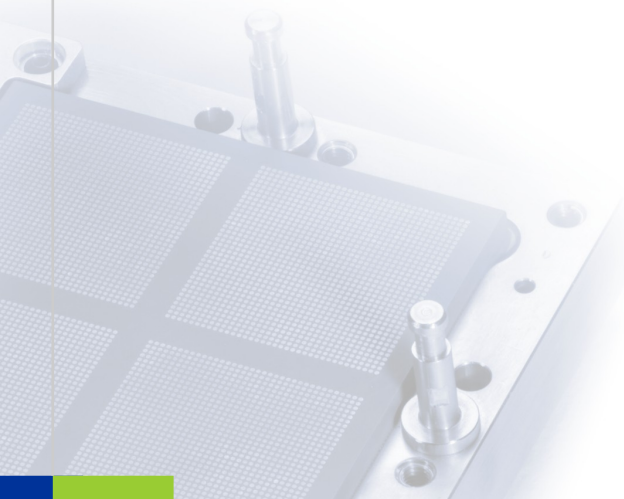


VBA (Visual Basic for Application)

D(ata) A(ccess) O(bjects)



■ DAO (Data Access Object)

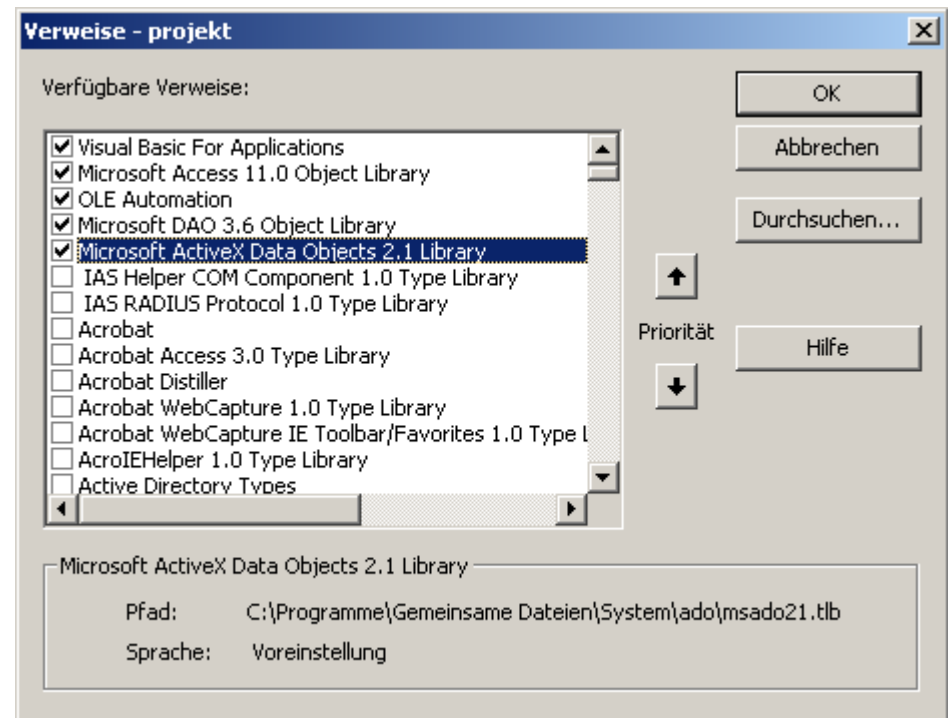
- ... wird für die Programmierung und Manipulation von Access-Datenbanken und deren Objekte genutzt.
- ... ist speziell für die Jet (Joint Engine Technologie) entwickelt.
- ... besteht aus einer großen Anzahl von Laufzeitbibliotheken (.dll).
- ... kann auf ODBC-Datenbanken seit Version 3.5 zugreifen.
- Momentan wird die Version 3.6 genutzt. Access 2007 nutzt eine Neuentwicklung, die auf diese Version aufbaut.

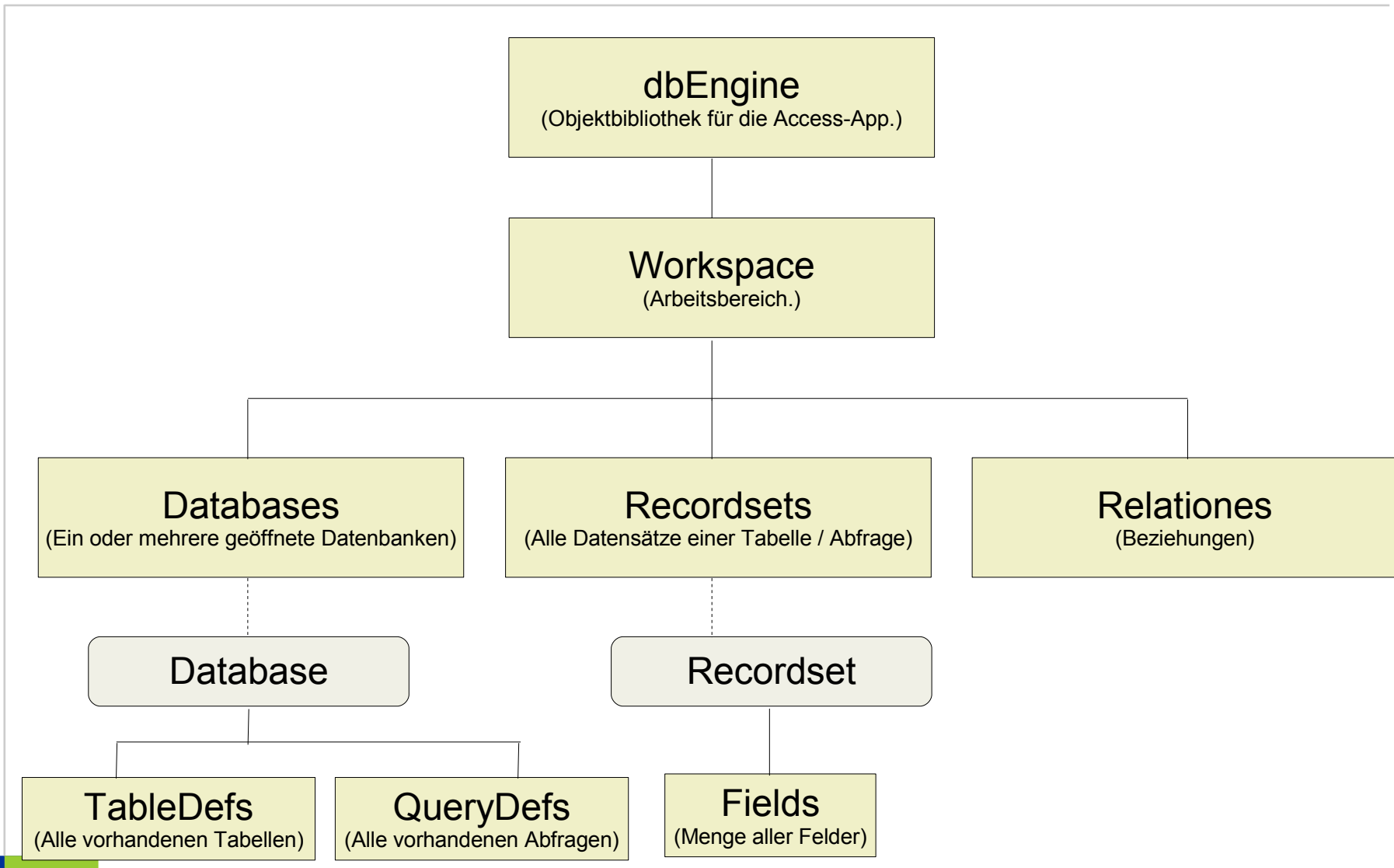
■ ADO (Active(X) Data Object)

