext, Move > 0	No change	True





Add New Record

Creates a new record for an updatable Recordset object.

Syntax

recordset.AddNew

The recordset placeholder is an object variable that represents an updatable Recordset object to which you want to add a new record.

Use the AddNew method to create and add a new record in the Recordset object named by recordset. This method sets the fields to default values, and if no default values are specified, it sets the fields to Null (the default values specified for a table-type Recordset).

After you modify the new record, use the Update method to save the changes and add the record to the Recordset. No changes occur in the database until you use the Update method.

Caution If you issue an AddNew and then perform any operation that moves to another record, but without using Update, your changes are lost without warning. In addition, if you close the Recordset or end the procedure that declares the Recordset or its Database object, the new record is discarded without warning.

Note When you use AddNew in a Microsoft Jet workspace and the database engine has to create a new page to hold the current record, page locking is pessimistic. If the new record fits in an existing page, page locking is optimistic.

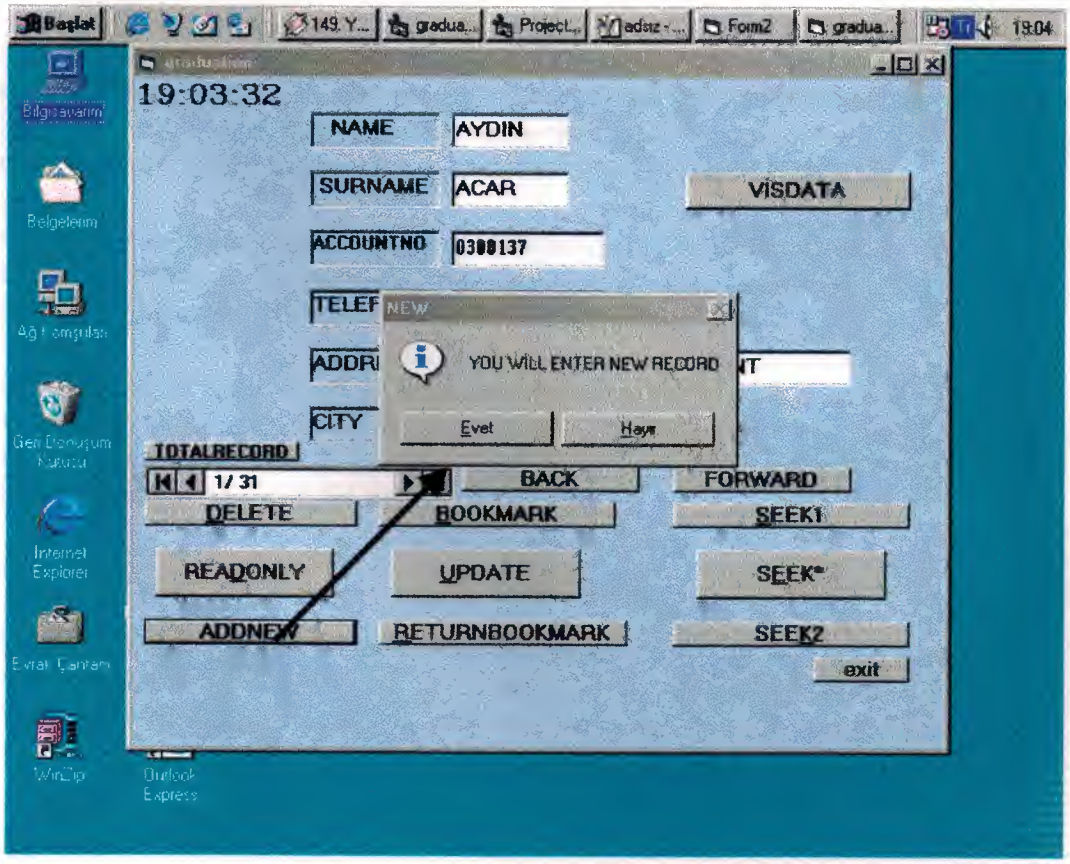
If you haven't moved to the last record of your Recordset, records added to base tables by other processes may be included if they are positioned beyond the current record. If you add a record to your own Recordset, however, the record is visible in the Recordset and included in the underlying table where it becomes visible to any new Recordset objects.

The position of the new record depends on the type of Recordset:

- In a dynaset-type Recordset object, records are inserted at the end of the Recordset, regardless of any sorting or ordering rules that were in effect when the Recordset was opened.
- In a table-type Recordset object whose Index property has been set, records are returned in their proper place in the sort order. If you haven't set the Index property, new records are returned at the end of the Recordset.

