Access [basics] 6/2011
www.access-basics.de
Das Access-Magazin für alle, die schnell von 0 auf 100 wollen

Access [basics]-News

In dieser Ausgabe dreht sich alles um zwei Themen. Das erste ist der Export von Daten im Textformat, das zweite das Aufteilen von Datenbanken.

Weil Artikel sich am leichtesten schreiben und sich auch toll nachvollziehen lassen, wenn ihnen ein Beispiel aus der Praxis zugrunde liegt, gingen die ersten beiden Artikel dieser Ausgabe mir ganz leicht von der Hand. Es geht um den Export von Daten in Text-Dateien, und zwar im sogenannten CSV-Format. Dieses Format zeichnet sich dadurch aus, dass die Datensätze durch Zeilenumbrüche und die einzelnen Felder durch ein Zeichen wie etwa ein Komma oder ein Semikolon voneinander getrennt werden.

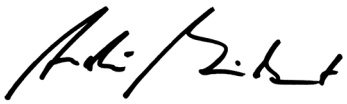
Genau dieses Format erwartet die Schnittstelle eines bekannten Versanddienstleisters. Und da bei mir gerade zum Zeitpunkt der Erstellung dieser Ausgabe eine größere Versandaktion anstand, schrieb ich auch gleich einen Artikel über das Thema. Wenn Sie erfahren möchten, wie Sie Ihre Adressdaten nach bestimmten Vorgaben exportieren, lesen Sie unter **CSV-Export für Adressdaten** weiter. Passend zu diesem Thema erfahren Sie unter dem Titel **Im- und Export-Spezifikationen definieren und verwenden**, wie Sie die Einstellungen solcher Exporte leicht speichern und dann mit einer einzigen Codezeile aufrufen können.

Der zweite Schwerpunkt dieser Ausgabe liegt im Aufteilen von Datenbanken. Gründe dafür gibt es genug: Entweder man möchte die Mehrbenutzerfähigkeiten von Access nutzen oder vielleicht auch nur die Wartbarkeit einer Datenbankanwendung verbessern. Beides gelingt, wenn Sie die Anwendung in Frontend und Backend aufteilen und vom Frontend aus über geeignete Verknüpfungen auf die im Backend gespeicherten Tabellen zugreifen.

Die Grundlagen zum Aufteilen und Verknüpfen liefert der Artikel **Datenbanken aufteilen**. Die Aufteilung einer Datenbank soll es in den meisten Fällen ermöglichen, mit mehreren Benutzern auf die Daten zuzugreifen. Was dabei zu beachten ist, erfahren Sie im Artikel **Datenbanken im Mehrbenutzerbetrieb**.

Und schließlich wird auch eine Backend-Datenbank gelegentlich verschoben. Access verfolgt den Standort von Backend-Datenbanken nicht automatisch, also müssen Sie selbst Hand anlegen. Alles Wichtige über das Neuverknüpfen aufgeteilter Datenbanken liefert der Artikel **Tabellenverknüpfungen pflegen**.

Und nun: Viel Spaß beim Lesen der neuen Ausgabe!



André Minhorst

Impressum

Access [basics] wird monatlich herausgegeben von:
André Minhorst | Fachverlag für Softwareentwicklung | Borkhofer Straße 17 | 47137 Duisburg

Redaktion:
André Minhorst (V.i.S.d.P) | Telefon: 0203/4495577 | E-Mail: info@access-basics.de | Internet: www.access-basics.de

Geschäftsführung, Herstellung, Text- und Schlussredaktion, Layout von Magazin und Webseite: André Minhorst

Autor: André Minhorst

ISSN: 2190-8761

ISSN: 2190-8761

CSV-Export für Adressdaten

Der Export von Daten ist eine sehr wichtige Funktion unter Access. Ein Export ist schnell erledigt – Sie rufen einfach den Export-Assistenten auf, tragen die notwendigen Daten ein und schreiben die Daten dann in eine entsprechende Text-, CSV- oder Excel-Datei. Im vorliegenden Beitrag wollen wir uns ansehen, welche Anforderungen es in der Praxis gibt. Dazu werfen wir einen genauen Blick auf einen Export, der genauen Spezifikationen genügen muss.

Beispieldatenbank

Die Beispiele zu diesem Artikel finden Sie in der Datenbank **1106_CSVExport.mdb**.

Zum Götterboten

In diesem Artikel geht es um den Paketversand mit Hermes. Hier soll keine Schleichwerbung gemacht werden, aber Hermes bietet eine Online-Schnittstelle, der Sie eine CSV-Datei mit Adressdaten übergeben können.

Diese Daten werden dann online gespeichert und können für den späteren Druck von Versandetiketten weiterverwendet werden.

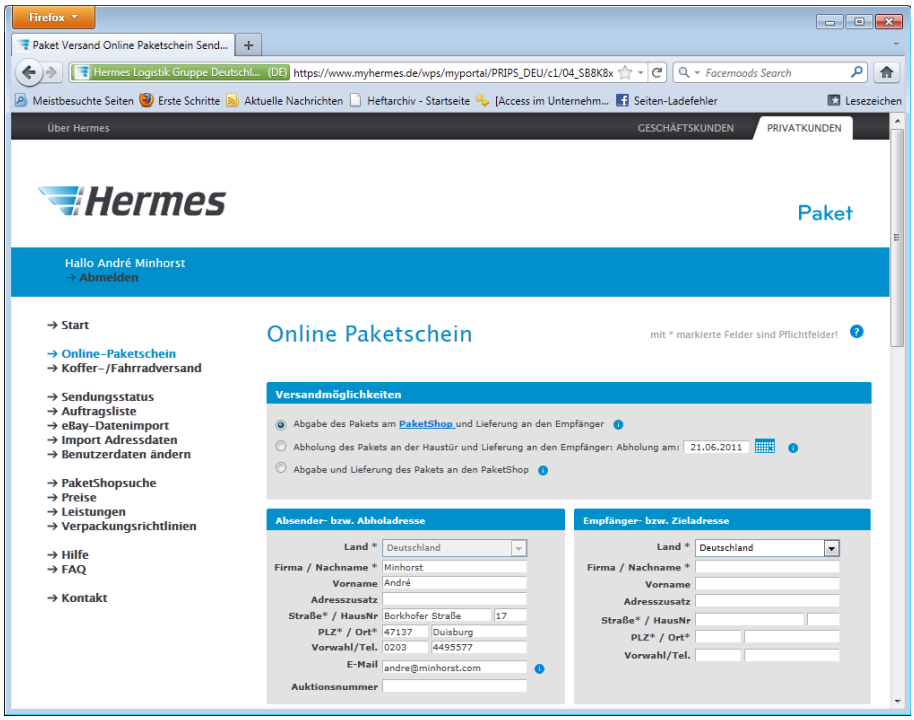


Bild 1: Beispiel für einen Online-Paketschein

Wie bei vielen anderen Anbietern können Sie bei Online-Beauftragung ein paar Cent oder gar einen Euro sparen. Sie geben dann die Empfänger- und die Absenderdaten online ein, drucken einen Paketschein aus, versehen Ihre Sendung damit und geben diese beim Postamt oder beim jeweiligen Dienstleister ab.

Im Beispiel ist dies nicht anders: Das Formular aus Bild 1 etwa ist bereits zur Hälfte vorausgefüllt, weil der Absender sich registriert und seine Adressdaten hinterlegt hat. Hier braucht man nur noch die Daten des Empfängers einzutragen, weiter unten im Formular die Paketgröße auszuwählen und zwei Häkchen zu setzen – etwa zum Akzeptieren der AGB.

Die vorliegenden Absenderdaten sparen schon einige Arbeit. Wenn Sie jedoch regelmäßig Sendungen an Ihre Kunden verschicken und deren Adressdaten in einer Datenbank vorliegen, macht es unnötig Arbeit, all diese Daten manuell in das Formular aus Bild 1 zu übertragen.

Ein Blick in die Navigationsleiste links offenbart jedoch einen Eintrag namens **Import Adressdaten**. Das hört sich gut an! Einen Klick weiter zeigt sich das Formular aus Bild 2. Hier können Sie eine Datei von Ihrer Festplatte angeben und diese zum Einlesen der enthaltenen Daten hochladen. Fehlt nur noch eine Information: Wie soll diese Datei aussehen und wie bekommen wir die vorliegenden Adressdaten aus der Datenbank dort hinein?

Import-Vorgaben

Normalerweise bieten Dienstleiter wie der aus unserem Beispiel eine Beschreibung der zu liefernden Daten an. Das diese nicht nach Gutdünken zusammengestellt werden können, liegt auf der Hand: Schließlich soll die Datei automatisch eingelesen werden und muss daher einem vorgegebenen Schema entsprechen. In diesem Fall bietet der Dienstleister zwei Varianten an – eine für nationale und eine für internationale Sendungen. Für nationale Sendungen sehen die Vorgaben so aus:

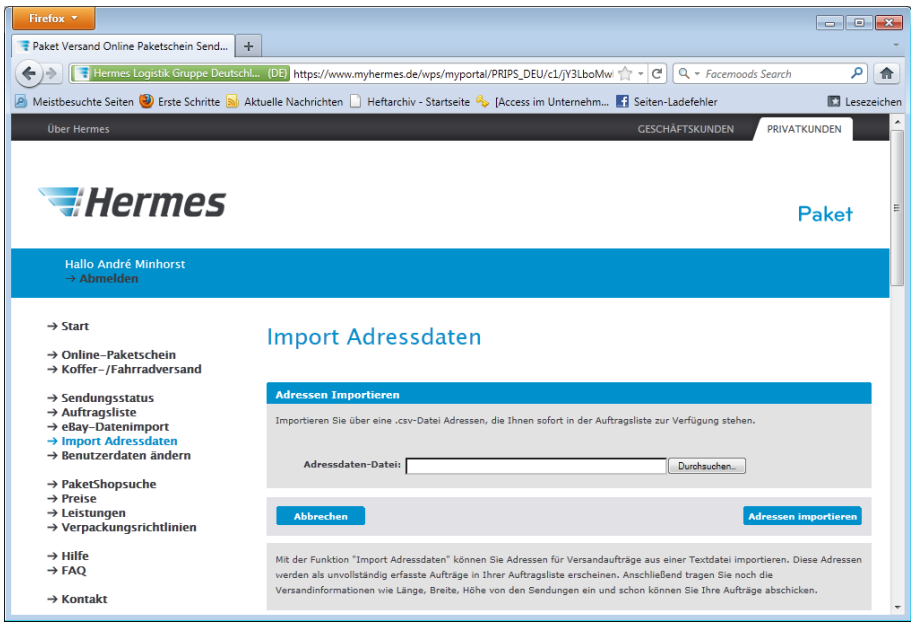


Bild 2: Schnittstelle zum Hochladen von CSV-Dateien mit Adressenlisten

- Jede Adresse landet in einer eigenen Zeile.
- Jede Zeile hat zehn Felder.
- Die Felder werden durch das Semikolon (;) getrennt, die Zeilen durch einen Zeilenumbruch.
- Es können maximal 150 Datensätze importiert werden.
- Die zehn Felder sind so aufgebaut (maximale Zeichenzahl in Klammern): **Vorname** (20), **Nachname** (25), **Adresszusatz** (20), **Straße** (27), **Hausnummer** (5), **PLZ** (5), **Ort** (25), **Tel. Vorwahl** (6), **Telefonnummer** (8), **E-Mail-Adresse** (80).

Als erstes fällt auf, dass Straße und Hausnummer getrennt angegeben werden sollen. Ups! In den bisherigen Beispieldatenbanken haben wir Straße und Hausnummer immer in einem einzigen Feld gespeichert! Aber keine Sorge: Straße und Hausnummer können auch zusammen in das Feld **Straße** eingetragen werden.

Auch für die Telefonnummer gibt es zwei Felder – aber diese ist wohl ohnehin nur den Fall gedacht, dass etwas Unvorhergesehenes mit der Lieferung geschieht und der Empfänger benachrichtigt werden muss.

Unter den zehn Feldern befindet sich keines, das einen Firmennamen aufnehmen kann – außer vielleicht das Feld **Adresszusatz**.

Ein Blick auf das Onlineformular zur Eingabe von Adressen zeigt jedoch, dass eine Firma gegebenenfalls mit im **Nachname**-Feld landet (**Firma/Nachname**).

Export vorbereiten

Die Daten zum Füttern der Online-Schnittstelle stammen aus der Tabelle **tblKunden** der Beispieldatenbank, deren Entwurf wie in Bild 3 aussieht. Das Problem bei dieser Tabelle ist, dass Sie nicht alle für den Export notwendigen Felder enthält. So können wir wie oben beschrieben nicht auf ein eigenes Feld für die Hausnummer zugreifen und auch das Feld **Adresszusatz** ist nicht vorhanden.

Außerdem sollen die Felder **Kunde** (als **Firma**) und **Nachname** in einem Feld der Exportdatei zusammengefasst werden – und das hier ebenfalls nicht der Fall. Die Lösung dieses Problems ist einfach: Sie

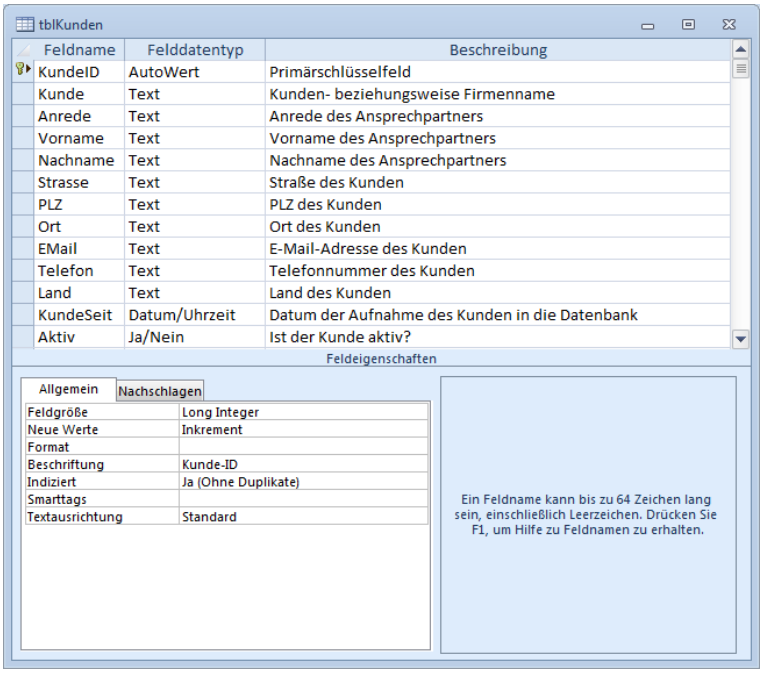


Bild 3: Diese Tabelle dient als Datenherkunft für den Export der Adressdaten.

erstellen eine Abfrage, welche die Felder der Tabelle **tblKunden** genau so ausgibt, dass daraus eine Exportdatei nach dem gewünschten Schema erstellt werden kann.