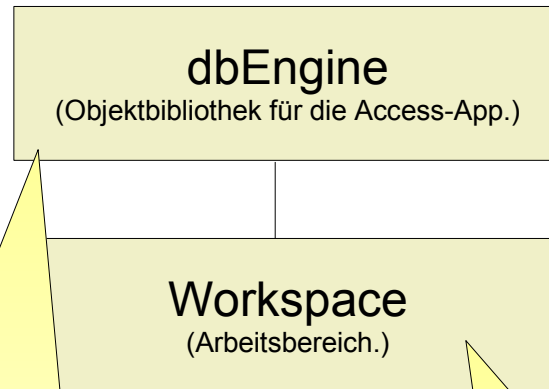
- ... ermöglicht ein Zugriff auf Datenquellen unterschiedlicher Art und Herkunft.
- ... baut auf OLE DB auf. OLE DB definiert Interfaces, die die Funktionalitäten für den Zugriff auf unterschiedliche Datenquellen kapseln.
- Access 2003 nutzt die Version ADO 2.1.

■ Vergleich unter <http://www.microsoft.com/germany/msdn/library/data/MigrationskonzepteUndDerenUmsetzung.msp?mfr=true>

- Klicken Sie auf das Menü Extras – Verweise im VBA-Editor.
- Der Verweis Microsoft *DAO 3.6 Object Library* ist aktiviert.

