Die wichtigsten Grundlagen zum Einsatz von DAO ADO Recordsets nutzen

Teil 1
Versionen: Access 2013, 2010, 2007, 2003 und 2002

Um programmatisch auf die Inhalte von Tabellen zuzugreifen, werden "Recordsets" eingesetzt. Ein Recordset ist einfach ausgedrückt eine Sammlung von Datensätzen, die in Schleifen durchlaufen und auf verschiedener Art verarbeitet werden kann. Alle notwendigen Grundlagen dazu vermittelt Ihnen unser Tipp.

Die Basis eines Recordsets kann eine Tabelle oder eine Abfrage, eine Kombination aus mehreren Tabellen und/oder Abfragen oder eine SELECT-Anweisung sein. Es bilden entweder alle Datensätze ein Recordset oder nur Datensätze, die einem bestimmten Kriterium wie "Land = 'Deutschland'" oder "Umsatz > 10000" entsprechen. In einem Recordset können neue Datensätze angelegt und gefüllt oder vorhandene Datensätze gelöscht werden, sofern ihm eine eindeutige, konsistente Datenbasis zugrunde liegt. Und natürlich können Sie auf einzelne Felder zugreifen und Inhalte ändern.

Als Datenzugriffsmethode kommt DAO (Data Access Objects) oder ADO (ActiveX Data Objects) zum Einsatz. ADO war ursprünglich als Ersatz für DAO geplant, Proteste aus der Entwicklergemeinde hatten aber zur Folge, dass DAO weiterhin unterstützt und kontinuierlich weiterentwickelt wird. ADO bringt eine Reihe an Erweiterungen beispielsweise für die Vereinfachung des Zugriffs auf beliebige lokale oder entfernte Datenquellen oder in Bezug auf die Datensicherheit mit. Für welche Datenzugriffsmethode Sie sich entscheiden, hängt hauptsächlich davon ab, auf welche Art von Daten Sie zugreifen möchten: Wenn es darum geht, auf die Tabellen der aktuellen Datenbanken zuzugreifen, empfiehlt sich DAO, das etwas einfacher zu handhaben und schneller in der Ausführung ist. Müssen Sie auf spezielle Datenquellen wie einen SQL-Server im Internet zugreifen oder sind besondere Anforderungen beispielsweise in Bezug auf Datensicherheit zu erfüllen, nutzen Sie ADO.

Um mit DAO oder ADO arbeiten zu können, muss zunächst ein entsprechender Verweis gesetzt sein. Access erledigt dies automatisch beim Anlegen einer neuen Datenbank. In Access 2003 und 2002 wird ein Verweis auf DAO eingerichtet, ab Access 2007 auf die Access Database Engine, die DAO enthält. Möchten Sie in einer Datenbank ADO nutzen, müssen Sie die Referenzen entsprechend ergänzen:

1. Wechseln Sie mit **Alt+F11** in die VBA-Entwicklungsumgebung.
2. Wählen Sie das Menü *Extras-Verweise* an.
3. Setzen Sie das Kontrollkästchen vor dem Eintrag "Microsoft ActiveX Data Objects x.x Library". Für "x.x" wählen Sie die höchste verfügbare Version.
4. Übernehmen Sie die Änderungen mit einem Klick auf *Ok*.

Wenn DAO und ADO parallel eingesetzt werden sollen, stellen Sie beim Deklarieren ein "DAO." bzw. ein "ADODB." vor den Objektbezeichner, also beispielsweise:

Dim rs As DAO.Recordset
Dim rs As ADODB.Recordset

Damit Sie per DAO oder ADO über ein Recordset auf Datensätze und Felder zugreifen können, muss es zunächst mit der gewünschten Datenbasis initialisiert werden. Das erfolgt in der Regel immer im Kontext der jeweiligen Datenbank, die bei DAO durch das "Database"-Objekt referenziert wird. Dieses Objekt verfügt über eine Methode "OpenRecordset" für die Initialisierung von Recordsets. Recordsets selbst stellen ebenfalls Methoden zur Verfügung, mit denen man beispielsweise durch Datensätze blättert (MoveFirst, MoveLast, MoveNext, MovePrevious), Datensätze anlegt (Add) oder Datensätze löscht (Delete). Ein erstes einfaches DAO-Beispiel greift auf die Tabelle "Kunden" der aktuellen Datenbank zu und gibt den Inhalt der Felder des ersten Datensatzes im Direktfenster aus:

Sub Test1()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Dim fld As DAO.Field