Gehen Sie dabei folgendermaßen vor:

- Erstellen Sie eine neue, leere Abfrage (**Abfrage erstellen**).
- Fügen Sie die Tabelle **tblKunden** als Datenherkunft hinzu.
- Legen Sie dann die benötigten Felder wie im Anschluss beschrieben an.

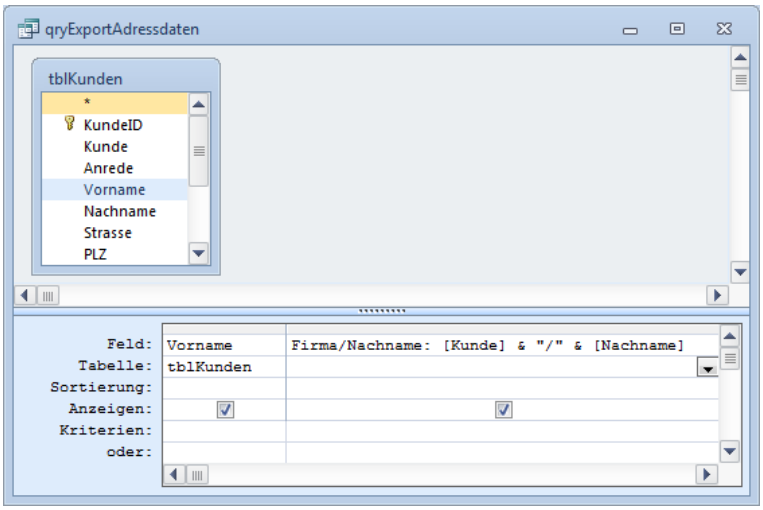


Bild 4: Entwurf der Abfrage zum Zusammenstellen der Adressdaten

Das Zusammenstellen der Felder ist die Hauptarbeit beim Export. Das erste Feld ist einfach: Den Vornamen liefert das Feld **Vorname**, also ziehen Sie dieses als erstes Feld in das Entwurfsraster.

Das Feld für **Firma/Nachname** ist gleich etwas schwieriger: Hier benötigen wir ein sogenanntes berechnetes Feld (siehe **Abfragen für die Datenauswahl – Teil VI: Berechnete Felder in Abfragen und Eigenschaften**). Zuerst legen Sie den Namen des Feldes fest, wie er in der Datenblattansicht der Abfrage erscheinen soll – in diesem Fall **Firma/Nachname**.

Dann folgt ein Doppelpunkt und schließlich der Ausdruck mit den benötigten Daten. Da die beiden Felder Kunde und Nachname zusammengeführt und durch einen Schrägstrich getrennt werden sollen, sieht der komplette Ausdruck so aus (siehe auch Bild 4):

```
Firma/Nachname: [Kunde] & "/" & [Nachname]
```

Ein Wechsel zur Datenblattansicht offenbart jedoch einige Schwächen des ersten Ansatzes (siehe Bild 5). Wenn **Kunde** und/oder **Nachname** fehlen, soll nämlich auch kein Schrägstrich angezeigt werden. Also verwenden wir die in **Access-Funktionen – Teil II: Bedingungen: Wenn(), Schalter() und Wahl()** vorgestellte **Wenn**-Funktion.

Der erste Parameter gibt den zu prüfenden Ausdruck an. Ist mindestens eines der beiden Felder **Kunde** oder **Nachname** leer? Die Prüfung sieht so aus: **IstNull([Kunde]) Oder IstNull([Nachname])**. Wenn die Bedingung eintritt, soll kein Schrägstrich ausgegeben werden, sonst ja. Also vervollständigen Sie den Ausdruck wie folgt:

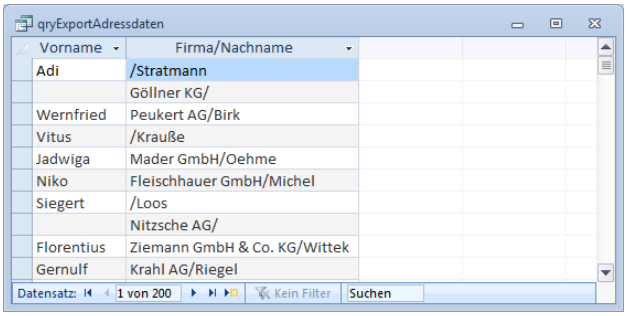


Bild 5: Der erste Versuch, Firma und Nachname zusammenzuführen

```
Firma/Nachname: [Kunde] & Wenn(IstNull([Kunde])
Oder IstNull([Nachname]);"";"/") & [Nachname]
```

Das lässt sich noch ein wenig kürzer formulieren. Dazu wenden Sie einen kleinen Trick an: Es gibt zum Verknüpfen von Texten nicht nur den Verknüpfungsoperator **&**, sondern auch das Plus-Zeichen (**+**). Wenn einer der beiden durch das Plus-Zeichen zusammengefassten Ausdrücke den Wert **Null** hat, wie es bei leeren Textfelder der Fall ist, dann ist auch die Verkettung beider Ausdrücke **Null**. Der Ausdruck sieht nun so aus:

```
Firma/Nachname: [Kunde] & Wenn(IstNull([Kunde]+[Nac
hname]);"";"/") & [Nachname]
```

Der Adresszusatz ist in unserem Fall immer leer. Dennoch muss dieses Feld im Export in Form eines leeren Feldes vorkommen. Es wird schlicht durch eine leere Zeichenkette zwischen zwei Semikola repräsentiert (**;;**). Das dritte Feld der Abfrage bezieht sich also nicht auf ein Feld der Herkunftstabelle, sondern verwendet schlicht den folgenden Ausdruck:

```
Adresszusatz: ""
```

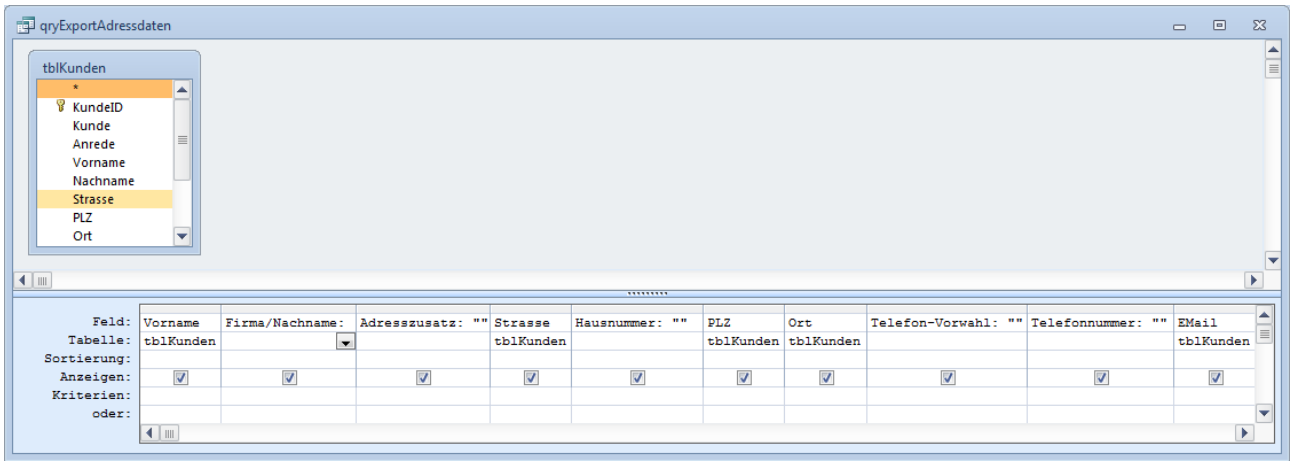


Bild 6: Die komplette Abfrage für den Datenexport

Die nächsten beiden Felder sind **Straße** und **Hausnummer**. Glücklicherweise bietet Hermes an, beide Informationen in das Feld **Straße** zu schreiben. Es ist nämlich alles andere als einfach, zuverlässig die Hausnummer aus einer entsprechenden Angabe zu ermitteln. Fügen Sie also einfach das Feld Straße zum Entwurfsraster hinzu und legen Sie für das Feld Hausnummer wieder ein Feld mit einer leeren Zeichenkette an:

```
Hausnummer: ""
```

Die Postleitzahl ist wieder einfach. Sie müssen hier lediglich sicherstellen, dass der Benutzer der Datenbank es nicht ausgenutzt hat, dass das Feld **PLZ** mit dem Datentyp **Text** definiert wurde und Einträge wie D-12345 vorgenommen hat. Hier ist nämlich die reine Postleitzahl gefragt – und außerdem darf diese ohnehin nur fünf Stellen belegen. Ziehen Sie das Feld PLZ in das Entwurfsraster – genau wie das Feld Ort, das direkt verwendet werden kann. Die beiden folgenden Felder bleiben wieder leer:

```
Telefon-Vorwahl: ""
Telefonnummer: ""
```

Mit dem Hinzufügen es Feldes **EMail** ist die Abfrage komplett (siehe Bild 6).

Export in eine CSV-Datei

Den Exportvorgang starten Sie unter den verschiedenen Access-Versionen wie folgt:

Access 2003 und älter: Klicken Sie mit der rechten Maustaste auf die zu exportierende Abfrage und wählen Sie den Eintrag **Exportieren...** aus. Wählen

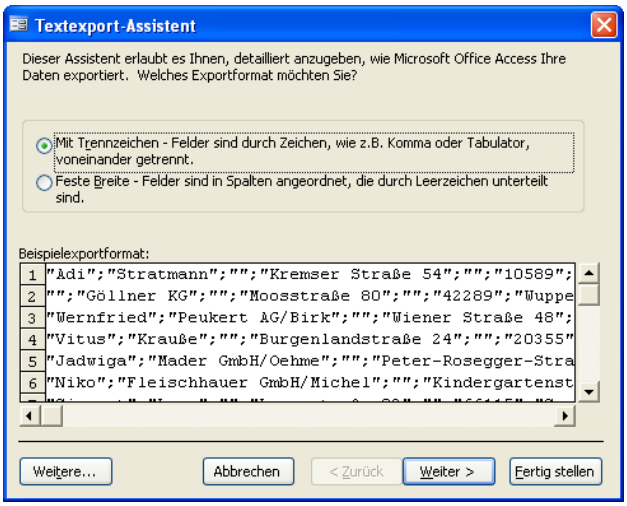


Bild 7: Trennzeichen oder feste Breite?

Sie als Dateityp den Eintrag **Textdateien** aus und geben Sie einen Dateinamen wie **ExportAdressdaten.csv** an. Es erscheint der Textexport-Assistent, mit dem Sie zunächst festlegen, dass die Ausgabe unter Verwendung von Trennzeichen erfolgen soll (siehe Bild 7). Einen Schritt weiter wählen Sie das gewünschte Trennzeichen aus – hier das Semikolon. Außerdem ist es wichtig, unter Textbegrenzungszeichen den Wert **{kein}** auszuwählen (siehe Bild 8). Im letzten Schritt bestätigen Sie noch die Zieldatei und starten den Export.

Access 2007 und jünger: Klicken Sie mit der rechten Maustaste auf die zu exportierende Abfrage im Navigationsbereich und wählen Sie den Eintrag **Exportieren|Textdatei** aus dem Kontextmenü aus. Im folgenden Dialog wählen Sie die Zieldatei aus, die übrigen Einstellungen sind nicht relevant (siehe Bild 9). Die folgenden Schritte entsprechen denen unter Access 2003 und älter.

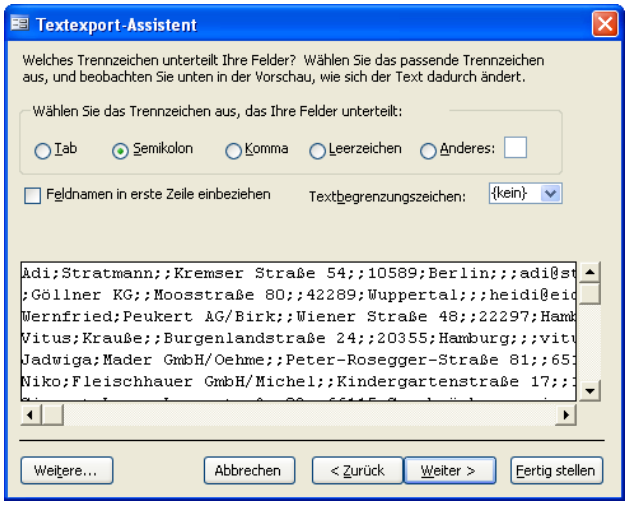


Bild 8: Welches Trennzeichen und welches Begrenzungszeichen?

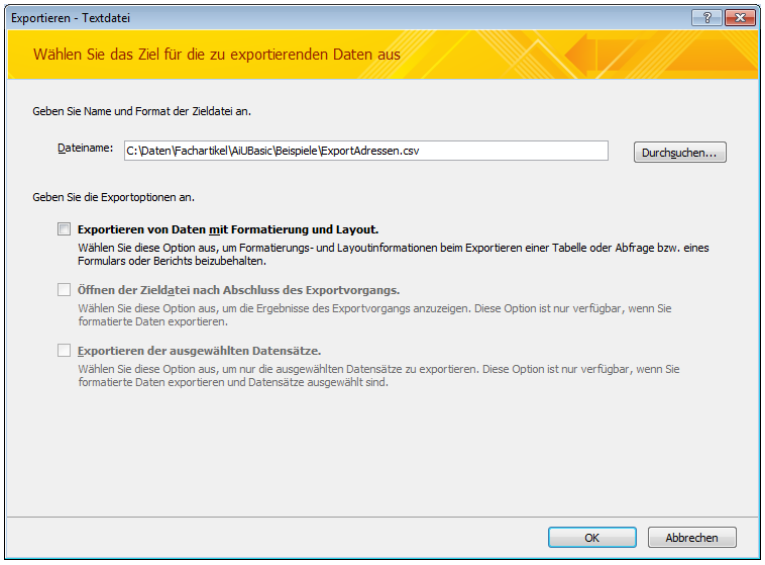


Bild 9: Festlegen der Zieldatei unter Access 2007 und jünger

Die exportierte Datei sieht im Texteditor nun wie in Bild 10 aus.