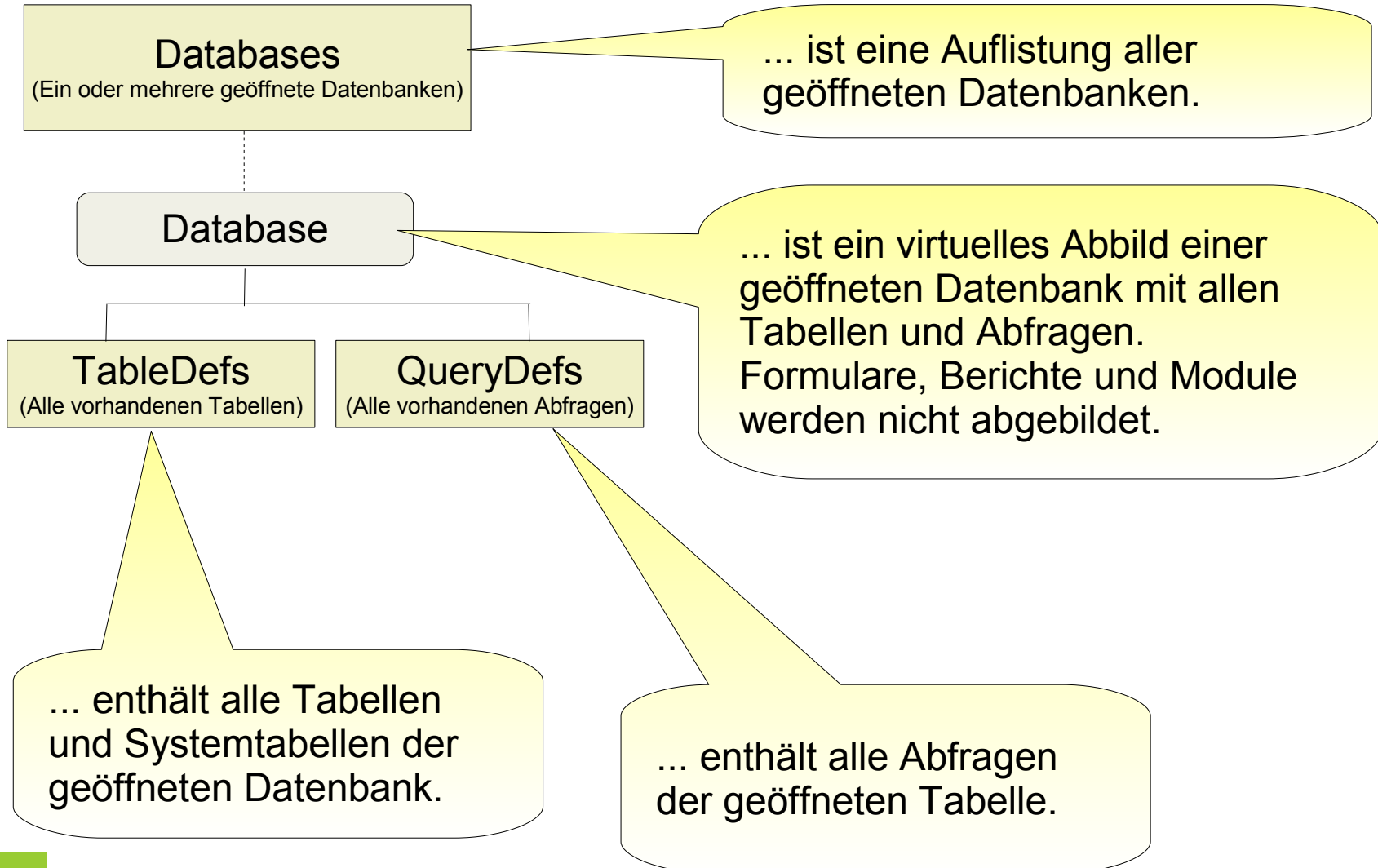


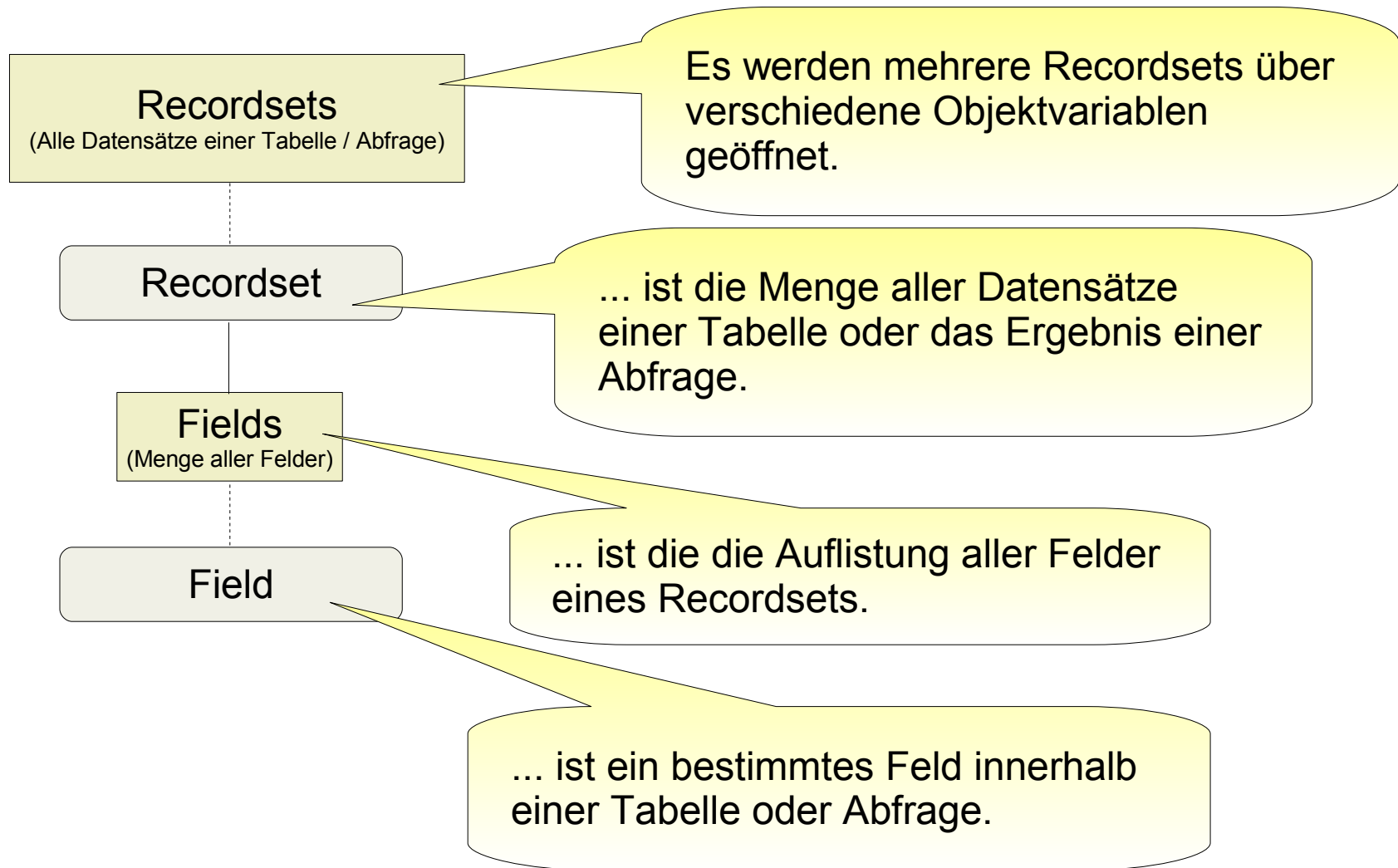




... ist die Basis für alle Datenbankobjekte in Access.
... wird von der Jet-Engine automatisch erstellt.
... stellt eine Objektbibliothek für Access-Anwendungen bereit.

... stellt den Arbeitsbereich dar. Neue Datenbanken können angelegt werden. Vorhandene Datenbanken können geöffnet und verwaltet werden.
... normalerweise wird der Arbeitsbereich der aktuellen Datenbank genutzt.





```
Dim db As Database
```

```
Set db = CurrentDb()
```

- In der ersten Zeile wird eine Variable vom Datentyp *Database* erstellt.
- Mit Hilfe von *Set* wird die Variable initialisiert. Hier wird der Variablen ein Verweis auf die aktuelle Datenbank übergeben. Falls die Verbindung zwischen Objekt und Objektvariable nicht benötigt wird, sollte diese mit Hilfe der Anweisung *Set db = Nothing* getrennt werden.
- *CurrentDb()*
 - ... öffnet eine neue aktuelle Instanz der momentan geöffneten Datenbank.
 - ... sollte, wenn möglich, nur einmal in einem Projekt einer Variablen zugewiesen werden. Anschließend sollte diese Objektvariable genutzt werden.

```
Dim db As Database
```

```
Set db = DBEngine.Workspace(0).Databases(0)
```

- Hier wird der komplette "Pfad" genutzt.
 - *DbEngine* bezeichnet das Objektmodell für Access-Anwendungen.
 - *Workspace(0)* bezeichnet den Standard-Arbeitsbereich der aktuellen Datenbank.
 - *Databases(0)* bezeichnet die aktuelle Datenbank.
- Der komplette Pfad kann auch durch die *DBEngine(0)(0)* ersetzt werden.

```
Dim db As Database
```

```
Set db = DBEngine.Workspace(0).Databases(0)
```

```
Debug.Print db.Name
```

```
Debug.Print db.Version
```

```
Debug.Print db.Properties("Build")
```

- Die Eigenschaft *Name* gibt den vollständigen Pfad der Datenbank zurück.
- Die Eigenschaft *Version* gibt über die genutzte Access-Version Auskunft.
- Die Eigenschaft *Build* gibt die Buildnummer der installierten Access-Anwendung zurück. Diese Eigenschaft kann nur über die Properties-Auflistung abgefragt werden.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Set ws = DBEngine.Workspace(0)
```

```
Set db = ws.OpenDatabase("C:\Eigene Dateien\work.mdb")
```

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Set ws = DBEngine.Workspace(0)
```

```
Set db = ws.OpenDatabase("C:\Eigen
```

Zu Anfang werden zwei Objektvariablen erstellt
Die erste Objektvariable ist vom Datentyp *Workspace*. Sie kann ein Verweis auf ein Arbeitsbereich enthalten.
Die zweite Objektvariable ist vom Datentyp *Database*. Sie kann ein Verweis auf eine Datenbank enthalten.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Set ws = DBEngine.Workspace(0)
```