The record that was current before you used AddNew remains current. If you want to make the new record current, you can set the Bookmark property to the bookmark identified by the LastModified property setting.

Note To add, edit, or delete a record, there must be a unique index on the record in the underlying data source. If not, a "Permission denied" error will occur on the AddNew, Delete, or Edit method call in a Microsoft Jet workspace, or an "Invalid argument" error will occur on the Update call in an ODBCDirect workspace.



Delete Record

Recordset objects — deletes the current record in an updatable Recordset object. For ODBCDirect workspaces, the type of driver determines whether Recordset objects are updatable and therefore support the Delete method.

- Collections — deletes a persistent object from a collection.

Syntax

recordset.Delete
collection.Delete objectname

The Delete method syntax has these parts.

Part	Description
recordset	An object variable that represents an updatable Recordset object containing the record you want to delete.
collection	An object variable that represents a collection from which you are deleting objectname.
objectname	A String that is the Name property setting of an object in collection.

You can use the Delete method to delete a current record from a Recordset or a member from a collection, such as a stored table from a database, a stored field from a table, or a stored index from a table.

Recordsets

A Recordset must contain a current record before you use Delete; otherwise, a run-time error occurs.
In an updatable Recordset object, Delete removes the current record and makes it inaccessible. Although you can't edit or use the deleted record, it remains current. Once you move to another record, however, you can't make the deleted record current again. Subsequent references to a deleted record in a Recordset are invalid and produce an error.

You can undo a record deletion if you use transactions and the Rollback method.
If the base table is the primary table in a cascading delete relationship, deleting the current record may also delete one or more records in a foreign table.

Note To add, edit, or delete a record, there must be a unique index on the record in the underlying data source. If not, a "Permission denied" error will occur on the AddNew, Delete, or Edit method call in a Microsoft Jet workspace, or an "Invalid argument" error will occur on the Update method call in an ODBCDirect workspace.

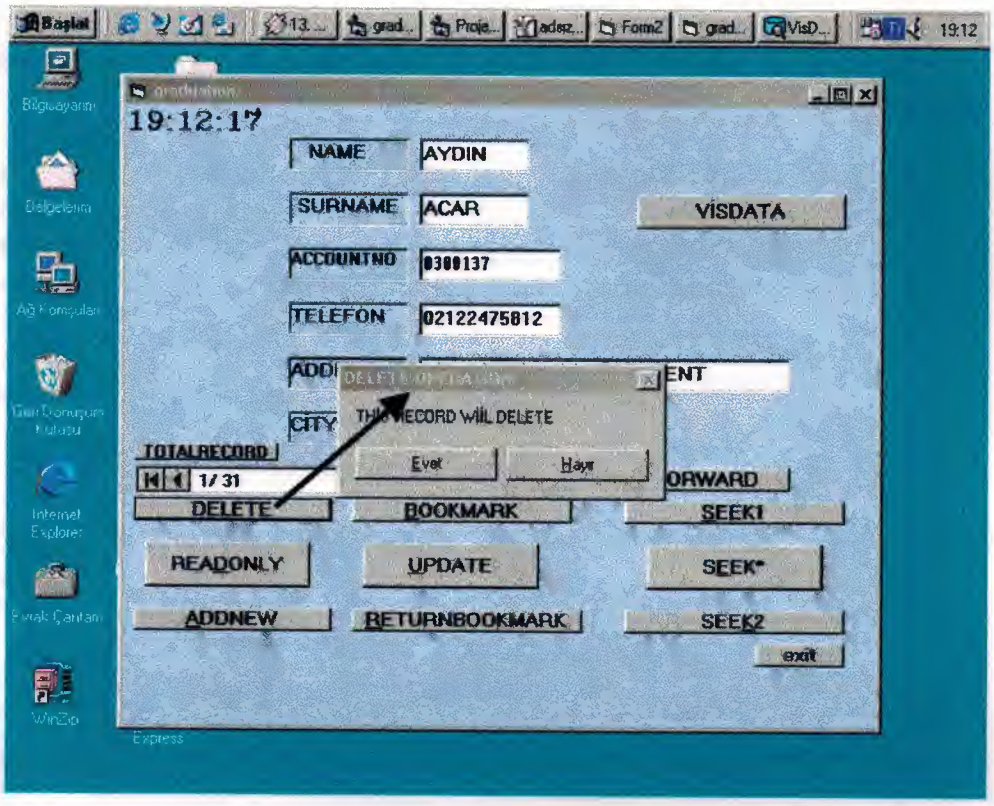
Collections

You can use the Delete method to delete a persistent object. However, if the collection is a Databases, Recordsets, or Workspaces collection (each of which is stored only in memory), you can remove an open or active object only by closing that object with the Close method.

