VBA Made Easy

Recordsets

10

www.accessallinone.com

This guide was prepared for AccessAllInOne.com by:
Robert Austin

This is one of a series of guides pertaining to the use of Microsoft Access.

Contents

Recordsets

Introduction: What are Recordsets

So, what is a recordset? You can think of a recordset as a table or query that we can utilise (read, update, delete, insert) but cannot see. When we open a recordset, the recordset itself is stored in memory and we are able to loop through the records one at a time, manipulating the data as we go.

Recordsets enable us to reference field names, search for records, filter records, count records and much more. With recordsets, we can truly interact with the data stored in our databases. There are two types of recordsets, DAO and ADODB. They both have similar functions and similar operation speeds (DAO is a bit faster) but as a general rule use DAO if you are referencing standard Access tables within your database and use ADODB when you are referencing tables held outside of your Access application (SQL Server for instance).

In order to use recordsets, we need to reference certain libraries. As standard, the DAO library is already referenced in Access whilst the ADO library is not.

Checking DAO is Referenced

Open a new module and enter the following code. In the immediate window, execute the function by entering `testDAO` and pressing the return key.

```
1   Sub testDAO()
2       'This sub-procedure loops through all current
3       'references and looks for a DAO reference
4       Dim A As Variant
5       For Each A In Application.References
6           If A.Name = "DAO" Then
7               MsgBox "DAO Library loaded!"
8               Exit Sub
9           End If
10      Next
11      MsgBox "DAO Library NOT loaded"
12  End Sub
```

Figure 10.1

Checking ADO is Referenced

Open a new module and enter the following code. In the immediate, window execute the sub by entering `testADO` and pressing the enter key.

```
1   Sub testADO()
2       'This sub-procedure loops through all current
3       'references and looks for a ADO reference
4       Dim A As Variant
5       For Each A In Application.References
6           If A.Name = "ADODB" Then
7               MsgBox "ADO Library loaded!"
8               Exit Sub
```

```
9           End If
10      Next
11      MsgBox "ADO Library NOT loaded"
12  End Sub
```

Figure 10.2

Adding Missing DAO and ADO References

If either DAO or ADO is missing we need to add them to the VBA IDE by selecting the Tools menu and References…

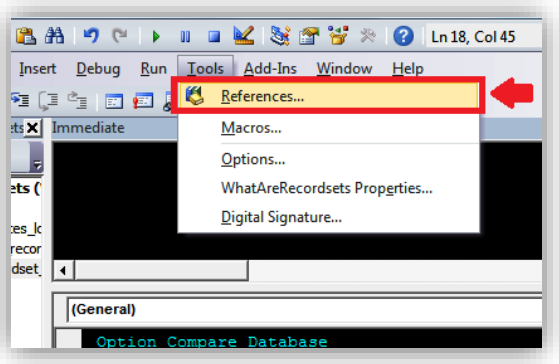

Figure 10.3

Adding DAO References

To set the DAO reference find the references below (they are dependent on your version of Access).

Microsoft Office 12.0 Access Database Engine Objects Library (Access 2007)

Microsoft Office 14.0 Access Database Engine Objects Library (Access 2010)

Microsoft Office 15.0 Access Database Engine Objects Library (Access 2013)

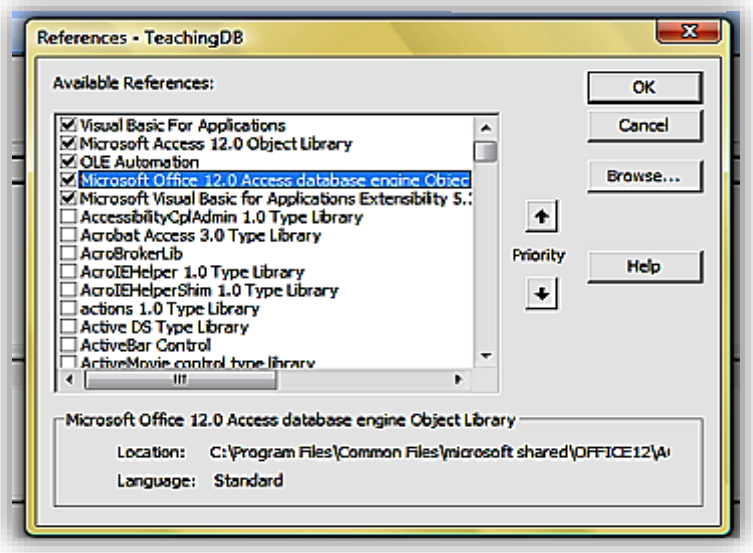

Figure 10.4

Adding ADO References

To set the ADO reference find the references below (they are dependent on your version of Access).