```
Set db = ws.OpenDatabase("C:\Eigene Dateien")
```

Der Variablen enthält einen Verweis auf den Standard-Arbeitsbereich von Access.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Set ws = DBEngine.Workspace(0)
```

```
Set db = ws.OpenDatabase("C:\Eigene Dateien\work.mdb")
```

In diesem Beispiel wird der Objektvariablen ein Verweis auf eine beliebige Datenbank übergeben. Die Datenbank wird durch die Angabe des vollständigen Pfades identifiziert. Die existierende Datenbank selber kann geöffnet oder geschlossen sein.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
If Dir$(pfad) <> "" Then
```

```
Set db = ws.OpenDatabase(pfad)
```

```
End If
```

- Der Dir-Funktion wird ein Suchmuster übergeben. Falls ein Dateiname oder Ordnername dem Suchmuster entspricht, gibt die Funktion die gefundene Bezeichnung zurück. Andernfalls wird ein leerer String zurückgegeben.
- Die Funktion gibt keine Auskunft darüber, ob eine Datei exklusiv geöffnet werden kann oder nicht.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
If Dir$(pfad) <> "" Then
```

```
    Set db = ws.OpenDatabase(pfad, True, False)
```

```
End If
```

- Der zweite, an *OpenDataBase()* zu übergebene Parameter ist true. Das heißt, die Datenbank wird exklusiv geöffnet. Die Datenbank kann innerhalb des Netzes nur einmal geöffnet werden, alle anderen Zugriffe werden gesperrt.
- Der dritte, an *OpenDataBase()* zu übergebene Parameter ist false. Die Datenbank wird nicht schreibgeschützt geöffnet.

```
Dim ws As Workspace  
Dim db As Database  
Dim pfad As String  
  
Set ws = DBEngine.Workspace(0)  
  
pfad = Application.CurrentProject.Path & "\work.mdb"  
  
If Dir$(pfad) <> "" Then  
    Set db = ws.OpenDatabase(pfad, False, False, "MS Access; pwd=passwort")  
End If
```

- *OpenDatabase()* kann als letztes Argument Verbindungsinformationen wie Kennwörter etc übergeben werden.
- Zwischen den verschiedenen Angaben wird ein Semikolon als Trennzeichen genutzt.
- In diesem Beispiel wird die Art der Datenbank sowie ein Passwort zum Öffnen der Datenbank übergeben.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
Set db = ws.OpenDatabase(pfad)
```

```
db.Close
```

```
'Andere Möglichkeit: Set db = NOTHING
```

- Mit Hilfe der Methode *Close* wird die Verbindung zu einer Datenbank geschlossen.
- Nach dem Verlassen der Prozedur wird automatisch die Datenbankverbindung getrennt.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
Set db = ws.OpenDatabase(pfad)
```

```
Set rs = db.OpenRecordset("tblMitarbeiter", dbOpenTable)
```

In diesem Beispiel wird eine, in der geöffneten Datenbank, existierende Tabelle als Recordset genutzt. Als Argument wird der Tabellename und die Art des Recordsets übergeben.

- Ein Recordset basiert immer auf
 - ... eine Tabelle,
 - ... eine Auswahlabfrage oder
 - ... eine SQL-Anweisung.
- Das Recordset stellt immer die Ergebnismenge von Datensätzen dar.
- Mit Hilfe von *OpenRecordset()* wird ein Recordset geöffnet. Wenn kein Typ angegeben wird, wird immer versucht auf eine lokale Tabelle zuzugreifen.
- Mit Hilfe von *Close* kann ein Recordset geschlossen werden. Anschließend sollte die Variable auf *Nothing* gesetzt werden, um einen Zugriff auf eine nicht bestehende Verbindung zu verhindern.

- **dbOpenTable**
 - ... greift auf eine lokale Tabelle in der geöffneten Datenbank zu.
 - Die Daten in der Tabelle können bearbeitet, aber nicht sortiert werden.
- **dbOpenDynaset**
 - ... wird für Abfragen und verknüpfte Tabellen genutzt.
 - ... lädt einen eindeutigen Schlüssel für jeden Datensatz in den Speicher.
 - Die Daten können häufig editiert werden.
 - Die Daten können beliebig gefiltert und sortiert werden.
- **dbOpenSnapshot**
 - ... erstellt eine Kopie der Daten zu einem bestimmten Zeitpunkt.
 - Die Daten können nicht bearbeitet werden.
- **dbOpenForwardOnly**
 - ... erstellt eine Kopie der Daten zu einem bestimmten Zeitpunkt.
 - Die Daten können nur von vorn nach hinten durchgeblättert werden.
- **dbOpenDynamic**
 - ... wird für OBCD-Abreitsbereiche genutzt.

```
Set rs = db.OpenRecordset("tblMitarbeiter", dbOpenTable, dbDenyWrite)
```

- Als drittes Argument können der Funktion zum Beispiel folgende Parameter übergeben werden:
 - *dbAppendOnly*: Die Benutzer können nur neue Datensätze hinzufügen.
 - *dbDenyWrite* verhindert Datensatzänderungen in einem geöffneten Recordset in einer Mehrbenutzer-Umgebung.
 - *dbDenyRead* verhindert Lesezugriffe von anderen Nutzern in einer Mehrbenutzer-Umgebung.
 - *dbSeeChanges* löst einen Laufzeitfehler in einem Dynaset aus, wenn ein Benutzer Daten ändert, die ein anderer Benutzer momentan bearbeitet.

```
Set rs = db.OpenRecordset("tblMitarbeiter", dbOpenTable, , dbReadOnly)
```

- Als viertes Argument können der Funktion Parameter für das Arbeiten in einer Mehrbenutzer-Umgebung übergeben werden:
 - *dbReadOnly*: Die Datensätze können nur gelesen werden.
 - *dbPessimistic*: Der erste Nutzer, der auf einen Datensatz zugreift, erhält Schreibrechte. Alle nachfolgenden Nutzer können nur lesend auf die Daten zugreifen.
 - *dbOptimistic*: Es ist keine Sperrung vorhanden.

```
Set rs = db.OpenRecordset("SELECT * FROM tblMitarbeiter")
```

- In diesem Beispiel wird eine SQL-Anweisung für die Angabe eines Recordsets genutzt.
- Mit Hilfe des Schlüsselwortes *SELET* werden die Tabellenfelder ausgewählt.
 - Durch das Sternchen werden alle Tabellenfelder ausgewählt.
 - Mit Hilfe von *SELECT vorname, nachname* werden nur die angegebenen Felder ausgewählt. Die einzelnen Feldnamen werden durch Kommata getrennt.
- Das Schlüsselwort *FROM* gibt Auskunft darüber, woher die Daten kommen. In diesem Beispiel kommen die Daten aus der Tabelle *tblMitarbeiter*.