The deletion of a stored object occurs immediately, but you should use the Refresh method on any other collections that may be affected by changes to the database structure.

When you delete a TableDef object from the TableDefs collection, you delete the table definition and the data in the table.
The following table lists some limitations of the Delete method. The object in the first column contains the collection in the second column. The third column indicates if you can delete an object from that collection (for example, you can never delete a Container object from the Containers collection of a Database object).

Object	Collection	Can you use the Delete method?
DBEngine	Workspaces	No. Closing the objects deletes them.
DBEngine	Errors	No
Workspace	Connections	No. Closing the objects deletes them.
Workspace	Databases	No. Closing the objects deletes them.
Workspace	Groups	Yes
Workspace	Users	Yes
Connection	QueryDefs	No
Connection	Recordsets	No. Closing the objects deletes them.
Database	Containers	No
Database	QueryDefs	Yes
Database	Recordsets	No. Closing the objects deletes them.
Database	Relations	Yes
Database	TableDefs	Yes
Group	Users	Yes
User	Groups	Yes
Container	Documents	No
QueryDef	Fields	No
QueryDef	Parameters	No
Recordset	Fields	No
Relation	Fields	Only when the Relation object is a new, unappended object.
TableDef	Fields	Only when the TableDef object is new and hasn't been appended to the database, or when the Updatable property of the TableDef is set to True.
TableDef	Indexes	Only when the TableDef object is new and hasn't been appended to the database, or when the Updatable property of the TableDef is set to True.
Index	Fields	Only when the Index object is new and hasn't been appended to the database.
Database, Field, Index, QueryDef, TableDef	Properties	Only when the property is user-defined.
DBEngine, Parameter, Recordset, Workspace	Properties	No



Update Any Records
Syntax

recordset.Update (type, force)

The Update method syntax has the following parts.

Part Description
recordset An object variable that represents an open, updatable Recordset object.
type Optional. A constant indicating the type of update, as specified in Settings (ODBCDirect workspaces only).
force Optional. A Boolean value indicating whether or not to force the changes into the database, regardless of whether the underlying data has been changed by another user since the AddNew, Delete, or Edit call. If True, the changes are forced and changes made by other users are simply overwritten. If False (default), changes made by another user while the update is pending will cause the update to fail for those changes that are in conflict. No error occurs, but the BatchCollisionCount and BatchCollisions properties will indicate the number of conflicts and the rows affected by conflicts, respectively (ODBCDirect workspaces only).
Settings

You can use the following values for the type argument. You can use the non-default values only if batch updating is enabled.

Constant	Description
dbUpdateRegular	Default. Pending changes aren't cached and are written to disk immediately.
dbUpdateBatch	All pending changes in the update cache are written to disk.
dbUpdateCurrentRecord	Only the current record's pending changes are written to disk.

Use Update to save the current record and any changes you've made to it.

Caution Changes to the current record are lost if:

- You use the Edit or AddNew method, and then move to another record without first using Update.
- You use Edit or AddNew, and then use Edit or AddNew again without first using Update.
- You set the Bookmark property to another record.
- You close recordset without first using Update.
- You cancel the Edit operation by using CancelUpdate.

To edit a record, use the Edit method to copy the contents of the current record to the copy buffer. If you don't use Edit first, an error occurs when you use Update or attempt to change a field's value.

In an ODBCDirect workspace, you can do batch updates, provided the cursor library supports batch updates, and the Recordset was opened with the optimistic batch locking option.

In a Microsoft Jet workspace, when the Recordset object's LockEdits property setting is True (
pessimistically locked) in a multiuser environment, the record remains locked from the time Edit is used until the Update method is executed or the edit is canceled. If the LockEdits property setting is False (optimistically locked), the record is locked and compared with the pre-edited record just before it is updated in the database. If the record has changed since you used the Edit method, the Update operation fails. Microsoft Jet-connected ODBC and installable ISAM databases always use optimistic locking. To continue the Update operation with your changes, use the Update method again. To revert to the record as the other user changed it, refresh the current record by using Move 0
.

