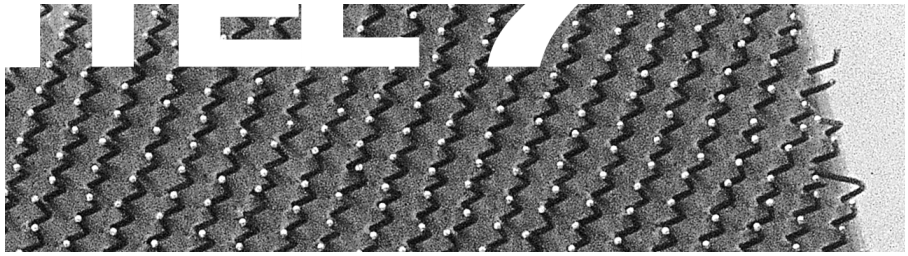
7

Access-Formulare



Access verfolgt als Datenbank-Programm sicherlich andere Konzepte als beispielsweise Excel. Dennoch richten sich in vielen Büros Mitarbeiter kleine oder große Listen in Access ein, da sie Funktionen wie Sortieren und Filtern schätzen. Auch bei der Dateneingabe liegt mit Access ein mächtiges Tool vor: Nicht nur, dass die Steuerelemente eine Reihe von Eigenschaften besitzen, mit deren Hilfe sie vielseitig eingesetzt werden können, sie sind in der Regel auch an Felder einer Tabelle oder Abfrage gebunden, an die wiederum bestimmte Voraussetzungen gestellt werden.
Da es bei Access-Formularen noch stärker als bei VBA-Formularen um Vorausplanung geht, wird auch dieses Programm hier vorgestellt.

Inhaltsübersicht Kapitel 7

7 Access-Formulare

Dieses Kapitel soll einen Überblick und einen Einblick in die Arbeitsweise von Access geben. Es ist unmöglich, auf den wenigen Seiten ein profundes Wissen der Datenbankprogrammierung zu vermitteln. Zunächst einige allgemeine Vorbemerkungen, dann die Besonderheiten der Access-Formulare.

7.1 Tabellen in Access

Access liegt ein relationales Datenbankmodell zugrunde. Das heißt, Daten werden nur einmal in einer Tabelle abgespeichert. Überall dort, wo sie benötigt werden, werden sie mit Hilfe von Verknüpfungen organisiert. Stellen Sie sich eine Mitgliederliste vor. In einer Tabelle werden alle Namen der Mitglieder gespeichert, in einer weiteren Tabelle befinden sich die Bankverbindungen. Natürlich könnte man – wie beispielsweise in Excel möglich – zu jedem Namen die Bankverbindung eingeben. Doch: stellen Sie sich vor, mehrere Leute pflegen die Daten: Der Bankname von fünf verschiedenen Leuten würde auf fünf verschiedene Arten eingegeben; bei der selben Bankleitzahl. Oder zwei Banken fusionieren (was wirklich kein aus der Luft geholtes Beispiel ist). Der Name einer der beiden Banken ändert sich. Damit nicht die gesamte Tabelle (mit „Suchen und Ersetzen“) nach dem Banknamen durchforstet werden muss, wird er nur ein Mal in der Bankenliste geändert. Jedes Mitglied, mit einer Bankverbindung bei dieser Bank, erhält nun automatisch die Änderungen. Wie eine solche Normalform (man spricht von der dritten Normalform) eingerichtet wird, soll im Folgenden beschrieben werden.
Wesen und Kern der Datenbank liegt in der Tabelle. Jede Tabelle einer Datenbank besteht aus Spalten (Datenfeldern) und Zeilen (Datensätzen). Sie enthalten die eigentlichen Daten. Dabei sind die Daten der Zeilen und der Spalten in der Regel unsortiert. Alle Datenwerte einer Spalte sind vom gleichen Typ. Die Namen der Spalten sind eindeutig und folgen gewissen Regeln: Sie dürfen keinen Punkt enthalten, ebenso kein „!“ und keine „[“ und „]“. Leerzeichen sind erlaubt, allerdings nicht immer von praktischem Wert.

7.2 Datentypen

In der Entwurfsansicht wird eine Tabelle definiert. In der Datenblattansicht kann man sich das Ergebnis ansehen. Man könnte dort Daten eingeben, ansehen und ändern. Doch dafür stehen Formulare zur Verfügung. Die Feldnamen, die in der Tabelle normalerweise nebeneinander stehen, werden untereinander geschrieben. Sie können nachträglich gelöscht, verschoben, umbenannt und kopiert werden.
Bevor man die Daten eingibt, sollte man sich Gedanken machen, um welchen Typ Daten es sich handelt. Folgende Typen stehen zur Verfügung:

Tabelle 7.1 Die Felddatentypen in Access

Datentyp	Größe	Beispiel, Verwendung
Text	max. 255 Zeichen	Personennamen, Straßennamen, Ortsnamen
Memo	längere Texte: maximal 64.000 Zeichen (≈ 32 Din-A-4-Seiten) Achtung: kann nicht sortiert werden!	Bemerkungen, Kommentare
Zahl	Zahlen, mit denen später gerechnet werden. 1, 2, 4 oder 8 Byte	Menge, Anzahl, Werte
Datum/Uhrzeit	Alle Datums- oder Zeitangaben zwischen den 01.01.100 und dem 31.12.9999. Benötigt werden 8 Byte.	Termine, Daten, Uhrzeiten, Arbeitsbeginn
Währung	Alle Zahlen, die als Währungswerte dargestellt werden. Sie werden auf vier Dezimalstellen genau berechnet, mit zwei Dezimalstellen dargestellt und automatisch auf zwei Nachkommastellen kaufmännisch gerundet. 8 Byte werden benötigt.	Preise, Kosten, Mehrwertsteuer
AutoWert	Eindeutiger Zähler, der vom Benutzer nicht verändert werden kann. 4 Byte werden benötigt.	Zähler
Ja/Nein	Diese Feld nimmt nur diese beiden Zustände an. 1 Bit	Bezahlt (ja/nein) Versicherung abgeschlossen (ja/nein)
OLE-Objekt	Eingebettete Objekte, beispielsweise Bilder, Sound, Videos. Bis zu 1 Gigabyte ist möglich.	Fotos
Hyperlink	Verknüpfungen zu anderen Dateien oder zu Seiten im Internet.	

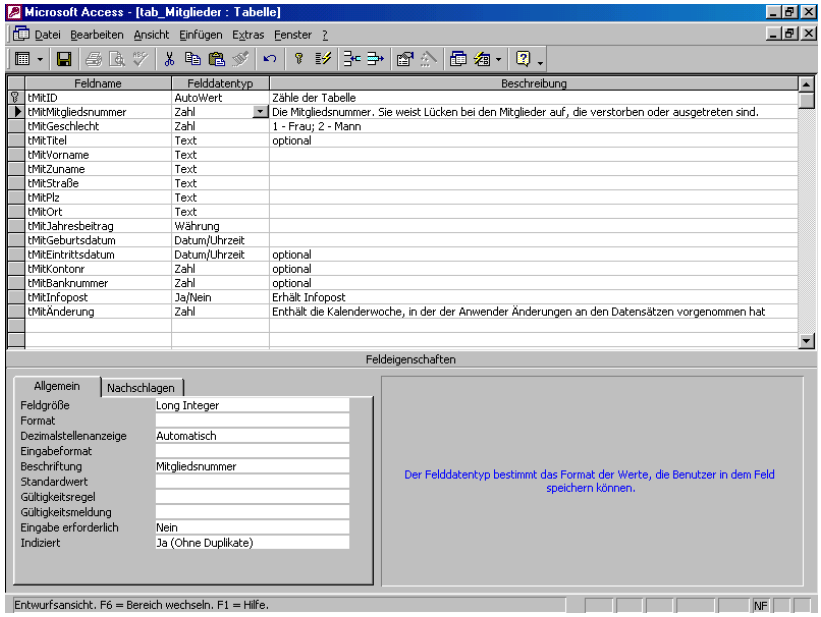


Abbildung 7.1 Die Entwurfsansicht einer Tabelle

Warnung

Auf den ersten Blick sieht man den Unterschied zwischen Text und Zahl sofort. Bei genauerem Hinsehen sind die Unterschiede zum Teil jedoch sehr diffizil. Es ist keine Frage, dass Vorname, Zuname, Straße und Ort Texte sind. Aber Postleitzahlen? Was auf den ersten Blick nach Zahlen aussieht, wird problematisch, wenn Sie eine Postleitzahl aus Sachsen eingeben. Die führende Null wird beim Verlassen gelöscht. Formatiert man nun die als Zahl definierte Postleitzahl vom Typ „00000“, bleiben die übrigen Postleitzahlen so, wie sie waren, die führende 0 der Sachsen wird intern abgeschnitten und per Formatierung wieder angezeigt. Der Nachteil liegt auf der Hand: Gesellen sich Mitglieder aus Österreich und der Schweiz zu den Deutschen, werden ihre vierstelligen Postleitzahlen zu fünfstelligen vervollständigt. Sie können nun mit den Sachsen verwechselt werden. Als Lösung bietet sich Text an.

Das gleiche Problem entsteht bei Telefonnummern. Will man die Durchwahl von Telefonnummern trennen, benötigt man ein Zeichen: eine Leerstelle, eine Klammer oder einen Gedankenstrich. Zwar könnte dieser hinzuformatiert werden, doch sind Telefonnummern und Durchwahlnummern unterschiedlich lang. Es kann keine feste

Position für das Trennzeichen angegeben werden. Über Vorwahlen braucht man wohl nicht zu sprechen: In Deutschland sind die Telefonvorwahlnummern drei-, vier- oder fünfziffrig. Alle besitzen eine führende Null. Es ist sehr schwierig, so etwas zu formatieren. Deshalb ist es besser, gleich auf Text zurückzugreifen.
Und warum nicht immer gleich Text? Die Antwort ist simpel: Gerechnet werden kann nur mit Zahlen. Für einfache Rechnungen, aber auch für komplexe Funktionen müssen die Daten intern als Zahlen vorliegen. Nur dann kann die Rechnung erfolgen. Sie sollten sich also vor Erstellen des Feldes genau Gedanken darüber machen, ob und wie die Daten weiterverarbeitet werden. Manche Banken fordern beispielsweise bei einem automatischen Bankeinzug als Gegenprobe die Summe der Kontonummern und Bankleitzahlen!

Hinweis

Ein weiterer Grund liegt im Speicherplatz: Je kleiner der Datentyp pro Feld ist, umso weniger Speicher wird benötigt. Für eine Integer-Zahl, beispielsweise 31.998, werden 2 Byte benötigt. 2 Byte entsprechen allerdings erst zwei Zeichen. Wird die gleiche Zahl als Text gespeichert, werden 5 Byte benötigt!

Eine andere Überlegung ist in diesem Zusammenhang relevant: Offensichtlich ist es vernünftig, für Vor- und Zunamen zwei verschiedene Felder anzulegen. Für Straßen wird in der Regel nur eines verwendet. Auch dies ist nicht unproblematisch. Text wird anders sortiert als Zahlen. Nehmen Sie die drei Vornamen „Sieglinde“, „Brünnhilde“ und „Siegmund“. Um sie in eine alphabetische Reihenfolge zu bringen, wird der erste Buchstabe (das erste Zeichen) aller drei Texte verglichen. Ergebnis: „B“ < „S“. Folgerung: „Brünnhilde“ ist die erste. Da „S“ = „S“, wird das zweite Zeichen genommen. Hier gilt „i“ = „i“, ebenso „e“ = „e“ und „g“ = „g“. Erst „l“ < „m“ ermöglicht die Aussage „Sieglinde“ < „Siegmund“. Zahlen werden anders sortiert. Von den drei Zahlen 7, 21 und13 wird nicht die Reihenfolge der Zeichen von links nach rechts betrachtet. Sonst würde gelten: 13 < 21 < 7. Und das ist falsch. Zahlen werden als ganze Zahlen betrachtet und verglichen. Ergebnis: 7 < 13 < 21. Was passiert nun mit Straßennamen. Angenommen, wir haben einige Mitglieder in der Wotanstraße. Und zwar in der Wotanstraße 2, Wotanstraße 5a, Wotanstraße 5b, Wotanstraße 14 Wotanstraße 17 – 19 und Wotanstraße 22. Da es sich hier um einen Text handelt, werden die Einträge wie folgt sortiert: Wotanstraße 14, Wotanstraße 17 – 19, Wotanstraße 2, Wotanstraße 22, Wotanstraße 5a und Wotanstraße 5b. Das ist falsch. Eine Lösung wäre die Aufteilung in zwei Spalten: Straßennamen und Hausnummer. Doch dann stellt sich die Frage, von welchem Typ die Hausnummer ist. Wird sie als Zahl deklariert, können Wotanstraße 5a und Wotanstraße 17 – 19 nicht eingetragen werden. Werden

Sie als Text deklariert, funktioniert das korrekte Sortieren wieder nicht. Ich selbst habe früher in Mannheim, in „C7,7b“ gewohnt. Wie sollte man das auf Straße und Hausnummer verteilen? Korrekt wäre: Straße: „C7“, Hausnummer: „7b“. Damit geht allerdings das Komma verloren. Verzichtet man auf das Komma, könnte es natürlich leicht zu Verwechslungen zwischen L1 22 und L12 2 kommen. Egal, wie Sie das Problem lösen: Auch hier sollten Hausnummern nicht als Zahlen deklariert werden, da es durchaus welche gibt, die keine „echten“ Zahlen sind.
Die gleichen Überlegungen müssen selbstverständlich bei jeder anderen Art von Nummern angestellt werden: Artikelnummern, Abteilungsnummern, Kfz-Kennzeichen, ISBN-Nummern, Sozialversicherungsnummern und so weiter.

7.2.1 Primärschlüssel

Ein wichtiges Kennzeichen von relationalen Datenbanken ist die Vergabe eines Schlüssels. Jeder Datensatz muss eindeutig durch einen Primärschlüssel identifizierbar sein. Ein Schlüssel kann ein eindeutiges Datenfeld darstellen oder eine Kombination aus mehreren, die Eindeutigkeit liefern. Dazu kann eine fortlaufende Zahl verwendet werden oder auch eine Zahl, die Eindeutigkeit aufweist: die Mitgliedsnummer. Nicht immer die eleganteste Lösung, doch funktioniert sie immer: Wählen Sie zu Beginn der Tabelle als erste Spalte eine ID, vergeben Sie den Datentyp „AutoWert“ und setzen Sie einen Primärschlüssel. Dann kann nichts mehr schief gehen!

7.2.2 Die Feldeigenschaften

Jeder Typ besitzt nun eine Reihe von Feldeigenschaften. Dort können weitere Kriterien eingestellt werden. Die wichtigste Eigenschaft ist sicherlich die Feldgröße. Je kleiner die Feldgröße gewählt wird, umso weniger Speicherplatz wird benötigt und umso schneller arbeitet die Datenbank.
Die Feldgröße kann für den Text festgelegt werden. Damit sind die Anzahl der Zeichen gemeint, die der Benutzer maximal eingeben darf. Auch für Zahlen gibt es eine Feldgröße. Folgende Werte stehen dabei zur Verfügung:

Tabelle 7.2 Die Zahlentypen von Access

Zahlentyp	Wertebereich	Dezimalstellen	Benötigter Speicherplatz
Byte	0 bis 255	keine	1 Byte
Integer	-32.768 bis 32.768	keine	2 Byte
Long Integer	-2.147.483.648 bis 2.147.483.648	keine	4 Byte
Single	$-3{,}4 * 10^{38}$ bis $3{,}4 * 10^{38}$	7	4 Byte
Double	$-1{,}797 * 10^{308}$ bis $1{,}797 * 10^{308}$	15	8 Byte

Warnung

Auch hier ist Vorsicht geboten. Was man bei den Datentypen bedenkt, muss auch für die Feldgröße bedacht werden. Angenommen, die Mitglieder einer Datenbank kommen alle aus einer norddeutschen Stadt, das heißt, kein Mitglied kommt aus Sachsen und hat damit eine führende Null in der Postleitzahl. Angenommen, man wollte für sie ein Zahlenfeld für die Postleitzahlen einrichten. Dann darf nicht „Integer“ verwendet werden, weil man damit alle Postleitzahlen südlich von Detmold (32701) ausschließen würde. Was auf den ersten Blick noch sehr groß aussieht, kann bei genauem Betrachten schon nicht genügen. Ebenso andere Zahlen. Dieses Buch hat die ISBN-Nummer 3446216472. Damit scheidet auch hier „Long Integer“ aus – es muss mindestens die Feldeigenschaft „Single“ verwendet werden. (Übrigens wird für einige ISBN-Nummern ein „x“ verwendet – also scheidet der Typ Zahl erneut aus: man muss Text verwenden.)

Weiter interessante Eigenschaften für die spätere Formulargestaltung sind die Dezimalstellen (selbstredend nur bei Zahlen), das Format (gemeint: das angezeigte Format), das Eingabeformat, der Standardwert, die Gültigkeitsregel und die Frage, ob eine Eingabe erforderlich ist.

7.2.3 Ausgabeformate

Einige Formate, die der späteren Anzeige der Daten in der Tabelle und im Formular dienen, finden sich in der Dropdownliste:

Tabelle 7.3 Die vordefinierten Zahlentypen

Datentyp	Vorhandene Formate
Zahlen, Währung	Allgemeine Zahl, Währung, Festkommazahl, Standardzahl, Prozentzahl und Exponentialzahl
Datum/Uhrzeit	Standarddatum, Datum, lang, mittel und kurz, Zeit lang, 12Std und 24Std
Ja/Nein	Wahr/Falsch, Ja/Nein, Ein/Aus

Daneben stehen die – mit Sicherheit von Excel her bekannten – Zahlen und Datumsformate zur Verfügung:

Tabelle 7.4 Die Zahlenformate

Zeichen	Bedeutung	Beispiel (1234,5678)
0	Ziffer	0 ergibt 1235
,	Dezimalzeichen	0,00 ergibt 1234,57
.	Tausendertrennzeichen	#.##0,00 ergibt 1.234,57
#	optionale Zahl	
%	Prozentzeichen (multipliziert mit 100)	0 % ergibt 123457 %
"Mio. t"	Maßeinheit, Währung	0 "Mio. t" ergibt 1235 Mio. t
0 "%"	hängt ein Prozentzeichen ohne Umrechnung an	0 "%" ergibt 1235 %

Tabelle 7.5 Die Textformate

Zeichen	Bedeutung	Beispiel (Feierabend)
@	Zeichen oder Leerzeichen	Feierabend
&	Zeichen, Leerzeichen oder gar nichts	Feierabend
<	Ausgabe in Kleinbuchstaben	feierabend
>	Großbuchstaben	FEIERABEND
"/"	Textausgabe, entspricht: "	/Feierabend

Die Datums- und Zeitformate

Zeichen	Bedeutung	Beispiel (01.02.2001)
t	Tag auf eine Ziffer gekürzt	t.m.jj ergibt 1.2.01
tt	Tag auf zwei Ziffern	tt.mm.jjjj ergibt 01.02.2001
ttt	Wochentag, abgekürzt	ttt ergibt Sa
tttt	Wochentag, ausgeschrieben	tttt ergibt Samstag
m	Monat auf eine Ziffer gekürzt	t.m ergibt 1.2
mm	Monat auf zwei Ziffern gekürzt	jjjj-mm-tt ergibt 2001-02-01
mmm	Monat, abgekürzt	mmm ergibt Feb
mmmm	Monat, ausgeschrieben	mmmm ergibt Februar
jj	Jahr, zweistellig	
jjjj	Jahr, vierstellig	
q	Quartal	
h	Stunde, auf eine Stelle	h:m ergibt 9:5
hh	Stunde, auf zwei Stellen	hh:mm ergibt 09:05
n	Minute auf eine Stelle	
nn	Minute auf zwei Stellen	
s	Sekunde auf eine Stelle	
ss	Sekunde auf zwei Stellen	

7.2.4 Eingabeformate

Soll dem Benutzer die Eingabe erleichtert werden, können verschiedene Zeichen vorgegeben werden, über die der Benutzer seine Zahlen, Texte oder Datumsangaben tippt. Folgende Werte stehen zur Verfügung:

Die Eingabeformate

Zeichen	Bedeutung	Eingabe
0	Platzhalter für eine Ziffer	erforderlich
9	Ziffer oder Leerzeichen	nicht erforderlich
#	Ziffer, Leerzeichen, Plus- oder Minuszeichen	nicht erforderlich
L	Buchstabe (A – Z)	erforderlich
?	Buchstabe (A – Z)	nicht erforderlich

Die Eingabeformate (Fortsetzung)

A	Buchstabe oder Ziffer	erforderlich
a	Buchstabe oder Ziffer	nicht erforderlich
&	beliebiges Zeichen	erforderlich
C	beliebiges Zeichen	nicht erforderlich
/	Dezimal-, Tausender-, Datums- oder Zeit-Trennzeichen	
\	das folgende Zeichen wird als Formatkonstante angezeigt	
<	Alle nachfolgenden Zeichen werden in Kleinbuchstaben verwandelt.	
>	Alle nachfolgenden Zeichen werden in Kleinbuchstaben verwandelt.	

