MS-Access Database Programming

MS-Access uses **Visual-Basic for Applications** (VBA) as its native programming language. The use of VBA allows the database developer to create sophisticated applications pulling their data from either MS-Access DB and/or other external data sources.

VBA is typically used in conjunction with the **DAO** and **ADODB** technology for applications entirely enclosed in an Access container. Microsoft developers are encouraged to use the newer **ADO.NET** technology and more powerful programming languages (such as Visual-Basic, C#, C++,...) to work in all other cases.

**A Quick Introduction to Classic ADO Technology**

The model consists of three fundamental classes: Command, Connection, and RecordSet

Reference:
You use Recordset objects to manipulate data in a database at the record level. All Recordset objects are constructed using records (rows) and fields (columns). There are five types of Recordset objects:

- **Table-type Recordset**— representation in code of a base table that you can use to add, change, or delete records from a single database table (Microsoft Access workspaces only).

- **Dynaset-type Recordset**— the result of a query that can have updatable records. A dynaset-type Recordset object is a dynamic set of records that you can use to add, change, or delete records from an underlying database table or tables. A dynaset-type Recordset object can contain fields from one or more tables in a database. This type corresponds to an ODBC keyset cursor.

- **Snapshot-type Recordset**— a static copy of a set of records that you can use to find data or generate reports. A snapshot-type Recordset object can contain fields from one or more tables in a database but can't be updated. This type corresponds to an ODBC static cursor.

- **Forward-only-type Recordset**— identical to a snapshot except that no cursor is provided. You can only scroll forward through records. This improves performance in situations where you only need to make a single pass through a result set. This type corresponds to an ODBC forward-only cursor.

- **Dynamic-type Recordset**— a query result set from one or more base tables in which you can add, change, or delete records from a row-returning query. Further, records other users add, delete, or edit in the base tables also appear in your Recordset. This type corresponds to an ODBC dynamic cursor (ODBCDirect workspaces only).

You can choose the type of Recordset object you want to create using the type argument of the OpenRecordset method.

- In a Microsoft Access workspace, if you don't specify a type, DAO attempts to create the type of Recordset with the most functionality available, starting with table. If this type isn't available, DAO attempts a dynaset, then a snapshot, and finally a forward-only type Recordset object.

- A new Recordset object is automatically added to the Recordsets collection when you open the object, and is automatically removed when you close it.

You can create as many Recordset object variables as needed. Different Recordset objects can access the same tables, queries, and fields without conflicting.

Dynaset-, snapshot-, and forward–only–type Recordset objects are stored in local memory. If there isn't enough space in local memory to store the data, the Microsoft Access database engine saves the additional data to TEMP disk space. If this space is exhausted, a trappable error occurs.

The default collection of a Recordset object is the Fields collection, and the default property of a Field object is the Value property. Use these defaults to simplify your code.

When you create a Recordset object, the current record is positioned to the first record if there are any records. If there are no records, the RecordCount property setting is 0, and the EOF and BOF property settings are True.

You can use the MoveNext, MovePrevious, MoveFirst, and MoveLast methods to reposition the current record. Forward-only–type Recordset objects support only the MoveNext method. When using the Move methods to visit each record (or “walk” through the Recordset), you can use the BOF and EOF properties to check for the beginning or end of the Recordset object.

With dynaset- and snapshot-type Recordset objects in a Microsoft Access workspace, you can also use the Find methods, such as FindFirst, to locate a specific record based on criteria. If the record isn't found, the NoMatch property is set to True. For table-type Recordset objects, you can scan records using the Seek method.

The Type property indicates the type of Recordset object created, and the Updatable property indicates whether you can change the object’s records.

Information about the structure of a base table, such as the names and data types of each Field object and any Index objects, is stored in a TableDef object.

To refer to a Recordset object in a collection by its ordinal number or by its Name property setting, use any of the following syntax forms:
**ADO Command Object**

Use a Command object to query a database and return records in a Recordset object, to execute a bulk operation, or to manipulate the structure of a database. Depending on the functionality of the provider, some Command collections, methods, or properties may generate an error when they are referenced.