Export-Datei hochladen

Nach dem Sie die Daten in die CSV-Datei exportiert haben, laden Sie diese zum Diensleister hoch. Dazu klicken Sie zunächst auf die **Durchsuchen**-Schaltfläche des Formulars aus Bild 3. Wählen Sie die soeben erstellte Datei aus und klicken Sie danach auf den Link **Adressen importieren**.

Und wie üblich funktioniert nicht alles beim ersten Versuch: Die Webseite moniert einige zu lange Nachnamen. Dies resultiert aus dem Zusammenführen von Kunde und Nachname. Sie können die Daten nun entweder so anpassen, dass die entsprechende Spalte nur noch die ersten 25 Zeichen enthält oder die Firma in den Adresszusatz verschieben. Wir wählen die letztere Lösung.

Außerdem stellen wir sicher, dass die Daten alle die gewünschte maximale Länge aufweisen. Dies geschieht, indem wir für alle Felder eine **Left**-Funktion hinzufügen. Diese Funktion liefert nur die ersten **n** Zeichen einer Zeichenkette, wobei die Zeichenkette mit dem ersten und die Anzahl der Zeichen mit dem zweiten Parameter übergeben wird. Sie lernen diese Funktion in einem Artikel in einer späteren Ausgabe von **Access [basics]** kennen.

Das Feld zur Ausgabe des Nachnamens sieht nun beispielsweise wie folgt aus:

```
Nachname1: Links([Nachname];25)
```

Mit diesen Modifikationen funktioniert der Import der Adressdaten beim Versanddienstleister bereits (siehe Bild 11). Allerdings kann es vorkommen, dass eine Adresse keinen Vor- und Nachnamen enthält. In diesem Fall soll die Spalte mit den Nachnamen doch den Firmennamen enthalten. Dazu ändern Sie diese wie folgt:

```
Nachname1: Wenn(IstNull([Nachname]);[Kunde];Links([Nachname];25))
```

Außerdem soll der Firmenname in diesem Fall natürlich nicht mehr als Adresszusatz angezeigt werden, was Sie mit diesem Ausdruck in der Abfrage realisieren:

```
Adresszusatz: Wenn(IstNull([Nachname]);"";Links([Kunde];20))
```

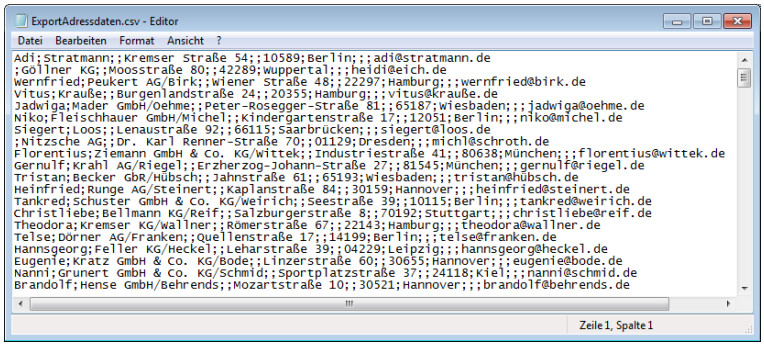


Bild 10: Die exportierten Adressen im Texteditor

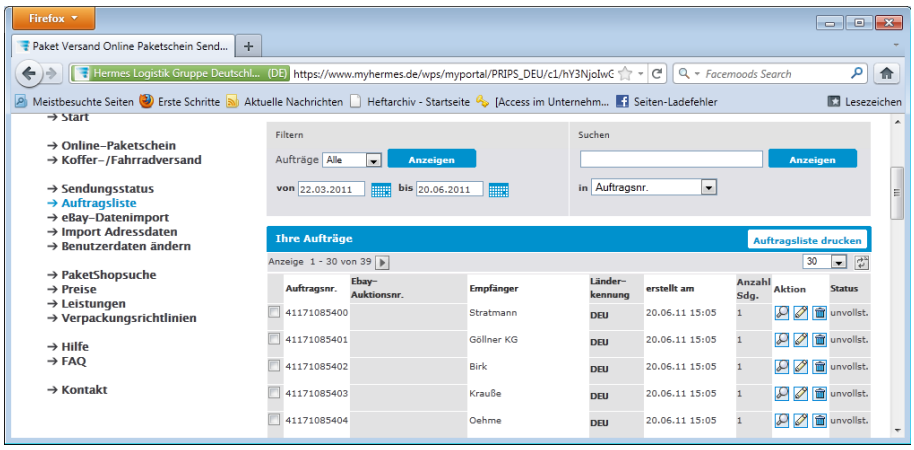


Bild 11: Die Adressen sind beim Versand-Dienstleister angekommen.

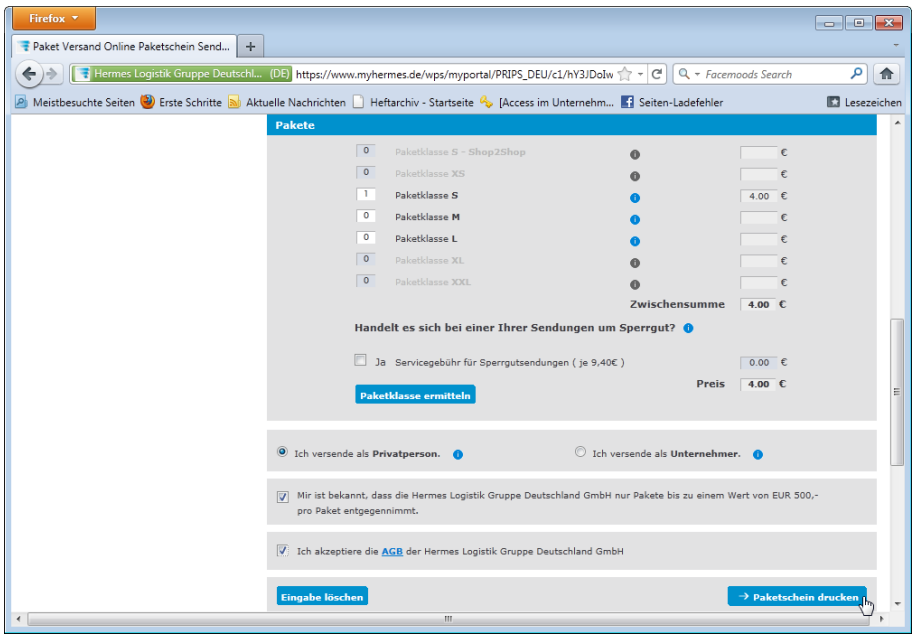


Bild 12: Ergänzen der fehlenden Informationen

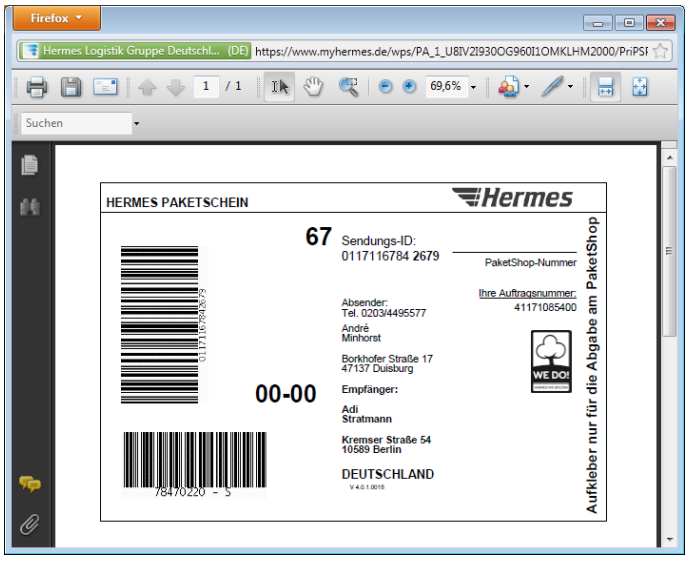


Bild 13: Der fertige Paketschein

Wenn Sie Daten einmal beim Dienstleister angekommen sind, können Sie diese dort weiterverarbeiten.

Dazu klicken Sie in diesem Fall etwa auf die Schaltfläche Auftrag bearbeiten, geben dort die fehlenden Daten wie etwa die Paketgröße ein (siehe Bild 12), haken die AGB ab und drucken schließlich den Paketschein aus (siehe Bild 13).

Zusammenfassung und Ausblick

Den Export-Vorgang können Sie natürlich automatisieren. Dazu legen Sie eine sogenannte Spezifikation an und verwenden einen speziellen VBA-Befehl, um die Daten entsprechend der Spezifikation in einer Datei zu speichern.

Wie dies funktioniert, erfahren Sie im Artikel **Im- und Export-Spezifikationen definieren und verwenden**.

Auch das Ausfüllen der fehlenden Daten im Paketschein-Formular können Sie theoretisch automatisiert von Access aus erledigen. Prinzipiell würden Sie dabei ein Formular mit einem Webbrowser-Steuerelement ausstatten, mit dem Sie Webseiten in einem Access-Formular darstellen können.

Die Access-Anwendung würde dann die entsprechende Seite des Versand-Dienstleisters aufrufen und die Eingabe der notwendigen Daten sowie das Anklicken der entsprechenden Links automatisiert durchführen. Diese Lösung würde allerdings den Rahmen dieses Artikels sprengen.

Im- und Export-Spezifikationen definieren und verwenden

Wer oft Daten mit anderen Anwendungen austauscht und dabei die Import- und Export-Funktionen von Access nutzt, arbeitet meist mit den dafür vorgesehenen Assistenten. Dumm nur, wenn man die notwendigen Einstellungen für jeden Im- und Export neu vornehmen muss. Das muss nicht sein: Access bietet die Möglichkeit, die Einstellungen als Spezifikation zu speichern und diese später auf verschiedene Arten zu nutzen.

Beispieldatenbank

Die Beispiele zu diesem Artikel finden Sie in der Datenbank **1106_Spezifikationen.mdb**.

Spezifikationen speichern

Die Assistenten für den Import und Export von Daten kennen Sie bereits aus verschiedenen Artikeln. In **Daten importieren – Teil I: Daten aus CSV-Dateien** lernen Sie etwa den Import kennen, in **CSV-Export für Adressdaten** den Export von Daten.

Die Assistenten zum Im- und Exportieren besitzen eine unscheinbare, oft übersehene Schaltfläche namens **Erweitert...**.

Ein Klick darauf öffnet einen weiteren Dialog, der nicht nur alle Einstellungen übersichtlich darstellt, sondern der auch noch die Möglichkeit zum Speichern von Spezifikationen bietet, sondern auch eine zum erneuten Öffnen bereits gespeicherter Spezifikationen (siehe Bild 1).

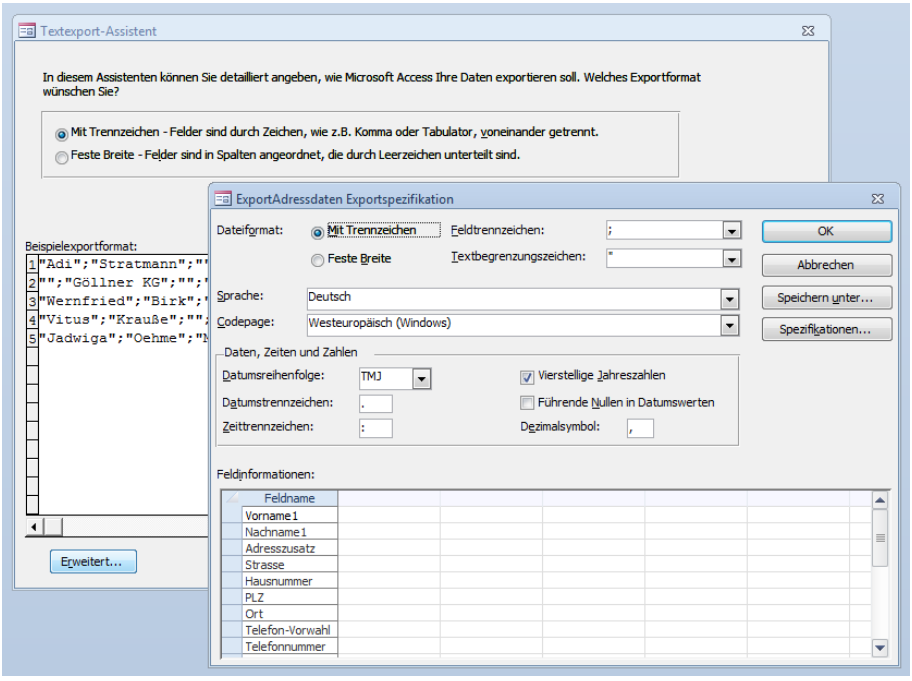


Bild 1: Der Dialog mit der Zusammenfassung der Einstellungen

Wenn Sie die gewünschten Einstellungen vorgenommen haben, klicken Sie auf die Schaltfläche **Speichern unter ...** und geben im nun erscheinenden Dialog **Import/Export-Spezifikation speichern** den Namen ein, unter dem Access die Spezifikation speichern soll.

Nachdem dies geschehen ist, klicken Sie auf die Schaltfläche **Spezifikationen ...**, um sich die Liste der verfügbaren Spezifikationen anzusehen (siehe Bild 2).

Import/Export nach Spezifikationen

Der Vorteil einer Spezifikation ist dieser: Sie können Daten damit importieren und exportieren, ohne jedesmal alle Einstellungen neu eingeben zu müssen. Ein kleiner Wermutstropfen bleibt dennoch: Sie können die Quell-/Zieldatei nicht mit der Spezifikation speichern, sondern müssen diese jedes Mal neu auswählen.

Das heißt, dass Sie beim Verwenden einer gespeicherten Spezifikation den Assistenten starten, die Quell-/Zieldatei festlegen, im Assistenten zur **Erweitert...**-Sektion wechseln und dort über die Schaltfläche Spezifikationen die gewünschte Spezifikation auswählen.

Spezifikationen anpassen

Ein weiterer Vorteil beim Einsatz von Spezifikationen ist, dass Sie diese nachträglich anpassen können. Dazu öffnen Sie diese ebenfalls über den Assistenten, laden die gewünschte Spezifikation, nehmen die Änderung vor und speichern die Spezifikation wieder.