Note To add, edit, or delete a record, there must be a unique index on the record in the underlying data source. If not, a "Permission denied" error will occur on the AddNew, Delete, or Edit method call in a Microsoft Jet workspace, or an "Invalid argument" error will occur on the Update call in an ODBCDirect workspace.

If you want to include all the records in your search — not just those that meet a specific condition — use the Move methods to move from record to record. To locate a record in a table-type Recordset, use the Seek method.
If a record matching the criteria isn't located, the current record pointer is unknown, and the NoMatch property is set to True. If recordset contains more than one record that satisfies the criteria, FindFirst
locates the first occurrence, FindNext locates the next occurrence, and so on.

Each of the Find methods begins its search from the location and in the direction specified in the following table.

Find method	Begins searching at	Search direction
FindFirst	Beginning of recordset	End of recordset
FindLast	End of recordset	Beginning of recordset
FindNext	Current record	End of recordset
FindPrevious	Current record	Beginning of recordset

When you use the FindLast method, the Microsoft Jet database engine fully populates your Recordset before beginning the search, if this hasn't already happened.
Using one of the Find methods isn't the same as using a Move method, however, which simply makes the first, last, next, or previous record current without specifying a condition. You can follow a Find operation with a Move operation.
Always check the value of the NoMatch property to determine whether the Find operation has succeeded. If the search succeeds, NoMatch is False. If it fails, NoMatch is True and the current record isn't defined. In this case, you must position the current record pointer back to a valid record.

Using the Find methods with Microsoft Jet-connected ODBC-accessed recordsets can be inefficient. You may find that rephrasing your criteria to locate a specific record is faster, especially when working with large recordsets.
In an ODBCDirect workspace, the Find and Seek methods are not available on any type of Recordset object, because executing a Find or Seek through an ODBC connection is not very efficient over the network. Instead, you should design the query (that is, using the source argument to the OpenRecordset method) with an appropriate WHERE clause that restricts the returned records to only those that meet the criteria you would otherwise use in a Find or Seek method.

When working with Microsoft Jet-connected ODBC databases and large dynaset-type Recordset objects, you might discover that using the Find methods or using the Sort or Filter property is slow. To improve performance, use SQL queries with customized ORDER BY or WHERE clauses, parameter queries, or QueryDef objects that retrieve specific indexed records.
You should use the U.S. date format (month-day-year) when you search for fields containing dates, even if you're not using the U.S. version of the Microsoft Jet database engine; otherwise, the data may not be found. Use the Visual Basic Format function to convert the date. For example:

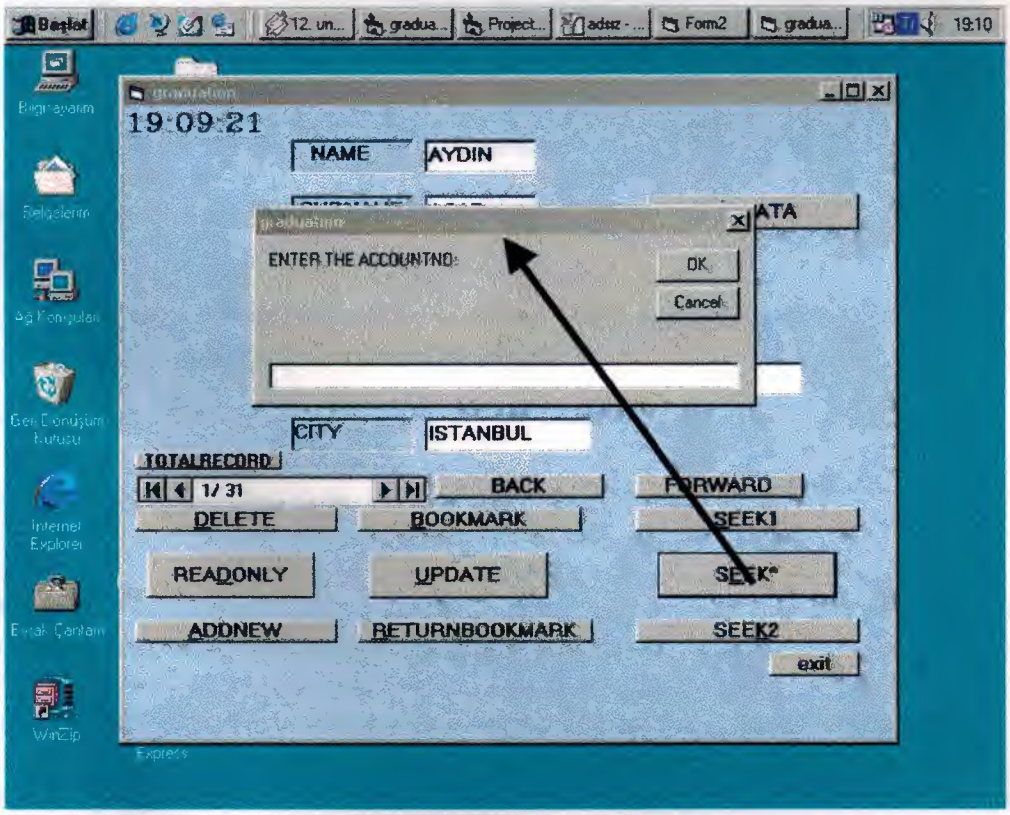
```
rstEmployees.FindFirst "HireDate > #" _
        & Format(mydate, 'm-d-yy' ) & "#"
```