With the collections, methods, and properties of a Command object, you can do the following:

- Define the executable text of the command (for example, an SQL statement) with the CommandText property. Alternatively, for command or query structures other than simple strings (for example, XML template queries), define the command with the CommandStream property.
- Optionally, indicate the command dialect used in the CommandText or CommandStream with the Dialect property.
- Define parameterized queries or stored-procedure arguments with Parameter objects and the Parameters collection.
- Indicate whether parameter names should be passed to the provider with the NamedParameters property.
- Execute a command and return a Recordset object if appropriate with the Execute method.
- Specify the type of command with the CommandType property prior to execution to optimize performance.
- Control whether the provider saves a prepared (or compiled) version of the command prior to execution with the Prepared property.
- Set the number of seconds that a provider will wait for a command to execute with the CommandTimeout property.
- Associate an open connection with a Command object by setting its ActiveConnection property.
- Set the Name property to identify the Command object as a method on the associated Connection object.
- Pass a Command object to the Source property of a Recordset to obtain data.
- To create a Command object independently of a previously defined Connection object, set its ActiveConnection property to a valid connection string. ADO still creates a Connection object, but it does not assign that object to an object variable. However, if you are associating multiple Command objects with the same connection, you should explicitly create and open a Connection object; this assigns the Connection object to an object variable. Make sure that the Connection object was opened successfully before you assign it to the ActiveConnection property of the Command object, because assigning a closed Connection object causes an error. If you do not set the ActiveConnection property of the Command object to this object variable, ADO creates a new Connection object for each Command object, even if you use the same connection string.
- To execute a Command, call it by its Name property on the associated Connection object. The Command must have its ActiveConnection property set to the Connection object. If the Command has parameters, pass their values as arguments to the method.

**ADO Connection Object**

A Connection object represents a unique session with a data source. In a client/server database system, it may be equivalent to an actual network connection to the server. Depending on the functionality supported by the provider, some collections, methods, or properties of a Connection object may not be available.

With the collections, methods, and properties of a Connection object, you can do the following:

- Configure the connection before opening it with the ConnectionString, ConnectionTimeout, and Mode properties. ConnectionString is the default property of the Connection object.
- Set the CursorLocation property to client to invoke the Microsoft Cursor Service for OLE DB, which supports batch updates.
- Set the default database for the connection with the DefaultDatabase property.
- Set the level of isolation for the transactions opened on the connection with the IsolationLevel property.
- Specify an OLE DB provider with the Provider property.
- Establish, and later break, the physical connection to the data source with the Open and Close methods.
- Execute a command on the connection with the Execute method and configure the execution with the CommandTimeout property.
- Manage transactions on the open connection, including nested transactions if the provider supports them, with the BeginTrans, CommitTrans, and RollbackTrans methods and the Attributes property.
- Examine errors returned from the data source with the Errors collection.
- Read the version from the ADO implementation used with the Version property.
- Obtain schema information about your database with the OpenSchema method.

**Allowing MS-Access to Run VBA code**

MS-Access 2007 and 2010 requires you to explicitly permit the use of VB macros. To allow them do this:

1. Click on the symbol > Access Options > Trust Center > Trust Center Settings
2. Check option “Enable all macros”
INCLUDING ADODB IN YOUR ACCESS PROJECT

Access 2007 (2010) doesn’t include the ADO reference as a default; you need to add it to your application as follows:

Home > Database Tools > Visual Basic > Tools > Reference
Check entry “Microsoft ActiveX Data Objects 6.0 Library” (use 6.0 or newer)
Click OK button
## ADO Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddNew</td>
<td>Creates a new record for an updatable Recordset object.</td>
</tr>
<tr>
<td>Append</td>
<td>Appends an object to a collection. If the collection is Fields, a new Field object may be created before it is appended to the collection.</td>
</tr>
<tr>
<td>AppendChunk</td>
<td>Appends data to a large text or binary data Field or Parameter object.</td>
</tr>
</tbody>
</table>
| BeginTrans, CommitTrans, and RollbackTrans | Manages transaction processing within a Connection object as follows:  
<p>|                               | BeginTrans – Begins a new transaction.                                      |
|                               | CommitTrans – Saves any changes and ends the current transaction. It may also start a new transaction. |
|                               | RollbackTrans – Cancels any changes and ends the current transaction. It may also start a new transaction. |
| Cancel                        | Cancels execution of a pending, asynchronous Execute or Open method call.   |
| Cancel (RDS)                  | Cancels the currently running asynchronous execution or fetch.              |
| CancelBatch                   | Cancels a pending batch update.                                             |
| CancelUpdate (ADO)            | Cancels any changes made to the current record or to a new record prior to calling the Update method. |
| CancelUpdate (RDS)            | Discards all the pending changes associated with the specified Recordset object, thus restoring the values since the last Refresh method call. |
| Clear                         | Removes all of the objects in a collection.                                |
| Clone                         | Creates a duplicate Recordset object from an existing Recordset object.     |
|                               | Optionally, specifies that the clone be read-only.                         |
| Close                         | Closes an open object and any dependent objects.                           |
| CompareBookmarks              | Compares two bookmarks and returns an indication of their relative values.  |
| ConvertToString               | Converts a Recordset to a MIME string that represents the recordset data.  |
| CopyRecord                    |Copies a file or directory, and its contents, to another location.          |
| CopyTo                        | Copies the specified number of characters or bytes in the Stream to another Stream object. |
| CreateObject (RDS)            | Creates the proxy for the target business object and returns a pointer to it.|
| CreateParameter               | Creates a new Parameter object with the specified properties.               |
| CreateRecordset (RDS)         | Creates an empty, disconnected Recordset.                                  |
| Delete (ADO Parameters Collection) | Deletes an object from the Parameters collection.                        |
| Delete (ADO Fields Collection) | Deletes an object from the Fields collection.                             |
| Delete (ADO Recordset)        | Deletes the current record or a group of records.                          |
| DeleteRecord                  | Deletes a file or directory, and all its subdirectories.                   |
| Execute (ADO Command)         | Executes the query, SQL statement, or stored procedure specified in the CommandText property. |
| Execute (ADO Connection)      | Executes the specified query, SQL statement, stored procedure, or provider-specific text. |
| Find                          | Searches a Recordset for the record that satisfies the specified criteria.  |
| Flush                         | Forces the contents of the Stream remaining in the ADO buffer to the underlying object with which the Stream is associated. |
| GetChildren                   | Returns a Recordset whose rows represent the files and subdirectories in the directory represented by this Record. |
| GetChunk                      | Returns all or a portion of the contents of a large text or binary data Field object. |
| GetRows                       | Retrieves multiple records of a Recordset object into an array.            |
| GetString                     | Returns the Recordset as a string.                                         |
| LoadFromFile                  | Loads the contents of an existing file into a Stream.                      |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>Moves the position of the current record in a <strong>Recordset</strong> object.</td>
</tr>
<tr>
<td>MoveFirst, MoveLast, MoveNext, MovePrevious (ADO)</td>
<td>Moves to the first, last, next, or previous record in a specified <strong>Recordset</strong> object and makes that record the current record.</td>
</tr>
<tr>
<td>MoveFirst, MoveLast, MoveNext, MovePrevious (RDS)</td>
<td>Moves to the first, last, next, or previous record in a displayed <strong>Recordset</strong>.</td>
</tr>
<tr>
<td>MoveRecord</td>
<td>Moves a file, or a directory and its contents, to another location.</td>
</tr>
<tr>
<td>NextRecordset</td>
<td>Clears the current <strong>Recordset</strong> object and returns the next <strong>Recordset</strong> by advancing through a series of commands.</td>
</tr>
<tr>
<td>Open (ADO Connection)</td>
<td>Opens a connection to a data source.</td>
</tr>
<tr>
<td>Open (ADO Record)</td>
<td>Opens an existing <strong>Record</strong> object, or creates a new file or directory.</td>
</tr>
<tr>
<td>Open (ADO Recordset)</td>
<td>Opens a cursor.</td>
</tr>
<tr>
<td>Open (ADO Stream)</td>
<td>Opens a <strong>Stream</strong> object to manipulate streams of binary or text data.</td>
</tr>
<tr>
<td>OpenSchema</td>
<td>Obtains database schema information from the provider.</td>
</tr>
<tr>
<td>Query (RDS)</td>
<td>Uses a valid SQL query string to return a <strong>Recordset</strong>.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified number of bytes from a <strong>Stream</strong> object.</td>
</tr>
<tr>
<td>ReadText</td>
<td>Reads a specified number of characters from a text <strong>Stream</strong> object.</td>
</tr>
<tr>
<td>Refresh (ADO)</td>
<td>Updates the objects in a collection to reflect objects available from and specific to the provider.</td>
</tr>
<tr>
<td>Refresh (RDS)</td>
<td>Requeries the ODBC data source specified in the <strong>Connect</strong> property and updates the query results.</td>
</tr>
<tr>
<td>Requery</td>
<td>Updates the data in a <strong>Recordset</strong> object by re-executing the query on which the object is based.</td>
</tr>
<tr>
<td>Reset (RDS)</td>
<td>Executes the sort or filter on a client-side <strong>Recordset</strong> based on the specified sort and filter properties.</td>
</tr>
<tr>
<td>Resync</td>
<td>Refreshes the data in the current <strong>Recordset</strong> object from the underlying database.</td>
</tr>
<tr>
<td>Save (ADO Recordset)</td>
<td>Saves (persists) the <strong>Recordset</strong> in a file or <strong>Stream</strong> object.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>Saves the binary contents of a <strong>Stream</strong> to a file.</td>
</tr>
<tr>
<td>Seek</td>
<td>Searches the index of a <strong>Recordset</strong> to quickly locate the row that matches the specified values, and changes the current row position to that row.</td>
</tr>
<tr>
<td>SetEOS</td>
<td>Sets the position that is the end of the stream.</td>
</tr>
<tr>
<td>SkipLine</td>
<td>Skips one entire line when reading a text <strong>Stream</strong>.</td>
</tr>
<tr>
<td>SubmitChanges (RDS)</td>
<td>Submits pending changes of the locally cached updatable <strong>Recordset</strong> to the ODBC data source specified in the <strong>Connect</strong> property.</td>
</tr>
<tr>
<td>Supports</td>
<td>Determines whether a specified <strong>Recordset</strong> object supports a particular type of functionality.</td>
</tr>
<tr>
<td>Update</td>
<td>Saves any changes you make to the current record of a <strong>Recordset</strong> object.</td>
</tr>
<tr>
<td>UpdateBatch</td>
<td>Writes all pending batch updates to disk.</td>
</tr>
<tr>
<td>Write</td>
<td>Writes binary data to a <strong>Stream</strong> object.</td>
</tr>
<tr>
<td>WriteText</td>
<td>Writes a specified text string to a <strong>Stream</strong> object.</td>
</tr>
</tbody>
</table>
EXAMPLES :  MS-Access VBA Code Fragments