- ... werden immer von links nach rechts gelesen.
- ... ignorieren Groß- und Kleinschreibung bei Schlüsselwörtern, Tabellennamen und Spaltennamen.
- Schlüsselwörter werden in einer SQL-Anweisung standardmäßig groß geschrieben.
- Abfragen werden intern als SQL-Anweisungen interpretiert. Die SQL-Anweisung zu einer Abfrage wird in der SQL-Ansicht angezeigt.
- Zeichenketten in SQL-Anweisungen werden in Hochkommata gesetzt.
- Datumswerte werden in SQL-Anweisungen durch das Hash-Zeichen begrenzt.
- Mit Hilfe des Ausdrucks *tabelle.feld* wird ein Feld einer Tabelle zugeordnet. Der Tabellename wird vom Feldnamen mit Hilfe eines Punktes getrennt.
- Leer- oder Sonderzeichen in benutzerdefinierten Bezeichnungen wie zum Beispiel Feldnamen müssen in eckige Klammern gesetzt werden.

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'M*'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

- Mit Hilfe der Bedingung *WHERE* wird eine Bedingung angegeben.
- Hier werden alle Mitarbeiter angezeigt, die mit M beginnen.

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'Meier'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'M*er'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'Me?er'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE tblArtikel.artikelPreis > 10"
```

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE"  
sqlAnweisung = sqlAnweisung & "(tblArtikel.artikelPreis > 10) AND"  
sqlAnweisung = sqlAnweisung & "(tblArtikel.artikelPreis < 100)"
```

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE"  
sqlAnweisung = sqlAnweisung & "tblArtikel.artikelPreis BETWEEN 10 AND 100"
```

```
Dim ws As Workspace  
Dim db As Database  
Dim rs As Recordset  
Dim pfad As String  
  
Set ws = DBEngine.Workspace(0)  
  
pfad = Application.CurrentProject.Path & "\work.mdb"  
Set db = ws.OpenDatabase(pfad)  
  
Set rs = db.OpenRecordset("tblMitarbeiter", dbOpenTable)  
  
If (rs.BOF AND rs.EOF) Then  
    MsgBox("Es sind keine Datensätze vorhanden")  
Else  
    MsgBox("Es sind x Datensätze vorhanden")  
End If
```

BOF (Begin of)

Datensatz 01

Datensatz 02

Datensatz 03

Datensatz 04

Datensatz 05

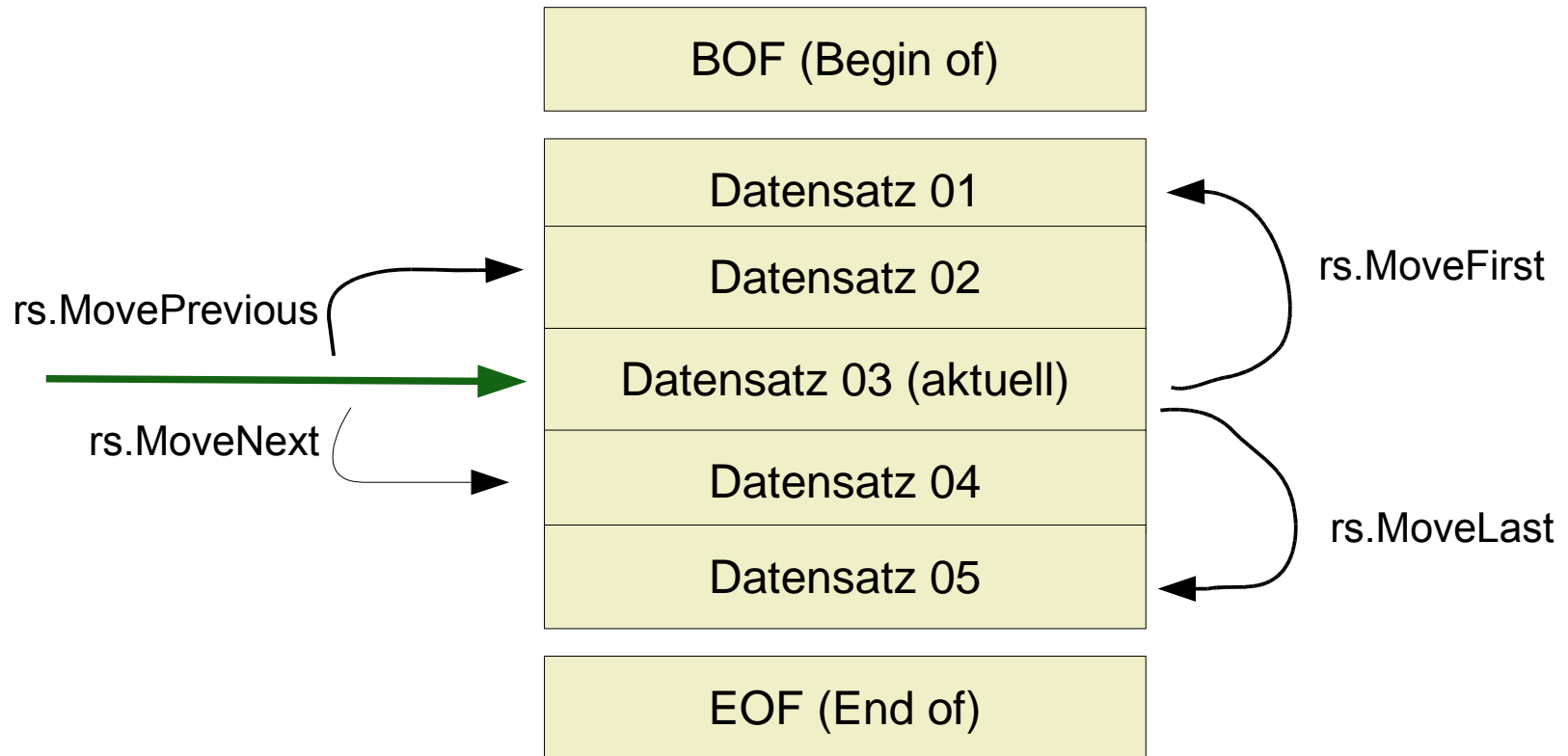
EOF (End of)

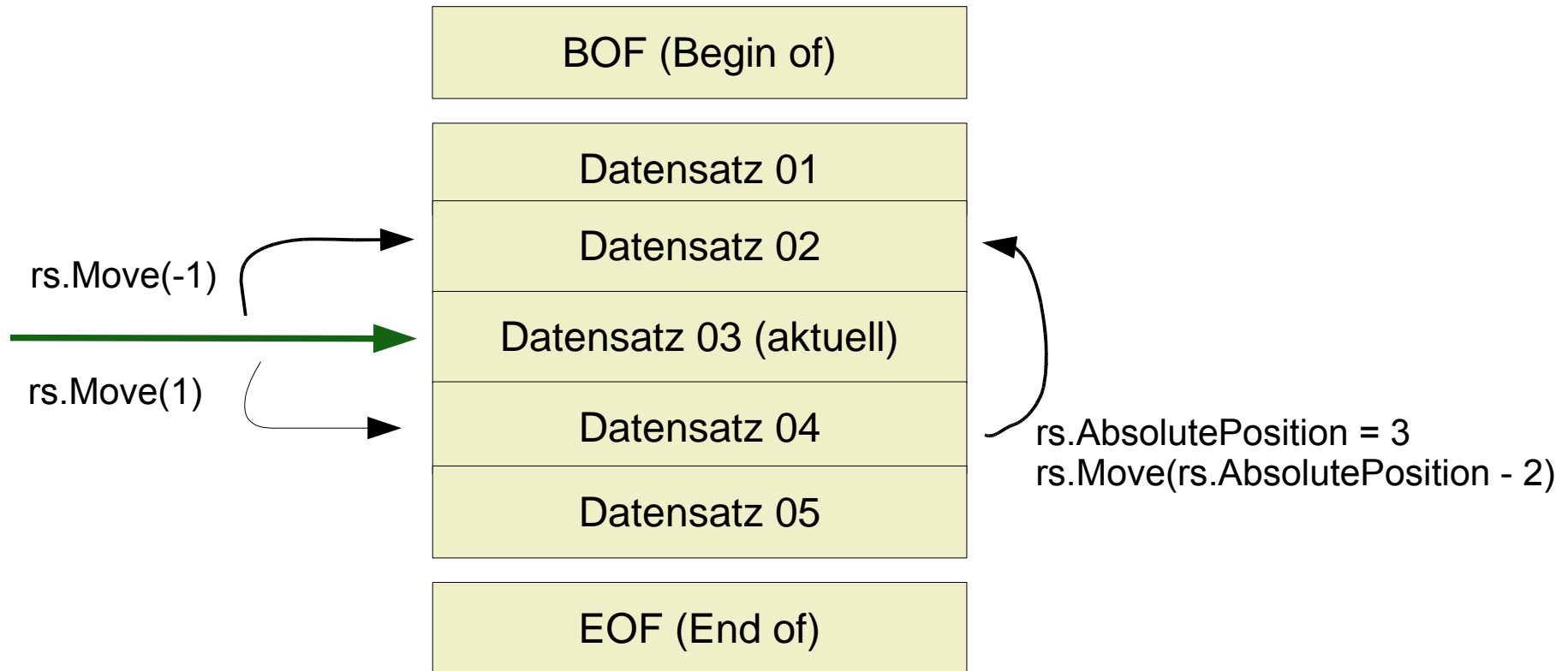
```
Dim ws As Workspace  
Dim db As Database  
Dim rs As Recordset  
Dim pfad As String  
  