  Set db = CurrentDb()
  Set rs = db.OpenRecordset("Kunden", dbOpenDynaset)
  For Each fld In rs.Fields
    Debug.Print fld.Value & ";";
  Next
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Mit "CurrentDb()" wird zunächst eine Referenz auf die aktuelle Datenbank ermittelt und darüber per "OpenRecordset" ein Recordset "rs" initialisiert, das alle Datensätze der Tabelle "Kunden" umfasst. In einer Schleife gehen wir dann alle Felder des aktuellen Datensatzes durch und schreiben die Feldinhalte in das Direktfenster. Der Aufruf der Methode "Close" schließt das Recordset; die letzten beiden Anweisungen geben die verwendeten Objektvariablen wieder frei.

Die Vorgehensweise beim Initialisieren von Recordsets ist bei ADO ein wenig komplizierter, da hier die "Zuständigkeiten" etwas umgestellt sind. Während ein Recordset bei DAO von der Methode "OpenRecordset" des Database-Objektes initialisiert wird, verfügt ein Recordset bei ADO über eine eigene Methode "Open" und initialisiert sich darüber sozusagen selbst. Als Parameter ist die Verbindung anzugeben, über die auf die betreffende Tabelle oder Abfrage zugegriffen werden soll.

Sub Test2()
  Dim conn As New ADODB.Connection
  Dim rs As New ADODB.Recordset
  Dim fld As ADODB.Field

  Set conn = CurrentProject.Connection
  rs.Open "Kunden", conn
  For Each fld In rs.Fields
    Debug.Print fld.Value & ";";
  Next
  rs.Close

  Set rs = Nothing
  Set conn = Nothing

End Sub

Über "CurrentProject" wird zunächst eine Referenz auf die aktuelle Datenbank ermittelt, der Objektvariablen "conn" zugewiesen und diese der Methode "Open" als Parameter übergeben. In einer Schleife werden danach alle Felder des aktuellen Datensatzes durchlaufen und deren Feldinhalte ins Direktfenster geschrieben. "Close" schließt das Recordset und die letzten beiden Anweisungen geben die verwendeten Objektvariablen wieder frei.

In der nächsten Ausgabe erläutern wir weitere Techniken für die Navigation in Recordsets und die Suche nach bestimmten Datensätzen.

Teil 2

Um programmatisch auf die Inhalte von Tabellen zuzugreifen, werden "Recordsets" eingesetzt. Ein Recordset ist einfach ausgedrückt eine Sammlung von Datensätzen, die in Schleifen durchlaufen und auf verschiedener Art verarbeitet werden kann. Alle notwendigen Grundlagen dazu vermittelt Ihnen unser Tipp.

Nachdem wir im ersten Teil Woche Grundlagen zu Recordsets und die Initialisierung erläutert haben, geht es nun darum, wie in Recordsets navigiert und gesucht wird. Zunächst zur Navigation in Recordsets: Um sich durch Recordsets zu bewegen oder den Status von Recordsets abzufragen, werden sowohl mit DAO als auch mit ADO die folgenden Methoden und Eigenschaften verwendet:

* rs.RecordCount: Anzahl Datensätze im Recordset ermitteln
* rs.MoveFirst: Auf den ersten Datensatz springen
* rs.MoveLast: Auf den letzten Datensatz springen
* rs.MoveNext: Zum nächsten Datensatz gehen
* rs.MovePrevious: Zum vorhergehenden Datensatz gehen

Das folgende DAO-Beispiel verdeutlicht dies:

Sub Test3()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Dim Anz As Long, I As Long

  Set db = CurrentDb()
  Set rs = \_
    db.OpenRecordset("SELECT \* FROM Kunden " & \_
                     "WHERE Land = 'Deutschland'")
  Anz = rs.RecordCount
  For I = 1 To Anz
    Debug.Print rs("Firma")
    rs.MoveNext
  Next
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Hier wird zunächst ein Recordset initialisiert, das aus der Tabelle "Kunden" der aktuellen Datenbank alle Datensätze selektiert, bei denen das Feld "Land" den Inhalt "Deutschland" hat. Es muss bei "OpenRecordset()" also nicht immer eine Tabelle oder eine Abfrage angegeben werden, eine SQL-Abfrage wie die oben gezeigte ist ebenfalls möglich. Über "RecordCount" wird dann die Anzahl der Datensätze im Recordset abgefragt und einer Variablen zugewiesen. Bei über ODBC verknüpften Tabellen liefert "RecordCount" manchmal falsche Ergebnisse. Nutzen Sie dann die folgende Konstruktion:

  rs.MoveLast
  Anz = rs.RecordCount
  rs.MoveFirst

"MoveLast" positioniert explizit auf den letzten Datensatz, so dass über "RecordCount" zuverlässig die Anzahl der Datensätze im Recordset abgefragt werden kann. Anschließend wird per "MoveFirst" wieder auf den ersten Datensatz gesprungen.