Access zeigt bei der Eingabe immer den Unterstrich als Platzhalter an. Wenn Sie ein anderes Zeichen haben möchten, geben Sie es als dritten Teil des Eingabeformats an. Der zweite Teil gibt an, ob Access die angezeigten Literalzeichen in der Tabelle speichert, wenn Sie Daten eingeben. Wenn Sie für diesen Bereich den Wert 0 verwenden, werden alle Literale (zum Beispiel die Klammern in einem Telefonnummern-Eingabeformat) zusammen mit dem Wert gespeichert. Wenn Sie den Wert 1 oder keinen Wert in diesem Bereich eingeben, werden nur die Zeichen gespeichert, die tatsächlich im Steuerelement eingegeben werden.

7.2.5 Gültigkeitsprüfung

Eingegebene Werte können sofort auf Gültigkeit überprüft werden. Dies kann bereits in der Entwurfsansicht der Tabellendefinition geschehen. Auf Gültigkeit könnte auch erst im Formular geprüft werden.
Bei Zahlen stehen die Vergleichsoperatoren =, <, >, <=, >= und <> zur Verfügung. Sie können mit `und`, beziehungsweise `oder` verknüpft werden.
Sind beispielsweise nur die vier Zahlen 1,2,3 und 4 zugelassen, könnte die Bedingung für eine Byte-Zahl wie folgt formuliert werden:

```
>0 Und <4
>=1 Und <=4
1 Oder 2 Oder 3 Oder 4
```

Es steht auch der Befehl `Zwischen` zur Verfügung:

```
Zwischen 1 Und 4
```

`Zwischen` meint dabei einschließlich des ersten und des letzten Wertes. Da Datumsangaben intern als Zahlen gespeichert werden, gilt für sie das Gleiche. Nur dass das Datum durch Zahlenzeichen eingeschlossen ist:

```
>=#01.01.2000#
```

oder:

```
Zwischen #01.01.00# Und #31.12.01#
```

```
>=#01.01.00# Und <=#31.12.01#
```

Soll ein eingegebenes Datum mit dem heutigen Datum verglichen werden, kann man für „heute“ die Funktion `Datum()` verwenden. Beispielsweise:

```
<=Datum()
```

Für Texte steht der Vergleichsoperator `Wie` zur Verfügung, der in Verbindung mit den oben beschriebenen Eingabezeichen verwendet werden kann:

Einige Beispiele zu Gültigkeitsregeln für Texte

Gültigkeitsregel	Bedeutung
Wie "A????"	Der eingegebene Text muss fünf Zeichen lang sein. Das erste Zeichen ist ein „A“.
Wie "Z*"	Der eingegebene Text muss mit einem Text beginnen.
Wie "#####"	Nur fünfstellige Zahlen sind erlaubt
Wie "*@*"	Der Text muss das Zeichen „@“ enthalten

Nicht nur einzelne Felder haben Gültigkeitsregeln – auch für die Tabelle können Gültigkeiten definiert werden. Soll beispielsweise in einer Leihtabelle das Rückgabedatum stets nach dem Ausleihdatum stehen, muss

```
[Rückgabedatum] >= [Ausleihdatum]
```

definiert werden. Allerdings nicht in einer der beiden Felder, sondern in den Eigenschaften der Tabelle. Sie gelangen über den Menüpunkt ANSICHT – EIGENSCHAFTEN dorthin. Dort könnte die obige Feldrestriktion eingegeben werden.

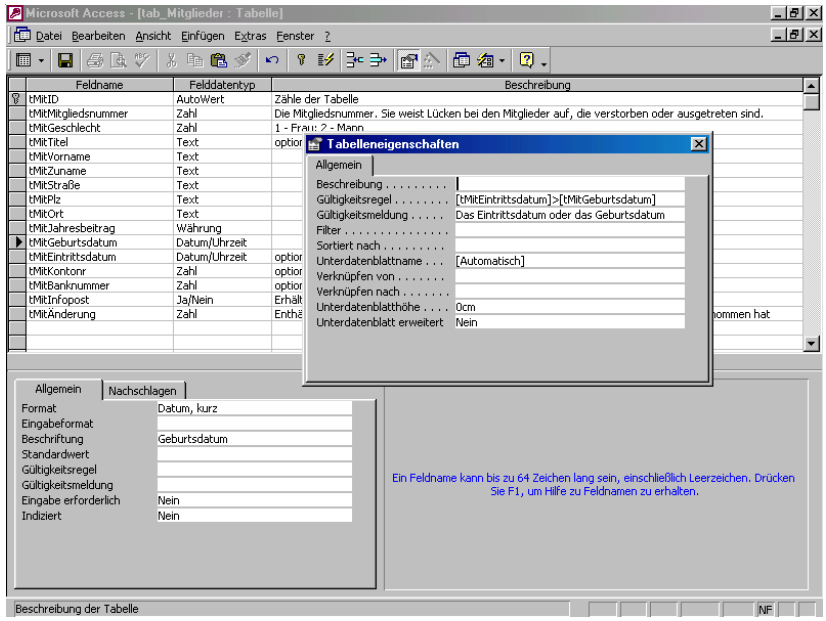


Abbildung 7.2 Die Eigenschaften der Tabelle

7.2.6 Gültigkeitsmeldung

Wird eine Gültigkeitsregel verletzt, kann der Benutzer darauf hingewiesen werden. Bleibt diese Zeile leer, erhält er beim Verletzten der Gültigkeit eine Standardmeldung.

7.2.7 Standardwert

In dieser Zeile kann festgelegt werden, ob der Benutzer einen Vorgabewert erhält, den er überschreiben kann. Dies kann ein Text oder eine Zahl sein, die häufig vorkommt oder das aktuelle Datum:

```
Datum()
```

7.2.8 Beschriftung

Die Beschriftung ist sekundär: Damit kann ein anderer Feldname als jener, der verwendet wird, angezeigt werden. Werden beispielsweise alle Feldnamen der Tabelle „tab_Mitglied“ mit dem Präfix „tMit“ versehen, also beispielsweise: tMitID, tMitTi-

tel, tMitVorname, ... dann könnte ihnen eine Beschriftung zugewiesen werden: ID, Titel, Vorname, ... Der Vorteil der Beschriftung besteht darin, dass alle Abfragen und Formulare (und natürlich auch Berichte) die Beschriftung als Standardtext verwenden. Man muss sie nicht neu beschriften. Allerdings gibt es noch eine Reihe anderer Möglichkeiten auf Formularen zu beschriften.

7.2.9 Eingabe erforderlich

Sollte im Feld „Gültigkeitsregel“ noch kein Wert festgelegt worden sein, kann hier bestimmt werden, ob der Benutzer immer etwas in ein bestimmtes Feld eintragen muss.

7.2.10 Ändern von Datentypen

Beim Erstellen einer neuen Tabelle ist es problemlos möglich, Datentypen zu ändern. Stehen schon Werte in der Tabelle, kann es zu Problemen kommen. Wird die Feldeigenschaft vergrößert, kommt es in der Regel zu keinen Problemen. Wird sie allerdings verkleinert, werden Texte, die zu lang waren, abgeschnitten und zu große Zahlen werden gelöscht. Wird ein Text in eine Zahl verwandelt und steht „echter“ Text in einem Feld, wird dieses gelöscht. Wird umgekehrt ein Datum oder eine Zahl in Text verwandelt, geschieht nichts. AutoWerte können in Zahlen verwandelt werden, Zahlen allerdings nicht in AutoWerte. Ist Letzteres gewünscht, muss die Spalte gelöscht und neu erzeugt werden.
Datentypen werden normalerweise geändert, wenn festgestellt wurde, dass der vorgegebene Typ zu klein ist – wenn beispielsweise die Zahlen doch den Byte-Bereich von 255 überschreiten. Oder wenn „Zahlen“ auch führende Nullen oder Buchstaben enthalten und sie nun als Text formatiert werden müssen. Dann wird der Datentyp oder die Feldeigenschaft vergrößert.
Verkleinert werden Typen und Eigenschaften in der Regel dann, wenn externe Daten (Exceltabellen, Textdateien, andere Datenbankformate) in eine Access-Datenbank importiert werden. Dann sollte man die vorgeschlagene Größe überprüfen und in der Regel verkleinern. Access schlägt beim Textimport immer 255 Zeichen vor und bei Zahlen den Typ „Double“. Dies kann beim Importieren im Assistenten verändert werden oder im Nachhinein in der Entwurfsansicht der Tabelle. Überprüfen Sie die geänderten Daten. Sollte es zu Verlusten kommen, müssen die Daten gelöscht und erneut importiert werden.

7.2.11 Umgang mit Tabellen

Wenn eine Tabelle in der Entwurfsansicht erstellt wurde, muss sie gespeichert werden. Nun könnten die Daten in der Datenblattansicht eingegeben werden. Sollte gegen Gültigkeitsregeln verstoßen werden, muss der Benutzer die Daten entweder korrekt eingeben oder die Dateneingabe mit der <Esc>-Taste beenden.
Die Tabelle, das heißt die Entwurfsansicht und die Daten, kann gelöscht (Taste <Entf>) oder umbenannt werden (<F2> oder zwei Mausklicks). Auch die Tabelle kann kopiert werden (BEARBEITEN – KOPIEREN, <Strg>+<C> oder Kontextmenü). Wird sie nun wieder eingefügt, fragt Access, ob nur die Struktur oder die Struktur und die Daten eingefügt werden sollen.

7.2.12 Namenskonvention

Es gehört nicht nur zum guten Programmierstil, in Access eine einheitliche Namenskonvention zu verwenden. Es erleichtert ungemein das nachträgliche Überarbeiten von eigenen und fremden Datenbanken. Sie müssen sich nicht an die folgende Konvention halten; Sie können auch ohne Namenskonvention auskommen oder sich eine eigene ausdenken. Dennoch: Es ist sinnvoll, mit der 1993 von Stan Leszynski und Greg Reddick herausgegebenen Empfehlung zu arbeiten, die schon mehrfach angesprochen wurde. Hier einige Auszüge:
Objekte in Access besitzen ein Präfix. Es lautet: „tbl“ bei der Tabelle, „qry“ bei der Abfrage, „frm“ beim Formular, „rpt“ beim Bericht, „mct“ für Makros und „bas“ bei Modulen. Das kann verfeinert werden: Man könnte im Präfix kenntlich machen, ob es sich um eine Datentabelle, Schlüsseltabelle oder Beziehungstabelle handelt, man könnte erkennbar machen, ob es sich um eine Anfüge-Abfrage, um eine Aktualisierungsabfrage, eine Lösch-Abfrage, eine Kreuztabelle oder um eine Auswahlabfrage handelt. Auch bei Formularen könnte durch ein „Sub“ kenntlich gemacht werden, ob es sich um ein Unterformular handelt. Das heißt: Eine Tabelle könnte „tabMitglieder“ heißen, eine Abfrage, die darauf basiert, „qryMitgliederSortZuname“ und ein Formular „frmMitglieder“. Der Basisname (hier: „Mitglieder“) wird in allen drei Objekten verwendet und deutet an, dass es sich um dasselbe Basisobjekt handelt. Es wird deutlich, dass eine Beziehung zwischen diesen Objekten besteht.
Bei den Feldnamen gibt es verschiedene Vorschläge, von denen sich noch keiner durchsetzen konnte. Einige Datenbankprogrammierer nehmen den Felddatentyp in den Namen auf, andere verzichten auf eine einheitliche Konvention, wieder andere verwenden in den Präfixen die Bedeutung für das Datenbankobjekt. Ich ziehe es vor, Feldnamen in Tabellen mit einem „t“ beginnen zu lassen und dann mit zwei oder drei Buchstaben den Namen der Tabelle abzukürzen. Also beispielsweise „tMiVorname“, „tMiZuname“, „tMiGeburtsdatum“ und so weiter. Das hat den Vorteil, dass

es in der gesamten Datenbank, das heißt in allen Tabellen, keine doppelt vorkommenden Namen gibt.

7.3 Abfragen

Formulare können entweder auf Tabellen oder auf Abfragen gesetzt werden. Dienen die Formulare zur reinen Dateneingabe, können sie direkt auf eine Tabelle aufgesetzt werden. Sollen Daten beim Weiterblättern in einer bestimmten Reihenfolge angezeigt werden, sollte das Formular eine Abfrage verwenden, die auf den Daten einer Tabelle basieren.
Abfragen sind das eigentliche Zentrum einer Datenbank. In ihnen wird sortiert, gefiltert, gerechnet, mit ihrer Hilfe kann man neue Tabellen erstellen, Datensätze aus Tabellen löschen oder anfügen. Da dies nicht zum Thema der Formulare gehört, beschränken wir uns auf zwei Funktionen der Abfragen: Filtern und Sortieren.

7.3.1 Sortieren in einer Abfrage

Wählen Sie die Entwurfsansicht für eine neue Abfrage, dann werden Sie nach der Tabelle gefragt, auf der sie basiert. Abfragen können auch auf Abfragen aufgesetzt werden.

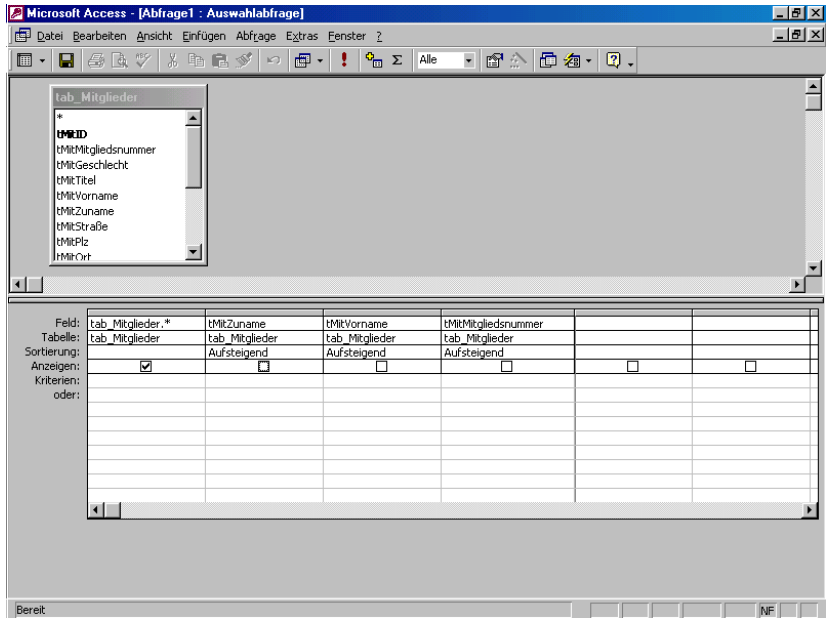


Abbildung 7.3 Eine neue Abfrage wird erstellt

In der Entwurfsansicht der Abfrage findet sich im oberen Teil die Feldliste. In ihr stehen die Namen der Felder der Tabelle. Dieses Fenster kann verschoben und vergrößert werden. Ist eines der Felder fett formatiert, bedeutet dies, dass es sich hier um den Primärschlüssel handelt. Die einzelnen Felder können mit einem Doppelklick auf ihren Namen oder mit der Maus per Drag & Drop in den Entwurfsbereich heruntergezogen werden. Ebenso kann der Feldname auch aus der Dropdownliste ausgewählt werden. So könnte man beispielsweise alle Felder einer Tabelle in die Abfrage aufnehmen (mit gedrückter ⟨Shift⟩- oder ⟨Strg⟩-Taste können mehrere Felder markiert werden). Nun könnte aus dem Dropdownfeld „Sortierung“ die Sortierung (auf- oder absteigend) eingeschaltet werden. Diese Lösung erweist sich allerdings als nicht sehr glücklich. Soll beispielsweise in einer Namensliste zuerst nach dem Zunamen und anschließend nach dem Vornamen sortiert werden, versagt die eben beschriebene Lösung. Geschickter ist es, statt jedes Feld einzeln in die Abfrage zu ziehen, die gesamte Tabelle nach unten zu holen. Sie wird in der Feldliste durch ein „*“ repräsentiert.
Nun kann zur Tabelle zusätzlich ein Feld oder mehrere Felder hinzugefügt werden. Für sie wird nun ein Sortierkriterium festgelegt. Access liest die Abfragekriterien von links nach rechts. Das bedeutet: Soll eine Namensliste zuerst nach dem Zunamen, dann nach dem Vornamen sortiert werden, sind diese beiden Felder in dieser Reihenfolge nebeneinander einzugeben. Damit sie in der Datenblattansicht nicht zwei Mal auftauchen (in der Tabelle und als Sortierkriterium), sollte das Feld „Anzeigen“ deaktiviert werden.
Das Ergebnis der Abfrage kann jederzeit über das Symbol oder den Menüpunkt ANSICHT – DATENBLATTANSICHT angesehen werden (falls kein Fehler gemacht wurde). Als dritte „Ansicht“ steht Ihnen noch der interne SQL-Befehl zur Verfügung, der beispielsweise in der Programmierung von Interesse ist. Dieser Befehl repräsentiert die eigentliche Abfrage. Glücklicherweise muss er nicht eingegeben werden, sondern kann in der Entwurfsansicht per Drag & Drop erzeugt werden. Umgekehrt könnte die Abfrage über die SQL-Ansicht verändert werden.