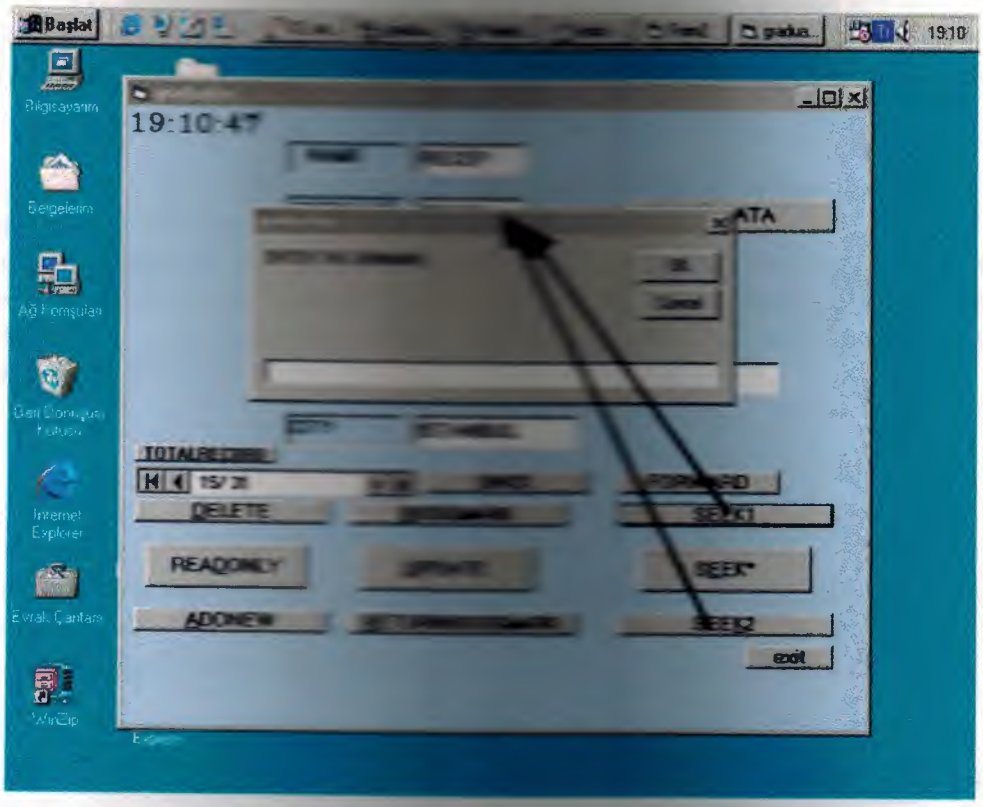
If criteria is composed of a string concatenated with a non-integer value, and the system parameters specify a non-U.S. decimal character such as a comma (for example, strSQL = "PRICE > " & lngPrice, and lngPrice = 125,50), an error occurs when you try to call the method. This is because during concatenation, the number will be converted to a string using your system's default decimal character, and Microsoft Jet SQL only accepts U.S. decimal characters.

Notes

· For best performance, the criteria should be in either the form "field = value" where field is an indexed field in the underlying base table, or "field LIKE prefix" where field is an indexed field in the underlying base table and prefix is a prefix search string (for example, "ART*").

· In general, for equivalent types of searches, the Seek method provides better performance than the Find methods. This assumes that table-type Recordset objects alone can satisfy your needs.