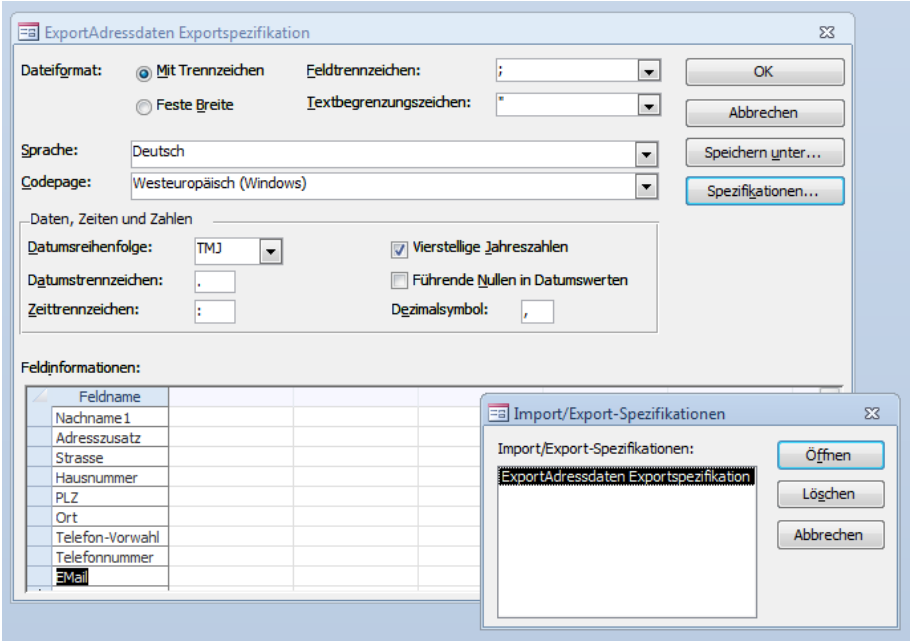


Bild 2: Öffnen oder Löschen bestehender Spezifikationen

Import/Export per VBA starten

Am meisten nützen Spezifikationen denjenigen, die regelmäßig die gleichen Import/Export-Operationen durchführen. Wenn Sie einmal einen Import oder Export als Spezifikation gespeichert haben, können Sie die Operation mit einer einzigen VBA-Zeile aufrufen.

Dazu verwenden Sie die **TransferSpreadsheet** oder die **TransferText**-Anweisung des **DoCmd**-Objekts. Lassen Sie sich von Ausdrücken wie **DoCmd**-Objekt an dieser Stelle nicht durcheinanderbringen und nehmen Sie die nachfolgend dokumentierten Befehle vorerst einfach hin.

Die **TransferText**-Anweisung dient generell dem Import-, dem Export und dem Verknüpfen von Textdateien. Die **TransferSpreadsheet**-Anweisung ist prinzipiell eine Spezialisierung von **TransferText**, denn es bezieht sich auf Excel-Tabellen und enthält noch einige auf Excel ausgelegte Besonderheiten.

Wir schauen uns an dieser Stelle jedoch nur die **TransferText**-Anweisung an. Sie können diese zu Testzwecken im Direktfenster des VBA-Editors ausprobieren.

Das Direktfenster öffnen Sie von Access aus am schnellsten mit der Tastenkombination **Strg + G**.

Wenn Sie dort **DoCmd** gefolgt von einem Punkt eingeben, bietet der VBA-Editor gleich alle möglichen Anweisungen dieses Objekts an. Das Eintippen der ersten Buchstaben der gewünschten Anweisung führt dann schnell zum Eintrag **TransferText** (siehe Bild 3).

Wenn Sie diese Anweisung eingegeben haben und noch ein Leerzeichen hinterher schicken, erscheint gleich die Parameterliste der **TransferText**-Anweisung (siehe Bild 4).

Die Anweisung verwendet die folgenden Parameter (es gibt noch zwei weitere Parameter, die aber in diesem Fall nicht zum Einsatz kommen):

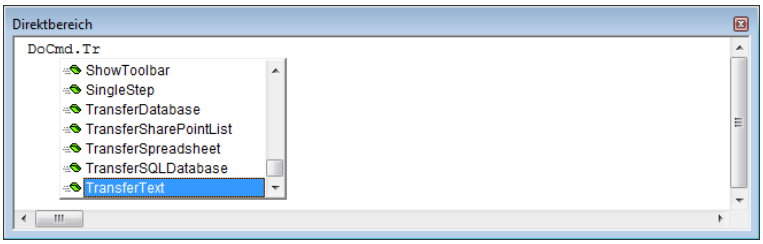


Bild 3: Der VBA-Editor unterstützt Sie bei der Eingabe von **DoCmd**-Anweisungen.

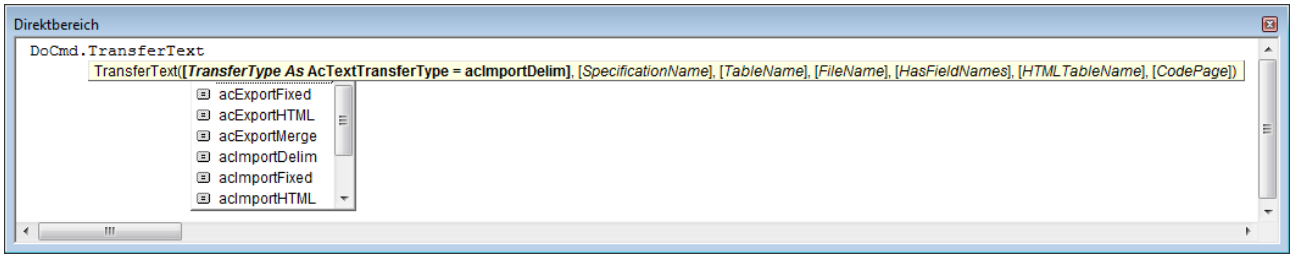


Bild 4: Auch beim Festlegen der Parameter hilft der VBA-Editor mit IntelliSense weiter.

- **TransferType**: Art des Transfers. Grundsätzlich gibt es drei Arten, nämlich Import, Export und Verknüpfen. Zu jeder dieser drei Arten fügt dieser Parameter noch die in diesem Fall wichtige Einstellung hinzu, ob die Daten in Spalten mit fester Breite untergebracht sind oder ob diese durch ein Begrenzungszeichen wie ein Semikolon et cetera voneinander getrennt werden. Die Konstante **acExportDelim** bedeutet also, dass die Daten exportiert und je Zeile durch Begrenzungszeichen getrennt werden.

- **SpecifikationName**: Erwartet den Namen einer gespeicherten Spezifikation, wie wir sie oben angelegt haben.

- **TableName**: Name der Tabelle, aus der die Daten exportiert werden sollen beziehungsweise die das Ziel eines Imports ist. Bei einer Verknüpfung tragen Sie hier den Namen ein, unter dem die Verknüpfung im Datenbankfenster/Navigationsbereich angezeigt werden soll.

- **Filename**: Pfad und Name der Ziel/Quell-Datei.

- **HasFieldNames**: Gibt an, ob beim Export die Feldnamen der Datenherkunft in die erste Zeile der Zieldatei geschrieben werden sollen oder ob beim Import der Inhalt der ersten Zeile als Feldnamen und nicht als Daten verarbeitet werden sollen.

Sollten Sie also nun eine Export-Spezifikation gespeichert haben, können Sie diese über das Direktfenster wie folgt aufrufen (in einer Zeile):

```
DoCmd.TransferText  acExportDelim, _
    "ExportAdressdaten", "qryExportAdressdaten", _
  CurrentProject.Path & "\ExportAdressdaten.csv", _
                                          False
```

Der Ausdruck **CurrentProject.Path** liefert hier das Verzeichnis, in der sich die aktuelle Datenbank befindet (wenn die Datei **c:\test\Beispiel.mdb** heißt, kommt beispielsweise **c:\text** dabei heraus).

Sie sehen hier bereits, dass Informationen wie die Quelltabelle oder die Zieldatei nicht mit der Spezifikation gespeichert werden, sondern mit der **TransferText**-Anweisung festgelegt werden müssen – genau wie es auch beim Einsatz des Assistenten geschieht. Auch dort müssen Sie die Quelltabelle oder -abfrage sowie die Zieldatei manuell auswählen. Die TransferText-Anweisung wäre also ein Fortschritt – aber nur, wenn man diese nicht jedesmal im Direktfenster des VBA-Editors eingeben muss.

Import/Export per VBA-Prozedur

Das ist natürlich auch nicht nötig: Sobald Sie einen Import oder Export einmal als Spezifikation gespeichert und die **TransferText**-Methode eingegeben haben, können Sie diese auch einer VBA-Prozedur hinzufügen.

Dazu legen Sie, sofern noch keines vorhanden ist, ein neues VBA-Modul an (**VBA-Modul anlegen**) und fügen die Anweisung in eine eigens dafür angelegte VBA-Prozedur ein. Dies sieht dann so aus:

```
Public Sub ExportKunden()
    DoCmd.TransferText acExportDelim, _
     "ExportAdressdaten", "qryExportAdressdaten", _
                            CurrentProject.Path _
              & "\ExportAdressdaten_1.csv", False
End Sub
```

Die Prozedur können Sie dann von verschiedenen Orten aus aufrufen – beispielsweise von einer **Beim Klicken**-Ereignisprozedur einer Schaltfläche aus.

TransferText flexibel aufrufen

Nun fehlt nur noch ein wenig Flexibilität. Es kommt zum Beispiel vor, dass Daten in bestimmten Zeitintervallen exportiert werden sollen. Es ist dann sinnvoll, etwa dem Dateinamen einen Hinweis auf den Zeitpunkt des Exports hinzuzufügen.

Der Inhalt der obigen Prozedur würde dann etwa so aussehen:

```
Dim strDateiname As String
strDateiname = CurrentProject.Path _
                        & "\ExportAdressdaten_" _
             & Format(Date, "yyyymmdd") & ".csv"
DoCmd.TransferText acExportDelim, _
    "ExportAdressdaten", "qryExportAdressdaten", _
                              strDateiname, False
```

Der Dateiname wird dabei noch um das Datum etwa in der Form **yyyymmdd** ergänzt. Dies liefert beispielsweise Dateinamen wie **C:\Beispiele\Export-Adressdaten_20110620.csv**.

Datenbanken aufteilen

Wer seine Daten mit einer Datenbank verwaltet, greift nicht unbedingt nur allein auf diese Daten zu. Es kommt auch vor, dass die enthaltenen Daten von mehreren Mitarbeiteren bearbeitet werden sollen – und dies unter Umständen auch noch gleichzeitig. Dieser Artikel liefert die Grundlagen zum Einsatz von Access als Datenlieferant im Mehrbenutzerbetrieb.

Beispieldatenbank

Die Beispiele zu diesem Artikel finden Sie in der Datenbank **1106_Multiuser.mdb**, **1106_Multiuser_FE.mdb** und **1106_Multiuser_BE.mdb**.

Wie alles beginnt

Es kommt nicht selten vor, dass Daten in einer Excel-Datei gespeichert werden, die von mehreren Personen bearbeitet wird. Dies erfordert bestimmte Voraussetzungen: Zunächst müssen alle beteiligten Mitarbeiter Zugriff auf die Excel-Tabelle erhalten. Dies kann man so regeln, dass diese per E-Mail von Mitarbeiter zu Mitarbeiter geschickt wird oder man legt die Excel-Datei einfach auf einem Rechner ab, der von den betroffenen Mitarbeitern erreichbar ist.

Die erste Lösung löst schnell Chaos aus: Wenn die Datei auf mehreren Rechnern liegt und hin- und hergeschickt wird, gibt es schnell parallel bearbeitete Dateien. Das Ablegen auf einem Server ist sinnvoll, wenn man direkt auf diese Datei zugreift. Sobald diese jedoch auf den eigenen Rechner kopiert wird, können theoretisch auch gleich mehrere Mitarbeiter verschiedene Versionen bearbeiten – auch das ist problematisch.

Die einzige Lösung für diesen Fall ist ein Dokumentenmanagementsystem wie etwa SharePoint: Hier darf nur jeweils ein Mitarbeiter ein Dokument auschecken und dieses bearbeiten. Erst wenn dieser das Dokument wieder eingecheckt hat, darf der nächste Mitarbeiter darauf zugreifen.

Wenn die Excel-Datei dann noch Daten enthält, auf die viele Mitarbeiter oft schreibend zugreifen müssen, sinkt die Produktivität schnell, weil immer nur eine Person gleichzeitig auf das Dokument zugreifen kann.

In solchen Fällen ist es sinnvoll, wenn irgendwo ein Access-Programmierer in der Nähe ist und den Ernst der Lage erkennt: Dies ist ein Einsatzbereich für Access! Dabei sei vorausgesetzt, dass sich die Anzahl der Benutzer und ihrer Zugriffe in einem gewissen Rahmen hält.

Nun kommt es darauf an, dass der Access-Programmierer sich auch wirklich auskennt und nicht nur ein Gelegenheits-Benutzer ist. In diesem Fall geschieht nämlich oft Folgendes: Die Daten werden aus der Excel-Datei in eine Tabelle einer Access-Datenbank übertragen.

Die Access-Datenbank landet auf einem für alle beteiligten Mitarbeiter zugänglichen Rechner, zum Beispiel einem Server. Die Mitarbeiter starten die Access-Datenbank durch einen Doppelklick auf diese Datei und sehen ihren Traum von der mehrbenutzerfähigen Datenbank verwirklicht (Access muss dazu auf allen Arbeitsplatzrechnern installliert sein).

Dies funktioniert, denn so können tatsächlich mehrere Benutzer gleichzeitig auf die Daten der Tabelle zugreifen (okay, wenn der gleiche Datensatz von mehreren Personen bearbeitet wird, gibt es je nach Einstellung Probleme – mehr dazu später). Allerdings müssen nicht nur die zu bearbeitenden Daten, sondern auch die Formulare, die Berichte und mehr auf den jeweiligen Arbeitsplatz übertragen werden.

Datenbank aufteilen

Also gehen Sie auch noch den letzten Schritt: Sie teilen die Datenbank in ein Frontend und ein Backend auf. Das Frontend enthält die Abfragen, Formulare, Berichte und Module/Makros.

Das Backend bewahrt nur noch die Tabellen mit den Daten auf. Damit das Frontend auf die im Backend gespeicherten Tabellen zugreifen kann, erstellen Sie sogenannte Verknüpfungen auf diese Tabellen. Darüber können Sie genauso auf die verknüpften Daten zugreifen als wenn Sie direkt mit den Tabellen arbeiten.