Microsoft ActiveX Data Objects 6.0 Library (Access 2007)

Microsoft ActiveX Data Objects 6.1 Library (Access 2010/2013)

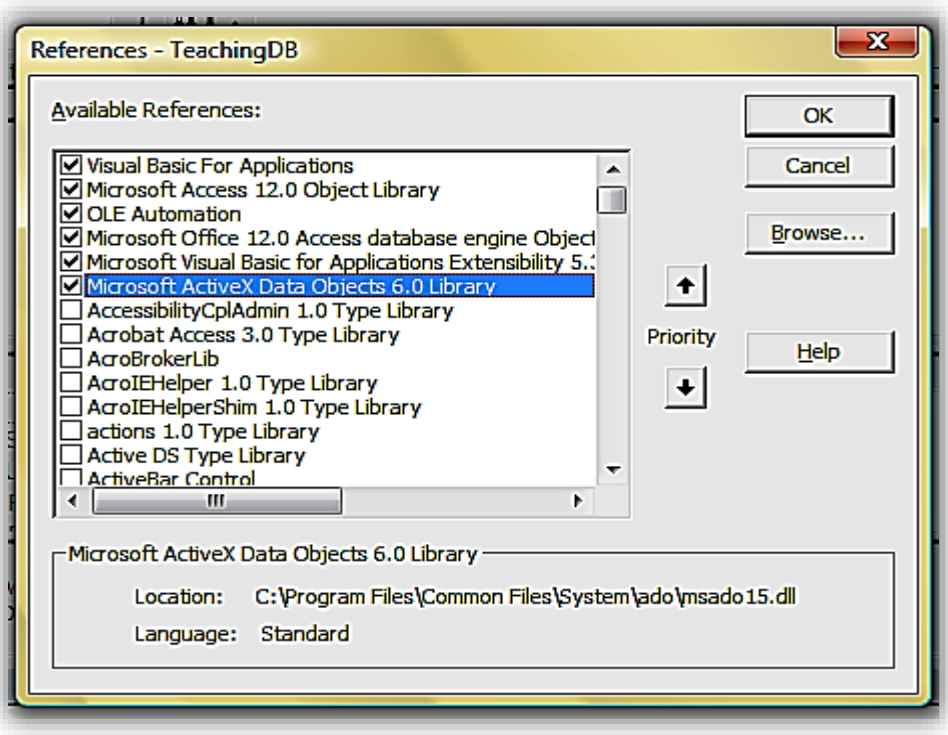

Figure 10.5

Again, Check DAO and ADO References

To check if the above referencing has worked, rerun the two test methods.

Declaring a Recordset Object

In order to use a recordset, you will need to declare it! Here we provide you with the most standard way to declare a recordset object.

DAO Recordsets

To declare a DAO Recordset object in a module, use the following code:

```
1   Sub declareDAORecordset()
2
3       Dim db As DAO.Database
4       Dim rs As DAO.Recordset
5       Set db = CurrentDb
6       Set rs = db.OpenRecordset("tblStudents")
7       'Opens tblStudents in memory. We can access all the
8       'data stored in tblStudents but we cannot see the
9       'table itself
10
11      With rs
12          .MoveLast
13          .MoveFirst
14          'These two lines of code are necessary to ensure
15          'that the recordcount property works correctly
16
17          Debug.Print .RecordCount
18          'Here we are printing the number of records in
19          'tblStudents to the immediate window
20      End With
21
22  End Sub
```

Figure 10.6

The code in Figure 10.6 is the most standard way to declare a recordset.

ADODB Recordsets

To declare an ADODB Recordset object in a module we can use the following code:

```
1    Sub declareADODBRecordset()
2
3        Dim rs As New ADODB.Recordset
4     rs.Open "tblStudents", CurrentProject.Connection, _
5    adOpenKeyset, adLockOptimistic
6        'Opens tblStudents in memory. We can access all the
7        'data stored in tblStudents but we cannot see the
8        'table itself
9
10       With rs
11           .MoveLast
12           .MoveFirst
13           'These two lines of code are necessary to ensure
14           'that the recordcount property works correctly
15
16           Debug.Print .RecordCount
17           'Here we are printing the number of records in
18           'tblStudents to the immediate window
19       End With
20
21   End Sub
```

Figure 10.7

In an Access module, ADO Recordsets can be obtained from the CurrentProject object which is in global scope so you can access it from anywhere in your project, even forms.