Record Count and Record No

Returns the number of records accessed in a Recordset object, or the total number of records in a table-type Recordset or TableDef object.

Return Values

The return value is a Long data type.

Use the RecordCount property to find out how many records in a Recordset or TableDef object have been accessed. The RecordCount property doesn't indicate how many records are contained in a dynaset-, snapshot-, or forward-only-type Recordset object until all records have been accessed. Once the last record has been accessed, the RecordCount property indicates the total number of undeleted records in the Recordset or TableDef object. To force the last record to be accessed, use the MoveLast method on the Recordset object. You can also use an SQL Count function to determine the approximate number of records your query will return.

Note Using the MoveLast method to populate a newly opened Recordset negatively impacts performance. Unless it is necessary to have an accurate RecordCount as soon as you open a Recordset, it's better to wait until you populate the Recordset with other portions of code before checking the RecordCount property.

· Setting the AbsolutePosition property to a value greater than zero on a newly opened but unpopulated Recordset object causes a trappable error. Populate the Recordset object first with the MoveLast method.
· The AbsolutePosition property isn't available on forward-only–type Recordset objects, or on Recordset objects opened from pass-through queries against Microsoft Jet-connected ODBC databases.

Bookmark and Return Bookmark

Sets or returns a bookmark that uniquely identifies the current record in a Recordset object.

Settings and Return Values

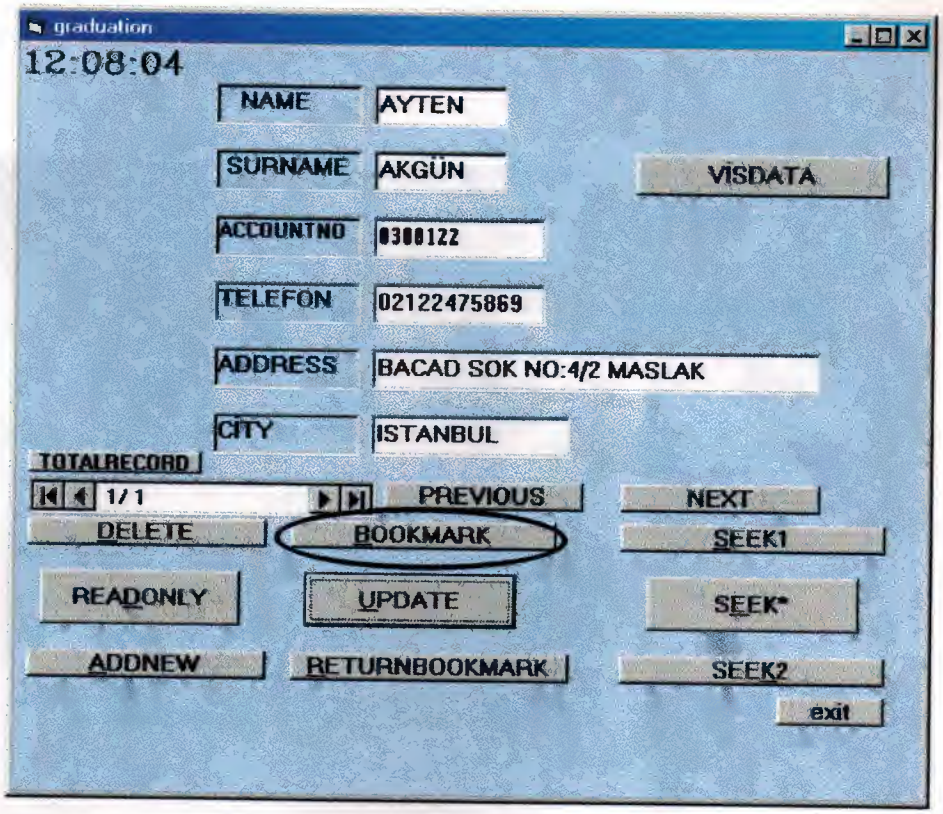
The setting or return value is a string expression or variant expression that evaluates to a valid bookmark. The data type is a Variant array of Byte data.

For a Recordset object based entirely on Microsoft Jet tables, the value of the Bookmarkable property is True, and you can use the Bookmark property with that Recordset. Other database products may not support bookmarks, however. For example, you can't use bookmarks in any Recordset object based on a linked Paradox table that has no primary key.
When you create or open a Recordset object, each of its records already has a unique bookmark. You can save the bookmark for the current record by assigning the value of the Bookmark property to a variable. To quickly return to that record at any time after moving to a different record, set the Recordset object's Bookmark property to the value of that variable.