Ex1. Creating a Connection Object using CurrentProject Object

Sub CreateConnection1()
  'PROBLEMS: must split front&back end
  Dim cnnCompany As ADODB.Connection
  Set cnnCompany = New ADODB.Connection
  cnnCompany.Open CurrentProject.Connection
  'Your code goes here ...
  cnnCompany.Close
  Set cnnCompany = Nothing
End Sub

Ex2. Creating a Connection Object using a Connection String

Sub CreateConnection2()
  Dim myConnectionString
  Dim cnnCompany As ADODB.Connection
  Set cnnCompany = New ADODB.Connection

  '-- Connection string reading a NEW Access container (.accdb format) ------------------------
  myConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" & _
                     "Data Source=C:\Users\1002125\Documents\CSU\CIS-430\CompanyDB\Company2007.accdb"

  '-- Connection string reading an OLD Access container (.mdb format) ------------------------
  '  myConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" & _
  '    "Data Source=C:\Users\1002125\Documents\CSU\CIS-430\CompanyDB\CompanyXP.mdb;"

  cnnCompany.ConnectionString = myConnectionString
  cnnCompany.Open

  cnnCompany.BeginTrans
  'Your code goes here ...
  Dim rst As Recordset
  Set rst = New Recordset
  rst.Open "select * from Project", cnnCompany
  While Not (rst.EOF)
    MsgBox rst("Pname")
    rst.MoveNext
  Wend

  MsgBox "... CreateConnection2 ... working "
  cnnCompany.CommitTrans 'or cnnCompany.RollbackTrans

  MsgBox "Done CreateConnection2"
  cnnCompany.Close
  Set cnnCompany = Nothing
End Sub
### Ex3. Creating a Recordset using the CurrentProject Object

```
Sub CreateRecordset1()
    Dim rst As ADODB.Recordset
    Set rstCompany = New ADODB.Recordset
    rstCompany.Open "Select * From Employee", CurrentProject.Connection
    MsgBox rstCompany.GetString
    rstCompany.Close
    Set rstCompany = Nothing
End Sub
```

### Ex4. Creating a Recordset using the ActiveConnection Property

```
Sub CreateRecordset2()
    Dim rstCompany As ADODB.Recordset
    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.Open "Select * From Employee"
    MsgBox rstCompany.GetString
    rstCompany.Close
    Set rstCompany = Nothing
End Sub
```