Zur Tat

Genug der Theorie: Im Download zu dieser Ausgabe finden Sie die Datenbank **1106_Multiuser.mdb**, die wir in den folgenden Schritten in eine Frontend- und eine Backend-Datenbank aufteilen werden (das Ergebnis finden Sie ebenfalls im Download).

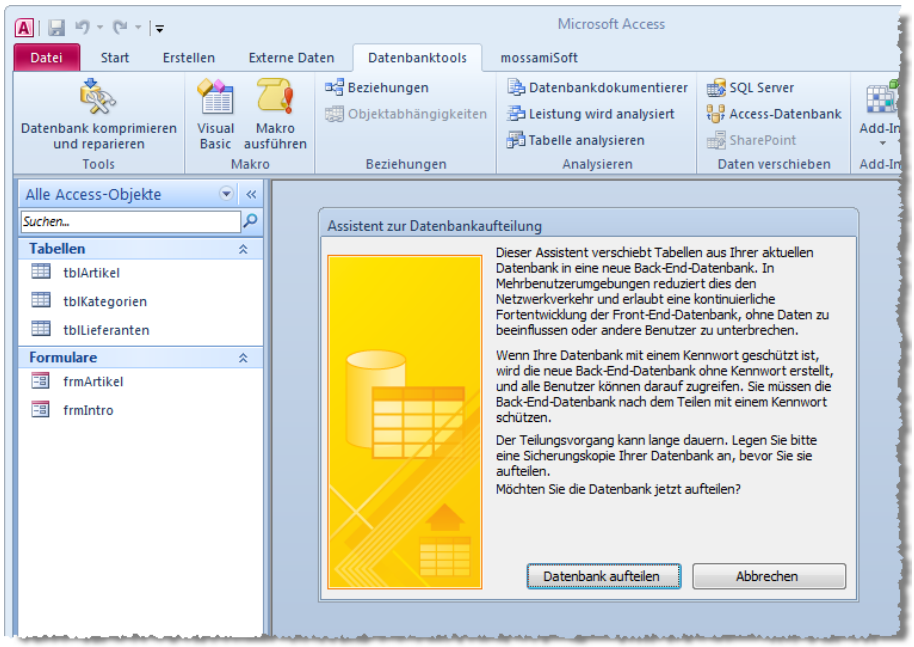


Bild 1: Start des Assistenten zur Datenbankaufteilung

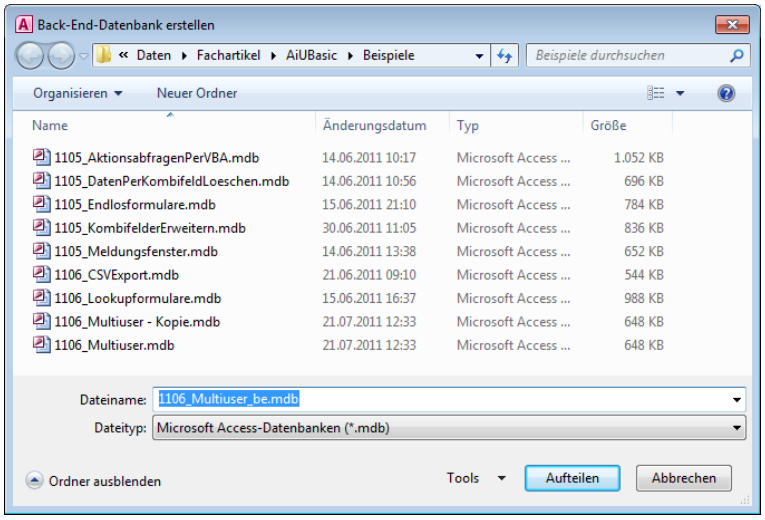


Bild 2: Festlegen des Namens der neuen Backend-Datenbank

Die Datenbank enthält die drei Tabellen **tblArtikel**, **tblKategorien** und **tblLieferanten** sowie ein Formular namens **frmArtikel** zur Anzeige dieser Daten.

Aufteilen per Assistent

Sie können die Aufteilung der Datenbank mit dem dafür vorgesehenen Assistenten durchführen oder aber selbst die notwendigen Schritte erledigen.

Den Assistenten finden Sie unter Access 2010 beispielsweise im Ribbon unter Datenbanktools|Daten verschieben|Access-Datenbank (siehe Bild 1). Unter Access 2003 und älter verbirgt sich diese Funktion hinter dem Menüeintrag **Extras|Datenbank-Dienstprogramme|Assistent zur Datenbankaufteilung**.

Der Assistent fragt im zweiten Schritt den Dateinamen der Datenbankdatei ab, welche die Tabellen aufnehmen soll, also den Part des Backends übernimmt (siehe Bild 2). In diesem Fall hängen Sie einfach die Zeichenfolge **_be** an den Dateinamen an. Die Datei soll zunächst im gleichen Verzeichnis gespeichert werden.

Das war es schon – je nach der Anzahl der Tabellen und der enthaltenen Daten meldet Access mehr oder weniger schnell Vollzug.

Aufteilen von Hand

Für spätere Arbeiten ist es unabdingbar, die einzelnen Schritte, die der Assistent im Hintergrund durchführt, zu kennen. Diese sehen kurz gefasst so aus:

- Importieren der Tabellen in einer neuen Backend-Datenbank
- Löschen der Tabellen aus der Frontend-Datenbank
- Erstellen von Verknüpfungen auf die Tabellen im Backend im Frontend

Quelle des manuell gesteuerten Aufteilens der Datenbank ist wiederum die Datei **1106_Multiuser.mdb**. Als erstes erstellen Sie eine neue Backend-Datenbank namens **1106_Multiuser_be.mdb**. Diese lassen Sie nach dem Erstellen geöffnet, um die Tabellen aus der Originaldatei zu importieren.

Den Import starten Sie je nach Access-Version wie folgt:

Access 2003 und älter: Betätigen Sie den Menüeintrag **Datei|Externe Daten|Importieren**. Wählen Sie im folgenden Dialog die Quelldatei aus, also **1106_Multiuser.mdb**. Im folgenden Dialog markieren Sie alle Tabellen, die in das Backend importiert werden

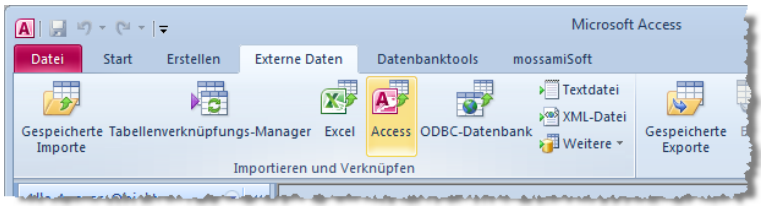


Bild 3: Start des Import unter Access 2010

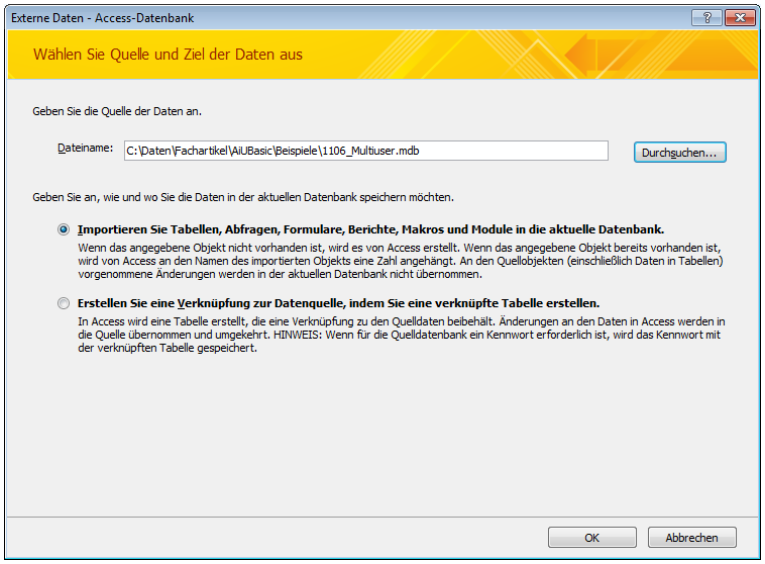


Bild 4: Importieren oder verknüpfen?

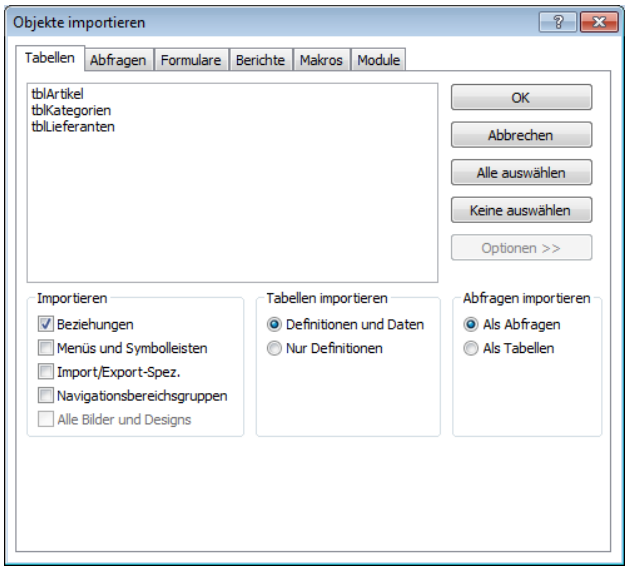


Bild 5: Festlegen der zu importierenden Tabellen

sollen. Wenn dies alle Tabellen sind, erledigen Sie dies am schnellsten mit einem Mausklick auf die Schaltfläche **Alle auswählen**. Ein weiterer Klick auf **OK** startet den Import-Vorgang.

Access 2007 und jünger: In den neueren Access-Versionen finden Sie die entsprechende Funktion im Ribbon unter **Externe Daten|Importieren und Verknüpfen|Access** (siehe Bild 3). Es erscheint der Dialog aus Bild 4, mit dem Sie zunächst die Quelldatei auswählen, dann festlegen, ob Sie Daten importieren oder verknüpfen möchten und dann mit einem Klick auf die **OK**-Schaltfläche die Aufnahme der weiteren Import-Informationen starten. Im Dialog **Objekte importieren** (siehe Bild 5) wählen Sie dann die Tabellen aus, die in das Backend übertragen werden sollen und klicken auf **OK**.

Wenn es in der Quelldatenbank Beziehungen zwischen den importierten Tabellen gab, werden diese standardmäßig auch in die neue Datenbank übernommen. Stellen Sie dazu sicher, dass die Option **Importieren|Beziehungen** aktiviert ist. Die Optionen blenden Sie versionsübergreifend mit der Schaltfläche **Optionen >>** des Dialogs **Objekte importieren** ein.

Nachdem Sie sich versichert haben, dass alle benötigten Tabellen erfolgreich in die Backend-Datenbank kopiert wurden, folgt der nächste Schritt: Sie löschen alle Tabellen aus der Quelldatenbank. Dazu markieren Sie die jeweiligen Elemente und löschen diese.

Schließlich folgt der letzte Schritt: Das Verknüpfen und somit das Einbinden der Tabellen der Backend-Datenbank in der Frontend-Datenbank.

Access 2003 und älter: Verwenden Sie den Menüeintrag **Datei|Externe Daten|Tabellen verknüpfen...**, um den Verknüpfungsvorgang zu starten. Danach wählen Sie mit dem entsprechenden Dialog die Quelldatei aus, also die soeben erstellte Backend-Datenbank. Schließlich legen Sie im folgenden Dialog die zu verknüpfenden Tabellen fest und starten das Verknüpfen mit einem Mausklick auf die Schaltfläche **OK**.

Access 2007 und jünger: Hier starten Sie das Verknüpfen wie beim Importieren über den Ribbon-Eintrag **Externe Daten|Importieren und Verknüpfen|Access**. Auch hier wählen Sie die Quelldatenbank aus, wählen dann aber die Option **Erstellen Sie eine Verknüpfung zur Datenquelle, indem Sie eine verknüpfte Tabelle erstellen**. Anschließend wählen Sie aus dem Dialog aus Bild 6 die zu verknüpfenden Tabellen aus.

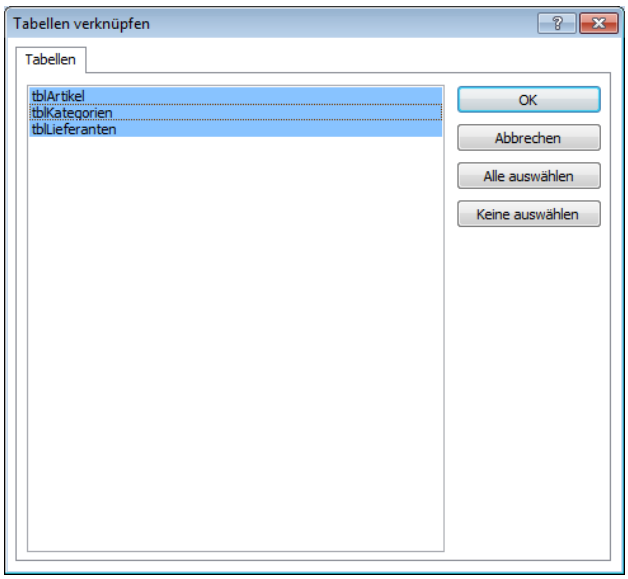


Bild 6: Festlegen der zu verknüpfenden Tabellen

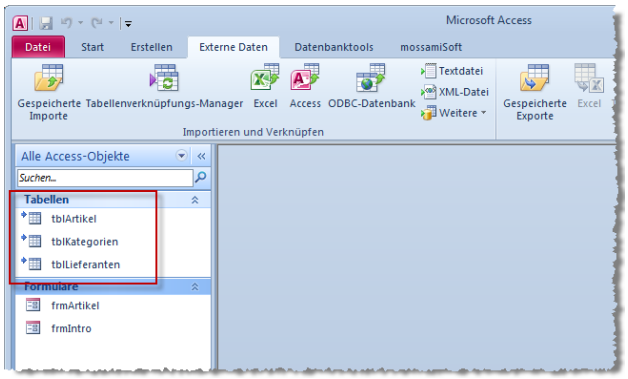


Bild 7: Verknüpfte Tabellen im Navigationsbereich

Unter Access 2007 und jünger finden Sie die Verknüpfungen dann in Form entsprechender Einträge im Bereich **Tabellen** des Navigationsbereichs vor (siehe Bild 7).