There is no limit to the number of bookmarks you can establish. To create a bookmark for a *record other than the current record, move to the desired record and assign the value of the Bookmark property to a String variable that identifies the record.*
To make sure the Recordset object supports bookmarks, check the value of its Bookmarkable property before you use the Bookmark property. If the Bookmarkable property is False, the Recordset object doesn't support bookmarks, and using the Bookmark property results in a trappable error.

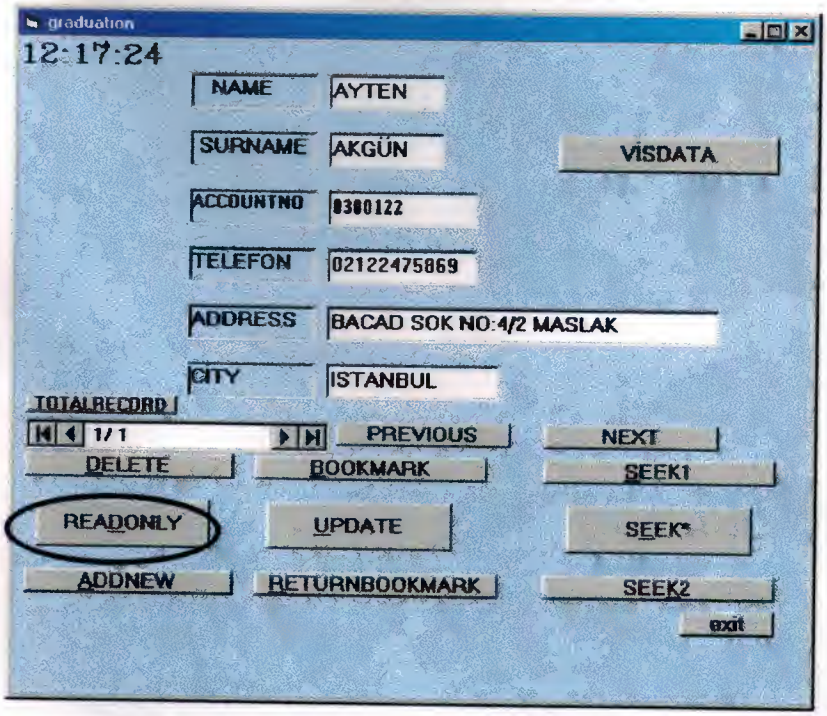
If you use the Clone method to create a copy of a Recordset object, the Bookmark property settings for the original and the duplicate Recordset objects are identical and can be used interchangeably. However, you can't use bookmarks from different Recordset objects interchangeably, even if they were created by using the same object or the same SQL statement.
If you set the Bookmark property to a value that represents a deleted record, a trappable error occurs.

The value of the Bookmark property isn't the same as a record number.