Set ws = DBEngine.Workspace(0)  
  
pfad = Application.CurrentProject.Path & "\work.mdb"  
Set db = ws.OpenDatabase(pfad)  
  
Set rs = db.OpenRecordset("tblMitarbeiter", dbOpenTable)  
  
If (rs.RecordCount > 0) Then  
    MsgBox("Es sind keine Datensätze vorhanden")  
Else  
    MsgBox("Es sind x Datensätze vorhanden")  
End If
```

- Die Eigenschaft *RecordCount* liefert nur beim Typ *dbTable* die Gesamtanzahl der Datensätze zurück.
- Falls der Typ *Snapshot* oder *Dynaset* genutzt wird, wird der Wert Eins zurückgeliefert.
- Für den Typ *Snapshot* oder *Dynaset* kann mit folgenden Codeschnipsel die korrekte Anzahl von Datensätzen ermittelt werden.

```
rs.Requery  
rs.MoveLast  
Debug.Print rs.RecordCount
```

- Mit Hilfe von *.Requery* werden die Datensätze aktualisiert.
- Anschließend wird zum letzten Datensatz gesprungen.
- Durch diesen Sprung werden die Datensätze gezählt und die Eigenschaft *.RecordCount* initialisiert.





- ... kann nur beim Typ *Dynaset* oder *Snapshot* genutzt werden.
- ... positioniert den Datensatz-Zeiger neu oder gibt die aktuelle Position relativ zu 0 zurück.
- Der erste Datensatz hat den Index 0.
- ... kann keinen Wert größer als (*.RecordCount* – 1) annehmen.

```
Set rs = db.OpenRecordset("SELECT * FROM tblMitarbeiter")
```

```
With rs
```

```
  .MoveFirst
```

```
  While Not .EOF
```

```
    .MoveNext
```

```
  Wend
```

```
End With
```

```
Set rs = db.OpenRecordset("SELECT * FROM tblMitarbeiter")
```

```
With rs
```

```
  .MoveFirst
```

```
  Do While .EOF = FALSE
```

```
    .MoveNext
```

```
  Loop
```

```
End With
```

```
...  
Set rs = db.OpenRecordset("SELECT * FROM tblMitarbeiter")  
  
Debug.Print rs![nachname]  
Debug.Print rs!nachname  
  
Debug.Print rs("nachname")  
Debug.Print rs.Fields("nachname")  
  
Dim feldNachname As Field  
Set feldNachname = rs![nachname]
```

- Falls in benutzerdefinierten Bezeichnungen ein Leerzeichen oder Sonderzeichen genutzt wird, muss der Name in eckige Klammern gesetzt werden.
- Mit Hilfe von `.Fields` wird die Auflistung der Spalten eines Datensatzes angesprochen.
- Tabellenfelder mit NULL-Werten müssen vom Entwickler abgefangen werden.

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer
```

```
...
```

```
Set rs = db.OpenRecordset("SELECT * FROM tblMitarbeiter")
```

```
With rs
```

```
.Requery
```

```
.MoveLast
```

```
anzahl = .RecordCount
```

```
.MoveFirst
```

```
daten = .GetRows(anzahl)
```

```
End With
```

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer
```

```
...
```

```
Set rs = db.OpenRecordset("SELECT * FROM ...")
```

```
With rs  
  .Requery  
  .MoveLast  
  anzahl = .RecordCount  
  
  .MoveFirst  
  daten = .GetRows(anzahl)  
End With
```

Die korrekte Anzahl der Datensätze ermittelt.
Bei einer großen Anzahl von Datensätzen wird
die Laufzeit des Programms durch die
Berechnung beeinträchtigt.

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer
```

...

```
Set rs = db.OpenRecordset
```

```
With rs
```

```
.Requery
```

```
.MoveLast
```

```
anzahl = .RecordCount
```

```
.MoveFirst
```

```
daten = .GetRows(anzahl)
```

```
End With
```

.GetRows gibt eine bestimmte Anzahl von Datensätze zurück.