DAO vs ADODB

As stated above, use DAO if you are referencing standard Access tables within your database and use ADODB when you are referencing tables held outside of your Access application (SQL Server for instance). You should try to become familiar with both DAO and ADODB objects as this will give you more flexibility going forward.

Cursors and Lock Types

In the ADODB code, we have the line:

```
 rs.Open "tblStudents", CurrentProject.Connection, adOpenKeyset,
adLockOptimistic
```

We know that "tblStudents" refers to the table we wish to open. We also know that CurrentProject.Connection refers to the connection from the current database (remember that ADODB recordsets can be used with connections outside of the current database such as SQL Server). But what do adOpenKeyset and adLockOptimistic refer to?

adOpenKeyset is what we refer to as a cursor and adOpenKeyset as a lock type.

Cursors

A cursor is a mechanism that gives VBA a view of the data, points to what it is currently looking at, and determines if we can move forward or backward through the data.

- Forward Only - cursor lets you move only to the next line, so no backwards mouse movements are allowed. That type of cursor is great just to look at data. Forward Only also only gets a portion of the data at a time – i.e. only that which fits on screen; it will fetch the rest when needed. This cursor uses the least amount of memory and CPU time.
    - The ADODB constant for this is **adOpenForwardOnl**y.
- Static Cursor - downloads the whole set of data and lets you move the cursor back and forth with ease. But you can't change the data; great for reports, but not so great for updating data. This cursor uses more memory than CPU resources.
    - The ADODB constant for this is **adOpenStatic**.
- Keyset Cursor - also downloads the whole set of data, lets you move back and forth, and also lets you see data that has been updated by other users and, when deleted, hides that data as well. Keyset also allows you to perform updates and inserts of records. You don't though get to see others' added rows. This cursor uses a little more memory as Static cursors and definitely more CPU time.
    - The ADODB constant for this is **adOpenKeyset**.
- Dynamic Cursor - offers everything the keyset cursors does, plus lets you see inserted records, deleted entries, lets you update and add and delete for yourself too. But this cursor uses much more memory and a lot more CPU resources.
    - The ADODB constant for this is **adOpenDynamic**

Because Access is a file-based system and will usually operate across a small network, Access can afford to use dynamic cursors as default, but over an internet connection drive performance could be hit.

Locks

Locking is only a consideration when inserting, updating or deleting data.

- Read Only - whilst your cursor is reading the data nobody else can change it
- Pessimistic locking - locks all records you are using or have used since the form or recordset has opened. This type of lock guarantees your data will be saved.
- Optimistic locking - doesn't lock anything until the moment you want to make an update, insert or delete. This type of lock guarantees your data will be saved if nobody has updated a record you have used in making the decision to update, insert or delete.

Lock Types

Locking involves stopping other users making alterations to the record we are looking at or working with. You need only be concerned with locking if you are intending to update the database in any way. So, on forms that don't update or insert they should not use locking, and as such should only use cursors that do not allow locking; forward-only or static. If, on the other hand, you need to perform updates or inserts then we must use a Keyset or Dynamic cursor.

A lock is placed on a row of data that either tells all other users that you are either reading the record or writing to the record.

If you have a lot of users updating lots of records and it takes a while to perform a task, use **adLockPessimistic**.

If that causes you problems or your data isn't updated much - irrespective of the number of users - use **adLockOptimistic**.

The major difference between these two lock types is that Pessimistic locking prevents all other users from changing the data being currently modified. Optimistic locking locks nothing until the save button is pressed.

Recordsets and SQL

In the above examples, we have used "tblStudents" in the **.open** statements of the DAO and ADODB recordsets. We don't just have to use a simple reference to a table; we can use a Structured Query Language (SQL) statement instead.