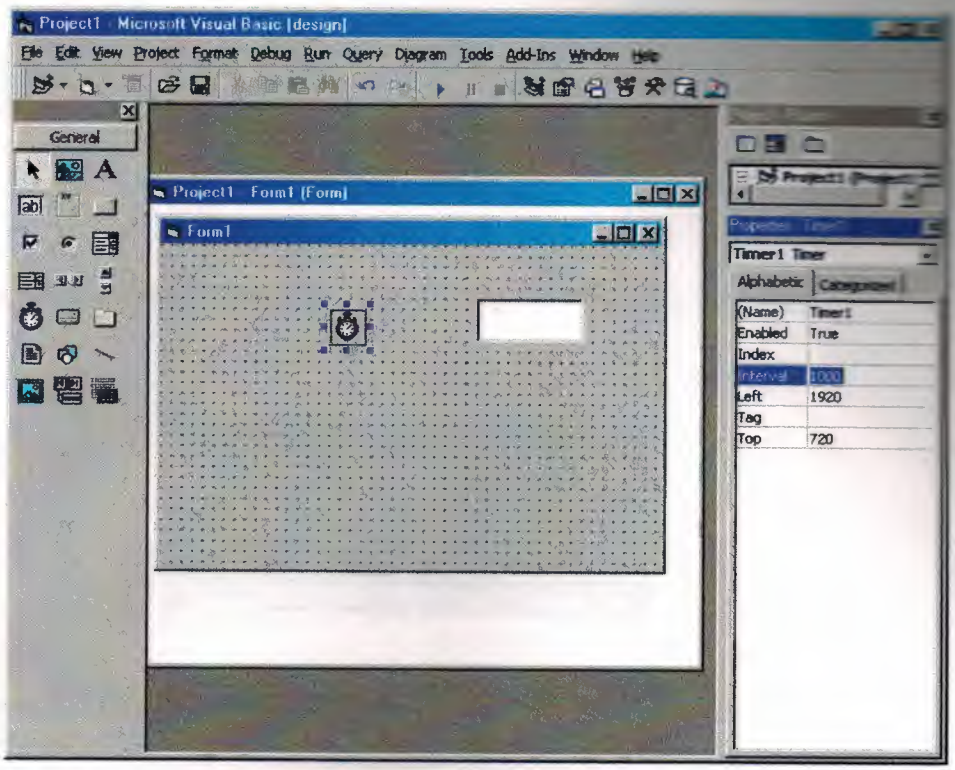


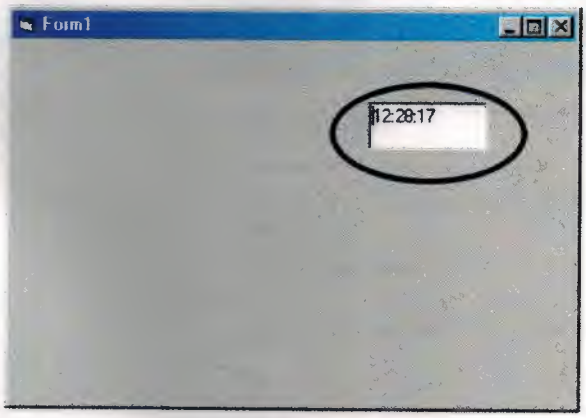
Readonly Method

A Boolean value that is True if the connection is to be opened for read-only access and False if the connection is to be opened for read/write access.



Timer





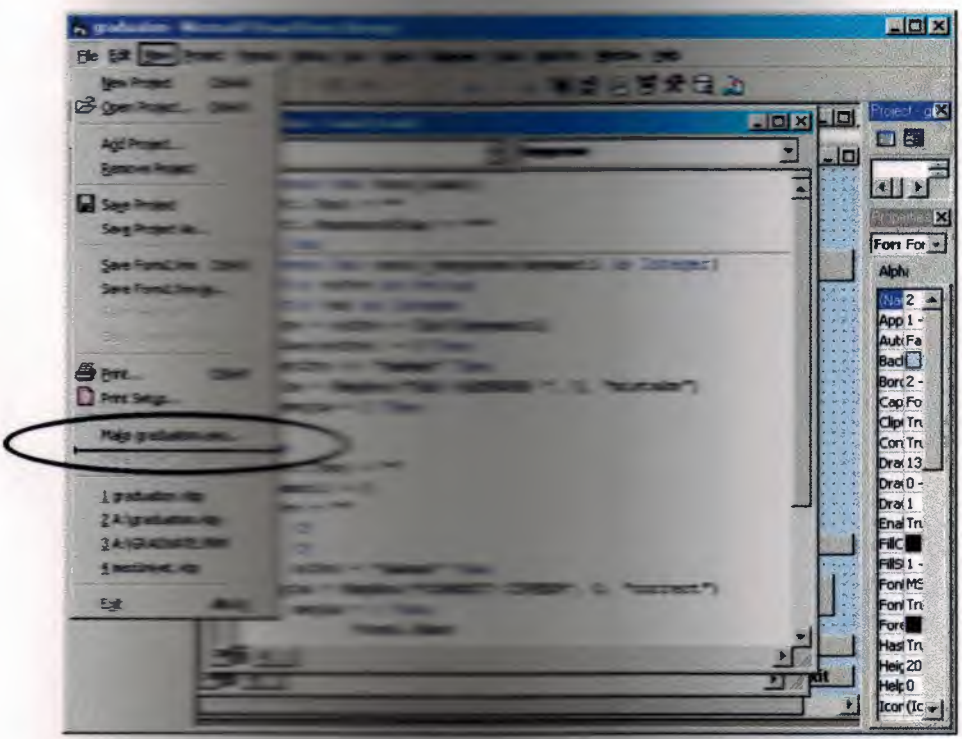
Password

For security I wrote password program.

If you write correct password then coming the following msgbox

If you write uncorrect password then coming the following msgbox

Execute the Graduation Program



Exclusive

If you don't want to use your program in multiuser system Exclusive properties should be **true**