Hinweis

Übrigens kann das Sortieren der Tabelle beschleunigt werden, wenn auf das Feld ein Index gelegt wird. Dabei gibt es Indizes mit und ohne Duplikate. Die Liste sämtlicher gesetzter Indizes einer Tabelle kann über den Menüpunkt ANSICHT – INDIZES eingesehen werden.

Abfragen können (und sollten) gespeichert werden. Dann können Formulare darauf aufbauen.

Hinweis

Jede Änderung in einer Tabelle wird in die Abfrage übernommen. Änderungen in sortierten Abfragen werden in die Tabelle geschrieben. Sollen in einem Formular nicht alle Datensätze verwendet werden, können die benötigten Informationen in der Abfrage gefiltert werden.

7.3.2 Filtern in einer Abfrage

Zum Filtern ist es nötig, den Typ des Feldes zu kennen, nach dem gefiltert wird. Zugegeben: Access erlaubt bei der Eingabe gewisse Unschärfen, die automatisch korrigiert werden, doch, wenn Sie Datum mit Text verwechseln, kann es zu Fehlern kommen.

Gefiltert wird ebenso wie sortiert: Man kann das Symbol für die gesamte Tabelle („*“) nach unten ziehen und daneben einzelne Spalten wiederholen. In ihnen werden die Filterkriterien eingestellt. Dabei stehen folgende Möglichkeiten zur Verfügung:

Tabelle 7.6 Die Filterkriterien der Abfrage

Datentyp	Filterkriterien	Beispiel
Zahl	= (oder nichts), <, >, <=, >=	7 >7
	Zwischen, und, oder, nicht	Zwischen 1 Und 4 1 Oder 2 Oder 3 Oder 4 >=1 Und <=4
Datum, Uhrzeit	wie Zahlen. Das Datum kann normal eingegeben werden und wird in der Regel als solches erkannt. Es wird in Access mit Zahlenzeichen eingeschlossen (#)	#01.04.2001# Zwischen #01.01.2001# Und #31.12.2001#
Text	" "	"Maier" "Maier" Oder "Müller" Oder "Moshammer"
	* (Platzhalter für beliebig viele Zeichen) ? (Platzhalter für ein Zeichen)	Wie "M*" liefert alle Namen, die mit „M“ beginnen
	[-] (Bereiche)	Wie " [A-M]* " liefert alle Namen, die mit einem Buchstaben von A bis M beginnen
Ja/Nein	Ja oder Nein	Ja filtert alle Wahrwerte
Zahl, Text, Datum	Ist Null	Liefert alle leeren Felder
	Ist nicht Null	Liefert alle Felder, in denen irgend etwas steht

7.4 Beziehungen zwischen Tabellen

Das Kapitel „Beziehungen zwischen Tabellen“ ist für Formulare nicht essenziell notwendig. Werden die Eingabemasken so gestaltet, dass dem Benutzer nur bestimmte Eingabemöglichkeiten zur Verfügung stehen, muss man sich wenig Gedanken über Beziehungen machen. Dennoch sollen hier der Logik und der Vollständigkeit halber einige Gedanken zu diesem Thema folgen.
Nehmen wir das obige Beispiel: eine Liste mit Namen (Mitglieder) und eine Liste mit Bankverbindungen. Zwei Tabellen liegen vor. In der Namenstabelle existiert eine Spalte „BankNr“, in der Zahlen (Long Integer) stehen. Die Banktabelle besitzt einen Autowert, auf dem der Primärschlüssel liegt. Jede Zahl, die in der Namenstabelle vorkommt, existiert eindeutig in der Banktabelle. Die Eindeutigkeit ist durch den Autowert gegeben. Jede Banknummer kann mehrmals in der Namenstabelle auftauchen. Eine Nummer kann in der Banktabelle vorkommen, aber nicht in der Namensliste verwendet werden.
Will man nun als Eigenschaft zwischen den Tabellen definieren, dass auf der Mitgliederseite ein n-seitiges Feld vorliegt, welches in der Banktabelle (1-Seite) erklärt wird, kann man über das Menü EXTRAS – BEZIEHUNGEN eine Beziehung zwischen diesen beiden Feldern und damit zwischen den beiden Spalten herstellen. Man zieht dazu eines der beiden Felder (welches, ist egal) auf das andere.

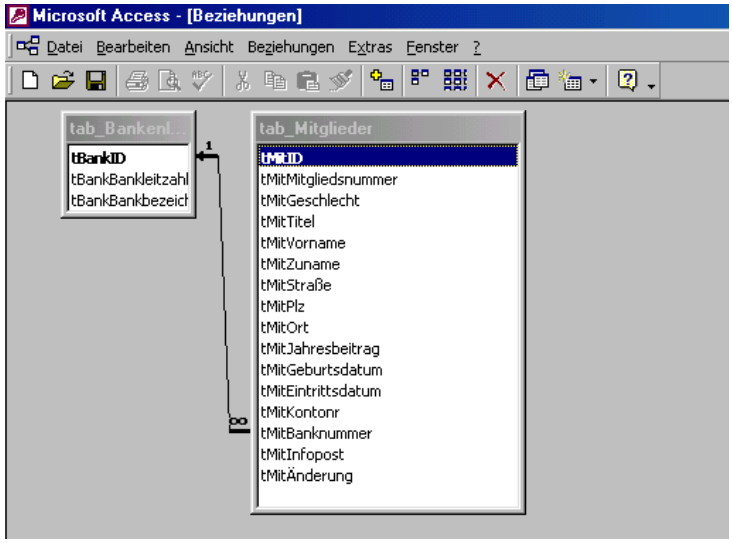


Abbildung 7.4 Eine Beziehung wird erstellt

Um die Gültigkeit zwischen beiden Tabellen herzustellen, muss die „Referentielle Integrität“ eingeschaltet werden. Ohne referentielle Integrität macht die Beziehung

keinen Sinn. Damit wird sichergestellt, dass in der Namensliste keine Banknummer verwendet werden kann, die in der Bankliste nicht existiert. Umgekehrt kann keine Bank gelöscht werden, bei der (noch) Mitglieder sind.

Hinweis

Sie können referentielle Integrität zuweisen, wenn die folgenden Bedingungen erfüllt sind:

- Die übereinstimmende Spalte in der Primärtabelle ist ein Primärschlüssel oder hat eine eindeutige Einschränkung.
- Die miteinander in Beziehung stehenden Spalten haben denselben Datentyp und dieselbe Größe.
- Beide Tabellen gehören zur selben Datenbank.

7.5 Formulare

Der von Access vorgegebene Begriff Formular (Form) ist ein wenig verwirrend. Es geht nicht nur um die Dateneingabe, sondern auch um die Steuerung. Der Schwerpunkt liegt an dieser Stelle auf dem ersten Teil: Der Benutzer kann sich eine Anzahl Daten ansehen, einzelne Informationen ändern oder neue hinzufügen. Formulare sind gleichzeitig die Steuerungstools, mit deren Hilfe der Benutzer andere Formulare oder Berichte öffnen oder schließen kann. Er kann sich auf diese Weise durch die verschiedenen Arbeitsebenen bewegen.

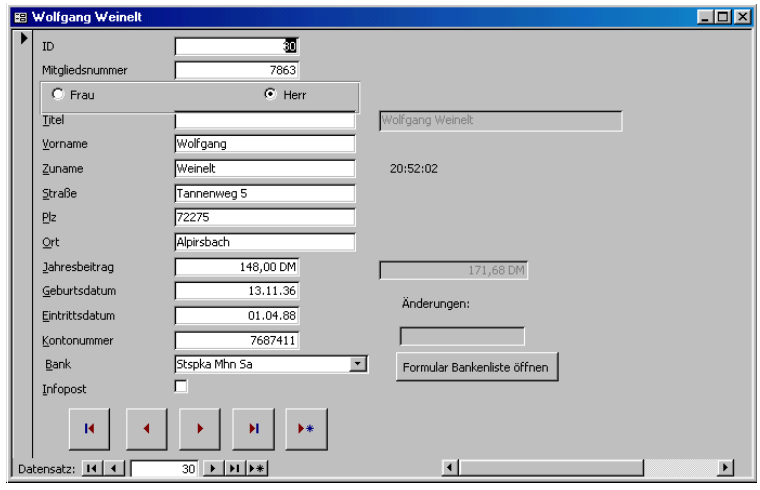


Abbildung 7.5 Der Benutzer gibt in seine Eingabemaske die Daten ein

Formulare können direkt an Tabellen oder Abfragen gebunden sein. Das bedeutet, dass Änderungen in einem Formular in eine Tabelle geschrieben werden, beziehungsweise dass im Formular jeweils der aktuelle Stand der Tabelle angezeigt wird. Und eben darum soll es im Folgenden gehen.

7.6 Der Formular-Assistent

Ein Formular ist mit Hilfe des Assistenten schnell erstellt – für Ungeübte eine einfache und effektive Methode, schnell ein Formular zu erstellen. Dabei kann ein „Auto-Formular“ erstellt werden: ein einspaltiges, ein tabellarisches oder ein Datenblatt. Das gleiche Ergebnis wird erzielt, wenn man den Vorgaben des Formular-Assistenten folgt. Das Layout bestimmt die Form des Formulars. Das Datenblatt und das tabellarische Formular weisen sehr viele Ähnlichkeiten auf, das einspaltige Formular erinnert an das Formular „In Blöcken“. Der Hauptunterschied besteht darin, dass das einspaltige Formular alle Felder von einem Datensatz darstellt, während das tabellarische mehrere Datensätze auf eine Seite bringt. Um sich durch die Datensätze zu bewegen, wird am unteren Rand eine Navigationsschaltfläche eingefügt, mit deren Hilfe der Benutzer auf den ersten, vorherigen, nächsten oder letzten Datensatz springen kann. Das Symbol rechts außen dient dazu, einen neuen Datensatz zu erzeugen.

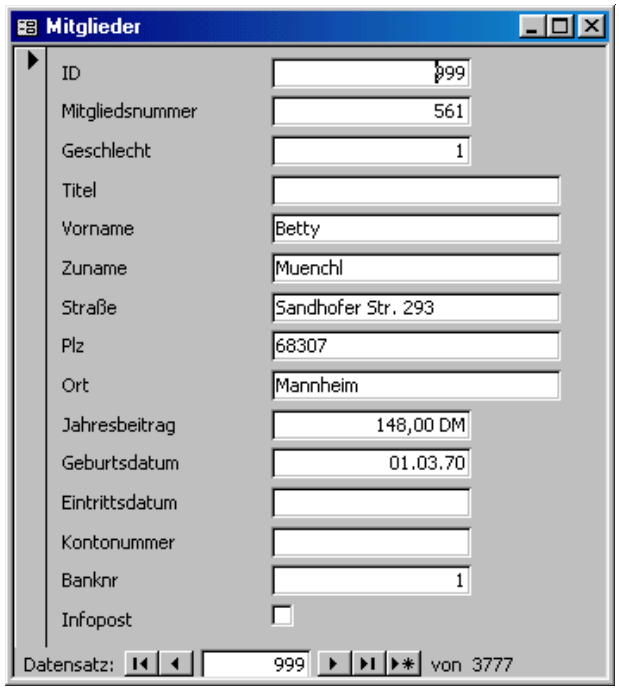


Abbildung 7.6 Ein einfaches Formular, das mit Hilfe des Assistenten erzeugt wurde

In der Entwurfs-Ansicht können nun die vorhandenen Eigenschaften verändert werden.

7.6.1 Die Entwurfsansicht

Die Entwurfsansicht wird entweder beim Bearbeiten eines fertigen Formulars oder beim Erstellen eines neuen Formulars verwendet. Hier soll gezeigt werden, wie ein neues Formular mit Hilfe der Entwurfsansicht erstellt wird.
Wenn Sie mittels der Entwurfsansicht ein neues Formular erstellen, ohne dabei eine Tabelle oder Abfrage auszuwählen, müssen Sie in der Entwurfsansicht zuerst eine Tabelle oder Abfrage an das Formular binden, nachdem das leere Formular erstellt wurde. Das geschieht über die Eigenschaften des Formulars. Rufen Sie für den Eigenschaften-Dialog das Menü ANSICHT – EIGENSCHAFTEN auf oder verwenden Sie das Kontextmenü oder das Symbol. In der Titelzeile muss „Formular“ stehen, da das gesamte Formular und nicht nur ein Teilbereich an eine Tabelle oder Abfrage gebunden wird. Sollte sich ein anderer Text in der Titelzeile befinden, klicken Sie mit der Maus in das Quadrat links neben dem Lineal oder wählen es aus der linken Dropdownliste aus.
In den Eigenschaften im Registerblatt „Daten“ wird die Datenherkunft ausgewählt („Datenherkunft“). Alle bis jetzt erstellten Abfragen und Tabellen findet man in dieser Liste. Man könnte nun eine Tabelle verwenden und auf diese einen Filter oder eine Sortierung anwenden. Stattdessen empfiehlt es sich, eine Abfrage zu verwenden, da diese klarer strukturierbar und leichter veränderbar ist.
Bei tabellarischen Formularen kann ein Kopf oder ein Fuß interessant sein. Sie werden über den Menüpunkt ANSICHT – SEITENKOPF/FUSS beziehungsweise ANSICHT – FORMUALRKOPF/FUSS oder über die rechte Maustaste aktiviert. Zwischen einem Einzelformular und einem tabellarischen Formular kann in den Eigenschaften im Registerblatt „Format“ in der Standardansicht umgeschalten werden. Dort finden sich die Einträge „Einzelnes Formular“, „Endlosformular“ und „Datenblatt“.
Beim Endlosformular ist der Formularkopf interessant. Dort steht eine Überschrift, die für das gesamte Formular oder genauer für jeden einzelnen Datensatz gültig ist. Er bleibt beim „Endlosformular“ und „Datenblatt“ am oberen Bildschirmrand (genauer: Formularrand) stehen, wenn über die Datensätze gescrollt wird.
In den Bereichen selbst stehen keine nennenswerten Eigenschaften zur Verfügung: die Höhe könnte mit der Maus geändert werden, Hintergrundfarbe und Spezialeffekt können verändert werden.

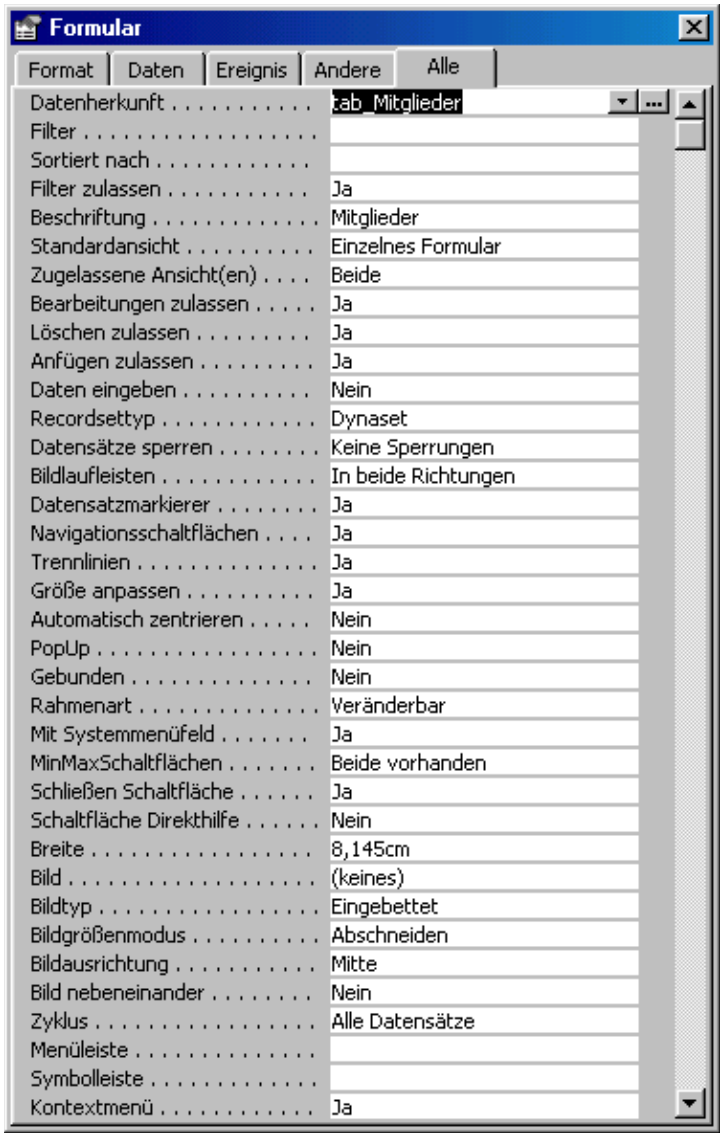


Abbildung 7.7 Die Eigenschaften eines Formulars

7.6.2 Die Steuerelemente

Nun können die einzelnen Datenfelder auf das Formular gezogen werden. Dazu öffnen Sie die Feldliste (Menü ANSICHT) und ziehen ein oder mehrere Felder auf das Formular. Mit der ‹Strg›- oder mit der ‹Shift›-Taste können mehrere Felder in der Feldliste markiert werden.

Warnung

Wird ein Feld aus der Feldliste auf das Zeichenblatt gezogen, werden immer zwei Steuerelemente auf dem Zeichenblatt erstellt: ein Textfeld und ein Bezeichnungsfeld. Um sie zu verändern (Formatieren, Lage verschieben, ...) muss das richtige von beiden markiert werden. Das korrekt markierte steht namentlich in der Dropdownliste am oberen, linken Rand. Wenn das Steuerelement verschoben werden soll, kann entweder das Textfeld oder das Bezeichnungsfeld oder beide verschoben werden. Wird das Textfeld markiert, erscheint an seinen vier Rändern eine offene Hand. Damit kann die Position von beiden Feldern verschoben werden. Soll nur eines von beiden anders positioniert werden, muss das Quadrat in der linken, oberen Ecke angefasst werden.

Abbildung 7.8 Ein Steuerelement wird verschoben

7.6.3 Textfeld

Statt ein Feld aus der Feldliste zu ziehen, können Sie auch das Steuerelement „Textfeld“ aus der Toolbox verwenden. Doch Achtung: Es wird nicht auf das Formular gezogen, sondern angeklickt. Danach kann man mit der Maustaste ein Rechteck auf dem Formular ziehen. In diesem befindet sich dann das Steuerelement. Um das Textfeld an ein Tabellenfeld zu binden, wird wieder der Eigenschaftsdialog benötigt, wo dies eingestellt werden kann. Dort kann bei „Steuerelementinhalt“ das korrekte Feld ausgewählt werden.

Tipps & Tricks

So können gleichzeitig oder nacheinander mehrere Textfelder erzeugt werden: Sollen verschiedene Steuerelemente zur gleichen Zeit verschoben werden, können sie mit gedrückter ⟨Shift⟩-Taste zusammen markiert werden. Man kann ebenso mit gedrückter Maus ein Rechteck um die Steuerelemente ziehen, um sie zu markieren. Dann können sie mit gedrückter Maustaste verschoben werden. Ein exaktes Verschieben ist

auch über die Eigenschaften möglich, wo bei „Links“ oder bei „Oben“ ein Wert eingegeben werden kann. Dann sitzen die Felder auf einer Linie oder in einer Spalte.