Mit einem Doppelklick auf einen dieser Einträge öffnen Sie die Datenblattansicht der jeweiligen Tabelle. Diese sieht aus wie bei einer lokal gespeicherten Tabelle und verhält sich auch genau so. Sie können vorhandene Datensätze ändern oder löschen und neue Datensätze anfügen.

Die verknüpften Tabellen dienen auch als Datenherkunft für Abfragen, Formulare und Berichte und auch per VBA greifen Sie auf die verknüpften Tabellen zu, als wenn diese in der lokalen Datenbank gespeichert wären.

Der einzige Unterschied ist, dass je nach Netzwerkverbindung die Geschwindigkeit der einzelnen Operationen leiden kann.

Zusammenfassung und Ausblick

Die Verwendung aufgeteilter Datenbanken ist in kleinen Betrieben sehr sinnvoll, weil so mehrere Benutzer auf die gleichen Daten zugreifen und diese bearbeiten können, ohne dass die Daten in Form einer Datei herumgereicht oder hin- und herkopiert werden müssen.

Es gibt noch einen anderen, sehr wichtigen Anlass zum Aufteilen einer Datenbank: Es kommt sehr selten vor, dass eine Anwendung wirklich komplett fertig entwickelt wird. Es gibt immer wieder neue Anforderungen und Wünsche, die vom Benutzer/Kunden angefragt und durch den Entwickler umgesetzt werden.

Es ist jedoch gar nicht so einfach, eine Datenbank gegen eine neue Version auszutauschen, wenn die vorherige Version bereits Daten enthält. Dies bedeutet entweder, dass Sie den aktuellen Datenbestand in die neue Version kopieren müssen oder dass Sie die Änderungen auf die in Betrieb befindliche Datenbank übertragen. Beides ist aufwendig und in der Regel fehlerbehaftet.

Wenn Sie hingegen mit einer aufgeteilten Datenbank arbeiten, können Sie in aller Ruhe am Frontend feilen, während die aktuelle Version sich im Einsatz befindet. Sind die Änderungen durchgeführt, tauschen Sie einfach das alte Frontend gegen das neue aus.

Unannehmlichkeiten können nur entstehen, wenn auch das Datenmodell und somit die im Backend gespeicherten Tabellen geändert werden. Solche Änderungen müssen Sie dann tatsächlich an Ort und Stelle durchführen – beispielsweise, indem Sie das neue Backend mit den Daten des alten Backends füllen oder die Änderungen an den Tabellen manuell oder codegesteuert durchführen (Letzteres wird irgendwann Thema in **Access [basics]** werden, aber noch nicht zu diesem Zeitpunkt).

In zwei weiteren Artikeln gehen wir auf weitere interessante Themen in Zusammenhang mit aufgeteilten Datenbanken ein: **Datenbanken im Mehrbenutzerbetrieb** und **Tabellenverknüpfungen pflegen**.

Datenbanken im Mehrbenutzerbetrieb

Wenn Sie eine Datenbank aufteilen und die Benutzer von mehreren Backends aus auf die Daten im Backend zugreifen, gibt es eine neue Herausforderung: Wie schnell sollen Änderungen an Daten auf den anderen Rechnern angezeigt werden? Was geschieht, wenn mehrere Benutzer gleichzeitig auf einen Datensatz zugreifen und diesen ändern? Diese und weitere Fragen beantwortet dieser Artikel.

Beispieldatenbanken

Die Beispieldatenbanken zu diesem Artikel heißen **1106_Multiuser_fe1.mdb**, **1106_Multiuser_fe2.mdb** und **1106_Multiuser_be.mdb**. Sie finden diese im Download zu diesem Artikel. Die Datenbanken sind das Ergebnis der Aufteilung einer Datenbank in Frontend und Backend. Details hierzu finden Sie im Artikel **Datenbanken aufteilen**.

Mehrbenutzertest

Die Mehrbenutzerfähigkeiten einer Frontend-Backend-Konstellation können Sie leicht testen. Dazu brauchen Sie noch nicht einmal mehrere Rechner. Erstellen Sie einfach ein weiteres Frontend, indem Sie die Datei **1106_Multiuser.mdb** kopieren und unter einem anderen Namen wieder einfügen. Öffnen Sie nun beide Instanzen dieser Datenbank. Ändern Sie einen Datensatz in der ersten Instanz und betrachten Sie, was mit dem gleichen Datensatz in der zweiten Instanz geschieht. Es dauert eine Weile, aber dann wird die Änderung dort automatisch angezeigt.

Anzeigeaktualisierung

Je nach Anforderungen soll die Anzeige möglichst schnell aktualisiert werden. Dies können Sie über entsprechend Optionen beeinflussen:

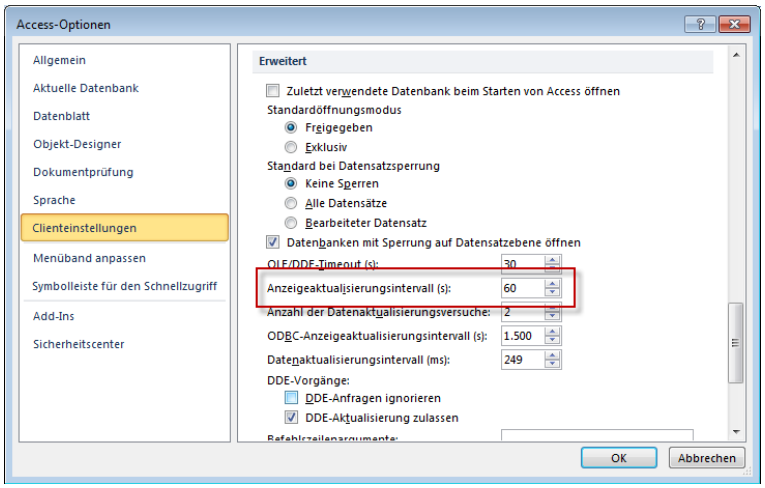


Bild 1: Einstellen des Aktualisierungsintervalls

Access 2003 und älter: Öffnen Sie den Optionen-Dialog über den Menüeintrag **Extras|Optionen** und stellen Sie die Eigenschaft **Weitere|Intervall für Anzeigeaktualisierung (s)** auf die Anzahl Sekunden ein, nach der die Daten erneut abgefragt werden sollen. Wenn Sie hier beispielsweise eine **1** eintragen, wird die Anzeige jede Sekunde aktualisiert.

Access 2007: Öffnen Sie den Optionen-Dialog über die Schaltfläche **Access-Optionen** des Office-Menüs. Die gesuchte Eigenschaft finden Sie unter **Erweitert|Erweitert|Anzeigeaktualisierungsintervall (s)**.

Access 2010: Hier öffnen Sie die Access-Optionen mit dem Ribbon-Eintrag **Datei|Optionen**. Die Option lautet **Clienteinstellungen|Erweitert|Anzeigeaktualisierungsintervall (s)**. Sie finden diese in Bild 1.

Diese und weitere Einstellungen, die Sie in den Access-Optionen vornehmen, wirken sich erst nach dem Schließen und dem erneuten Öffnen der aktuellen Datenbank aus.

Gleichzeitiges Bearbeiten

Sie können das Verhalten von Access beim gleichzeitigen Bearbeiten eines Datensatzes durch zwei Benutzer beeinflussen. Standardmäßig können mehrere Benutzer den gleichen Datensatz bearbeiten, ohne das etwas geschieht. Früher oder später könnte jedoch Folgendes geschehen:

- Benutzer 1 ändert ein Feld des Datensatzes, ohne diesen zu speichern.
- Benutzer 2 ändert ebenfalls ein Feld dieses Datensatzes. Auch er speichert die Änderungen nicht.
- Benutzer 1 speichert den Datensatz (beispielsweise durch den Wechseln zu einem anderen Datensatz).
- Benutzer 2 möchte seine Änderungen ebenfalls speichern. Er enthält allerdings die Meldung aus Bild 2.

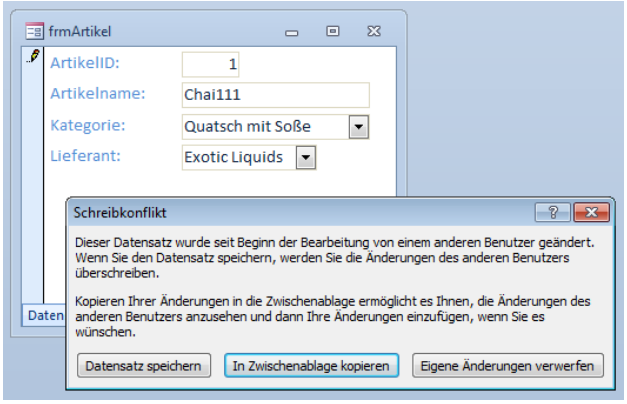


Bild 2: Meldung eines Schreibkonflikts

Diese Meldung verlangt vom Benutzer eine Entscheidung: Soll er seine Version des Datensatzes speichern und somit gegebenenfalls die Änderungen des anderen Benutzers überschreiben? Oder soll er den aktuellen Stand des durch ihn geänderten Datensatzes in die Zwischenablage kopieren, um zu prüfen, wie die Änderungen des anderen Benutzers aussehen? Oder verwirft er seine eigenen Änderungen zugunsten der Änderungen des anderen Benutzers?

Je weniger Datensätze es gibt und je mehr Benutzer diese bearbeiten, desto höher ist das Risiko eines Schreibkonflikts.

Im Idealfall bearbeitet jeder Mitarbeiter seinen eigenen Kreis von Daten, beispielsweise Bestellungen für bestimmte Kunden. Dann ist das Auftreten eines Schreibkonfliktes sehr unwahrscheinlich.

Das Zulassen von Änderungen von Datensätzen, die sich bereits in einem geänderten Zustand befinden, nennt sich übrigens **Optimistisches Sperren**. Das Gegenteil dazu erläutern die folgenden Abschnitte.

Pessimistisches Sperren

Sie können Schreibkonflikte jedoch auch auf andere Weise unterbinden. Dazu verwenden Sie eine weitere Option namens **Standard bei Datensatzsperrung**, die sich im Optionen-Dialog der verschiedenen Access-Versionen in unmittelbarer Nähe der soeben vorgestellten Option zum Einstellen der Aktualisierungsgeschwindigkeit befindet (siehe Bild 3).

Die Standardeinstellung lautet **Keine Sperren** und wirkt sich durch nachträgliche Schreibkonflikte bei gleichzeitiger Änderung des gleichen Datensatzes aus. Wenn Sie hier die Einstellung **Alle Datensätze** wählen, wird die komplette Tabelle beim Bearbeiten eines Datensatzes gesperrt, bei Bearbeiteter Datensatz nur der aktuelle Datensatz.

Aber Achtung: Die Optionen wirken sich nur auf das Erstellen neuer Formular aus! Wenn Sie also ein bestehendes Formular verwenden und die Option **Standard bei Datensatzsperrung** wählen, ändert sich die Sperrungsstrategie für Datensätze in bestehenden Formularen nicht. Sie müssen dann zunächst das betroffene Formular in der Entwurfsansicht öffnen und die Option **Daten|Datensätze sperren** wie in Bild 4 einstellen.

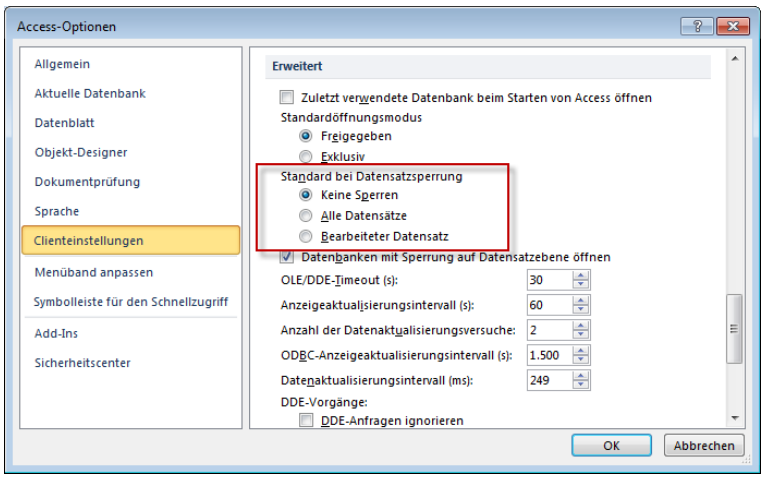


Bild 3: Arten der Sperrung von Datensätzen

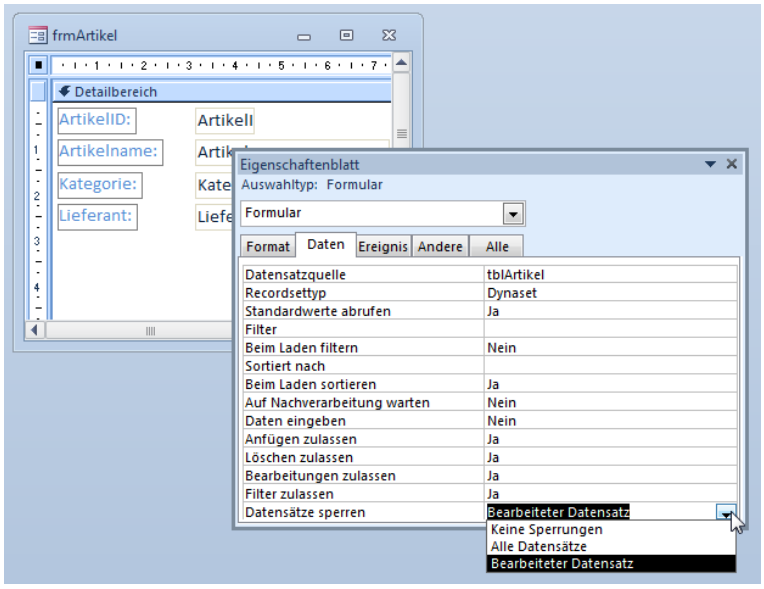


Bild 4: Sperrungsoptionen im Formularentwurf

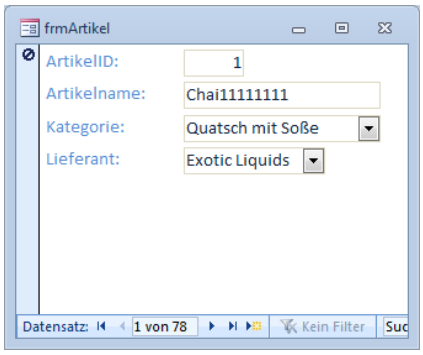


Bild 5: Sperrung eines Datensatzes, der gerade in einer anderen Instanz bearbeitet wird

Wenn der bearbeitete Datensatz gesperrt wird, kann Folgendes geschehen:

- Benutzer 1 ändert den Wert eines Feldes dieses Datensatzes.