### Ex5. Creating Multiple Recordsets using a Connection Object

```
Sub CreateRecordset3()
    Dim rstCompany1 As ADODB.Recordset
    Dim rstCompany2 As ADODB.Recordset

    Set rstCompany1 = New ADODB.Recordset
    Set rstCompany2 = New ADODB.Recordset

    rstCompany1.ActiveConnection = CurrentProject.Connection
    rstCompany2.ActiveConnection = CurrentProject.Connection

    rstCompany1.Open "Select * From Employee"
    rstCompany2.Open "Select * From Project"

    MsgBox rstCompany1.GetString
    MsgBox rstCompany2.GetString

    rstCompany1.Close
    rstCompany2.Close

    Set rstCompany1 = Nothing
    Set rstCompany2 = Nothing
End Sub
```
**Ex6. Creating Multiple Recordsets using a Connection Object**

Sub RecordsetMovements()
   "To see Debug.Print click on tool bar View > Immediate Window (Ctrl G)

   Dim rstCompany As ADODB.Recordset

   Set rstCompany = New ADODB.Recordset
   rstCompany.ActiveConnection = CurrentProject.Connection
   rstCompany.CursorType = adOpenStatic
   rstCompany.Open "Select * from Projects"

   Debug.Print rstCompany("PName")
   rstCompany.MoveNext
   Debug.Print rstCompany("PName")
   rstCompany.MoveLast
   Debug.Print rstCompany("PName")
   rstCompany.MovePrevious
   Debug.Print rstCompany("PName")
   rstCompany.MoveFirst
   Debug.Print rstCompany("PName")
   rstCompany.Close
   Set rstCompany = Nothing
End Sub

**Ex7. Traversing a Static Recordset Sequentially**

Sub DetermineLimits()
   Dim rstCompany As ADODB.Recordset

   Set rstCompany = New ADODB.Recordset
   rstCompany.ActiveConnection = CurrentProject.Connection
   rstCompany.CursorType = adOpenStatic

   rstCompany.Open "Select * from Employee"

   Do Until rstCompany.EOF
       MsgBox rstCompany("Lname")
       rstCompany.MoveNext
   Loop

   rstCompany.Close
   Set rstCompany = Nothing
End Sub
## Ex7. Counting how many Records are held in a Recordset

Sub HowManyRecords()
    Dim rstCompany As ADODB.Recordset
    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorType = adOpenStatic
    rstCompany.Open "Select * from Employee where ssn = 0 "
    MsgBox "absolute position: " & rstCompany.AbsolutePosition 'returns -1 when empty
    MsgBox "record count: " & rstCompany.RecordCount 'returns Recordcount
    rstCompany.Close
    Set rstCompany = Nothing
End Sub

## Ex8. Sorting-filtering-finding records on an existing Recordset

Sub SortFilterFindRecordset()
    Dim rstCompany As ADODB.Recordset
    Dim intCounter As Integer
    Dim thePerson, strCondition
    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorLocation = adUseClient
    rstCompany.Open "Select * from Employee"
    'without filtering - Consider all employees
    rstCompany.Sort = "Salary ASC"
    MsgBox "Lowest Salary: " & rstCompany("Salary")
    rstCompany.Sort = "Salary DESC"
    MsgBox "Highest Salary: " & rstCompany("Salary")
    'filtering records - Consider only female employees
    rstCompany.Filter = "sex = 'F' "
    rstCompany.Sort = "Salary ASC"
    MsgBox "Female Lowest Salary: " & rstCompany("Salary")
    rstCompany.Sort = "Salary DESC"
    MsgBox "Female Highest Salary: " & rstCompany("Salary")
    'Find Jennifer's record
    thePerson = "Jennifer"
    strCondition = "[Lname] = 'Jennifer' & thePerson & 'lict""
    rstCompany.Find strCondition
    If rstCompany.EOF Then
        MsgBox thePerson & " Not Found"
    Else
        MsgBox thePerson & " Found"
    End If
    rstCompany.Close
    Set rstCompany = Nothing
End Sub
**Ex9. Recordsets: Using BookMarks**

Sub UseBookMark()
    Dim rstCompany As ADODB.Recordset
    Dim strConditionAs As String
    Dim marker1 As Variant
    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorType = adOpenStatic
    rstCompany.Open "Select * from Employee"
    marker1 = rstCompany.Bookmark
    Do Until rstCompany.EOF
        MsgBox rstCompany("Lname")
        rstCompany.MoveNext
    Loop
    rstCompany.Bookmark = marker1
    MsgBox "using bookmark: " & rstCompany("Lname")
    rstCompany.Close
    Set rstCompany = Nothing
End Sub

**Ex10. Passing Parameters to an Executable Command Object**

Sub CallParameterizedCommand()
    ‘List ladies working for dept 4
    Call ParameterizedCommand(4, "F")
End Sub