Um ein exaktes Ausrichten zu erreichen, kann auch der Menüpunkt FORMAT – AUSRICHTEN verwendet werden. Dort kann „linksbündig“, „rechtsbündig“, „oben“ oder „unten“ eingestellt werden. Doch Achtung: Es müssen die richtigen Steuerelemente markiert sein. Zum Glück steht die Rückgängig-Funktion zur Verfügung!
Eine gute Hilfe für das exakte Positionieren der Steuerelemente auf einem Formular stellt das Raster dar. Normalerweise ist es aktiviert, wenn nicht, kann es über den Menüpunkt FORMAT – AM RASTER AUSRICHTEN eingeschaltet werden.
Auch die Größe kann für eines oder mehrere Steuerelemente verändert werden. Ziehen Sie dazu mit der Maus an einem der Anfasser an einem der vier Seiten oder an einer der drei Ecken (nicht links oben). Fürs Vergrößern ist der Menüpunkt FORMAT – GRÖSSE zuständig. Dort kann festgelegt werden, ob die Textfelder am höchsten, niedrigsten, breitesten oder schmalsten Textfeld angepasst werden. Die numerische Einstellung findet sich unter „Breite“ und „Höhe“.

7.6.4 Die Eigenschaften

Die meisten Eigenschaften sind bekannt. Bei Textfeldern können im Registerblatt „Format“ oder „Alle“ die üblichen Einstellungen bezüglich der Schrift vorgenommen werden: Textfarbe, Schriftart, Schriftgrad, Schriftbreite, Kursiv, Unterstrichen, Textausrichtung und das Verhältnis von Text zu den vier Rändern. Daneben wird die Lage und Größe des Steuerelements festgelegt (Links, Oben, Breite, Höhe), Gestaltungen, die den Inhalt betreffen (Format, Dezimalstellenanzeige) und Gestaltungen, die das Aussehen betreffen (Hintergrundart, Hintergrundfarbe, Spezialeffekt, Rahmenart, Rahmenfarbe, Rahmenbreite).
Soll das Textfeld eine andere Farbe haben, wenn es den Fokus besitzt, kann dies über das Menü FORMAT – BEDINGTE FORMATIERUNG geändert werden. Dort wird in der Bedingung der Eintrag auf „Feld hat Fokus“ gesetzt. Nun können Schriftmodi (Fett, Kursiv und Unterstrichen) eingeschalten, ebenso die Schriftfarbe und Hintergrundfarbe geändert werden.
Im Registerblatt „Daten“ finden Sie Einstellungen, die ebenfalls in der Tabelle vorgenommen werden können: Eingabeformat, Standardwert, Gültigkeitsregel und Gültigkeitsmeldung. Werden diese Einstellungen in der Tabelle vorgenommen, sind sie für alle Formulare gültig. Werden sie auf einem Formular verwendet, gelten sie, selbstredend, nur auf diesem. So können lokale und globale Eigenschaften einer Tabelle gesetzt werden.

Einige interessante Eigenschaften finden sich im Blatt „Andere“: Als Hilfe können „Statusleistentext“ und „SteuerelementTip-Text“ verwendet werden. Das Drücken der <Enter>-Taste kann einen Sprung auf das nächste Steuerelement bewirken („Standard“) oder im Textfeld eine neue Zeile schaffen („Neue Zeile im Feld“). Ebenso könnte die AutoKorrektur verwendet werden. Man könnte eine Kontextmenüleiste erstellen, die an dieses Steuerelement gebunden wird. Und schließlich sollten die einzelnen Felder, in die der Benutzer etwas eingibt, in einer bestimmten Reihenfolge stehen, die eingehalten wird, wenn <Tab> oder <Enter> gedrückt wird. Dazu muss im Feld „In Reihenfolge“ „Ja“ eingeschaltet sein. Die Position kann darunter („Reihenfolgenposition“) eingegeben werden oder über das Menü ANSICHT – AKTIVIERREIHENFOLGE.

7.6.5 Die Aktivierreihenfolge

Damit der Benutzer bei der Eingabe bequem zwischen den einzelnen Feldern mit der Tabulatortaste hin- und herspringen kann, muss in der Entwurfsansicht eines Formulars die Aktivierreihenfolge eingestellt werden. Wechseln Sie hierzu in das Menü ANSICHT – AKTIVIERREIHENFOLGE, markieren Sie dort ein oder mehrere Steuerelemente am grauen Kästchen und ziehen Sie es nach oben oder nach unten. Befinden sich auf dem Formular Bezeichnungsfelder (die keine Eingabe zulassen), können diese vor das nächste Textfeld gezogen werden, in welches die entsprechenden Daten eingegeben werden.

Abbildung 7.9 Die Aktivierreihenfolge

7.6.6 Berechnete Textfelder

Textfelder sind nicht nur zum Ändern von Daten da. Mit ihrer Hilfe können auch Berechnungen ausgeführt werden. Statt des Feldnamens wird eine Berechnung durchgeführt. Also: Statt

```
Jahresbeitrag
```

der Eintrag

```
=[Jahresbeitrag]*1,16
```

Das Feld muss in eckigen Klammern geschrieben sein, davor steht das Gleichheitszeichen und danach die Rechenoperation. Man könnte auch mehrere Felder miteinander verketten:

```
=[Vorname] & " " & [Zuname]
```

Auch im Formularfuß (und -kopf) kann eine Berechnung durchgeführt werden. Hierzu wird eine Funktion (Summe, Anzahl, Mittelwert, Maximum, Minimum, ...) verwendet. Ein Ausdruck für ein tabellarisches Formular könnte wie folgt aussehen:

```
=Summe([Anzahl]*[Einzelpreis])
```

7.6.7 Das Bezeichnungsfeld

Das Bezeichnungsfeld hat weitgehend dieselben Formatierungseigenschaften wie das Textfeld. Ihm fehlen hingegen „Daten“ und „Ereignisse“. Lediglich ein „SteuerelementTip-Text“ kann ihm zugewiesen werden (Registerblatt „Alle“). Soll dagegen einer der Buchstaben unterstrichen sein, damit der Benutzer mit gedrückter <Alt>-Taste das nächste Textfeld anspringen kann, muss das Et-Zeichen („&“) vor dem zu unterstreichenden Buchstaben verwendet werden.
Bezeichnungsfelder können verwendet werden, um einzelne Steuerelemente, die der Dateneingabe dienen, zu beschriften oder um das Formular zu gestalten. Das Formular kann so einen formatierten Titel erhalten.

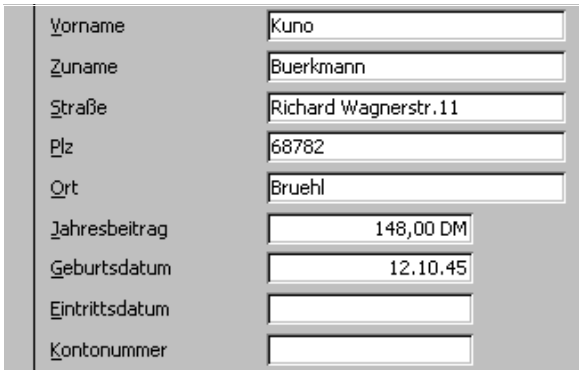


Abbildung 7.10 Textfelder und Bezeichnungsfelder

7.6.8 Kontrollkästchen und Umschaltflächen

Wurde in der Tabelle ein Ja/Nein-Feld definiert, kann für dieses ein Kontrollkästchen verwendet werden. Es wird auf dem Formular erzeugt und an ein solches Feld gebunden. Hier finden sich dieselben Formatierungskriterien, wie sie bei den Textfeldern auftauchen. Auch hier muss zwischen dem Kontrollkästchen und dem begleitenden Bezeichnungsfeld unterschieden werden.
Es existiert kein nennenswerter Unterschied zwischen Kontrollkästchen und Umschaltflächen – lediglich das Aussehen ist anders.

7.6.9 Optionsgruppe und Optionsfeld

Um Optionsfelder zu verwenden, kann der Assistent benutzt werden. Im Folgenden soll die Funktionsweise der Optionsfelder allerdings ohne Assistent gezeigt werden.
Gegeben sei ein Feld in einer Tabelle, in der nur einige wenige unterschiedliche Daten stehen – beispielsweise das Feld „Geschlecht“, in dem 1 für Frauen und 2 für Männer festgelegt wurde. Man könnte es beliebig erweitern, beispielsweise 3 für Firmen und so weiter.
Zuerst wird auf das Formular eine Optionsgruppe gezogen. Auch sie besitzt ein Bezeichnungsfeld, welches beschriftet und verschoben werden kann. Es ist nicht nötig, könnte also, wie alle Bezeichnungsfelder, gelöscht werden. Nun wird das erste Optionsfeld auf dem Formular in der Optionsgruppe aufgezogen. Dabei muss die Optionsgruppe andersfarbig aufleuchten. Dies ist ein Zeichen dafür, dass das Optionsfeld nun in der Gruppe steht. Analog werden die übrigen Optionsfelder in der Gruppe erstellt. Auch sie bestehen aus zwei Teilen: aus den eigentlichen Optionsfeldern und Bezeichnungsfeldern. Sie werden verschoben, ausgerichtet, beschriftet und formatiert. Schließlich wird die Optionsgruppe über den „Steuerelementinhalt“ an das entsprechende Feld der Tabelle gebunden, beispielsweise an „Geschlecht“. Im zweiten Schritt wird nun jedem der Optionsfelder der zugehörige Wert zugewiesen – im obigen Beispiel 1 für Frauen und 2 für Männer. Die Werte werden korrekt beim Weiterblättern von Datensatz zu Datensatz angezeigt. Wird ein Wert geändert, das heißt, wird der andere Optionsbutton ausgewählt, wird der entsprechende Wert in die Tabelle geschrieben.

Abbildung 7.11 Eine Optionsgruppe

7.6.10 Kombinationsfeld und Listenfeld

Eine weitere bequeme Möglichkeit, Daten in eine Tabelle über ein Formular einzugeben, steht Ihnen über das Kombinationsfeld oder über das Listenfeld zur Verfügung. Für gebundene Formulare besteht der Hauptunterschied im Aussehen: Listenfelder sind in der Regel geöffnet, das heißt, der Benutzer sieht schon eine Reihe von Auswahlmöglichkeiten, die ihm gegeben werden. Das Kombinationsfeld muss dagegen erst geöffnet werden. Im Gegensatz dazu benötigt das Kombinationsfeld weniger Platz auf dem Formular.

Beispiel

Gehen wir vom Fall eines gebundenen Formulars aus, beispielsweise einer Namensliste. Angenommen, jedem Namen (Kunden, Mitglieder oder ähnliches) steht eine Reihe an Bankverbindungen zur Verfügung. Dann sind diese Banken in einer zweiten Tabelle gespeichert. Das Formular soll also für neue Datensätze (neue Mitglieder) alle zur Verfügung stehenden Banken anzeigen. Wird nun eine Bankverbindung ausgewählt, wird eine interne Nummer (in der Regel der Schlüssel) in der Namenstabelle in eine dafür eingerichtete Spalte eingetragen. Lässt sich der Benutzer umgekehrt einen Namen anzeigen, wird nicht die interne Nummer, sondern die zugehörige Bankbezeichnung angezeigt. Die Nummer ist schließlich nur ein interner Wert, der keinerlei Bedeutung für denjenigen hat, der die Daten pflegt.

Normalerweise wird bei solch einer 1:n-Beziehung eine referentielle Integrität im Beziehungsfenster erstellt. Dies ist für mögliche andere Eingabearten nötig und verhindert falsche Löschungen in der Bankentabelle. Um ein Dropdownfeld auf einem Formular zu erstellen, besteht jedoch keine Notwendigkeit, mit einer solchen Beziehung zu arbeiten. Allerdings sollten Sie präventiv dennoch eine referentielle Integrität erstellen, da in neuen Abfragen, in denen beide Tabellen verwendet werden, dann schon die Beziehung zwischen beiden Tabellen besteht.

Um ein Kombinations- oder Listenfeld zu erzeugen, verwenden Sie am besten den Assistenten. Mit seiner Hilfe wird der dazugehörige SQL-Code erzeugt. Ist der Assistent eingeschaltet, kann ein neues Kombinationsfeld oder Listenfeld auf dem Formular erzeugt werden. Die erste Frage lautet, ob das Feld die Werte einer Tabelle oder Abfrage entnehmen soll. Selbstverständlich soll es das. Danach wird die entsprechende Tabelle ausgewählt. Schließlich folgt die Frage nach den Feldern, die in das Listenfeld oder in das Kombinationsfeld einbezogen werden sollen. Wählen Sie auch den Schlüssel aus, dessen Wert in einem Tabellenfeld gespeichert werden soll. Zusätzlich alle Informationen, die angezeigt werden sollen. Der erste Wert, der gewählt wurde, steht sichtbar im Kombinationsfeld. Beim Listenfeld ist die Reihen-

folge weniger wichtig, da alle Informationen sichtbar sind. Theoretisch könnte man beliebig viele Spalten angezeigt bekommen, praktisch macht dies wenig Sinn. Wichtig sind die Informationen, über die eindeutig eine Zuordnung zum gewünschten Element getroffen werden kann. Im Beispiel der Banken sind dies die Bankleitzahl und Bankbezeichnung.
Im nächsten Schritt wird in einer Vorschau angezeigt, wie der Benutzer das Ergebnis sehen wird. Hier könnte man die Spalten verbreitern oder verschmälern. Der Assistent empfiehlt, die Schlüsselspalte auszublenden, was sinnvoll ist. Im vorletzten Schritt wird gefragt, in welchem Feld der Wert gespeichert werden soll. Hier muss natürlich das Feld ausgewählt werden, in welches der Wert hineingeschrieben wird. Der letzte Wert betrifft die Beschriftung des Feldes.

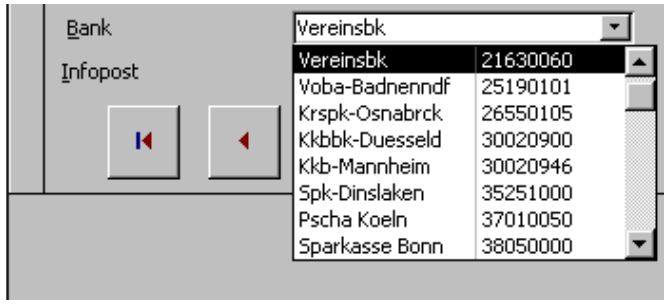


Abbildung 7.12 Ein Kombinationsfeld hilft bei der Eingabe von Daten

Was geschieht intern, wie könnte man dieses Feld ohne Assistenten erstellen? Ein Blick auf die Eigenschaften gibt die Antwort: Wurden im Bankenbeispiel eine ID-Spalte und zwei Informationsspalten ausgewählt, so zeigen die Eigenschaften im Registerblatt „Format“ drei Spalten an. Die Gesamtspaltenbreite steht unter „Breite“, sie zerfällt in „0 cm“ für die erste Spalte und zwei weitere Zahlen, deren Summe die Gesamtfeldbreite ergeben. Ist die Summe größer, wird der überstehende Rest nicht mehr angezeigt. Ist die Summe kleiner, geschieht nichts. Selbstredend sollte die Summe der einzelnen Spaltenbreiten die Breite des Feldes ergeben. Bei einem Kombinationsfeld könnten die Feldnamen als Spaltenbeschriftung verwendet werden. Er bleibt auch beim Blättern immer oben sichtbar stehen. In einem Listenfeld wird der erste Datensatz hergenommen, was wenig Sinn macht. Der Inhalt des Feldes steht im Registerblatt „Daten“ im Feld „Datensatzherkunft“. Im oben beschriebenen Beispiel könnte der SQL-Code wie folgt aussehen:

```
SELECT [tab_Bankenliste].[tBankID],
[tab_Bankenliste].[tBankBankbezeichnung],
[tab_Bankenliste].[tBankBankleitzahl] FROM tab_Bankenliste;
```

Die gebundene Spalte ist natürlich die Nummer 1. Diesen SQL-Code erhält man

auch, wenn man im Abfragefenster aus der Tabelle tab_Bankenlisten die drei Felder „BankenID“, „Bankleitzahl“ und „Bankbezeichnung“ herauszieht.
Was geschieht nun, wenn ein neuer (Namens-)Datensatz eingetragen wird, der allerdings Mitglied bei einer Bank ist, die noch nicht in der Liste steht. Man könnte den Benutzer das Formular schließen und das Bankenformular öffnen lassen, dort einen neuen Datensatz generieren, schließlich dieses Formular schließen und das erste wieder öffnen lassen. Es ist offensichtlich, dass dies keine komfortable Lösung ist, vor allem, wenn diese Aktion oft erforderlich ist. Man könnte natürlich auch ein Textfeld oder mehrere Textfelder zur Verfügung stellen, in die der Benutzer einen neuen (Banken-)Datensatz eingibt, der dann in die richtige Tabelle geschrieben wird. Diese Lösung erfordert eine Menge Programmierarbeit. Eine schnelle Lösung, die wenig Programmierung erfordert, sieht wie folgt aus:
Für die Bankennamen, die über das Kombinationsfeld zur Verfügung stehen, wird ein Eingabeformular kreiert. Mittels Assistent kann eine Befehlsschaltfläche erzeugt werden, die das Bankenformular öffnet, wenn der Benutzer darauf klickt. Dies funktioniert. Man könnte das Formular allerdings auch über einen Doppelklick auf das Steuerelement öffnen lassen. Dann wird das Ereignis „Beim Doppelklicken“ verwendet. Der dahinterliegende Code könnte wie folgt aussehen:

```
DoCmd.OpenForm "frm_Bankenliste"
DoCmd.GoToRecord , , acNewRec
Forms("frm_Bankenliste").Controls("txtBLZ").SetFocus
```

Das andere Formular wird geöffnet und kann wieder geschlossen werden. Allerdings werden neu eingegebene Daten nicht im anderen Formular im Listenfeld oder in der Kombinationsschaltfläche angezeigt. Der Benutzer müsste die Funktionstaste ⟨F9⟩ zur Aktualisierung drücken. Dieser Befehl kann allerdings automatisiert werden.
Dem neuen Formular wird im Ereignis „Beim Schließen“ folgender Befehl eingefügt:

Beispiel

```
Private Sub Form_Close()
Forms("frmMitglieder").Refresh
End Sub
```

Wird allerdings das Bankenformular geöffnet, ohne dass das Mitgliederformular offen ist, führt dieser Befehl natürlich zu einer Fehlermeldung. Man könnte dieses Banken-Formular nur dem Mitgliederformular zur Verfügung stellen und für andere Zwecke andere Formulare kreieren. Keine elegante Lösung. Oder man fügt den Befehl

```
On Error Resume Next
```

vor den Refresh-Befehl. Was natürlich nicht sehr clever ist. Etwas geschickter ist es,

den Refresh-Befehl in dem Ereignis „Beim Aktivieren“ des Mitgliederformulars unterzubringen. Also folgendermaßen:

Beispiel

```
Private Sub Form_Activate()
  Me.Refresh
End Sub
```

Dateizugriff in Access

Normalerweise ist es nicht nötig, auf Daten zuzugreifen, die außerhalb von Access liegen, da alles in Tabellen gespeichert wird. Normalerweise ist es auch nicht nötig, Listenfelder füllen zu lassen, ohne dass eine Tabelle verwendet wird. Es gibt allerdings einen Fall, der ein solches Vorgehen rechtfertigt.
Aus einem Großrechner oder einer anderen „großen“ Datenbank (SAP?) werden regelmäßig Daten in ein Textformat exportiert, die in Access zur Verfügung stehen sollen. Diese Dateien tragen nun unterschiedliche Namen und befinden sich (meistens) in einem bestimmten Ordner. Auf einem Formular in einem Listenfeld sollen diese Dateien angezeigt werden, sodass der Benutzer eine (oder mehrere?) davon auswählen kann, die mit einem bestimmten Filter importiert werden.
Ein Standardpfad wird in einer Tabelle gespeichert:

Abbildung 7.13 Laufwerk und Pfad sind in einer Tabelle gespeichert

Von dort können sie nun (über ein Formular) geändert werden:

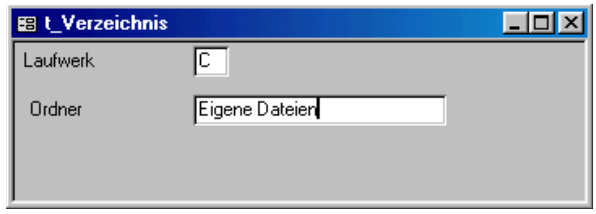


Abbildung 7.14 ... nicht schön, aber es funktioniert ...