- Benutzer 2 betrachtet gerade genau diesen Datensatz im Formular. Bei der nächsten Aktualisierung der Anzeige erscheint als Datensatzmarkierer das Gesperrt-Symbol wie in Bild 5. Spätestens aber nach der ersten Änderung des Datensatzes erscheint dieses Symbol und es ertönt ein akustisches Signal.

- Benutzer 2 kann in diesem Fall keine Änderungen am Datensatz vornehmen, bis Benutzer 1 den Datensatz gespeichert hat. Der Datensatzmarkierer zeigt dann wieder das übliche Dreieck an.

Satz- oder seitenweise Sperrung

Es gibt noch eine weitere Option, die das Sperrverhalten beeinflusst. Voraussetzung ist, dass Sie die Option **Datensätze sperren** auf **Bearbeiteter Datensatz** eingestellt haben. In diesem Fall macht sich die Option **Datenbanken mit Sperrung auf Datensatzebene öffnen** bemerkbar, die Sie nur in den Access-Optionen, nicht aber in Formularen einstellen können. Der Hintergrund ist, dass Access früher immer nur komplette Speicherseiten sperren konnte, in denen sich ein bearbeiteter Datensatz befand.

Eine Speicherseite war früher 2.048 Zeichen groß (mittlerweile 4.096 Zeichen). Wenn ein bearbeiteter Datensatz also nur Zeichen groß ist, wird dennoch die gesamte Speicherseite gesperrt, in der sich der Datensatz befindet – und somit einige benachbarte Datensätze. Sie können dies ausprobieren, indem Sie die Option **Datenbanken mit Sperrung auf Datensatzebene öffnen** aktivieren, im ersten Frontend einen Datensatz bearbeiten und dann im zweiten Frontend versuchen, den gleichen Datensatz und die umliegenden Datensätze zu editieren. Wenn Sie im ersten Frontend den ersten Datensatz der Tabelle **tblArtikel** bearbeiten, sind im zweiten Beispielfrongend immerhin rund 50 Datensätze gesperrt!

Wenn dies für Ihren Anwendungsfall nicht akzeptabel ist, aktivieren Sie die Option **Datenbanken mit Sperrung auf Datensatzebene öffnen**. In diesem Fall wird nur der tatsächlich in Bearbeitung befindliche Datensatz gesperrt.

Performance

Die verschiedenen Einstellungen wirken sich auch auf die Performance einer Access-Anwendung im Netzwerk aus. Die optimistische Sperrung ist die Einstellung mit den geringsten Einbußen bezüglich der Performance.

Bei der pessimistischen Sperrung ist in performance-kritischen Umgebungen die Variante mit der seitenweisen Sperrung vorzuziehen. Das Sperren auf Datensatzebene ist die langsamste der genannten Optionen.

Beachten Sie, dass sich das Intervall der Anzeigeaktualisierung ebenfalls auf die Performance auswirkt.

Zusammenfassung und Ausblick

Access bietet einige Alternativen für den Umgang mit Datenbanken in Mehrbenutzerumgebungen an. Welche Sie davon wählen, hängt vom konkreten Anwendungsfall ab.

Tabellenverknüpfungen pflegen

Wenn Sie eine Datenbank in Frontend und Backend aufgeteilt haben, greift das Frontend über eine Verknüpfung auf die Tabellen im Backend zu. Wo sich das Backend befindet, wird in einer Systemtabelle gespeichert. Diese wird beim Verschieben des Backends jedoch nicht automatisch aktualisiert, was zur Folge hat, dass das Frontend sein Backend und die enthaltenen verknüpften Tabellen nicht mehr findet. Dieser Artikel stellt Strategien vor, um die Verknüpfungen stets aktuell zu halten.

Beispieldatenbank

Die Beispieldatenbanken zu diesem Artikel heißen **1106_Verknuepfungen_FE.mdb** und **1106_Verknuepfungen_BE.mdb**. Sie finden diese im Download zu diesem Artikel.

Anwendungsfälle

Für das Aufteilen von Datenbanken gibt es verschiedene Anlässe. Die wichtigsten sind die bessere Möglichkeit der Wartung und der Mehrbenutzerzugriff. Eine aufgeteilte Datenbank lässt sich besser warten, weil man die Anwendungslogik, also Abfragen, Formulare, Berichte und VBA/Makros anpassen und anschließend einfach die neue Version des Frontends mit dem Backend verknüpfen kann.

Der Mehrbenutzerzugriff erlaubt es, das mehrere Benutzer mit jeweils einem auf ihrem Arbeitsplatz installiertem Frontend auf die im Backend auf dem Server gespeicherten Daten zugreifen können.

Natürlich lassen sich die verbesserten Wartungsmöglichkeiten auch in Mehrbenutzerumgebungen einsetzen.

Beim einfachen Aufteilen einer Datenbank zur besseren Wartung einer Einzelplatz-Anwendung liegen Frontend und Backend optimalerweise im gleichen Verzeichnis der lokalen Festplatte. Hier kann es vorkommen, dass Frontend- und Backend-Datenbank zusammen in ein anderes Verzeichnis oder sogar auf einen anderen Rechner übertragen werden.

Beim einer Mehrbenutzer-Datenbank liegen die Frontends jeweils auf den Arbeitsplätzen der Mitarbeiter, das Backend befindet sich auf einem Server-Rechner. Das entsprechende Netzwerkverzeichnis ist normalerweise auf dem jeweiligen Arbeitsplatzrechner gemappt. Dies geschieht entweder individuell, das heißt, das jeder Benutzer das Verzeichnis unter einem selbst gewählten Buchstaben mappt, oder jeder Benutzer verwendet standardmäßig den gleichen Pfad für den Zugriff auf den Server.

Verknüpfungen erneuern

Wann immer Sie bei einer Kombination aus Frontend- und Backend-Datenbank die Backend-Datenbank verschieben, müssen Sie die Verknüpfungen anpassen.

Daher ist Variante mit der individuellen Festlegung des Laufwerkbuchstabens die ungünstigste: Sie müssen dann auch für jedes Frontend die Verknüpfungen individuell erneuern.

Wenn alle Benutzer den gleichen Pfad verwenden, beispielsweise **e:\Datenbanken\Backend.mdb**, brauchen Sie die Verknüpfungen nur einmal auf Basis dieses Pfades zu erstellen und können die Frontend-Datenbank dann von jedem Arbeitsplatz aus problemlos einsetzen. Selbst bei einem Verschieben des Backends muss nur das Mapping aktualisiert werden.

Das Verschieben von Frontend- und Backend einer Einzelplatzlösung ist ein Zwischending: Auch hier müssen die Verknüpfungen erneuert werden, aber da beide Dateien immer im gleichen Verzeichnis liegen, lässt sich dies einfach realisieren.

Grundlagen der Verknüpfung

Die Daten einer Verknüpfung werden in der Systemtabelle **MSysObjects** gespeichert. Diese Tabelle ist standardmäßig nicht sichtbar. Um sie sichtbar zu machen, führen Sie in Abhängigkeit von der Access-Version Folgendes durch:

Access 2003 und älter: Klicken Sie auf den Menüeintrag **Extras|Optionen**. Dort finden Sie im Bereich Ansicht unter Anzeigen den Eintrag **Systemobjekte**. Aktivieren Sie diese Option und schließen Sie den Dialog wieder.

Access 2007 und jünger: Die richtige Option ist hier ein wenig schwieriger zu finden. Klicken Sie mit der rechten Maustaste auf die Titelleiste des Navigationsbereichs und wählen Sie dort den Eintrag **Naviga-**

tionsoptionen aus (siehe Bild 1). Im nun erscheinenden Dialog **Navigationsoptionen** finden Sie untern links den Eintrag **Systemobjekte** anzeigen, den Sie aktivieren (siehe Bild 2). Der Navigationsbereich beziehungsweise das Datenbankfenster zeigt nun einige Einträge in grauer Schrift an, darunter auch **MSysObjects**.

Wenn Sie diese Tabelle öffnen und die enthaltenen Spalten ein wenig umstellen, erhalten Sie eine Ansicht wie in Bild 3. Wichtig sind die Spalten **Name**, **Type** und **Database**. **Name** zeigt den Namen des jeweiligen Objekts an, **Type** den Typ (**6** steht für verknüpfte Tabellen) und **Database** liefert den Pfad zu der Datenbank, welche die verknüpften Tabellen enthält.

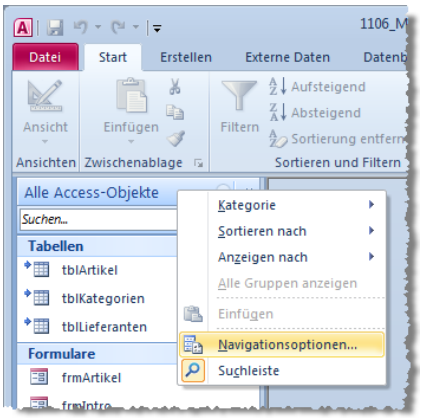


Bild 1: Aufrufen der Navigationsoptionen unter Access 2007 und jünger

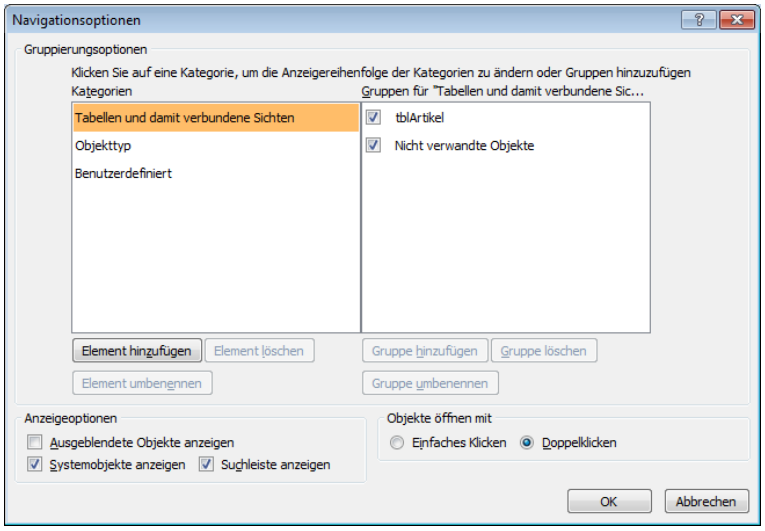


Bild 2: Aktivieren der Anzeige der Systemobjekte unter Access 2007 und jünger

Nun mögen Sie denken, dass Sie beim Verschieben der Backend-Datenbank einfach den Pfad in dieser Tabelle ändern können, damit die Datenbank wieder auf die verknüpften Tabellen zugreifen kann. Aber weit gefehlt: Es handelt sich um eine Systemtabelle, deren Daten nicht geändert werden können. Genau genommen lassen sich noch nicht einmal die Änderungen an der Reihenfolge der Spalten speichern.

Verknüpfung mit Backend im gleichen Verzeichnis

Befindet sich das Backend immer im gleichen Verzeichnis wie das Frontend, weil die Datenbank nur aus Gründen der besseren Wartbarkeit aufgeteilt wurde, können Sie dem Frontend eine kleine Prozedur hinzufügen, die bei jedem Start die Verknüpfung erneuert. Dies beugt dem Umbenennen der übergeordneten Verzeichnisse oder dem Verschieben von Frontend und Backend gemeinsam in ein anderes Verzeichnis vor. Die dazu verwendete Prozedur nutzt einige VBA-Befehle, die wir in **Access [basics]** noch nicht vorgestellt haben, aber im Rahmen dieser kleinen Lösung greifen wir einfach einmal den folgenden Ausgaben vor. Zum besseren Verständnis haben wir den

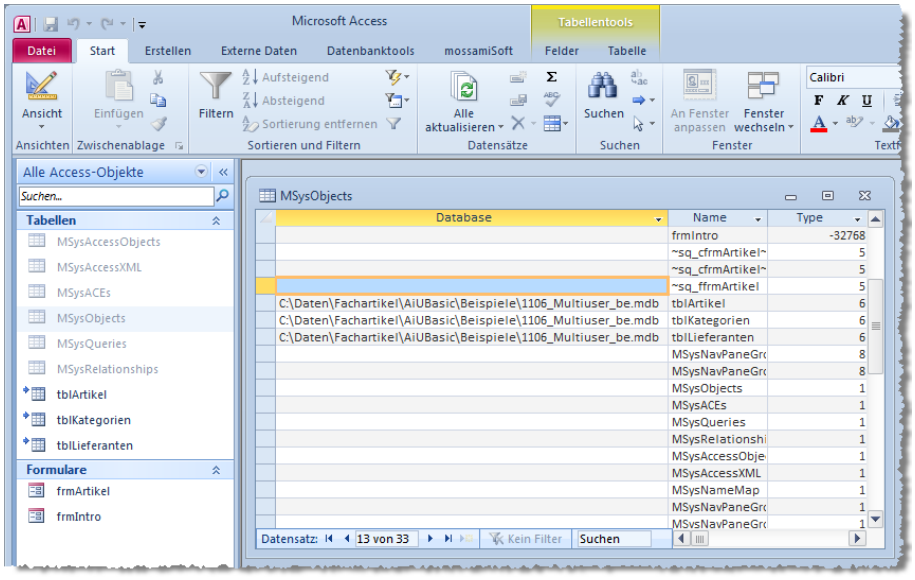


Bild 3: Die Tabelle **MSysObjects** zeigt den Pfad der verknüpften Tabellen an.