In einer Schleife durchlaufen wir dann alle Datensätze und geben den Inhalt des Feldes "Firma" für jeden Datensatz im Direktfenster aus, wobei "MoveNext" auf den jeweils nächsten Datensatz positioniert.

Alternativ kann eine Schleife zum Durchlaufen aller Datensätze auch mit Hilfe der folgenden Eigenschaften aufgebaut werden:

* rs.EOF: Letzter Datensatz erreicht?
* rs.BOF: Erster Datensatz erreicht?

Mit ADO sieht das zum Beispiel so aus:

Sub Test4a()
  Dim conn As New ADODB.Connection
  Dim rs As New ADODB.Recordset

  Set conn = CurrentProject.Connection
  rs.Open "select \* from Kunden " & \_
           "where Land= 'Belgien'", \_
           conn, \_
           adOpenKeyset
  While Not rs.EOF
    Debug.Print rs("Firma")
    rs.MoveNext
  Wend
  rs.Close

  Set rs = Nothing
  Set conn = Nothing

End Sub

Anstelle einer "For-Next"-Schleife wird hier eine "While-Wend"-Schleife mit Kriterium "rs.EOF" verwendet. "rs.EOF" liefert solange "False", bis der letzte Datensatz erreicht ist, dann wechselt es auf "True", so dass die Schleife terminiert wird. Eine weitere Besonderheit: das Recordset wird mit einem zusätzlichen Parameter "adOpenKeyset" initialisiert, so dass er bei Bedarf auch rückwärts durchblättert werden kann:

Sub Test4b()
  Dim conn As New ADODB.Connection
  Dim rs As New ADODB.Recordset

  Set conn = CurrentProject.Connection
  rs.Open "select \* from Kunden " & \_
           "where Land= 'Belgien'", \_
           conn, \_
           adOpenKeyset
  rs.MoveLast
  While Not rs.BOF
    Debug.Print rs("Firma")
    rs.MovePrevious
  Wend
  rs.Close

  Set rs = Nothing
  Set conn = Nothing

End Sub

Kommen wir zum Suchen in Recordsets: Mit DAO können Sie die Methoden "FindFirst", "FindNext", "FindPrevious" und "FindLast" nutzen, um Datensätze mit bestimmten Inhalten zu suchen. Die Eigenschaft "NoMatch" gibt dabei Auskunft, ob ein passender Datensatz gefunden wurde oder nicht. Das folgende DAO-Beispiel durchsucht einen Recordset nach Kunden aus "Frankreich" und gibt den Firmenamen im Direktfenster aus:

Sub Test5a()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset

  Set db = CurrentDb()
  Set rs = db.OpenRecordset("Kunden", dbOpenDynaset)
  rs.FindFirst "Land= 'Frankreich'"
  Do
    Debug.Print rs("Firma")
    rs.FindNext "Land= 'Frankreich'"
  Loop Until rs.NoMatch
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Die Besonderheit: Das Recordset wird mit dem zusätzlichen Parameter "dbOpenDynaset" initialisiert, so dass auch eine Suche rückwärts möglich wäre:

.....
  rs.FindLast "Land= 'Frankreich'"
  Do
    Debug.Print rs("Firma")
    rs.FindPrevious "Land= 'Frankreich'"
  Loop Until rs.NoMatch
  rs.Close
.....

ADO kennt demgegenüber nur eine Methode "Find", kein "FindFirst", "FindNext" etc. – die Art der Suche wird hier über einen zusätzlichen Parameter gesteuert:

.....
  rs.MoveFirst
  rs.Find "Land = 'Frankreich'", 0
  While Not rs.NoMatch
    Debug.Print rs.Fields("Ort").Value
    rs.Find "Land = 'Frankreich'", 1
  Wend
.....