In einem zweiten Formular werden nun in einem Listenfeld alle Dateien oder wahlweise alle Textdateien angezeigt, die sich in diesem Ordner befinden. Die Dateien stehen am Ende in der Form
A.txt;B.txt;C.txt
in der Datensatzherkunft. Der „Herkunftstyp“ wird auf „Werteliste“ gestellt.

Beispiel

```
Sub ListeFüllen(fAlle As Boolean)
Dim db As DAO.Database
Dim rs As DAO.Recordset
Dim strInhalt As String
Dim strName1 As String
Dim strListe As String

Set db = CurrentDb
Set rs = db.OpenRecordset("t_Verzeichnis")

strInhalt = rs.Fields(1).Value & ":\" & rs.Fields(2).Value


If Right(strInhalt, 1) <> "\" Then
  strInhalt = strInhalt & "\"
End If

If Dir(strInhalt) = "" Then
  MsgBox "Das Laufwerk oder der Ordner ist ungültig!",
  vbInformation
  Exit Sub
End If

strName1 = Dir(strInhalt, vbNormal)

Do While strName1 <> ""
  If strName1 <> "." And strName1 <> ".." Then
    If (GetAttr(strInhalt & strName1) And _
      vbNormal) = vbNormal Then
      If fAlle = True Then
        strListe = strListe & ";" & strName1
      Else
        If Right(strName1, 3) = "txt" Then
```

```
                strListe = strListe & ";" & strName1
              End If
            End If
          End If
        End If

        strName1 = Dir

    Loop
    strListe = Right(strListe, Len(strListe) - 1)
    lstDateien.RowSource = strListe

    Set rs = Nothing
    Set db = Nothing

    End Sub
```

Diese Prozedur wird beim Starten aufgerufen:

```
Private Sub Form_Load()
Me.fraDateityp.Value = 1
ListeFüllen (True)
End Sub
```

Damit er die Auswahl zwischen allen Dateien und den Textdateien hat, wurden zwei Optionsfelder eingefügt:

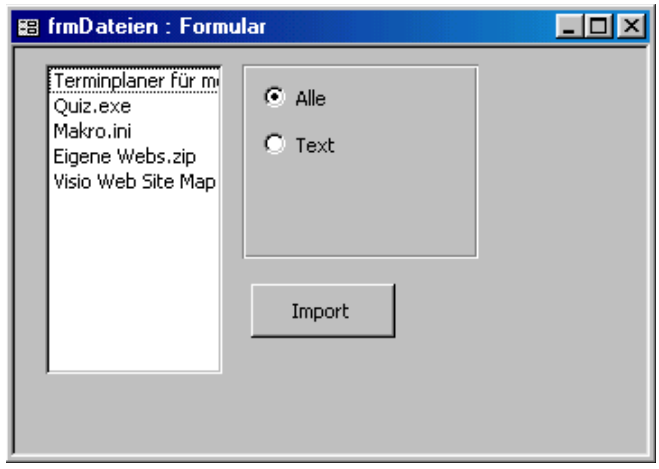


Abbildung 7.15 Die Liste der Dateien

Hinter den beiden Optionsfeldern liegt folgender Code:

```
Private Sub optAlle_MouseUp(Button As Integer, _
   Shift As Integer, X As Single, Y As Single)
ListeFüllen (True)
End Sub

Private Sub optText_MouseUp(Button As Integer, _
   Shift As Integer, X As Single, Y As Single)
ListeFüllen (False)
End Sub
```

Nun kann der Import starten, für den allerdings ein Filter definiert werden muss:

```
Private Sub cmdImport_Click()
If Me.lstDateien.ListIndex = -1 Then
   MsgBox "Bitte erst einen Eintrag auswählen!", vbInformation
Else
   MsgBox "Jetzt wird die Datei " & _
      Me.lstDateien.Value _
      & " importiert"
   ' -- und hier folgen nun die Befehle zum Import:

   ' --   DoCmd.TransferText TransferType:=acLinkDelim, _
   ' -- SpecificationName:="FilterName",
   ' --TableName:="NameDerNeuenTabelle", _
         FileName:=Me.lstDateien.Value, HasFieldNames:=True
   ' -- Der FilterName und der NameDerNeuenTabelle müssen noch
   ' -- angepasst werden

End If
End Sub
```

7.6.11 Bedingte Formatierung

In Access steht Ihnen die Möglichkeit zur Verfügung, Daten in Abhängigkeit von bestimmten Kriterien anzuzeigen. Sie wählen dann aus dem Menü FORMAT – BEDINGTE FORMATIERUNG eine der drei Optionen: „Feldwert ist“, „Ausdruck ist“ oder „Feld hat Fokus“. Daneben muss das zugehörige Feld gewählt werden. Schließlich werden noch die Werte benötigt, gegen die der Wert geprüft wird, und die zugehörige Formatierung, die Sie wünschen.

7.6.12 Tipps zu den Steuerelementen

Tipps & Tricks

Wenn ein Tool aus der Toolbox aktiv gehalten wird, brauchen Sie nicht wiederholt darauf zu klicken, um eine bestimmte Aktion mehrmals auszuführen. Wenn Sie zum Beispiel einem Formular mehrere Bezeichnungsfelder hinzufügen möchten, können Sie das Tool Bezeichnung durch einen Doppelklick in der Toolbox aktiv halten.

Sie können für Felder vom Datentyp Text, Zahl und Ja/Nein einen Standardsteuerelementtyp festlegen. Wenn Sie für ein Feld einen Standardsteuerelementtyp festlegen, zeigt Microsoft Access dieses Steuerelement im Datenblatt der Tabelle an und erstellt diesen Steuerelementtyp automatisch immer dann, wenn Sie dieses Feld zu einem Formular oder Bericht hinzufügen. Für Felder vom Datentyp Text und Zahl kann als Standardsteuerelementtyp ein Textfeld, ein Kombinationsfeld oder ein Listenfeld (nur in Formularen) festgelegt werden. Für Felder vom Datentyp Ja/Nein kann als Standardsteuerelementtyp ein Kontrollkästchen, ein Textfeld oder ein Kombinationsfeld festgelegt werden.
Wenn Sie das Feld im Datenblatt einer Tabelle als Kombinationsfeld, im Datenblatt einer Abfrage jedoch als Textfeld anzeigen möchten, können Sie zusätzlich zur Tabelleneigenschaft die Eigenschaft „Nachschlagen“ für das Feld im Abfrageentwurf einstellen.

7.6.13 Dateneingabe begrenzen

Auch ohne Programmierung kann die Dateneingabe begrenzt werden. Das Formular stellt die Eigenschaften „Bearbeiten zulassen“, „Anfügen zulassen“ und „Löschen zulassen“ zur Verfügung.
Ebenso kann ein einzelnes Steuerelement gesperrt werden. Sie finden die Eigenschaft „Gesperrt“. Dies bedeutet, dass das Feld zwar den Fokus besitzen (Inhalte können herauskopiert werden), aber nicht verändert werden kann. Wird die Eigenschaft „Aktiviert“ auf Nein“ gesetzt, wird auch beim Drücken der Tabulatortaste das Feld übergangen. Der Benutzer sieht lediglich den Inhalt des Feldes, ohne den Cursor hinein platzieren zu können. Bei diesen beiden Möglichkeiten kann die Hintergrundfarbe des Feldes auf Grau gesetzt werden. Damit weiß der Benutzer, dass die Eingabe von neuen oder das Verändern vorhandener Daten nicht möglich ist.

Tipps & Tricks

Noch einfacher ist es, das Feld auf unsichtbar zu setzen. Damit stehen für die Programmierung noch die Eigenschaften des Feldes zur Verfügung (Werte können hineingeschrieben werden), allerdings sieht der Benutzer nicht, was verborgen „im Hintergrund“ abläuft.

7.6.14 Unterformulare

Eine besondere Art der Formulare stellen in Access Unterformulare dar. Der zugrunde liegende Gedanke ist folgender: Gegeben seien zwei Tabellen, die in einer 1:n-Beziehung zueinander stehen. Ein einspaltiges Hauptformular zeigt jeweils einen Datensatz der Daten der 1-seitigen Tabelle an. Im Beispiel Mitglieder-Banken wäre dies die Bankentabelle. Alle Datensätze sollen nun auf dem Formular angezeigt werden, die mit der anderen Tabelle verknüpft sind. Wenn also 20 Personen Mitglieder bei Bank Nummer 1 sind, sollen diese angezeigt werden. Wenn niemand Mitglied bei Bank Nummer 2 ist, wird bei Nummer 2 niemand angezeigt und bei Nummer 3 gibt es nur ein Mitglied, also wird nur dieses eine angezeigt. Dazu müssen zwei Formulare erstellt werden: ein einspaltiges für die 1-Seite (Banken) und ein tabellarisches Formular für die n-Seite (Mitglieder). Man könnte aus dem Datenbankfenster das Symbol des tabellarischen Formulars auf die Entwurfsansicht des einspaltigen Formulars ziehen. Man könnte aber auch in der Entwurfsansicht des einspaltigen Formulars das Steuerelement Unterformular/-bericht aus der Toolbox auf dem Formular erzeugen. Ohne Assistent muss in den Eigenschaften eine Einstellung im Registerblatt „Daten“ vorgenommen werden: das Herkunftsobjekt ist das „neue“ Formular, das heißt, das Formular, das eingebettet wird. Bei „Verknüpfen von“ oder bei „Verknüpfen nach“ müssen die beiden Felder eingegeben werden, die verknüpft sind. Dabei ist „Verknüpfen von“ in der Regel der Primärschlüssel der Tabelle, auf der das tabellarische Formular basiert. „Verknüpfen nach“ ist der n-seitige Wert der anderen Tabelle des „Oberformulars“. Es genügt, in einem der beiden Felder den Wert zu ändern, da die Änderungen im anderen Feld automatisch erscheinen.

Warnung

Es können keine Formatierungen des Unterformulars im Hauptformular vorgenommen werden; sollen die Spalten vergrößert oder verkleinert werden, dann muss dies in dem Formular gestaltet werden, das an dieser Stelle verwendet wird.

Abbildung 7.16 Ein Unterformular

7.6.15 Ungebundene Formulare

Einfache Steuerungen auf einem ungebundenen (oder gebundenen) Formular können mit Hilfe des Assistenten erzeugt werden. Wird eine Befehlsschaltfläche auf dem Formular geschaffen, fragt der Assistent nach der Aktion, die ausgelöst wird, wenn der Benutzer auf die Schaltfläche klickt.

Wird die Navigationsschaltfläche ausgeschaltet, könnte man vier (oder fünf) Befehlsschaltflächen erzeugen, mit denen man zum ersten, vorherigen, nächsten und zum letzten Datensatz springen kann. Mit einem fünften könnte man einen neuen Datensatz erzeugen lassen.

Die einzig nicht ganz sauber arbeitende Befehlsschaltfläche ist diejenige, mit der man einen Datensatz zurück springen kann. Der automatisch eingefügte Code lautet:

```
Private Sub cmdVor_Click()
On Error GoTo Err_cmdVor_Click

    DoCmd.GoToRecord , , acPrevious

Exit_cmdVor_Click:
    Exit Sub

Err_cmdVor_Click:
    MsgBox Err.Description
    Resume Exit_cmdVor_Click
End Sub
```

Das bedeutet: Steht der Datensatzzeiger auf dem ersten Datensatz, versucht der Befehl

```
    DoCmd.GoToRecord , , acPrevious
```

einen Datensatz zurückzuspringen. Da dies nicht möglich ist, ist eine Fehlermeldung die Folge. Sie wird abgefangen und angezeigt:

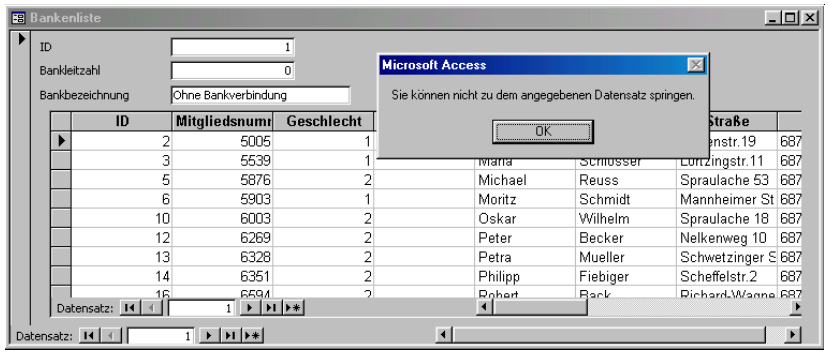


Abbildung 7.17 Die Fehlermeldung

Programmiertechnisch nicht elegant, aber durchaus effektiv ist folgende Lösung: Bei einer Fehlermeldung geschieht nichts.

```
Private Sub cmdVor_Click()
On Error Resume Next
    DoCmd.GoToRecord , , acPrevious
End Sub
```

Damit wird der Fehler übergangen – und nichts passiert! Eleganter natürlich wäre zu überprüfen, ob das Formular den ersten Datensatz anzeigt. Dies geschieht mit folgenden Befehlen:

Beispiel

```
Private Sub cmdVorhergehender_Click()
If Me.CurrentRecord = 1 Then
  MsgBox "Sie befinden sich auf dem ersten Datensatz"
Else
  DoCmd.GoToRecord , , acPrevious
End If
End Sub
```

Für ungebundene Formulare sind die Formularoperationen wichtig. Man könnte mit Hilfe einer Befehlsschaltfläche ein Formular öffnen oder (das aktuelle) schließen. Einige weitere Optionen, die hier nicht behandelt werden, stehen zur Verfügung: Berichte öffnen, Berichte oder Tabellen drucken oder eine andere Anwendung starten.

Die dahinterliegenden Befehle finden sich in den Eigenschaften im Registerblatt „Ereignis“ in der Zeile „Beim Klicken“. Klicken Sie dort auf die drei Pünktchen, die sich hinter der Zeile befinden, gelangen Sie in den Visual Basic-Editor, in dem sich der Befehl befindet. Die folgende Tabelle listet alle Methoden des DoCmd-Befehls auf:

Tabelle 7.7 Die Methoden des DoCmd-Befehls

Befehl	Beschreibung
Close	schließt ein Objekt (beispielsweise ein Formular) Syntax: DoCmd.Close [Objekttyp, Objektname], [Speichern]
FindRecord	springt auf einen bestimmten Datensatz Syntax: DoCmd.FindRecord Suchen nach[, Vergleichen] [, Groß-/Kleinschreibung][, Suchen][, Wie formatiert] [, Nur aktuelles Feld][, Am Anfang beginnen]
GoToRecord	wechselt zu einem bestimmten Datensatz Syntax: DoCmd.GoToRecord [*Objekttyp*, *Objektname*][, *Datensatz*] [, *Offset*] Die Methode GoToRecord verwendet die folgenden Argumente: ArgumentBeschreibung *Objekttyp* Eine der folgenden eingebauten Konstanten: acActiveDataObject (Standardwert) acDataForm acDataQuery acDataTable *Objektname* Ein Zeichenfolgenausdruck, der den gültigen Namen eines Objekts vom Typ *Objekttyp* angibt. *Datensatz* Eine der folgenden eingebauten Konstanten: acFirst acGoTo acLast acNewRec acNext (Standardwert) acPrevious *Offset* Ein numerischer Ausdruck, der die Anzahl der Datensätze angibt, um die vorwärts oder rückwärts geblättert werden soll, wenn Sie für das Argument Datensatz acNext oder acPrevious angegeben haben, oder der Datensatz, zu dem Sie gehen möchten, wenn Sie für das Argument Datensatz acGoTo angegeben haben. Der Ausdruck muss eine gültige Datensatznummer ergeben.

Tabelle 7.7 (Fortsetzung) Die Methoden des DoCmd-Befehls

Befehl	Beschreibung
OpenForm	öffnet ein Formular Syntax DoCmd.OpenForm Formularname [, Ansicht][, Filtername] [, Bedingung][, Datenmodus][, Fenstermodus] [, Öffnungsargumente]
AddMenu	fügt ein Menü hinzu
ApplyFilter	führt einen Filter aus
Beep	führt einen Signalton aus
CancelEvent	bricht Ereignisse ab
CopyObject	kopiert ein Objekt (beispielsweise eine Tabelle)
DoMenuItem	führt einen Befehl aus einem Menü aus
DeleteObject	löscht ein Objekt
Echo	schaltet die Aktualisierung der Bildschirmanzeige ein oder aus
FindNext	öffnet den Suchen-Dialog
GoToControl	setzt den Focus auf ein bestimmtes Steuerelement
GoToPage	blättert zu einer bestimmten Seite
Hourglass	zeigt die Sanduhr oder schaltet auf den Standardmauszeiger um
Maximize	vergrößert ein Fenster
Minimize	verkleinert ein Fenster
MoveSize	verschiebt ein Fenster
OpenDataAccessPage	öffnet eine Datenzugriffsseite
OpenDiagram	öffnet ein Diagramm
OpenModule	öffnet ein Modul
OpenQuery	öffnet eine Abfrage
OpenReport	öffnet einen Bericht
OpenStoredProcedure	öffnet eine Prozedur
OpenTable	öffnet eine Tabelle
OpenView	öffnet eine Sicht
OutputTo	exportiert eine Tabelle oder Abfrage
PrintOut	druckt
Quit	beendet Access
Rename	benennt ein Objekt um
RepaintObject	führt Bildschirmaktualisierungen eines Formulars aus
Requery	aktualisiert Daten
Restore	stellt die ursprüngliche Größe eines Fensters wieder her

Tabelle 7.7 (Fortsetzung) Die Methoden des DoCmd-Befehls

Befehl	Beschreibung
RunCommand	führt einen eingebauten Menübefehl aus
RunMacro	startet ein Makro
RunSQL	startet eine SQL-Abfrage
SelectObject	setzt den Focus auf ein bestimmtes Objekt (Tabelle, Abfrage, Formular, …)
Save	speichert ein Objekt
SendObject	versendet ein Objekt als Anhang einer email
SetMenuItem	ändert einen Menüeintrag
SetWarnings	schaltet Warnmeldungen ein oder aus
ShowAllRecords	zeigt alle Datensätze an und schaltet somit den Filter aus
ShowToolbar	zeigt eine Symbolleiste an
TransferDatabase	importiert, exportiert oder verknüpft eine Tabelle oder Abfrage
TransferSpreadsheet	importiert eine Tabelle
TransferText	importiert Text

Der Assistent schreibt einen Code in den VBA-Editor, der wie folgt aussieht:

```
Private Sub Befehl43_Click()
On Error GoTo Err_Befehl43_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "frm_Bank"

    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Befehl43_Click:
    Exit Sub

Err_Befehl43_Click:
    MsgBox Err.Description
    Resume Exit_Befehl43_Click

End Sub
```

Die Zeile

```
DoCmd.OpenForm "frm_Bank"
```

hätte genügt. Der Rest sind Befehle, die mögliche Fehler abfangen, die zur Laufzeit entstehen.