Note: Structured Query Language is outside the scope of this series but we will provide you with a couple of examples to get started.

```
1   …
2       Set db = CurrentDb
3       Set rs = db.OpenRecordset("SELECT * FROM  tblStudents")
4
5       With rs
6   …
```

Figure 10.8

Figure 10.8 will return exactly the same recordset as ***Set rs = db.OpenRecordset("tblStudents")*** because "***SELECT * FROM tblStudents***" means "*select all records from tblStudents*". We can use a where statement to restrict any records that are being returned.

```
1   SELECT * FROM   tblStudents WHERE  studentid < 10
```

Figure 10.9

In Figure 10.9, we have added ***WHERE studentid < 10*** to the original SQL statement. The literal translation of Figure 10.9 is "*select all records from tblStudents where the studentid field is less than 10*".

If you are going to be referring to text fields in the WHERE statement, you will need to use single quotation marks.

```
1   SELECT * FROM   tblStudents WHERE  FirstName = 'Anna'
```

Figure 10.10

In Figure 10.10, all records in tblStudents where the FirstName field is equal to Anna will be returned. Without the single quotation marks, this statement would throw an error.

Looping Through a Recordset

Recordsets act like a cursor or a ruler underneath a row of data. They only operate on one row at a time so to access the data returned by the database we must *Move* the cursor Next or Previous, First or Last.

The EOF and BOF properties

Recordsets have two important properties when looping through data, EOF (End-Of-File) and BOF (Beginning-Of-File). Recordsets are like tables and when you loop through one, you are literally moving from record to record in sequence. As you move through the records the EOF property is set to false but after you try and go past the last record, the EOF property becomes true. This works the same in reverse for the BOF property.

These properties let us know when we have reached the limits of a recordset.

```
1   Sub DAOLooping()
2   On Error GoTo ErrorHandler
3
4   Dim strSQL As String
5   Dim rs As DAO.Recordset
6
7   strSQL = "tblTeachers"
8   'Instead of "tblTeachers"
9   'you could use a full SELECT statement such as:
10  'SELECT * FROM tblTeachers (this would produce the same result in fact).
11  'You could also add a Where clause to filter which records are returned:
12  'SELECT * FROM tblTeachers Where ZIPPostal = '98052'
13  ' (this would return 5 records)
14
15  Set rs = CurrentDb.OpenRecordset(strSQL)
16  'This line of code instantiates the recordset object!!!
17  'In English, this means that we have opened up a recordset
18  'and can access its values using the rs variable.
19
20  With rs
21
22
23      If Not .BOF And Not .EOF Then
24      'We don't know if the recordset has any records,
25      'so we use this line of code to check. If there are no records
26      'we won't execute any code in the if..end if statement.
27
28          .MoveLast
29          .MoveFirst
30          'It is not necessary to move to the last record and then back
31          'to the first one but it is good practice to do so.
32
33          While (Not .EOF)
34          'With this code, we are using a while loop to loop
35          'through the records. If we reach the end of the recordset, .EOF
36          'will return true and we will exit the while loop.
37
38         Debug.Print rs.Fields("teacherID") & " " & rs.Fields("FirstName")
39              'prints info from fields to the immediate window
40
41              .MoveNext
42              'We need to ensure that we use .MoveNext,
```

```
43              'otherwise we will be stuck in a loop forever…
44              '(or at least until you press CTRL+Break)
45          Wend
46
47      End If
48
49      .close
50      'Make sure you close the recordset...
51  End With
52
53  ExitSub:
54      Set rs = Nothing
55      '..and set it to nothing
56      Exit Sub
57  ErrorHandler:
58      Resume ExitSub
59  End Sub
```

```
The output in the immediate window will be:

1 Anna
2 Antonio
3 Thomas
4 Christina
5 Martin
6 Francisco
7 Ming-Yang
8 Elizabeth
9 Sven
```

Figure 10.11

Figure 10.11 is an excellent recordset template to use for any code you write.

Looping Through An ADODB Recordset

The same concept for an ADODB recordset would look like this:

```
1   Sub ADOLooping()
2
3   On Error GoTo ErrorHandler
4
5   Dim strSQL As String
6   Dim rs As New ADODB.Recordset
7
8   'we will be opening tblTeachers
9   strSQL = "tblTeachers"
10
11  rs.Open strSQL, CurrentProject.Connection, adOpenDynamic,
12  adLockOptimistic
13
14  With rs
15
16      'Ensure recordset is populated
17      If Not .BOF And Not .EOF Then
18
19          'not necessary but good practice
20          .MoveLast
21          .MoveFirst
22
23          While (Not .EOF)
24          'print info from fields to the immediate window
25         Debug.Print rs.Fields("teacherID") & " " & rs.Fields("FirstName")
26              .MoveNext
27          Wend
28
29      End If
30      .close
31  End With
32
33  ExitSub:
34      Set rs = Nothing
35      Exit Sub
36  ErrorHandler:
37      Resume ExitSub
38  End Sub
```

Figure 10.12

Updating, Adding and Deleting Records

Recordsets wouldn't be a whole lot of use if we couldn't manipulate the data. In this section, we show you how to update, add and delete records in a recordset.