Sub ParameterizedCommand(theDept As Integer, theGender As String)
    Dim cmd As ADODB.Command
    Dim rstCompany As ADODB.Recordset
    Dim myLine
    Set cmd = New ADODB.Command
    cmd.ActiveConnection = CurrentProject.Connection
    cmd.CommandText = "Select * from Employee " & _
    " Where Dno = " & theDept & _
    " and Sex = " & theGender & ""
    cmd.CommandType = adCmdText
    Set rstCompany = cmd.Execute(Parameters:=Array(theDept, theGender))
myLine = ""
Do Until rstCompany.EOF
    myLine = myLine & rstCompany("Dno") & " " & rstCompany("Fname") & vbCrLf
    rstCompany.MoveNext
Loop
MsgBox myLine
rstCompany.Close
Set rstCompany = Nothing
Set cmd = Nothing
End Sub

Ex11. Updating a Recordset

Sub raiseLadiesSalary()
    Dim rstCompany As ADODB.Recordset
    Set rstCompany = New ADODB.Recordset
    Dim strSQL As String
    Dim intUpdated As Integer
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorType = adOpenDynamic
    rstCompany.LockType = adLockOptimistic
    rstCompany.Open "Select * from Employee where Sex = 'F' "
    'increase by $1 the salary of female employees
    While Not rstCompany.EOF
        intUpdated = intUpdated + 1
        rstCompany("Salary") = rstCompany("Salary") + 1
        rstCompany.Update
        rstCompany.MoveNext
    Wend
    MsgBox intUpdated & " Records Updated"
    rstCompany.Close
    Set rstCompany = Nothing
End Sub

Ex12. Deleting Rows from a Recordset

Sub DeleteFemaleEmployees()
    Dim rstCompany As ADODB.Recordset
    Dim intCounter As Integer
    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorType = adOpenDynamic
    rstCompany.LockType = adLockOptimistic
rstCompany.Open "Select * from Employee"

intCounter = 0
While Not rstCompany.EOF
    If rstCompany("Sex") = "F" Then
        rstCompany.Delete
        intCounter = intCounter + 1
    End If
    rstCompany.MoveNext
Wend

MsgBox intCounter & " Employee(s) Deleted"

rstCompany.Close
Set rstCompany = Nothing
End Sub

Ex13. Inserting a new row into a Recordset

Sub setNewWorkLoad()
    Call newWorkLoad(123456789, 99, 40)
End Sub

Sub newWorkLoad(theSSN As Long, theProject As Integer, theHours As Integer)
    Dim rstCompany As ADODB.Recordset
    Dim intCounter As Integer

    Set rstCompany = New ADODB.Recordset
    rstCompany.ActiveConnection = CurrentProject.Connection
    rstCompany.CursorType = adOpenStatic
    rstCompany.LockType = adLockOptimistic

    rstCompany.Open "Select * from Works_On"
    MsgBox rstCompany.RecordCount
    rstCompany.AddNew
    rstCompany("ESSN") = theSSN
    rstCompany("Pno") = theProject
    rstCompany("Hours") = theHours
    rstCompany.Update
    MsgBox rstCompany.RecordCount
    rstCompany.Close

    Set rstCompany = Nothing
End Sub
**Ex14. Bulk-Update of a Recordset**

Sub bulkUpdate()

    Dim mySQL, countRecAffected

    mySQL = "Update Employee set Salary = Salary - 1 where sex = 'F'"

    CurrentProject.Connection.Execute mySQL, countRecAffected

    MsgBox countRecAffected & " records changed"

End Sub

**Ex18. Running a Stored SQL-Query**

Sub RunUpdateQuery()

    CurrentProject.Connection.Execute "qryRaiseSalary"

End Sub

Assume the following query has been added to the Company database Query collection as: qryRaiseSalary:

    update EMPLOYEE set salary = salary * 1.10 where sex = 'F'

**Ex19. Exploring the Database Structure Using ADOX**

Sub exploringADOX()

    '--- To include ADOX do as follows
    '--- Tools | References | Choose "Microsoft ADO Ext. 6.0 for DDL and Security"

    Dim catalog As ADOX.catalog
    Set catalog = New ADOX.catalog
    Dim msg as String

    catalog.ActiveConnection = CurrentProject.Connection
    For Each t In catalog.Tables
        msg = t.Name & vbTab & t.Type & vbCrLf
        For Each f In t.Columns
            msg = msg & f.Type & vbTab & f.Name & vbCrLf
        Next
        MsgBox msg
    Next