Um ein Suchenfeld zu erstellen, ist allerdings noch etwas Wissen und Überblick über die Access-Programmierung nötig.

7.7 Access-Programmierung

Access stellt eine ganze Reihe von Hilfsmitteln zur Verfügung, die das Arbeiten erleichtern. Gerade für Anfänger der Datenbankprogrammierung stellen diese wertvolle Hilfen dar. Dennoch gibt es einige wichtige und grundlegende Dinge bei der Datenbankprogrammierung zu beachten:

Access unterscheidet zwei verschiedene Typen von Objekten: die Microsoft-Access-Objekte und die Datenzugriffsobjekte DAO (Data Access Object). Mit Access 2000 wird eine weitere Datenzugriffstechnologie eingeführt: ActiveX Database Objects (ADO). Obwohl ADO Zugriff auf eine größere Anzahl von Datenquellen bietet als DAO und sogar einige Funktionen der Jet 4.0-Datenbank-Engine offen legt, die über DAO nicht verfügbar sind, gibt es bei der Verwendung von ADO in Verbindung mit Access-Datenbanken einige Beschränkungen, die die Verwendung von DAO weiterhin erfordern. Außerdem ist es nicht möglich, Informationen zwischen ADO- und DAO-Code auszutauschen. Deshalb behandeln die folgenden Seiten in erster Linie das DAO-Objektmodell, das bei Access-Programmierern bekannt ist. ADO wird nur am Rande gestreift.

Die DAO-Objekte greifen auf die Datenbank zurück. Mit ihnen werden Datensätze gesucht, gelöscht, verschoben, neue Daten werden eingetragen oder vorhandene verändert. Das oberste Objekt (oder der Kern) stellt die Datenbank-Engine (DBEngine) dar.

Die MS-Access-Objekte greifen auf die Datenbank und nicht auf die Daten zu. Das heißt, in ihnen werden Einstellungen wie Menüleiste, Symbolleisten, Standarddialoge und sonstige Eigenschaften von Access selbst geregelt. Oberstes Objekt ist Application.

7.8 Die Objekte der Datenbank

Auf die aktuelle Datenbank kann man mit

```
CurrentDb
```

zugreifen.

Wie schon gezeigt, besteht die Datenbank aus einer Reihe von Elementen: aus Tabellen, Abfragen und Beziehungen. Die Kollektionen lauten:
`TableDefs`, `QueryDefs` und `Relations`
Folgendes Programm listet alle Tabellen, Abfragen und Beziehungen auf:

Beispiel

```
Sub ElementeDerDatenbank()
Dim i As Integer
Dim dbAktuelleDB As Database
Dim strAusgabeTextTabellen As String
Dim strAusgabeTextAbfragen As String
Dim strAusgabeTextBeziehungen As String

On Error Resume Next

Set dbAktuelleDB = CurrentDb

For i = 0 To dbAktuelleDB.TableDefs.Count - 1
  strAusgabeTextTabellen = strAusgabeTextTabellen & _
    ", " & dbAktuelleDB.TableDefs(i).Name & " /"
Next

For i = 0 To dbAktuelleDB.QueryDefs.Count - 1
  strAusgabeTextAbfragen = strAusgabeTextAbfragen & _
    dbAktuelleDB.QueryDefs(i).Name & " /"
Next

For i = 0 To dbAktuelleDB.Relations.Count - 1
  strAusgabeTextBeziehungen = _
    strAusgabeTextBeziehungen & _
      dbAktuelleDB.Relations(i).Name & " / "
Next

MsgBox "Die Datenbank " & dbAktuelleDB.Name & _
  vbCr & vbCr & " enthält folgende Tabellen: " & vbCr & _
  vbCr & strAusgabeTextTabellen & vbCr & vbCr & _
  " und folgende Abfragen: " & vbCr & vbCr & _
    strAusgabeTextAbfragen & vbCr & vbCr & _
    " und folgende Beziehungen: " & vbCr & vbCr & _
    strAusgabeTextBeziehungen

End Sub
```

Hinweis

`CurrentDB` und `DBEngine(0)(0)` sind synonym.

Man hätte jede Schleife auch über Objekte laufen lassen können:

```
Dim dbTabelle As TableDef
Dim dbAbfrage As QueryDef
Dim dbBeziehung As Relation
For Each dbTabelle In dbAktuelleDB.TableDefs
  strAusgabeTextTabellen = strAusgabeTextTabellen & _
", " & dbTabelle.Name & " /"
Next
[...]
```

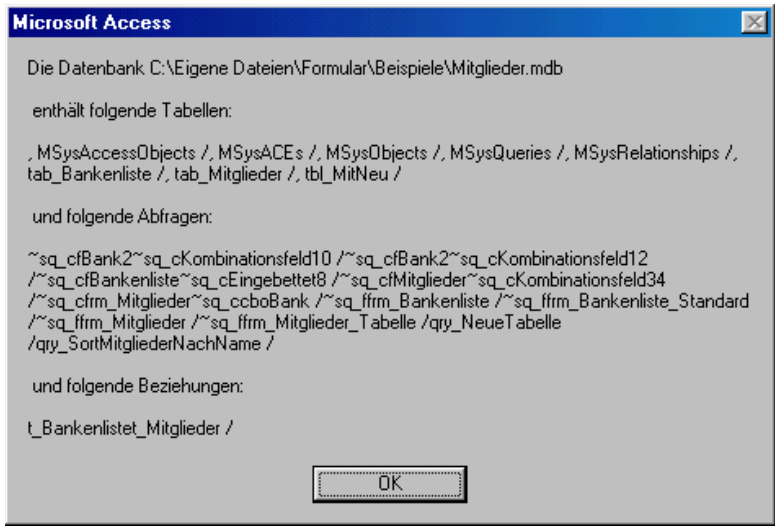


Abbildung 7.18 Alle Tabellen, Abfragen und Beziehungen

7.8.1 Field

Tabellen und Abfragen wiederum besitzen Spalten und Zeilen oder Felder und Datensätze. Auch sie können abgefragt werden:

Beispiel

```
Sub Felder()
Dim dbAktuelleDB As Database
Dim dbTabelle As TableDef
Dim dbFeld As Field
Dim i As Integer
```

```
Dim strTabname As String, strFelder As String
On Error Resume Next

Set dbAktuelleDB = CurrentDb
Set dbTabelle = dbAktuelleDB.TableDefs("t_Mitglieder")

strTabname = dbTabelle.Name

For Each dbFeld In dbTabelle.Fields
   strFelder = strFelder & dbFeld.Name & " / "
Next

MsgBox "Die Tabelle """ & strTabname & _
   """ enthält folgende Spalten:" & vbCr & vbCr & strFelder
End Sub
```

Abbildung 7.19 Die Auflistung der Datenfelder mit dem Objekt Fields

Und ebenso können die Anzahl der Datensätze einer bestimmten Tabelle angezeigt werden:

```
CurrentDb.TableDefs(0).RecordCount
```

7.8.2 Deklaration von DAO-Objekten

Tipps & Tricks

Statt der langen Tipparbeit können auch Objektvariablen deklariert werden, an welche die einzelnen Objekte übergeben werden. Dabei stehen für eine Tabelle, Abfrage oder Beziehung die Befehle `TableDefs` (`TableDef`), `QueryDefs` (`QueryDef`) und `RelationDefs` (`RelationDef`) zur Verfügung. Daneben existieren noch `Fields` (`Field`), `Forms` (`Form`) für die Formulare, `Reports` (`Reports`) für die Berichte, `Users` (`User`) und `Errors` für die einzelnen Fehler.

Man kann per Programmierung neue Datenbanken erzeugen lassen. Das folgende Beispiel generiert eine Filmdatendank mit zwei Tabellen, die über eine 1:n-Beziehung verknüpft sind. Jede der beiden Tabellen verfügt über Felder unterschiedlichen Typs:

Beispiel

```
Sub Neue_Datenbank()
Dim dbNeu As Database
Dim dbFilmliste As TableDef
Dim dbKategorienListe As DAO.TableDef
Dim dbFeldNeu As DAO.Field
Dim IndexNr As DAO.Index
Dim dbBeziehung As DAO.Relation

If Dir("c:\Filme.mdb") <> "" Then
  Kill "c:\Filme.mdb"
End If

'Die Datenbank wird neu erzeugt
Set dbNeu = DBEngine.Workspaces(0).CreateDatabase("Filme", _
  dbLangGeneral)

'Die erste Tabelle wird neu erzeugt
Set dbFilmliste = dbNeu.CreateTableDef("t_Filme")

'Ein Zählerfeld wird erzeugt
Set dbFeldNeu = dbFilmliste.CreateField("film_ID", dbLong)
dbFeldNeu.Attributes = dbAutoIncrField
dbFilmliste.Fields.Append dbFeldNeu

'Die übrigen Felder werden erzeugt
Set dbFeldNeu = dbFilmliste.CreateField("film_titel", _
  dbText, 100)
dbFilmliste.Fields.Append dbFeldNeu
Set dbFeldNeu = dbFilmliste.CreateField("film_jahr", _
  dbInteger)
dbFilmliste.Fields.Append dbFeldNeu
Set dbFeldNeu = dbFilmliste.CreateField("film_regisseur", _
  dbText, 50)
dbFilmliste.Fields.Append dbFeldNeu
```

```
Set dbFeldNeu = dbFilmliste.CreateField("film_kategorie", _
  dbLong)
dbFilmliste.Fields.Append dbFeldNeu
'Die Tabelle wird an die Datenbank angehängt
dbNeu.TableDefs.Append dbFilmliste

'Ein Index wird erzeugt
Set IndexNr = dbFilmliste.CreateIndex("film_ID")
Set dbFeldNeu = IndexNr.CreateField("film_ID", dbLong)
IndexNr.Primary = True
IndexNr.Required = True
IndexNr.Fields.Append dbFeldNeu
dbFilmliste.Indexes.Append IndexNr

'Die zweite Tabelle
Set dbKategorienListe = dbNeu.CreateTableDef("t_Kategorien")

'Der Zähler
Set dbFeldNeu = dbKategorienListe.CreateField("kat_ID", _
  dbLong)
dbFeldNeu.Attributes = dbAutoIncrField
dbKategorienListe.Fields.Append dbFeldNeu

'Hier nur ein Feld
Set dbFeldNeu = _
  dbKategorienListe.CreateField("Kategorien_Beschreibung", _
  dbText, 100)
dbKategorienListe.Fields.Append dbFeldNeu

'Die Tabelle wird angehängt
dbNeu.TableDefs.Append dbKategorienListe

'Auch die zweite Tabelle erhält einen Index
Set IndexNr = dbKategorienListe.CreateIndex("kat_ID")
Set dbFeldNeu = IndexNr.CreateField("kat_ID", dbLong)
IndexNr.Primary = True
IndexNr.Required = True
IndexNr.Fields.Append dbFeldNeu
dbKategorienListe.Indexes.Append IndexNr
```

```
' Eine 1:n-Beziehung wird zwischen beiden
' Datenbanken definiert
Set dbBeziehung = dbNeu.CreateRelation("relFilmKat", _
    "t_Kategorien", "t_Filme")
Set dbFeldNeu = dbBeziehung.CreateField("kat_ID")
dbFeldNeu.ForeignName = "film_kategorie"
dbBeziehung.Fields.Append dbFeldNeu
'Die Beziehung wird in die Datenbank eingefügt
dbNeu.Relations.Append dbBeziehung

'Die Datenbank wird geschlossen
dbNeu.Close

End Sub
```

Hinweis

Einige Erläuterungen hierzu: Wenn Sie eine Datenbank neu anlegen und das Programm ein zweites Mal ablaufen lassen, werden Sie eine Fehlermeldung erhalten, da die Datenbank bereits existiert. Sie muss zuerst gelöscht werden. Die Access-Endung *.mdb wird dabei automatisch vergeben. Eine neue Tabelle wird mit dem Befehl `CreateTableDef` erzeugt. Dies allein genügt jedoch nicht: Erst wenn die Felder definiert sind, kann und muss die Tabelle über die Methode `Append` angefügt werden. Das Gleiche gilt für die Felder. Jedes Feld wird definiert und dann über `Append` angefügt. Der ersten Spalte wird ein Index zugewiesen, dieser wird ebenfalls mit `Append` angefügt. Schließlich wird eine 1:n-Beziehung erzeugt, wobei in der Syntax

```
Set Relation = Datenbank.CreateRelation([Name _
   [,Tabelle [, Fremdtabelle [,Attribute]]]])
```

mit Tabelle die 1-Seite der Beziehung gemeint ist, mit Fremdtabelle die n-Seite. So sind die beiden folgenden Anweisungen zu lesen:

```
Set dbFeldNeu = dbBeziehung.CreateField("kat_ID")
dbFeldNeu.ForeignName = "film_kategorie"
```

`kat_ID` ist der Primärschlüssel, `film-kategorie` das n-seitige Feld von Zahlentyp `long`.

Tabelle 7.8 Die DAO-Objekte und ihre Hierarchie

Name	Bedeutung
DBEngine	Oberstes Objekt
Errors	Fehler. Objekt von DBEngine
Workspaces	Hier werden Datenbanken, Benutzer und Gruppen verwaltet. Objekt von DBEngine. Kommt in der Regel nur als Workspace(0) vor
Users	Der Benutzer. Objekt von Workspaces
Groups	Die Gruppe. Objekt von Workspaces
Databases	Zusammenfassung der Tabellen, Abfragen und Beziehungen. Objekt von `Workspaces(0).DBEngine.Workspaces(0).Databases(0)` entspricht `DBEngine(0)(0)` oder `CurrentDB`
TableDefs	Die Tabellen. Objekt von Databases
Fields und Indexes	Die Felder (Spalten) und die Indices. Objekte von TableDefs
QueryDefs	Die Abfragen. Objekt von Databases
Relations	Beziehungen. Objekt von Databases
RecordSets	Ermöglicht den Zugriff auf Tabellen. Objekt von Databases
Daneben: Containes, Connections	

7.8.3 RecordSet

Wenn Sie auf Tabellen oder Abfragen zurückgreifen wollen, so steht Ihnen das Objekt Recordset zur Verfügung. Es wird deklariert:

```
Dim dbRs As recordset
```

Da Recordset auch ein Objekt ist, wird mit `Set` an diese Variable (hier: `dbRs`) ein Objekt übergeben, das zuerst geöffnet wird. Das Öffnen erledigt die Methode `OpenRecordSet`

Es stehen Ihnen zwei Möglichkeiten zur Verfügung:

```
Set dbRs = Datenbank.OpenRecordset(Name, Typ, Optionen)
```

oder

```
Set dbRs = Objekt.OpenRecordset(Typ, Optionen)
```

Dabei sind Typ und Optionen optional.

Beispiel

Die Tabelle „tab_Mitglieder“ kann geöffnet werden mit:

```
Set dbRs = _
DBEngine(0)(0).OpenRecordset("tab_Mitglieder", dbOpenTable)
```

oder analog:

```
Set rs = CurrentDb.OpenRecordset("tab_Mitglieder", _
  dbOpenTable)
```

oder auch über das Objekt `TableDefs`, das eine Eigenschaft der `CurrentDB` darstellt mit seiner Methode `OpenRecordSet`:

```
Set dbRs = _
  CurrentDb.TableDefs("tab_Mitglieder").OpenRecordset
```

Dabei kann die Tabelle oder Abfrage auf verschiedene Weise geöffnet werden:

Tabelle 7.9 Die vier möglichen Typen für die Methode OpenRecordset

Typ	Alternative (die älteren Variablen)	Beschreibung
dbOpenTable	DB_OPEN_TABLE	Zugriff auf die lokale Tabelle
dbOpenDynaSet	DB_OPEN_DYNASET	Zugriff auf die eingebundene Tabelle oder Abfrage
dbOpenSnapShot	DB_OPEN_SNAPSHOT	Zugriff auf die eingebundene Tabelle oder Abfrage, nicht editierbar!
dbOpenDynamic		Für ODBC-Beriche kann ein RecordSet dynamisch geöffnet werden

Beim Öffnen legen weitere Konstanten in den Optionen fest, wie das Objekt geöffnet wird:

Tabelle 7.10 Die möglichen Optionen für die Methode OpenRecordset

Option	Beschreibung
dbDenyWrite	Keine Datensätze können vom Benutzer hinzugefügt oder verändert werden
dbDenyRead	Keine Datensätze können vom Benutzer hinzugefügt werden
dbReadOnly	Nur Leserecht
dbAppendOnly	Es können nur neue Datensätze angefügt, alte aber nicht verändert werden
dbInconsistent	Inkonsistente Aktualisierungen
dbConsistent	Konsistente Aktualisierungen
dbForwardOnly	Die Datensätze dürfen nur vorwärts durchlaufen werden
dbSQLPassThrough	eine SQL-Abfrage wird als SQL-Pass-Through weitergegeben
dbSeeChanges	löst eine Fehlermeldung aus, wenn ein anderer Benutzer die Daten ändert, die in Bearbeitung sind
dbRunAsync	Asynchrone Abfrage
dbExecDirect	Führt Abfrage durch Überspringen von SQLPrepare sowie dem direkten Aufrufen von SQLExecDirect aus

Im folgenden Beispiel wird die Tabelle „tab_Mitglieder“ nur mit Leserechten geöffnet:

Beispiel

```
Dim dbRs As recordset
Dim db As Database
Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", ,dbReadOnly)
```

Nun kann der erste Eintrag angezeigt werden:

```
MsgBox dbRs.Fields("Name").Value
```

7.8.4 Methoden und Eigenschaften des Recordset

Access stellt eine Reihe von Methoden zur Verfügung, um sich durch die Datensätze zu bewegen. Dabei wird ein Datensatzzeiger auf einen bestimmten Datensatz gesetzt:

Methoden von Recordset zum Bewegen über eine Tabelle

Methode	Beschreibung
MoveFirst	Geht zum ersten Datensatz
MoveLast	Geht zum letzten Datensatz
MoveNext	Geht zum nächsten Datensatz
MovePrevious	Geht zum vorhergehenden Datensatz
Move Zeilen, Start	Geht die Anzahl der Zeilen nach unten (bei negativen Zahlen nach oben). Start ist optional und kennzeichnet ein Lesezeichen