Dieses Beispiel sucht ausgehend vom ersten Datensatz (rs.MoveFirst) Richtung Recordset-Ende und gibt bei Übereinstimmung den Ort im Direktfenster aus. Beim ersten "Find" steht der Parameter "0" für "FindFirst", in der Schleife steht der Parameter "1" für "FindNext". Um rückwärts zu suchen, kommt die folgende Routine zum Einsatz:

.....
  rs.MoveLast
  rs.Find "Land = 'Frankreich'", 0
  While Not rs.NoMatch
    Debug.Print rs.Fields("Ort").Value
    rs.Find "Land = 'Frankreich'", 1
  Wend
.....

Hier wird ausgehend vom letzten Datensatz (rs.MoveLast) Richtung Recordset-Anfang gesucht. Beim ersten "Find" steht der Parameter "0" für "FindLast", in der Schleife steht der Parameter "1" für "FindPrevious".

In der nächsten Ausgabe erläutern wir, wie über Recordsets Feldinhalte geändert und Datensätze hinzugefügt oder gelöscht werden können.

Teil 3
Versionen: Access 2013, 2010, 2007, 2003 und 2002

Um programmatisch auf die Inhalte von Tabellen zuzugreifen, werden "Recordsets" eingesetzt. Ein Recordset ist einfach ausgedrückt eine Sammlung von Datensätzen, die in Schleifen durchlaufen und auf verschiedener Art verarbeitet werden kann. Alle notwendigen Grundlagen dazu vermittelt Ihnen unser Tipp.

In den ersten beiden Teilen dieses Tipps haben wir gezeigt, wie Recordsets initialisiert werden, wie man in Recordsets navigiert, in einer Schleife alle Datensätze durchläuft und wie man Recordsets durchsucht. Im Folgenden erläutern wir, wie Sie über Recordsets Feldinhalte ändern und Datensätze hinzufügen oder löschen können.

Um einen Feldinhalt mit DAO zu ändern, sind grundsätzlich folgende Schritte notwendig:

rs.Edit: Datensatz für Bearbeitung vorbereiten
rs("Feld")= "NeuerInhalt": Neuen Inhalt zuweisen
rs.Update: Datensatz speichern

An einem konkreten Beispiel:

Sub Test6a()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Dim Anz As Long, I As Long

  Set db = CurrentDb()
  Set rs = \_
    db.OpenRecordset("Kunden", dbOpenDynaset)
  rs.MoveLast
  Anz = rs.RecordCount
  rs.MoveFirst
  For I = 1 To Anz
    rs.Edit
    rs("UmsatzAktJahr") = 0
    rs.Update
    rs.MoveNext
  Next
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Diese Routine durchläuft alle Datensätze einer Tabelle "Kunden" und setzt das Feld für den kumulierten Jahresumsatz auf "0" zurück.

Die Umsetzung mit ADO weicht davon in einigen Punkten ab: Um Feldinhalte ändern zu können, müssen Sie bei DAO zunächst wie oben gezeigt ein "rs.Edit" ausführen. Das ist bei ADO nicht notwendig. Nach der Initialisierung eines Recordsets können Sie sofort zum Beispiel per

rs("UmsatzAktJahr") = 0

eine Änderung vornehmen. Wenn Sie bei DAO nach einer Änderung per "Edit" beispielsweise mit "rs.MoveNext" auf einen anderen Datensatz positionieren, werden die Änderungen stillschweigend verworfen und sind somit verloren. ADO führt in diesen Fällen automatisch ein "Update" aus! Um zu verhindern, dass Änderungen gespeichert werden, müssen Sie bei ADO explizit die Methode "CancelUpdate" aufrufen, bevor Sie zu einem anderen Datensatz wechseln!

Eine Aktualisierung wie oben für DAO gezeigt, sieht dann in ADO zum Beispiel so aus:

Sub Test6b()
  Dim conn As New ADODB.Connection
  Dim rs As New ADODB.Recordset

  Set conn = CurrentProject.Connection
  rs.Open "Kunden", \_
           conn, \_
           adOpenKeyset, \_
           adLockOptimistic
  While Not rs.EOF
    rs("UmsatzAktJahr") = 0
    rs.MoveNext
  Wend
  rs.Close

  Set rs = Nothing
  Set conn = Nothing

End Sub

Hier kann man also auf "Edit" vor der Änderung und "Update" nach der Änderung verzichten, dafür muss aber bei "Open" ein vierter Parameter "adLockOptimistic" angegeben werden, damit das Recordset als änderbar angesehen wird.