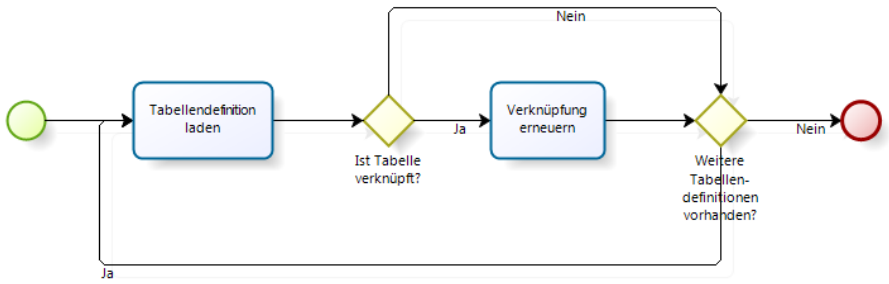


Bild 4: Ablauf beim Wiederverknüpfen von Tabellen eines Backends im gleichen Verzeichnis wie die Frontend-Datenbank

grundsätzlichen Ablauf in Bild 4 skizziert. Die Prozedur legen Sie in einem Standardmodul an. Sie sieht so aus:

```
Public Sub VerknuepfungAktualisieren()
    Dim strDB As String
    Dim strPath  As String
    Dim strConnect As String
    Dim db As DAO.Database
    Dim tdf As DAO.TableDef
    Set db = CurrentDb
    strPath = CurrentProject.Path
    For Each tdf In db.TableDefs
        strConnect = tdf.Connect
        If Len(strConnect) > 0 Then
            strDB = Mid(strConnect, 7
                        InStrRev(strConnect, "\"))
            tdf.Connect = 7
                    ";database=" & strPath & strDB
            tdf.RefreshLink
        End If
    Next tdf
End Sub
```

Die Prozedur deklariert zunächst einige Variablen, deren Bedeutung wir im Folgenden erläutern. Als erstes wird die Objektvariable **db** über die Funktion **CurrentDB** mit einem Verweis auf die aktuelle Datenbank gefüllt. Das bezieht sich in diesem Fall auf die Frontend-Datenbank. Als zweites ermittelt die **Path**-Eigenschaft des **CurrentProject**-Objekts den Pfad zur aktuellen Datenbank, also das Verzeichnis, in dem sich nicht nur das Frontend, sondern auch das Backend befindet. Liegt die Datenbank in einem Verzeichnis wie **c:\Beispiele\Beispieldatenbank.mdb**, liefert diese Funktion **c:\Beispiele** zurück.

Dann beginnt eine **For Each**-Schleife, die in der folgenden Ausgabe von **Access [basics]** erläutert wird, mit dem Durchlaufen der Objekte der Auflistung **TableDefs** des aktuellen Datenbank-Objekts. Die **TableDefs**-Auflistung enthält wiederum Objekte des Typs **TableDef**. Ein **TableDef**-Objekt liefert Informationen über die Definition einer Tabelle, also beispielsweise zu den enthaltenen Feldern und deren Datentypen.

Beim Durchlaufen der Auflistung **TableDefs** wird nun bei jedem Durchlauf ein einziges **TableDef**-Objekt mit der Objektvariablen **tdf** referenziert. Innerhalb der Schleife greift die Prozedur dann darauf zu, um den Wert der Eigenschaft **Connect** in die String-Variable **strConnect** zu schreiben. Diese Eigenschaft liefert für eine verknüpfte Access-Datenbank einen Ausdruck, der aus der Zeichenkette **;database=** und dem Namen der Backend-Datenbank besteht, also beispielsweise

```
;database=c:\Beispiele\Beispieldatenbank.mdb
```

Danach folgen einige Zeichenketten-Funktionen. Die erste heißt **Len** und ermittelt die Länge des Inhalts der Variablen **strConnect**. Ist die Länge größer als **0**, enthält die **Connect**-Eigenschaft einen Wert und es handelt sich beim aktuellen **TableDef**-Objekt um eine verknüpfte Datenbank. Diese Unterscheidung ist wichtig, weil die Datenbank ja auch lokale Tabellen enthält – zum Beispiel die Systemtabellen oder auch benutzerdefinierte Tabellen.

Nachdem der Status des **TableDef**-Objekts geklärt ist, kann sich die Prozedur an das Aktualisieren der Verknüpfung heranwagen Die erste der drei Zeilen innerhalb der **If...Then**-Bedingung ermittelt den reinen Dateinamen des Wertes der **Connect**-Eigenschaft, bei **;database=c:\Beispiele\Beispieldatenbank.mdb** also beispielsweise **\Beispieldatenbank.mdb**. Dabei ermittelt die Funktion **InStrRev** zunächst die Position des letzten Backslash-Zeichens der Zeichenkette (****). Damit wird dann der zweite

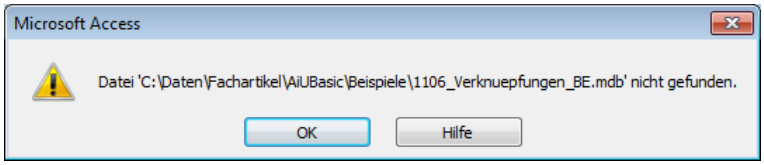


Bild 5: Diese Meldung erscheint, wenn Access die für eine Verknüpfung angegebene Datenbank nicht findet.

Parameter der **Mid**-Zeichenkette gefüttert, die alle Zeichen der im ersten Parameter angegebenen Zeichenkette ab der im zweiten Parameter angegebenen Position zurückliefert. Dabei wird das Backslash-Zeichen mit eingeschlossen.

Die zweite Anweisung innerhalb der **If...Then**-Bedingung setzt dann den Ausdruck für die **Connect**-Eigenschaft des **TableDef**-Attributs neu zusammen, und zwar aus drei Elementen:

- die Zeichenkette **;database=**,
- den aktuellen Pfad des Frontends aus **strPath** und
- den zuvor ermittelten und in **strDB** gespeicherten Dateinamen der Backend-Datenbank.

Die letzte Anweisung ist die Methode **RefreshLink** des aktuellen **TableDef**-Objekts. Sie sorgt dafür, dass die angegebene Verknüpfung aktualisiert wird.

Neuverknüpfung ausprobieren

Um dies zu testen, verschieben Sie einfach die beiden Datenbanken **1106_Verknuepfungen_FE.mdb** und **1106_Verknuepfungen_BE.mdb** gemeinsam in ein anderes Verzeichnis.

Wenn Sie dann das Frontend öffnen und doppelt auf eine der Verknüpfungen klicken, erscheint eine Meldung wie die aus Bild 5. Öffnen Sie dann das Modul **mdlVerknuepfungen** und starten Sie die soeben angelegte Prozedur. Direkt danach können Sie per Doppelklick auf die aktualisierten Verknüpfungen zugreifen.

Um diese Prozedur direkt beim Starten aufzurufen, verwenden Sie ein spezielles Makro namens **Autoexec**. Mehr dazu lesen Sie im Artikel **Das AutoExec-Makro** in der nächsten Ausgabe von **Access [basics]**.

Verknüpfung mit Backend in einem beliebigen Verzeichnis

Wesentlich interessanter wird es, wenn das Backend von mehreren Rechnern aus erreichbar sein soll und es keinen festen Laufwerksbuchstaben beziehungsweise Pfad zum Backend gibt. Dann muss jeder Benutzer zumindest einmal das Backend von Hand auswählen. Dummerweise gelingt auch dies nicht automatisch. Sie müssen also beim Öffnen der Anwendung prüfen, ob die Verknüpfungen alle funktionieren und diese gegebenenfalls erneuern. In diesem Fall kann das Backend aber an beliebiger Stelle liegen, sodass der Benutzer dieses selbst auswählen muss.

Um dies zu realisieren, haben wir zunächst die Prozedur **VerknuepfungAktualisieren** in eine Funktion umgewandelt, die als optionalen Parameter einen **Dateinamen** entgegennimmt und als Ergebnis entweder den Wert **True** oder **False** zurückliefert – je nachdem, ob die Verknüpfung aller Tabellen gelungen ist. Den Ablauf haben wir wiederum skizziert, und zwar in Bild 6. Die Funktion durchläuft genau wie die bereits beschriebene Prozedur alle **TableDef**-Objekte der Tabelle. Dabei wird zunächst geprüft, ob die Eigenschaft **Connect** des **TableDef**-Objekts einen Wert enthält, was darauf hindeutet, dass es sich um eine der zu überprüfenden verknüpften Tabellen handelt.

Falls ja, gibt es nun zwei Fälle: Entweder der Parameter **strFile** besitzt einen Wert oder nicht. Falls die

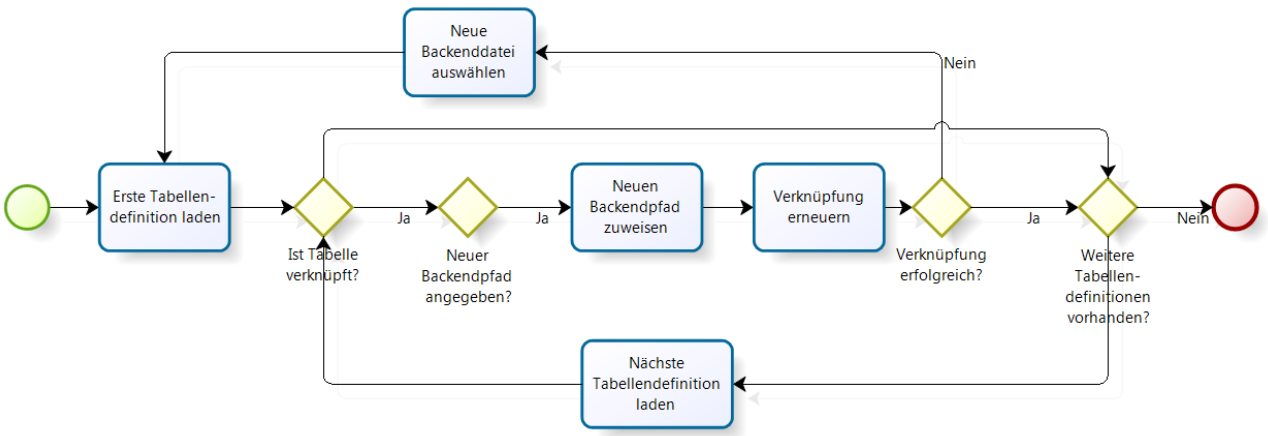


Bild 6: Ablauf beim Verknüpfen von Tabellen, wenn das Backend möglicherweise verschoben wurde

Funktion ohne diesen Wert aufgerufen wurde, geht diese davon aus, dass der aktuell in **Connect** enthaltene Dateiname korrekt ist und probiert, die Verknüpfung mit dem Befehl **RefreshLink** zu aktualisieren. Dabei verwenden wir einen kleinen Kunstgriff, der im Artikel **Fehlerbehandlung unter VBA** genauer erläutert wird. Hier wird die Fehlerbehandlung zunächst mit der Zeile **On Error Resume Next** ausgeschaltet. Dann wird die Anweisung **tdf.RefreshLink** ausgeführt, was zu einem Fehler führt, wenn die in **Connect** angegebene Datei nicht vorhanden ist beziehungsweise die gesuchte Tabelle dort nicht vorliegt.

Die folgenden Zeilen prüfen, ob es einen Fehler gegeben hat und brechen die Funktion im Fehlerfall ab. Diese liefert dann den Wert **False** zurück, was dem Standardwert der als Rückgabewert verwendeten Boolean-Variablen entspricht. Wenn jedoch alle Tabellen mit dem **RefreshLink**-Befehl der **TableDef**-Objekte erfolgreich verknüpft werden konnten, läuft die Funktion bis zur letzten Zeile durch, ohne zuvor abgebrochen zu werden. Dadurch wird der Rückgabewert in der letzten Zeile auf den Wert **True** eingestellt.

```
Public Function VerknuepfungAktualisieren(Optional
strFile As String) As Boolean
    Dim db As DAO.Database
    Dim tdf As DAO.TableDef
    Set db = CurrentDb
    For Each tdf In db.TableDefs
        If Len(tdf.Connect) > 0 Then
            If Len(strFile) > 0 Then
                tdf.Connect = ";database=" _
                                        & strFile
            End If
            On Error Resume Next
            tdf.RefreshLink
            Select Case Err.Number
                Case 3024, 3170
                    Exit Function
            End Select
            On Error GoTo 0
        End If
    Next tdf
    VerknuepfungAktualisieren = True
End Function
```

Nun fehlt noch der Aufruf dieser Funktion. Dies erledigt die Prozedur **Verknuepfung**, die wie folgt aussieht:

```
Public Function Verknuepfung()
    Dim strFile As String
    Do While Not VerknuepfungAktualisieren(strFile)
        If MsgBox("Neu verknüpfen oder abbrechen?", _
                            vbOKCancel) = vbOK Then
            strFile = OpenFileName( _
                CurrentProject.Path)
        Else
            DoCmd.Quit
        End If
    Loop
End Function
```

Die Prozedur enthält eine **Do While**-Schleife, die genau dann abbricht, wenn die Funktion **VerknuepfungAktualisieren** den Wert **True** zurückgibt. Am schnellsten geht dies, wenn die Verknüpfungen der Tabellen bereits auf die richtige Datenbank gemappt sind. Dann liefert bereits der erste Aufruf mit einer leeren Zeichenkette in **strFile** den Wert **True** zurück. Sollte der Benutzer das Backend jedoch zwischenzeitlich in ein anderes Verzeichnis verschoben haben, kann die Funktion **VerknuepfungAktualisieren** die Tabellen nicht neu verknüpfen. Also liefert diese im ersten Durchlauf den Wert **False** zurück. Nun tritt die Prozedur **Verknuepfen** richtig in Aktion:

Sie fragt den Benutzer dann in einem per MsgBox-Funktion erzeugten Meldungsfenster, ob die Verknüpfungen erneuert werden sollen. Falls ja, ruft die folgende Anweisung die Funktion **OpenFileName** auf, die einen **Datei öffnen**-Dialog anzeigt. Details zu dieser Funktion finden Sie im Artikel **Dialog zur Auswahl von Dateien anzeigen** – an dieser Stelle ist nur wichtig, dass Sie damit das Backend auswählen können und die Funktion den Dateinamen samt **Pfad** in die Variable **strFile** einträgt.

Da die **Do While**-Schleife noch nicht abgebrochen wurde, wird der in der ersten Zeile angegebene Ausdruck erneut geprüft. Diesmal wird die Funktion **VerknuepfungAktualisieren** jedoch mit einem konkreten Wert für den Parameter **strFile** aufgerufen. Und wenn der Benutzer die richtige Datei erwischt hat, sollte dies auch zum Aktualisieren der Verknüpfungen führen. Wenn Sie eine Anwendung einsetzen, deren Backend unter Umständen einmal den Speicherort wechseln könnte, fügen Sie die im Modul **mdlVerknuepfungen** enthaltenen Routinen einfach in die gewünschte Frontend-Datenbank ein und legen wie oben beschrieben ein entsprechendes **Autoexec**-Makro an. Die Anwendung prüft die Verknüpfungen dann bei jedem Öffnungsvorgang und lässt diese gegebenenfalls durch den Benutzer wieder herstellen.