Beispiel

Es soll der Inhalt des letzten Datensatzes der Tabelle Filmliste angezeigt werden:

```
Sub suchMöglichkeit1()

Dim rs  As DAO.Recordset
Dim db As DAO.Database

Set db = CurrentDb
Set rs = db.OpenRecordset("tab_Mitglieder", dbReadOnly)
```

```
rs.MoveLast
MsgBox rs.Fields("tMitZuname").Value

Set rs = Nothing
Set db = Nothing
End Sub
```

Ein Datensatz wird mit der Methode `Delete` gelöscht. Zum Beispiel der letzte:

Beispiel

```
Sub LöschMöglichkeit1()

Dim rs  As DAO.Recordset
Dim db As DAO.Database

Set db = CurrentDb
Set rs = db.OpenRecordset("tab_Mitglieder")
rs.MoveLast
rs.Delete

Set rs = Nothing
Set db = Nothing
End Sub
```

Weitere Methoden:

Tabelle 7.11 Weitere Methoden von Recordset

Methode	Beschreibung
AddNew	Fügt einen neuen, leeren Datensatz an
Edit	Wechselt in den Editiermodus
CancelUpdate	Schaltet den Editiermodus oder AddNew-Vorgang aus (entspricht der Taste <esc>)
Requery	Frischt den Recordset auf
Close	Schließt den Recordset

Einige Eigenschaften des RecordSets:

Tabelle 7.12 Eigenschaften von Recordset

Eigenschaft	Beschreibung
RecordCount	Die Anzahl der Datensätze
BOF	Begin Of File. BOF ist wahr, wenn der Datensatzzeiger auf dem ersten Datensatz steht
EOF	End Of File. EOF ist wahr, wenn der Datensatzzeiger auf dem letzten Datensatz steht
AbsolutePosition	Die absolute Position – die Datensatznummer
PercentPosition	Die relative Position in Prozent
EditMode	Ist wahr, wenn ein Datensatz bearbeitet wird

Die Methoden und Eigenschaften können auf folgende Weise geschrieben werden:

```
Set rs = db.OpenRecordset("tab_Mitglieder")
MsgBox rs.Fields("Name").Value
```

zeigt den Inhalt der Spalte „Filmtitel“ des aktuellen Datensatzes an. Das gleiche bewirkt:

```
Set rs = db.OpenRecordset("Filmliste")
MsgBox rs.Fields("Filmtitel")
```

oder auch:

```
Set rs = db.OpenRecordset("Filmliste")
MsgBox rs!FilmTitel
```

oder:

```
Set rs = db.OpenRecordset("Filmliste")
MsgBox rs![FilmTitel]
```

Hinweis

Die eckigen Klammern sind unbedingt nötig, wenn der Feldname ein oder mehrere Leerzeichen enthält. Statt des Namens kann auch die Nummer der Spalte verwendet werden. Wenn „Filmtitel“ die erste Spalte ist, so wird der Inhalt auch mit:

```
Set rs = db.OpenRecordset("Filmliste")
MsgBox rs.Fields(1).Value
```

angezeigt. Da Fields eine Aufzählung ist, können alle Daten des Datensatzes durchlaufen werden.

Beispiel

```
Sub zeigeAlle1()
On Error Resume Next
Dim db As DAO.Database
Dim rs As DAO.Recordset
Dim strInhalt As String
Dim i As Integer

Set db = CurrentDb
Set rs = db.OpenRecordset("tab_Mitglieder")

For i = 0 To rs.Fields.Count - 1
  strInhalt = strInhalt & rs.Fields(i).Value & "; "
Next

MsgBox strInhalt
Set rs = Nothing
Set db = Nothing

End Sub
```

Mit der Eigenschaft `Count` wird die Anzahl der Spalten bestimmt. Da die Zählung bei Null beginnt, muss der Zähler auf Null gesetzt werden und läuft von Null bis zur Anzahl – 1 hoch. Genauso funktioniert das Anzeigen mit einer For ... Each-Schleife:

```
Sub zeigeAlle2()
On Error Resume Next
Dim db As DAO.Database
Dim rs As DAO.Recordset
Dim fld As DAO.Field
Dim strInhalt As String

Set db = CurrentDb
Set rs = db.OpenRecordset("tab_Mitglieder")

For Each fld In rs.Fields
   strInhalt = strInhalt & fld.Value & "; "
Next
MsgBox strInhalt
Set rs = Nothing
Set fld = Nothing
Set db = Nothing
End Sub
```

Beide Routinen zeigen den Inhalt des ersten Datensatzes:

Abbildung 7.20 Der erste Datensatz wird durchlaufen und sein Inhalt angezeigt

Hinweis

Die Eigenschaft `Value` ist hierbei überflüssig, da es sich um die Standardeigenschaft handelt.

Und die Einträge einer Spalte? Genauso – nur umgekehrt:

Beispiel

```
Sub zeigeAlle3()
Dim db As DAO.Database
Dim rs As DAO.Recordset
Dim i As Integer
Dim strInhalt As String

On Error Resume Next
Set db = CurrentDb
Set rs = db.OpenRecordset("tab_Mitglieder")

rs.MoveFirst
For i = 1 To rs.RecordCount
   strInhalt = strInhalt & rs.Fields("tMitZuname").Value & ";
"
   rs.MoveNext
Next

MsgBox strInhalt
Set rs = Nothing
Set db = Nothing
End Sub
```

Dies führt zu:

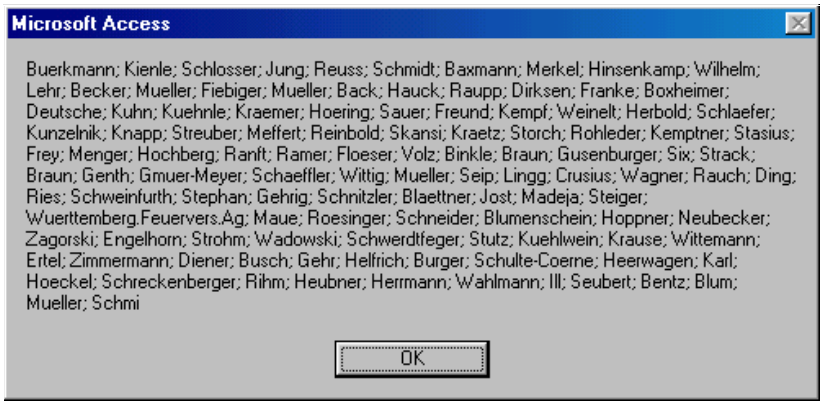


Abbildung 7.21 Leider passen nicht alle Daten der Spalte „Zuname“ der Tabelle „tab_Mitglieder“ in ein Meldungsfenster

7.8.5 Suchen von Datensätzen

Neben dem Ändern von vorhandenen Daten und dem Hinzufügen von neuen ist das Suchen (und Finden) von Daten sicherlich die wichtigste Datenbankoperation. Eine (wenig elegante) Möglichkeit besteht darin, alle Datensätze zu durchlaufen und mit einem Eintrag zu vergleichen. Im folgenden Beispiel wird der Benutzer aufgefordert, einen Namen einzugeben. Danach erhält er das Erscheinungsjahr:

Beispiel

```
Sub suchenNichtSchön()
Dim db As DAO.Database
Dim dbRS As DAO.Recordset
Dim strName As String

Set db = CurrentDb
Set dbRS = db.OpenRecordset("tab_Mitglieder")
strName = InputBox("welcher Name?")

Do While dbRS.EOF = False
  If dbRS.Fields("tMitZuname") = strName Then
    MsgBox Chr(187) & strName & Chr(171) & " wohnt in " _
      & "der " & dbRS.Fields("tMitStraße") & " in " & _
      dbRS.Fields("tMitOrt")
    Set rs = Nothing
```

```
        Set db = Nothing
        Exit Sub
      End If

      dbRS.MoveNext

    Loop

      MsgBox "sorry, aber " & strName & _
      " habe ich nicht gefunden!"
      Set rs = Nothing
      Set db = Nothing

    End Sub
```

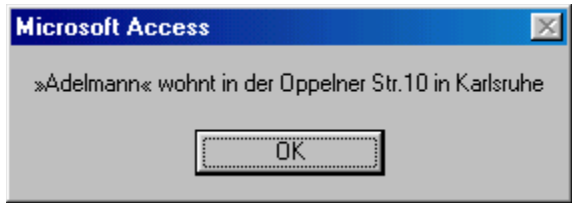


Abbildung 7.22 Ein Name wird gesucht ... und gefunden oder auch nicht

Eleganter – und vor allem schneller – wird mit den beiden Methoden `Find` und `Seek` gesucht:

Die Methode Find

Hier stehen Ihnen vier Suchmöglichkeiten zur Verfügung:

```
FindFirst
FindLast
FindNext
FindPrevious
```

Alle vier Methode benötigen eine Bedingung. Bei der Bedingung wird zwischen Text und Zahl unterschieden. Soll nach einer Zahl gesucht werden, so könnte die Bedingung folgendermaßen aussehen:

```
Bedingung = "Mitgliedsnummer = 4711"
```

oder:

```
Bedingung = "Mitgliedsbeitrag > 199"
```

Bei einer Zeichenkette kann abgefragt werden:

```
Bedingung = "Name = 'Lotte Lichtenberger'"
```

analog kann statt des Apostrophs ein doppeltes Anführungszeichen gesetzt werden:

```
Bedingung = "Name = ""Lotte Lichtenberger"""
```

soll eine Zeichenkette gesucht werden, so kann dies mit einem „*“ geschehen:

```
"Name like ""L*"""
```

findet alle Namen, die mit „L“ beginnen.
Im folgenden Beispiel trägt der Benutzer eine Postleitzahl ein und die Prozedur findet alle Namen, die in Städten mit dieser Postleistzahl wohnen, und zeigt sie an:

Beispiel

```
Sub suchen3()
Dim db As DAO.Database
Dim dbRs  As DAO.Recordset
Dim dblMN As Double
Dim strBedingung As String
Dim strAlle_Namen As String

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

dblMN = InputBox("Welche Mitgliedsnummer soll gesucht wer-
den?")
strBedingung = "tMitMitgliedsnummer = " & dblMN

dbRs.FindFirst strBedingung

Do While dbRs.NoMatch = False
  strAlle_Namen = strAlle_Namen & _
  dbRs.Fields("tMitZuname").Value & "; "
  dbRs.FindNext strBedingung
Loop

If strAlle_Namen = "" Then
```

```
    MsgBox "sorry, es wurde keine Nummer " _
    & dblMN & " gefunden."
  Else
    MsgBox "Die Nummer " & dblMN & _
    " hat: " & strAlle_Namen
  End If

  Set dbRs = Nothing
  Set db = Nothing

  End Sub
```

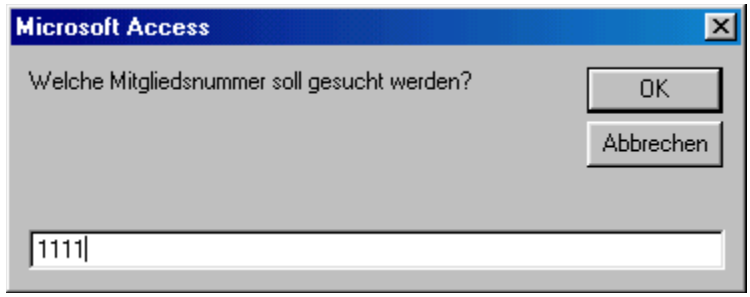


Abbildung 7.23 Gesucht ... und gefunden

Die Methode Seek

Wesentlich schneller als die Methode `Find` ist die Methode `Seek`. Sie kann allerdings nur in indizierten Feldern suchen. Außerdem müssen Sie deren Namen kennen. Die Syntax von `Seek` lautet:

```
Datensatzgruppe.Seek Vergleich, Schlüssel1, Schlüssel2...Schlüssel13
```

Dabei stellt die Datensatzgruppe das RecordSet-Objekt dar.
Vergleich ist einer der folgenden Ausdrücke: <, <=, =, >= oder >.
Schlüssel1, Schlüssel2, ... Schlüssel13 sind ein oder mehrere Werte, nach denen gesucht wird:

Beispiel

```
Sub suchen4()
Dim db As DAO.Database
Dim dbRs  As DAO.Recordset
Dim dbFeld As DAO.Field
Dim strName As String
Dim i As Integer

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder")

dbRs.Index = "tMitZuname"
strName = InputBox("Welcher Name?")

dbRs.Seek "=", strName

If dbRs.NoMatch Then
  MsgBox strName & " wurde leider nicht gefunden."
  Exit Sub
End If

MsgBox Chr(187) & strName & Chr(171) & " wohnt in " & _
  dbRs.Fields("tMitOrt").Value
Set db = Nothing
Set dbRs = Nothing
Set dbFeld = Nothing

End Sub
```

Abbildung 7.24 Der Name wird gefunden und angezeigt

7.8.6 Filtern und Sortieren

Sehr einfach lässt sich eine Datenbank mit dem Befehl `Sort` sortieren. Davor muss die Datenbank jedoch erneut geöffnet werden.

Hinweis

Das Sortierkriterium muss dabei in eckigen Klammern stehen. Im folgenden Beispiel werden die Datensätze einmal unsortiert und einmal sortiert ausgegeben:

```
Sub zeigen3()
Dim db As DAO.Database
Dim dbRs As DAO.Recordset
Dim i As Integer
Dim strInhalt As String
Dim strSQL As String

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder")

dbRs.MoveFirst

For i = 1 To dbRs.RecordCount
   If dbRs.Fields("tMitPlz").Value = 68526 Then
   strInhalt = strInhalt & dbRs.Fields("tMitZuname").Value & "; "
   End If
   dbRs.MoveNext
Next

MsgBox strInhalt

strInhalt = ""
dbRs.Close

Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)
dbRs.Sort = "[tMitZuname]"
Set dbRs = dbRs.OpenRecordset()
dbRs.MoveFirst

For i = 1 To dbRs.RecordCount
   If dbRs.Fields("tMitPlz").Value = 68526 Then
   strInhalt = strInhalt & dbRs.Fields("tMitZuname").Value & "; "
   End If
   dbRs.MoveNext
Next

MsgBox strInhalt

Set db = Nothing
Set dbRs = Nothing
End Sub
```

Warnung

Sie müssen die Tabelle zum Sortieren im Modus `dbOpenDynaset` öffnen, da sonst nur der erste Datensatz herausgeholt werden kann.

Ähnlich wie das Sortieren funktioniert das Filtern mit der Methode `Filter`. Dabei wird das Filterkriterium als String übergeben. Beispiele:

```
"[Ort] Like ""D*"""
"[Jahresbeitrag] > 179"
"[Ort] Like ""D*"" And [Jahresbeitrag] > 179"
```

Tipps & Tricks

Wenn Sie Schwierigkeiten mit der Syntax haben, können Sie diese dem SQL-Fenster der Abfragen entnehmen. Doch Achtung: SQL verlangt am Ende ein Semikolon – dies darf beim Filterkriterium nicht stehen!

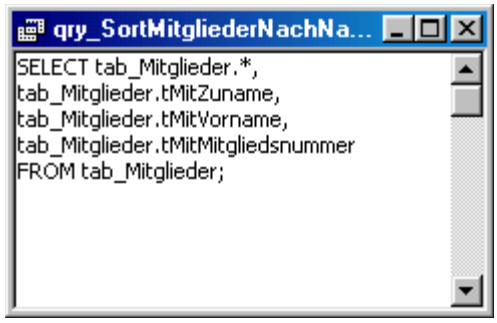


Abbildung 7.25 Das SQL-Fenster der Abfrage

Im folgenden Beispiel werden die drei Filterkriterien angewendet und das Ergebnis angezeigt:

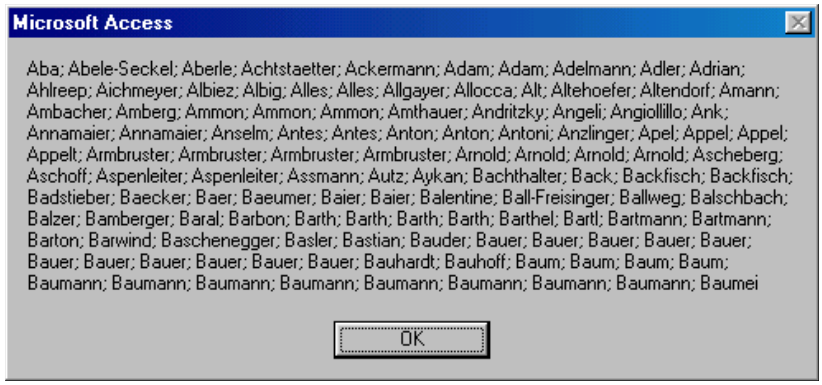
Beispiel

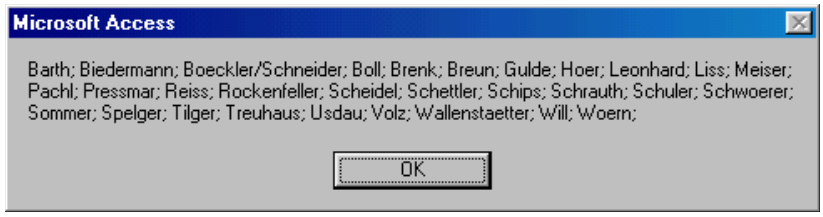
```
Sub zeigi4()
Dim db As DAO.Database
Dim dbRs As DAO.Recordset
Dim i As Integer
Dim strInhalt As String, strFilterkriterium As String

Set db = CurrentDb

strFilterkriterium = "[tMitOrt] Like ""M*"""
```

```
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)
dbRs.Sort = "[tMitZuname]"
dbRs.Filter = strFilterkriterium
Set dbRs = dbRs.OpenRecordset()

For i = 1 To dbRs.RecordCount
  strInhalt = strInhalt & dbRs.Fields("tMitZuname").
  Value & ";"
  dbRs.MoveNext
Next

MsgBox strInhalt
strInhalt = ""
dbRs.Close

Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

strFilterkriterium = "[tMitJahresbeitrag] > 179"

dbRs.Sort = "[tMitZuname]"
dbRs.Filter = strFilterkriterium
Set dbRs = dbRs.OpenRecordset()

For i = 1 To dbRs.RecordCount
  strInhalt = strInhalt & dbRs.Fields("tMitZuname").
  Value & "; "
  dbRs.MoveNext
Next
MsgBox strInhalt
strInhalt = ""
dbRs.Close

Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

strFilterkriterium = _
"[tMitOrt] Like ""D*"" And [tMitJahresbeitrag] > 179"
dbRs.Sort = "[tMitZuname]"

dbRs.Filter = strFilterkriterium
Set dbRs = dbRs.OpenRecordset()
```

```
For i = 1 To dbRs.RecordCount
  strInhalt = strInhalt & dbRs.Fields("tMitZuname").
  Value & "; "
  dbRs.MoveNext
Next
MsgBox strInhalt
Set db = Nothing
Set dbRs = Nothing
End Sub
```

Die drei Ergebnisse:





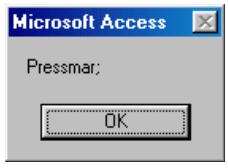


Abbildung 7.26 Drei Filter: Suche die Mitglieder, die in einem Ort wohnen, der mit „D“ beginnt, deren Mitgliedsbeitrag mehr als 179 (DM) beträgt und die sowohl mit „D“ beginnen als auch mehr als 179 (DM) Mitgliedsbeitrag bezahlen.