Um neue Datensätze zu einem Recordset hinzuzufügen, verwenden Sie sowohl mit DAO als auch mit ADO die Methode "AddNew". Danach können die einzelnen Felder wie bei einer Änderung mit "rs(<Feldname>) = <NeuerInhalt>" gefüllt werden. Anschließend ist bei DAO ein "Update" notwendig, damit der neue Datensatz gespeichert wird. Hier zunächst ein DAO-Beispiel:

Sub Test7a()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset
  Dim I As Long

  Set db = CurrentDb()
  Set rs = \_
    db.OpenRecordset("Kunden", dbOpenDynaset)
  rs.AddNew
  rs("Kunden-Code") = "ABCDE"
  rs("Firma") = "Testfirma"
  .....
  rs("Telefon") = "030/123 456"
  rs.Update
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Bei ADO können Sie genauso vorgehen, es gibt allerdings die Möglichkeit, bei "AddNew" mit Parametern zu arbeiten und so direkt neue Feldinhalte zu übergeben:

Sub Test7b()
  Dim conn As New ADODB.Connection
  Dim rs As New ADODB.Recordset

  Set conn = CurrentProject.Connection
  rs.Open "Kunden", \_
           conn, \_
           adOpenKeyset, \_
           adLockOptimistic
  rs.AddNew Array("Kunden-Code", "Firma"), \_
            Array("ABCDE", "Testfirma")
  rs.Update
  rs.Close

  Set rs = Nothing
  Set conn = Nothing

End Sub

Als ersten Parameter übergeben Sie ein Array mit den Namen der Felder, deren Inhalte gesetzt werden sollen, der zweite Parameter ist ein Array der Werte, die in die betreffenden Felder zu schreiben sind.

Das Löschen eines Datensatzes erfolgt sowohl bei DAO als auch bei ADO mit der Methode "rs.Delete", hier an einem DAO-Beispiel demonstriert:

Sub Test8()
  Dim db As DAO.Database
  Dim rs As DAO.Recordset

  Set db = CurrentDb()
  Set rs = \_
    db.OpenRecordset("Artikel")
  While Not rs.EOF
    If rs("Auslaufartikel") = -1 And \_
       rs("Lagerbestand") = 0 Then
      If MsgBox("Artikel '" & \_
                rs("Artikelname") & \_
                "' löschen?", \_
                vbYesNo + vbQuestion) = vbYes Then
        rs.Delete
      End If
    End If
    rs.MoveNext
  Wend
  rs.Close

  Set rs = Nothing
  Set db = Nothing

End Sub

Es werden alle Datensätze einer Tabelle "Artikel" durchlaufen. Handelt es sich beim aktuellen Artikel um einen Auslaufartikel und ist dessen Lagerbestand "0", so wird der Datensatz nach einer Sicherheitsabfrage gelöscht.

Damit sind wir am Ende unseres Einstiegs in die Nutzung von Recordsets. Sie können nun Recordsets initialisieren, in deren Datensätzen navigieren und suchen sowie Feldinhalte ändern und Datensätze anlegen oder löschen. Spezielle Funktionen und Möglichkeiten von DAO- und ADO-Recordsets werden Inhalt zukünftiger Tipps sein.

Probleme mit DAO-Verweisen lösen
Versionen: Access 2013, 2010 und 2007

**FRAGE:** Bei der Umsetzung von Tipps oder Schritt-für-Schritt-Anleitungen habe ich bei der Aufforderung "Richten Sie einen Verweis auf 'Microsoft DAO 3.6 Object Library' ein" in einigen Datenbanken das Problem, dass die VBA-Entwicklungsumgebung nach dem Klick auf *Ok* folgende Fehlermeldung anzeigt:

Name steht im Konflikt mit vorhandenem Modul, Projekt oder vorhandener Objektbibliothek

Es ist aber kein Verweis auf DAO eingerichtet. Warum wird diese Fehlermeldung angezeigt und wie kann ich sie vermeiden?
*I. Hansen*

**ANTWORT:** In den Datenbanken, in denen Sie diese Fehlermeldung erhalten, gibt es einen Verweis auf die "Microsoft Office x.x Access Database Engine Object Library" (x.x= 15.0 bei Access 2013, 14.0 bei Access 2010, 12.0 bei Access 2007). Diese mit Access 2007 eingeführte Database Engine beinhaltet bereits eine DAO-Bibliothek, so dass ein Versuch, einen weiteren Verweis auf DAO einzurichten, zu der Fehlermeldung führt. Sie können in diesen Fällen auf den Verweis auf "Microsoft DAO 3.6 Object Library" verzichten; die Lösung wird auch ohne den Verweis funktionieren.