In diesem Beispiel werden ab dem ersten Datensatz alle Datensätze in einem Array gespeichert.

Die Daten werden in einer Variablen vom vom Datentyp *Variant* gespeichert. Diese Variable wird als Array ohne vorgegebene Dimension interpretiert.

Die Methode liest immer nur Datensätze bis zum Ende des Recordset ein. Falls eine zu große Anzahl übergeben wird, bricht die Methode ohne Meldung ab.

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer  
  
...  
  
Set rs = db.OpenRecordset("SELECT plz, ort FROM tblKunde")  
  
With rs  
    .MoveFirst  
  
    Do While .EOF = False  
        If (.Fields("ort") Like "Braunschweig") Then  
            .Edit  
            .Fields("plz") = "38100"  
            .Update  
        End If  
    Loop  
End With
```

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer  
  
...  
Set rs = db.OpenRecordset("SELEO  
  
With rs  
  .MoveFirst  
  
  Do While .EOF = False  
    If (.Fields("ort") Like "Braunschweig") Then  
      .Edit  
      .Fields("plz") = "38100"  
      .Update  
    End If  
  Loop  
End With
```

Der aktuelle Datensatz wird in ein Puffer geschrieben. Der Datensatz wird für die Bearbeitung geöffnet.

```
Dim rs As DAO.Recordset  
Dim daten As Variant  
Dim anzahl As Integer  
  
...  
  
Set rs = db.OpenRecordset("SELECT  
  
With rs  
    .MoveFirst  
  
    Do While .EOF = False  
        If (.Fields("ort") Like "Braunsel", Then  
            .Edit  
            .Fields("plz") = "38100"  
            .Update  
        End If  
    Loop  
End With
```

Der geänderte Datensatz wird in die Datenbank übernommen. Der Datensatz wird aus dem Puffer in die Tabelle geschrieben. Vor der Übernahme wird überprüft, ob die Datentypen korrekt sind und die angegebenen Gültigkeitsregeln eingehalten werden.

```
Set rs = db.OpenRecordset("SELECT IDKunde FROM tblKunde")
```

```
rs.MoveFirst
```

```
Do While rs.EOF = False
```

```
    kundeID = rs.Fields("IDKunde")
```

```
    For anzahl = 0 To Ubound(allKundeProjekt)
```

```
        if kundeID = allKundeProjekt(anzahl) Then
```

```
            rs.MoveNext
```

```
            Exit For
```

```
        End If
```

```
    Next anzahl
```

```
    If (anzahl > Ubound(allKundeProjekt)) Then
```

```
        If rs.Updatable Then
```

```
            rs.Delete
```

```
            rs.MoveNext
```

```
        End If
```

```
    End If
```

```
Loop
```

```
Set rs = db.OpenRecordset("SELECT IDKunde FROM tblKunde")
```

```
rs.MoveFirst
```

```
Do While rs.EOF = False
```

```
    kundelD = rs.Fields("IDKunde")
```

```
    For anzahl = 0 To Ubound(allKundeProjekt)
```

```
        if kundelD = allKundeProjekt(anzahl) Then
```

```
            rs.MoveNext
```

```
            Exit For
```

```
        End If
```

```
    Next anzahl
```

```
    If (anzahl > Ubound(allKundeProjekt)) Then
```

```
        If rs.Updatable Then
```

```
            rs.Delete
```

```
            rs.MoveNext
```

```
        End If
```

```
    End If
```

```
Loop
```

Wenn der Datensatz editierbar ist ...

```
Set rs = db.OpenRecordset("SELECT IDKunde FROM tblKunde")
```

```
rs.MoveFirst
```

```
Do While rs.EOF = False
```

```
    kundeID = rs.Fields("IDKunde")
```

```
    For anzahl = 0 To Ubound(allKundeProjekt)
```

```
        if kundeID = allKundeProjekt(anzahl) Then
```

```
            rs.MoveNext
```

```
            Exit For
```

```
        End If
```

```
    Next anzahl
```

```
    If (anzahl > Ubound(allKundeProjekt)) Then
```

```
        If rs.Updatable Then
```

```
            rs.Delete
```

```
            rs.MoveNext
```

```
        End If
```

```
    End If
```

```
Loop
```

... wird hier der aktuelle Datensatz gelöscht und zum nächsten Datensatz gegangen. Der Datensatz-Zeiger zeigt anschließend immer noch auf den gelöschten Datensatz. Hier wird mit Hilfe von `.MoveNext` der Zeiger auf den nächsten Datensatz gesetzt.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
Set db = ws.OpenDatabase(pfad)
```

```
Set rs = db.OpenRecordset("tblKunde")
```

```
With rs
```

```
    .AddNew
```

```
    .Fields(" firma") = "Molkerei Kuh Gut"
```

```
    .Fields("mail") = "sekretariat@molkereiKuhGut.de"
```

```
    .Update
```

```
    .Move 0, rs.LastModified
```

```
End rs
```

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb
```

```
Set db = ws.OpenDatabase(pfad)
```

```
Set rs = db.OpenRecordset("tblKunde")
```

```
With rs
```

```
  .AddNew
```

```
  .Fields(" firma") = "Molkerei Kuh Gut"
```

```
  .Fields("mail") = "sekretariat@molkereiKuhGut.de"
```

```
  .Update
```

```
  .Move 0, rs.LastModified
```

```
End rs
```

Hier wird mit Hilfe von `.AddNew` ein neuer Datensatz angelegt. Die Felder werden anschließend mit den entsprechenden Werten belegt.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
Set db = ws.OpenDatabase(pfad)
```

```
Set rs = db.OpenRecordset("tblKunde")
```

```
With rs
```

```
.AddNew
```

```
.Fields(" firma") = "Molkerei Kuh Gut"
```

```
.Fields("mail") = "sekretariat@molkereiKuhGut.de"
```

```
.Update
```

```
.Move 0, rs.LastModified
```

```
End rs
```

Der neue Datensatz wird am Ende der Tabelle eingefügt. Der Datensatzzeiger muss nicht neu gesetzt werden.