7.8.7 Daten ändern, hinzufügen und löschen

Eine der Hauptaufgaben neben der Anzeige von vorhandenen Daten ist das Verändern von Daten (dazu gehört auch das Löschen) und das Anhängen von neuen Daten. Hierzu stellt Access die drei Methoden `Edit`, `AddNew` und `Delete` zur Verfügung. Folgendes Programm fügt einen Datensatz an – unabhängig von der Position des Datensatzzeigers:

Beispiel

```
Sub neu()
Dim db As DAO.Database
Dim dbRs As DAO.Recordset
On Error Resume Next

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder")

dbRs.AddNew
dbRs.Fields("tMitZuname").Value = "Rene Martin"
dbRs.Fields("tMitOrt").Value = "Muenchen"
dbRs.Fields("tMitPlz").Value = 80337
dbRs.Update

Set db = Nothing
Set dbRs = Nothing

End Sub
```

Die Methode `Update` ist notwendig, denn sie bewirkt letztendlich, dass der Datensatz dauerhaft hinzugefügt wird. Die Methode `Edit` funktioniert analog:

```
Sub ändern()
Dim db As DAO.Database
Dim dbRs As DAO.Recordset
Dim strBedingung As String
On Error Resume Next

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

strBedingung = "tMitZuname = 'Rene Martin'"
```

```
dbRs.FindFirst strBedingung

dbRs.Edit
dbRs.Fields("tMitPlz").Value = 80339
dbRs.Update
Set db = Nothing
Set dbRs = Nothing

End Sub
```

Hinweis

Auch hier dürfen Sie die Methode `Update` nicht vergessen! `Delete` hingegen benötigt kein `Update`, wie das folgende Beispiel zeigt, in dem der letzte Datensatz (ohne Warnmeldung!) gelöscht wird:

Beispiel

```
Sub ändern()
Dim db As Database
Dim dbRs As Recordset
Dim strBedingung As String
On Error Resume Next

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

dbRs.MoveLast
dbRs.Delete
Set db = Nothing
Set dbRs = Nothing
End Sub
```

7.8.8 Suchen und Filtern von Datensätzen auf einem Formular

Mit diesen Befehlen (und ein wenig SQL-Kenntnissen) kann nun auf einem Formular gesucht und gefiltert werden. Angenommen, auf einem Formular befindet sich ein Textfeld zur Eingabe. Dann kann an eine Befehlsschaltfläche die Prozedur gebunden werden, mit der alle Datensätze gefiltert werden, bei denen der Text in einem Feld steht. Angenommen, er soll komplett darin stehen, dann lautet der Befehl:

```
DoCmd.ApplyFilter , "tMitZuname = """ & _
Me.txtZuname.Value & """"
```

Soll dagegen der eingegebene Text als Textteil gefiltert werden, kann dies mit folgendem Befehl geschehen:

```
DoCmd.ApplyFilter , "tMitZuname Like ""*" & _
Me.txtZuname.Value & "*"""
```

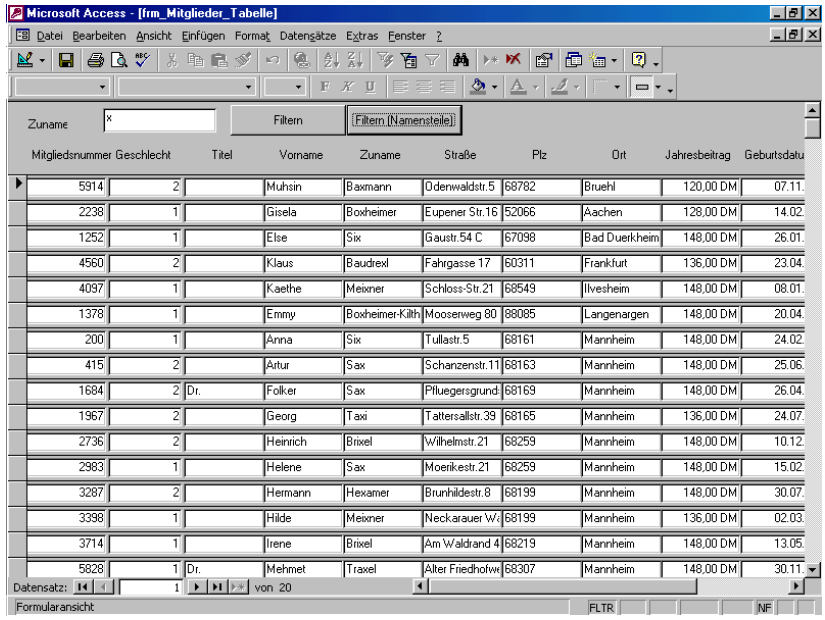


Abbildung 7.27 Die gefilterten Daten (alle Zunamen mit „x“)

Das Suchen von Datensätzen funktioniert ähnlich. In der Tabelle wird der Datensatz gesucht, die ID-Nummer wird an eine Variable übergeben und das Formular springt direkt diesen Datensatz an. Wird kein Datensatz gefunden, bleibt das Formular auf dem ursprünglichen stehen. Das kann abgefangen werden und dem Benutzer wird mitgeteilt, dass der gesuchte Datensatz nicht gefunden wurde. Sicherlich gibt es noch eine Reihe weiterer Lösungen für dieses Problem:

Beispiel

```
Private Sub cmdSuchen_Click()
Dim db As DAO.Database
Dim dbRs  As DAO.Recordset
Dim dblMN As Double
Dim strBedingung As String
```

```
Dim strAlle_Namen As String
Dim lngID As Long

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

strBedingung = "tMitZuname = """ & Me.txtZuname.Value & """"

dbRs.MoveFirst
dbRs.FindFirst strBedingung
lngID = dbRs.Fields("tMitID").Value

   DoCmd.GoToRecord acDataForm, "frm_Mitglieder", _
   acGoTo, lngID

If Me.tMitZuname.Value <> Me.txtZuname.Value Then
   MsgBox Me.txtZuname.Value & " wurde nicht gefunden."
End If

Set dbRs = Nothing
Set db = Nothing
End Sub
```

Das Weitersuchen, das heißt, die Suche ab dem aktuellen Datensatz kann auf ähnliche Weise erfolgen:

Beispiel

```
Private Sub cmdWeitersuchen_Click()
Dim db As DAO.Database
Dim dbRs  As DAO.Recordset
Dim dblMN As Double
Dim strBedingung As String
Dim strAlle_Namen As String
Dim lngID As Long

Set db = CurrentDb
Set dbRs = db.OpenRecordset("tab_Mitglieder", dbOpenDynaset)

strBedingung = "tMitZuname = """ & Me.txtZuname.Value & """"
dbRs.MoveFirst
```

```
dbRs.Move Me.tMitID.Value
dbRs.FindNext strBedingung
lngID = dbRs.Fields("tMitID").Value

  DoCmd.GoToRecord acDataForm, "frm_Mitglieder", _
  acGoTo, lngID

If Me.tMitZuname.Value <> Me.txtZuname.Value Then
  MsgBox Me.txtZuname.Value & " wurde nicht gefunden."
End If

Set dbRs = Nothing
Set db = Nothing

End Sub
```

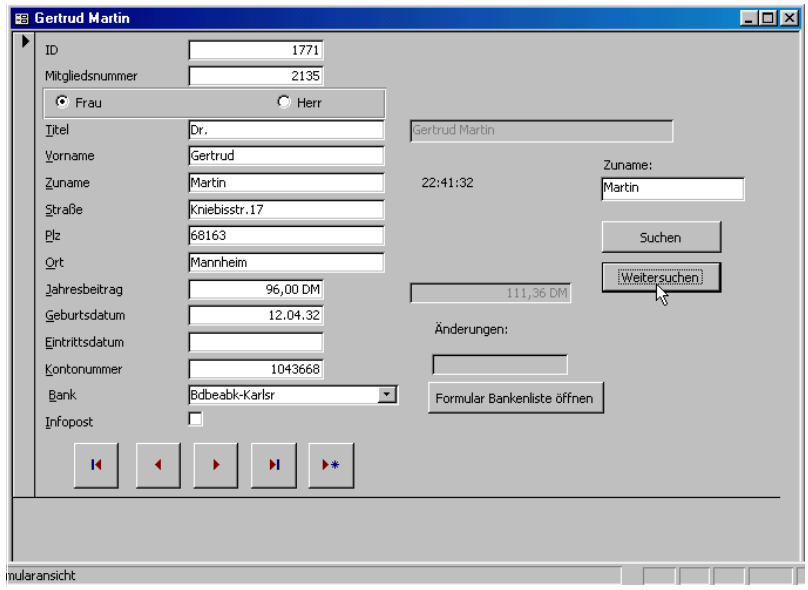


Abbildung 7.28 Ein gefundener Datensatz

7.8.9 Die Formularereignisse

Ähnlich wie bei den Formularen in VBA löst ein Access-Formular eine Reihe von Ereignissen aus, die abgefangen werden können und auf die spezifisch reagiert werden kann. Im Folgenden nun die wichtigsten Ereignisse:

Current (Beim Anzeigen)

Das Ereignis `Current` wird ausgelöst, wenn der Fokus von einem Datensatz zum nächsten wechselt. Damit kann beispielsweise der Inhalt eines Datensatzes in der Titelzeile angezeigt werden:

Beispiel

```
Private Sub Form_Current()
Me.Caption = Me.tMitVorname & " " & Me.tMitZuname
End Sub
```

Damit der Code funktioniert, muss auch

Load (Beim Laden)

oder

Open (Beim Öffnen)

für den ersten angezeigten Datensatz aktiviert werden. Während `Load` vor `Open` aktiviert wird, kann `Open` über den Parameter `Cancel` das Ereignis des Öffnens abbrechen, was `Load` nicht kann:

Beispiel

```
Private Sub Form_Open(Cancel As Integer)
Me.Caption = Me.tMitVorname & " " & Me.tMitZuname
End Sub
```

Activate (Bei Aktivierung)

wird aufgerufen, wenn das Formular den Fokus erhält. Dies kann beim Öffnen sein, oder wenn von einem anderen geöffneten Formular hinübergewechselt wird. Das Ereignis ist beim Aktualisieren des Kombinationsfeldes schon einmal aufgetreten:

Beispiel

```
Private Sub Form_Activate()
  Me.Refresh
End Sub
```

GotFocus (Bei Fokuserhalt)

ist sehr ähnlich. Es wird ausgelöst, wenn das Formular keine sichtbaren aktivierten Steuerelemente enthält.
Verliert das Formular den Fokus oder wird es geschlossen, stehen die gegenteiligen Befehle zur Verfügung:

LostFocus (Bei Fokusverlust)

ist das Gegenteil von GotFocus. Es wird aktiviert, wenn das Formular keine sichtbaren aktivierten Steuerelemente enthält.

Deactivate (Bei Deaktivierung)

ist das Gegenteil von Activate. Es wird ausgelöst, wenn das Formular geschlossen wird oder wenn zu einem anderen Formular (oder Bericht, einer anderen Tabelle oder Abfrage) gewechselt wird.
Beim Schließen werden die beiden Ereignisse

Unload (Beim Entladen)

und

Close (Beim Schließen)

verwendet. Analog zum Öffnen kann nur das Ereignis Unload abgefangen werden. Das folgende Beispiel fragt den Benutzer, ob er den aktuellen Stand der Tabelle in eine andere Tabelle speichern will, das heißt, ob er möchte, dass eine Tabellenerstellungsabfrage gestartet wird:

Beispiel

```
Private Sub Form_Unload(Cancel As Integer)
If MsgBox("Soll der aktuelle Stand " & _
  "gespeichert werden?", vbYesNo) = vbYes Then
    DoCmd.SetWarnings False
    DoCmd.OpenQuery "qry_NeueTabelle"
    DoCmd.SetWarnings False
End If
End Sub
```

Das folgende Beispiel überprüft beim Schließen, ob noch ein anderes, bestimmtes Formular offen ist, und schließt es ebenfalls.

Beispiel

```
Private Sub Form_Close()
If istOffen("frm_Bankenliste_Standard") = True Then
  DoCmd.Close acForm, "frm_Bankenliste_Standard"
End If
End Sub

Function istOffen(strFormname As String) As Boolean
Dim i As Integer
For i = 0 To Forms.Count - 1
  If Forms(i).Name = strFormname Then
    istOffen = True
    Exit Function
End If
Next
End Function
```

Auch für Änderungen an einem Datensatz stehen eine Reihe von Ereignissen zur Verfügung:

BeforeInsert (Vor Eingabe)

und

AfterInsert (Nach Eingabe)

AfterInsert wird ausgelöst, wenn der Datensatz vollständig eingegeben wurde und nun hinzugefügt wird. Bei Änderungen an einem bestehenden Datensatz heißen die drei Ereignisse:

BeforeUpdate (Vor Aktualisierung), Dirty (Bei Änderung) und AfterUpdate (Nach Aktualisierung)

Auf einem Formular können Änderungen in einer Tabelle vorgenommen werden. Um zu dokumentieren, wann diese Änderungen durchgeführt wurden, wird in ein Feld die Kalenderwoche der Änderung gespeichert.

Das heißt: An das Ereignis „Beim Ändern“ des Formulars wird folgender Code gebunden:

Beispiel

```
Private Sub Form_Ditry(Cancel As Variant)
Forms("tab_Mitglieder").Controls("txtÄnderung").Value _
  = Format(Date, "ww")
End Sub
```

Das folgende Makro überprüft, ob das eingegebene Mitglied bereits volljährig ist. Wenn nicht, wird der Benutzer gefragt, ob seine Daten korrekt sind. Bejaht er die Frage, wird der Datensatz gespeichert, verneint er ihn, werden seine Eingaben gelöscht:

Beispiel

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
If (Date - Me.Controls("txtGeburtsdatum").Value) / _
  365.25 < 18 Then
    If MsgBox("Das geänderte Mitglied ist noch " & _
    "nicht volljährig!" & _
    vbCr & "Ist das korrekt?", vbYesNo) = vbYes Then
      Cancel = False
    Else
      Me.Controls("txtGeburtsdatum").Value = Null
      Me.Controls("txtGeburtsdatum").SetFocus
  End If
End If
End Sub
```

Während das Ereignis

Resize (Bei Größenänderung)

auftritt, wenn das Formular in seiner Größe geändert wird, was sicherlich keine allzu bedeutende Rolle spielt, fangen die Ereignisse

7.8.10 BeforeDelConfirm (Vor Löschbestätigung), Delete (Beim Löschen) und AfterDelConfirm (Nach Löschbestätigung)

ab, ob der Benutzer einen Datensatz löschen möchte oder schon gelöscht hat. Das folgende Makro holt vom Benutzer die Bestätigung ein, wenn er versucht, ein weibliches Mitglied zu löschen:

Beispiel

```
Private Sub Form_BeforeDelConfirm(Cancel As Integer, _
  Response As Integer)
If Me.Controls("fraGeschlecht").Value = 1 Then
  If MsgBox("Sind Sie sicher, dass Sie diese Frau " & _
  "löschen möchten?", vbYesNo) = vbNo Then
    Cancel = True
  End If
End If
End Sub
```

Schließlich können noch Aktionen wie das Klicken auf das Formular mit `Click` (Beim Klicken) oder der Doppelklick `DblClick` (Beim Doppelklicken) abgefangen werden. Etwas differenzierter arbeiten `Keydown`, `KeyPress` und `KeyUp` (Bei Taste Ab, Bei Taste und Bei Taste Auf). Damit kann eine bestimmte Taste abgefangen werden, beispielsweise die Taste ‹Bild↑› und ‹Bild↓›. Mausbewegungen, bei denen nicht geklickt wird, werden mit `MouseMove` (Bei Mausbewegung) abgefangen. Besitzt beispielsweise ein Steuerelement die Eigenschaft, dass das Überfahren mit der Maus seine Textfarbe ändert:

Beispiel

```
Private Sub cmdBank_MouseMove(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
Me.Controls("cmdBank").ForeColor = 16711680
End Sub
```

kann diese Eigenschaft im Detailbereich des Formulars wieder ausgeschaltet werden:

Beispiel

```
Private Sub Detailbereich_MouseMove(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
Me.Controls("cmdBank").ForeColor = 0
End Sub
```

Daneben existieren noch die Befehle `MouseDown` (Bei Maustaste Ab) und `MouseUp` (Bei Maustaste Auf).
Wird ein Filter eingeschaltet, startet das Ereignis Filter (Bei Filter), ist der Filter schon aktiv, wird ApplyFilter (Bei angewendetem Filter) aktiviert. Interessanter ist das Ereignis

Error (Bei Fehler)

Tritt ein Fehler auf, weil der Benutzer sich vertippt (beispielsweise in ein Zahlenfeld Text eingibt), so kann auf diesen Fehler reagiert werden. Um nicht die internen Standardfehlermeldungen zu erhalten, muss `Response` auf 0 gesetzt werden:

Beispiel

```
Private Sub Form_Error(DataErr As Integer, _
  Response As Integer)
MsgBox "Bitte überprüfen Sie die Eingabe"
Response = 0
End Sub
```

Und schließlich der viel geliebte

Timer (Zeitgeber)

Wird er verwendet, dann wird die Eigenschaft Zeitgeberintervall auf einen Wert zwischen 0 und 2.147.483.647 gesetzt. Diese Zahl gibt die Häufigkeit in Millisekunden an, mit der die Timer-Eigenschaft aufgerufen wird. Im folgenden Beispiel wird in einem Bezeichnungsfeld laufend die aktuelle Uhrzeit angezeigt:

Beispiel

```
Private Sub Form_Timer()
Me.Controls("lblZeit").Caption = Format(Now, "hh:nn:ss")
End Sub
```

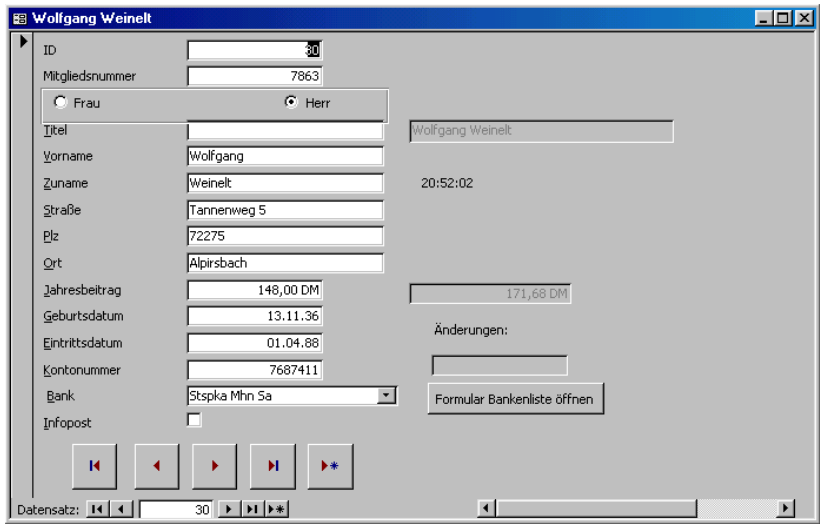


Abbildung 7.29 Alle Ereignisse finden Sie im Datenbankbeispiel „Mitglieder.mdb“ auf der CD-ROM