Updating A Record

DAO

```
1       1. Sub DAOUpdating()
2       2. On Error GoTo ErrorHandler
3       3. 'This sub-produre will add 'z' to the first name of
4       4. 'the record that corresponds to TeacherID 5
5       5. Dim sql As String
6       6. Dim rs As DAO.Recordset
7       7. sql = "SELECT * FROM tblTeachers WHERE TeacherID=5"
8   'We are using a select statement that will return only
9   'one record (TeacherID 5)
10  Set rs = CurrentDb.OpenRecordset(sql)
11  'Open RecordSet
12
13  With rs
14      If Not .BOF And Not .EOF Then
15      'Ensure that the recordset contains records
16      'If no records the code inside the if...end if
17      'statement won't run
18
19          .MoveLast
20          .MoveFirst
21          'Not necessary but good practice
22
23          If .Updatable Then
24          'It is possible that the record you want to update
25          'is locked by another user. If we don't check before
26          'updating, we will generate an error
27
28              .Edit
29              'Must start an update with the edit statement
30              ![FirstName] = "z" & ![FirstName]
31              'Another way of accessing the fields would be to use
32              '.fields("FirstName") = z" & .fields("FirstName")
33
34              .Update
35              'And finally we will need to confirm the update
36          End If
37      End If
38      .Close
39      'Make sure you close the recordset...
40  End With
41
42  ExitSub:
43      Set rs = Nothing
44      '...and set it to nothing
45      Exit Sub
46  ErrorHandler:
47      Resume ExitSub
48
49  End Sub
```

Figure 10.13

ADODB

```
1   Sub ADODBUpdating()
2   On Error GoTo ErrorHandler
3   'This sub-produre will add 'z' to the first name of
4   'the record that corresponds to TeacherID 5
5   Dim sql As String
6   Dim rs As adodb.Recordset
7
8   sql = "SELECT * FROM tblTeachers WHERE TeacherID=5"
9   'We are using a select statement that will return only
10  'one record (TeacherID 5)
11
12  Set rs = New adodb.Recordset
13  rs.Open sql, CurrentProject.Connection, adOpenDynamic, adLockOptimistic
14  'Open RecordSet
15
16  With rs
17
18      If Not .BOF And Not .EOF Then
19      'Ensure that the recordset contains records
20      'If no records the code inside the if...end if
21      'statement won't run
22
23          .MoveLast
24          .MoveFirst
25          'Not necessary but good practice
26
27          If .Supports(adUpdate) Then
28          'It is possible that the record you want to update
29          'is locked by another user. If we don't check before
30          'updating, we will generate an error
31              ![FirstName] = "x" & ![FirstName]
32              'Another way of accessing the fields would be to use
33              '.fields("FirstName") = z" & .fields("FirstName")
34              .Update
35              'And finally we will need to confirm the update
36          End If
37      End If
38
39      .Close
40      'Make sure you close the recordset...
41  End With
42
43  ExitSub:
44      Set rs = Nothing
45      '...and set it to nothing
46      Exit Sub
47  ErrorHandler:
48      Resume ExitSub
49
50  End Sub
```

Figure 10.14

Adding A Record

DAO

```
1   Sub DAOAdding()
2   On Error GoTo ErrorHandler
3   'This sub-produre will add a new record to tblTeachers
4   Dim sql As String
5   Dim rs As DAO.Recordset
6
7   sql = "tblTeachers"
8   'The table we will be adding the record to is tblTeachers
9
10  Set rs = CurrentDb.OpenRecordset(sql)
11  'Open RecordSet
12
13  With rs
14
15      .AddNew
16      'Must start an update with the AddNew statement
17
18      .Fields!FirstName = "Steve"
19      .Fields!LastName = "Evets"
20      .Fields!CreatedBy = 1 ' NOT NULL
21      'Here we are adding someone called Steve Evets and adding
22      'the number 1 to the CreatedBy field
23
24      .Update
25      'And finally we will need to confirm the update
26
27
28      .Close
29      'Make sure you close the recordset...
30  End With
31
32  ExitSub:
33      Set rs = Nothing
34      '...and set it to nothing
35      Exit Sub
36  ErrorHandler:
37      Resume ExitSub
38
39  End Sub
```