```
Dim ws As Workspace
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim pfad As String
```

```
Set ws = DBEngine.Workspace(0)
```

```
pfad = Application.CurrentProject.Path & "\work.mdb"
```

```
Set db = ws.OpenDatabase(pfad)
```

```
Set rs = db.OpenRecordset("tblKunde")
```

```
With rs
```

```
.AddNew
```

```
.Fields(" firma") = "Molkerei Kuh Gut"
```

```
.Fields("mail") = "sekretariat@molkereiKuhGut.de"
```

```
.Update
```

```
.Move 0, rs.LastModified
```

```
End rs
```

Hier wird der
Datenzeiger auf den
zuletzt geänderten
Datensatz gesetzt.

```
Private Sub Form_Current()  
  Dim rs As DAO.Recordset  
  Dim summeKM As Integer  
  Set rs = Me.subFahrten.Form.RecordsetClone  
  
  With rs  
    .MoveFirst  
    summeKM = 0  
  
    Do While Not .EOF  
      summeKM = summeKM + .Fields("Anzahl_KM")  
      .MoveNext  
    Loop  
  End With  
  
  If summeKM > 0 Then  
    Me.txtAnzahlKilometer = summeKM  
  End If  
End Sub
```

```
Private Sub Form_Current()  
  Dim rs As DAO.Recordset  
  Dim summeKM As Integer  
  Set rs = Me.subFahrten.Form.RecordsetClone  
  
  With rs  
    .MoveFirst  
    summeKM = 0  
  
    Do While Not .EOF  
      summeKM = summeKM + .Fields("AnzahlKilometer")  
      .MoveNext  
    Loop  
  End With  
  
  If summeKM > 0 Then  
    Me.txtAnzahlKilometer = summeKM  
  End If  
End Sub
```

Me ist das aktuell aktive Formular.
subFahrten ist ein Unterformular in dem aktuellen Formular. Mit Hilfe der Eigenschaft *Form* wird auf das Unterformular verwiesen.

```
Private Sub Form_Current()  
  Dim rs As DAO.Recordset  
  Dim summeKM As Integer  
  Set rs = Me.subFahrten.Form.RecordsetClone  
  
  With rs  
    .MoveFirst  
    summeKM = 0  
  
    Do While Not .EOF  
      summeKM = summeKM + .Fields("summeKM")  
      .MoveNext  
    Loop  
  End With  
  
  If summeKM > 0 Then  
    Me.txtAnzahlKilometer = summeKM  
  End If  
End Sub
```

Jedes Formular hat die Eigenschaft *RecordSetClone*, um ein Duplikat des bestehenden Recordset zu erstellen. Die Daten in dem kopierten Recordset können nur gelesen werden. Ein Schreibzugriff besteht nicht.

```
Private Sub Form_Current()  
  Dim rs As DAO.Recordset  
  Dim summeKM As Integer  
  Set rs = Me.subFahrten.Form.Re
```

```
  With rs  
    .MoveFirst  
    summeKM = 0
```

```
  Do While Not .EOF  
    summeKM = summeKM + .Fields("Anzahl_KM")  
    .MoveNext  
  Loop  
End With
```

```
If summeKM > 0 Then  
  Me.txtAnzahlKilometer = summeKM  
End If  
End Sub
```

Das geklonte Recordset wird durchlaufen. Die Werte des Feldes *Anzahl_KM* werden aufsummiert. Das Ergebnis wird in einem Steuerelement angezeigt.

```
Private Sub cmdFahrten_Click()  
  Const frmName = "frmMitarbeiterFahrten"  
  Dim rs As DAO.Recordset  
  
  Set rs = Forms(frmName).RecordsetClone  
  
  rs.FindFirst "IDMitarbeiter = " & Me!IDMitarbeiter.Value  
  
  If Not rs.NoMatch Then  
    Forms(frmName).Bookmark = rs.Bookmark  
  End If  
  
End Sub
```

```
Private Sub cmdFahrten_Click()  
  Const frmName = "frmMitarbeiterFahrten"  
  Dim rs As DAO.Recordset
```

Hier wird von einem Formular ein Duplikat seines Recordsets erstellt.

```
  Set rs = Forms(frmName).RecordsetClone
```

```
  rs.FindFirst "IDMitarbeiter = " & Me.IDMitarbeiter.Value
```

```
  If Not rs.NoMatch Then
```

```
    Forms(frmName).Bookmark = rs.Bookmark
```

```
  End If
```

```
End Sub
```

```
Private Sub cmdFahrten_Click()  
  Const frmName = "frmMitarbeiterFahrten"  
  Dim rs As DAO.Recordset  
  
  Set rs = Forms(frmName).RecordsetClone  
  
  rs.FindFirst "IDMitarbeiter = " & Me!IDMitarbeiter.Value  
  
  If Not rs.NoMatch Then  
    Forms(frmName).Bookmark = rs.Bookmark  
  End If  
  
End Sub
```

Innerhalb des Recordset des zweiten Formulars wird nach einem bestimmten Wert gesucht. In diesem Beispiel wird vom Anfang des Recordset der erste Datensatz gesucht, der die gleiche Mitarbeiterkennung besitzt wie im aktuellen Formular angezeigt.

- „, beginnend beim ersten Datensatz:
 - *.FindFirst* sucht den ersten übereinstimmenden Datensatz.
- „, beginnend beim letzten Datensatz:
 - *.FindLast* sucht den ersten übereinstimmenden Datensatz.
- „, beginnend beim aktuellen Datensatz:
 - *.FindPrevious* durchsucht das Recordset rückwärts.
 - *.FindNext* durchsucht das Recordset vorwärts.

```
Private Sub cmdFahrten_Click()  
  Const frmName = "frmMitarbeiterFahrten"  
  Dim rs As DAO.Recordset  
  
  Set rs = Forms(frmName).Recordset  
  
  rs.FindFirst "IDMitarbeiter = " & Me.IDMitarbeiter  
  
  If Not rs.NoMatch Then  
    Forms(frmName).Bookmark = rs.Bookmark  
  End If  
  
End Sub
```

Das Lesezeichen des Recordsets liegt auf dem aktuell gefundenen Datensatz. Diese Lesezeichen wird in das Lesezeichen des Formulars kopiert. Mit Hilfe dieser Lesezeichen wird ein Datensatz eindeutig identifiziert und kann in dem anderen Formular angezeigt werden.