End Sub
**Various Connection Strings**

**Microsoft Access 2000, XP**
Reaching the Company2007 database at local drive c.

```
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\VMatos\Documents\CSU\CIS-430\CompanyDB\Company2007.accdb
```

**TODO:** Check following connections—add MySql

**Microsoft SQL Server**
Reaching the NorthWind database placed on the (corporate) MSSQL server called SQLSRV01 from a networked workstation named MYMACHINE.

```
workstation id=MYMACHINE;packet size=4096;integrated security=SSPI;
data source=SQLSRV01;persist security info=True;initial catalog=Northwind
```

**Oracle Server**
User scott reaching a corporate Oracle server called VMORA9 from a networked workstation (the user’s password tiger is required at connection time)

Using Oracle driver:
```
Data Source=sancho.csuohio.edu;Persist Security Info=True;User ID=CSUPERSON;Password=***********;Unicode=True
```

Using Microsoft driver for Oracle:
```
Provider=MSDAORA;User ID=scott;Data Source=VMORA9;
```

**Sybase Adaptative SQL Anywhere 9.0**
Use name: dba (password sql) reaches the corporate server ASADemo from a networked workstation.
```
Provider="ASAProv.90";User ID=dba;Data Source=asademo;
```

MySQL .....
### Using MS Access

#### Demo0 Opening-Closing Forms

<table>
<thead>
<tr>
<th>Code on the zYellow form (made by the control wizard)</th>
<th>Code on the zBlue form (made by the control wizard &amp; by-hand)</th>
<th>Code on the zRed form (made by-hand)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Sub cmdOpenBlue_Click()</td>
<td>Private Sub cmdOpenRed_Click()</td>
<td>Private Sub cmdOpenBlue_Click()</td>
</tr>
<tr>
<td>On Error GoTo Err_cmdOpenBlue_Click</td>
<td>On Error GoTo Err_cmdOpenRed_Click</td>
<td>DoCmd.OpenForm “zBlue”</td>
</tr>
<tr>
<td>DoCmd.OpenForm “zBlue”</td>
<td>Dim stDocName As String</td>
<td>End Sub</td>
</tr>
<tr>
<td>Exit_cmdOpenBlue_Click:</td>
<td>stDocName = “zRed”</td>
<td>Private Sub cmdOpenYellow_Click()</td>
</tr>
<tr>
<td>Exit Sub</td>
<td>DoCmd.OpenForm stDocName</td>
<td>DoCmd.OpenForm “zYellow”</td>
</tr>
<tr>
<td>Err_cmdOpenBlue_Click:</td>
<td>Exit_cmdOpenRed_Click:</td>
<td>End Sub</td>
</tr>
<tr>
<td>MsgBox Err.Description</td>
<td>Exit Sub</td>
<td>Private Sub cmdOpenYellow_Click()</td>
</tr>
<tr>
<td>Resume Exit_cmdOpenBlue_Click</td>
<td>Err_cmdOpenRed_Click:</td>
<td>DoCmd.OpenForm “zYellow”</td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
<td>End Sub</td>
</tr>
<tr>
<td>Private Sub cmdExit_Click()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoCmd.Close</td>
<td></td>
<td>Private Sub cmdOpenYellow_Click()</td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
<td>DoCmd.OpenForm “zYellow”</td>
</tr>
</tbody>
</table>

#### NOTES:

1. Make sure the User Control Wizard is NOT active when you place the buttons on the screen.

2. To add VB code to a button (on the Design-View) Right-Click on the button > Build Events
Using MS ACCESS  Demo1:  A login form for the COMPANY database

Option Compare Database

Private Sub cmdClose_Click()
Dim theAnswer
    theAnswer = MsgBox("Do you really want to EXIT?", vbQuestion + vbYesNoCancel, "Terminating")
    If theAnswer = vbYes Then
        DoCmd.Quit
    End If
End Sub

Private Sub cmdLogin_Click()
    Dim storedPwd
    Dim theCondition
    'consult the stored PASSWORDTABLE for the current user
    theCondition = "[userName]= " & txtUser & ""
    storedPwd = DLookup("Password", "passwordTable", theCondition)
    'See if the given password matches the stored valued
    If (Len(storedPwd & "") = 0) Or _
        (UCase(storedPwd) <> UCase(txtPassword)) Then
        MsgBox "Invalid Password", vbCritical, "Login Error"
        txtPassword.SetFocus
        txtPassword = ""
        Exit Sub
    End If
    'if everything is fine show the main switchboard
    DoCmd.OpenForm "frmMySwitchBoard"
End Sub