Figure 10.15

ADODB

```
1   Sub ADOAdding()
2   On Error GoTo ErrorHandler
3   'This sub-produre will add a new record to tblTeachers
4   Dim sql As String
5   Dim rs As adodb.Recordset
6
7   sql = "tblTeachers"
8   'The table we will be adding the record to is tblTeachers
9
10  Set rs = New adodb.Recordset
11  rs.Open sql, CurrentProject.Connection, adOpenDynamic, adLockOptimistic
12  'Open RecordSet
13
14  With rs
15
16      .AddNew
17      'Must start an update with the AddNew statement
18
19      .Fields!FirstName = "Robert"
20      .Fields!LastName = "Trebor"
21      .Fields!CreatedBy = 1   ' NOT NULL
22      'Here we are adding someone called Robert Trebor and adding
23      'the number 1 to the CreatedBy field
24
25      .Save
26      'To confirm the addition we need to use the Save command
27
28      .Close
29      'Make sure you close the recordset...
30  End With
31
32  ExitSub:
33      Set rs = Nothing
34      '...and set it to nothing
35      Exit Sub
36  ErrorHandler:
37      Resume ExitSub
38  End Sub
```

Figure 10.16

Deleting Records

DAO

```
1   Sub DAODeleting()
2   On Error GoTo ErrorHandler
3   'This sub-produre will delete the record that
4   'corresponds to TeacherID 7
5   Dim sql As String
6   Dim rs As DAO.Recordset
7
8   sql = "SELECT * FROM tblTeachers WHERE TeacherID=7"
9   'We are using a select statement that will return only
10  'one record (TeacherID 7)
11
12  Set rs = CurrentDb.OpenRecordset(sql)
13  'Open RecordSet
14
15  With rs
16
17      If Not .BOF And Not .EOF Then
18      'Ensure that the recordset contains records
19      'If no records the code inside the if...end if
20      'statement won't run
21
22          .MoveLast
23          .MoveFirst
24          'Not necessary but good practice
25
26          If .Updatable Then
27          'It is possible that the record you want to update
28          'is locked by another user. If we don't check before
29          'updating, we will generate an error
30
31           .Delete
32           'The only command we need!
33           'Be careful!!! Once a record is deleted, it is gone forever...
34
35          End If
36      End If
37
38      .Close
39      'Make sure you close the recordset...
40  End With
41
42  ExitSub:
43      Set rs = Nothing
44      '...and set it to nothing
45      Exit Sub
46  ErrorHandler:
47      Resume ExitSub
48  End Sub
49
```

Figure 10.17

ADODB

```
1   Sub ADODeleting()
2   On Error GoTo ErrorHandler
3   'This sub-produre will delete the record that
4   'corresponds to TeacherID 7
5   Dim sql As String
6   Dim rs As adodb.Recordset
7
8   sql = "SELECT * FROM tblTeachers WHERE TeacherID=8"
9   'We are using a select statement that will return only
10  'one record (TeacherID 7)
11
12  Set rs = New adodb.Recordset
13  rs.Open sql, CurrentProject.Connection, adOpenDynamic, adLockOptimistic
14  'Open RecordSet
15
16  With rs
17
18      If Not .BOF And Not .EOF Then
19      'Ensure that the recordset contains records
20      'If no records the code inside the if...end if
21      'statement won't run
22
23          .MoveLast
24          .MoveFirst
25          'Not necessary but good practice
26
27          If .Supports(adDelete) Then
28          'It is possible that the record you want to update
29          'is locked by another user. If we don't check before
30          'updating, we will generate an error
31
32           .Delete
33           'The only command we need!
34           'Be careful!!! Once a record is deleted, it is gone forever...
35
36          End If
37      End If
38
39      .Close
40      'Make sure you close the recordset...
41  End With
42
43  ExitSub:
44      Set rs = Nothing
45      '...and set it to nothing
46      Exit Sub
47  ErrorHandler:
48      Resume ExitSub
49  End Sub
```

Figure 10.18

Questions

1) True or false?
    a. ADO is loaded by default when access is installed.
    b. DAO is the same as ADO.
    c. DAO.Net.
    d. We can use CurrentDB to get an ADO recordset.
    e. ADODB and DAO are interfaces for VBA to the database layer.

2) Which of the following are valid strings to open a recordset with?
    a. tblFurnatureManufacturers
    b. qryDogsbyDOB
    c. frmAnimalEditRecord
    d. rptInvoices
    e. select * from [home addresses]
    f. qryPivotYearMonth
    g. where [OEMid]=”00191JU1”

3) After opening an Access database table in a DAO or ADO object, which will correctly show the number of records? DAO or ADODB.

4) Why?

5) Can For Each be used on an DAO recordset object? Why or why not?

6) Can For Each be used on an ADO Fields object? Why or why not?

7) What might it mean if EOF and BOF are true on a recordset object?

8) Using the ADODB.Recordset object what are the commands to do the following
    a. Retrieve the last record
    b. Retrieve first record
    c. Retrieve the third record from the front
    d. Retrieve the second record from the rear of the table

9) When moving backwards through a set of records which property of the ADO and DAO object will signify no more records to read?

10) Rewrite the following code to include the With statement on object rsADO

```
1     rsADO.MoveFirst
2     rsADO.MoveNext
3     If rsADO.Supports(adUpdate) Then
4       rsADO![FirstName] = "x" & rsADO![FirstName]
5       rsADO.Update
6     End If
```

11) Make the following code work:

```
1   Dim sql As String : sql = "tblBuildings"
2   Dim rsADO As ADODB.Recordset
3   Set rsADO = openADODBRecordset()
4
5   Dim adoField As ADODB.field
6   For Each adoField In rsADO.Fields
7     Debug.Print adoField.Name
8   Next
```

12) Which would you need to use if you wanted to edit structures within an Access database, DAO or ADODB?

13) Which menu item would we need to use to add ADODB library?

14) How does a recordset differ from a table?

15) Do forms work with ADO or DAO recordsets?

16) Change the following code to work as expected:

```
1     Debug.Print "Teacher Records"
2     While (rsDAO.EOF)
3       Debug.Print rsDAO.Fields("teacherID"); rsDAO![FirstName]
4       rsDAO.MoveNext
5     Wend
```

17) What type of situations may prevent the following from executing?

```
1   Set rsADO = openADORecordset("tblTeachers")
2   With rsADO
3     .AddNew
4     .Fields!FirstName = "James"
5     .Fields!LastName = "Mustafa"
6     .Fields!CreatedBy = 2   ' NOT NULL
7     .Save
8   End With
```

18) Write the instructions to dispose of the ADO object.

Answers - Recordsets

1) True or false
    a. False
    b. False
    c. False
    d. False
    e. True

2) Yes or No
    a. Yes
    b. Yes
    c. No
    d. No
    e. Yes
    f. Yes
    g. No

3) DAO will have correct the recordcount property.

4) Because DAO is closer to the Access database layer and has functionality that ADO doesn't have.

5) No, recordset objects are not collections.

6) Yes, because Field objects are collections.

7) There are no records in the recordset or we are at the first record.

8) See below
    a. MoveLast
    b. MoveFirst
    c. MovcFirst, MoveNext, MoveNext
    d. MoveLast, Move Previous,

9) BOF

10) See below

```
1   With rsADO
2     .MoveFirst
3     .MoveNext
4     If .Supports(adUpdate) Then
5       ![FirstName] = "x" &![FirstName]
6       .Update
7     End If
8   End With
```

11) See below

```
1   Dim sql As String : sql = "tblBuildings"
2   Dim rsADO As ADODB.Recordset
3   Set rsADO = openADODBRecordset(sql)
4
5   Dim adoField As ADODB.field
6   For Each adoField In rsADO.Fields
7     Debug.Print adoField.Name
8   Next
```

12) DAO, because DAO is an AccessObject and implements the Access Data Model.

13) Tools and Reference...

14) A table is a set of rows and columns containing data.
A recordset contains at any one time just one or no rows of a table.

15) DAO

16) See below

```
1     Debug.Print "Teacher Records"
2     While (Not rsDAO.EOF)
3       Debug.Print rsDAO.Fields("teacherID"); rsDAO![FirstName]
4       rsDAO.MoveNext
5     Wend
```

17) Answers

a. rsADO doesn't have read access
b. rsADO is not open
c. Where another field has a NOT NULL property

20) See below

```
1   rsADO.Close: Set rsADO = Nothing
```