Equivalently:
    txtPassword.value = ""
    and you do not need to call setFocus first
Using MS Access Demo2 Programming Textboxes, Radio and Check Buttons

Private Sub cmdCancel_Click()
    'clear the name, phone, and summary textboxes
    Me.txtName = ""
    Me.txtPhone = ""
    Me.txtSummary = ""
    'clear the cheese selection
    Me.fraCheese.Value = 1
    'clear the topping selections
    Me.chkAnchovies = 0
    Me.chkChicken = 0
    Me.chkMushrooms = 0
    Me.chkOnios = 0
    Me.chkPepperoni = 0
End Sub

Private Sub cmdClose_Click()
    DoCmd.Close
End Sub

Private Sub cmdOK_Click()
    txtSummary = txtName & " - " & txtPhone & " - " &
    'cheese selection
    If fraCheese.Value = 1 Then
        txtSummary = txtSummary & "Single cheese"
    ElseIf fraCheese.Value = 2 Then
        txtSummary = txtSummary & "Double cheese"
    ElseIf fraCheese.Value = 3 Then
        txtSummary = txtSummary & "one side cheese"
    ElseIf fraCheese.Value = 4 Then
        txtSummary = txtSummary & "NO cheese"
    End If
    'topping selection
    If chkPepperoni.Value = vbTrue Then
        txtSummary = txtSummary & " - " & "Pepperoni"
    End If
    If chkOnios.Value = vbTrue Then
        txtSummary = txtSummary & " - " & "Onios"
    End If
    If chkMushrooms.Value = vbTrue Then
        txtSummary = txtSummary & " - " & "Mushroom"
    End If
    If chkChicken.Value = vbTrue Then
        txtSummary = txtSummary & " - " & "Chicken"
    End If
    If chkAnchovies.Value = vbTrue Then
        txtSummary = txtSummary & " - " & "Anchovies"
    End If
End Sub
1. Create the "EMPLOYEE" form using the "form Wizard" (Select the EMPLOYEE table on the Object Explorer panel > Create > Form

2. On 'Design View' mode add a command button. Make sure the Control Wizard is “ON”. Choose “Record Operation | Find Record”. Test the functionality of the button by positioning the cursor on different textboxes and clicking the “Find” button.

3. Add buttons (Control Wizard OFF) to locate employees by LAST NAME and SSN. Add to each button the following VB code

```vbnet
Private Sub cmdFindByLName_Click()
    On Error GoTo Exit_Routine
    'forces search to operated on the employee's LastName.
    LNAME.SetFocus
    DoCmd.RunCommand acCmdFind
    Exit_Routine
    Exit Sub
End Sub

Private Sub cmdFindBySSN_Click()
    On Error GoTo Exit_Routine
    'forces search to operated on the employee's SSN.
    SSN.SetFocus
    DoCmd.RunCommand acCmdFind
    Exit_Routine
    Exit Sub
End Sub
```
Using MS Access

**Problem:**
Display the (first & Last Name) of the three best paid employees.

**Steps**
1. Design a form similar to the following screen-shot.
2. Put the provided code behind the command button’s click event

```vba
Private Sub cmdTop3Salaries_Click()
    'using ADO control
    Dim myRS As New ADODB.Recordset
    Dim mySQL As String
    Dim empCounter As Integer
    Dim totalSalary As Double
    myRS.ActiveConnection = CurrentProject.Connection
    mySQL = "select Fname, Lname, Salary from EMPLOYEE order by Salary DESC"
    myRS.Open (mySQL)
    empCounter = 0
    While (Not myRS.EOF) And (empCounter < 3)
        empCounter = empCounter + 1
        totalSalary = totalSalary + myRS(“Salary”)
        Select Case empCounter
            Case 1:
                txtEmpName1.SetFocus
                txtEmpName1 = myRS(“Fname”) & “ “ & myRS(“Lname”)
                txtSalary1 = myRS(“Salary”)
            Case 2:
                txtEmpName2 = myRS(“Fname”) & “ “ & myRS(“Lname”)
                txtSalary2 = myRS(“Salary”)
            Case 3:
                txtEmpName3 = myRS(“Fname”) & “ “ & myRS(“Lname”)
                txtSalary3 = myRS(“Salary”)
        End Select
        myRS.MoveNext
    Wend
    txtSalary = totalSalary
End Sub
```