Inhaltsverzeichnis

Vorwort

Wenn Du dieses Heft in die Hand nimmst, hast Du Deine erste Begegnung mit Access vermutlich schon hinter Dir. Du hast gelernt, die eingängigen Methoden der Programmieranwendung Access zu benutzen, um eine Datenbank mit Tabellen, den dazugehörigen Eingabeformularen und vielleicht einigen Abfragen zu erstellen. Bislang hast Du dazu hauptsächlich die verschiedenen Assistenten benutzt. Bei der Arbeit und beim Durchsuchen der Menüs hat Dich möglicherweise das Gefühl beschlichen, dass es da noch mehr gibt. Einen Teil davon will ich in diesem Heft beschreiben – nur einen Teil, denn Access ist eine umfangreiche Programmier-Anwendung. Ich greife einige Beispiele heraus, um Dir einen ersten Zugang und vielleicht eine gewisse Sicherheit zu verschaffen, womit Du dann selbständig weiterarbeiten kannst.

Access ist ohnehin für eine solche schrittweise Annäherung konzipiert:

- Einen grossen Teil der Arbeit kannst Du mit den Assistenten abwickeln.
- Spezielle Aufgaben sowie die Automatisation über Menüs und Schaltflächen musst Du selbst erstellen.

Später, wenn Du eine moderne objektorientierte Datenbank entwerfen willst, musst Du Dich mit VBA beschäftigen, also mit Visual Basic for Applications. VBA ist ab Office 97 die gemeinsame Programmiersprache aller Microsoft Office-Programme. Doch bis zum Umgang mit VBA hast Du noch viel Zeit.

Wir werden uns hauptsächlich mit Teil 2 auseinandersetzen, denn Automatisation in der ersten Stufe heisst, den Umgang mit *Makros* kennenzulernen. Makros sind in vielen Programmieranwendungen gebräuchlich, weil sie die Arbeit weitestgehend erleichtern. Makros sind nichts anderes als VBA-Befehle, die gezielt zur Abwicklung bestimmter Aufgaben zusammengefasst werden.

Zu diesem Zweck werden wir eine Datenbank programmieren, wobei ich Dich mit den wesentlichen Werkzeugen in Access bekanntmachen möchte. Das Ziel ist, eine elegante Datenbank zu entwickeln, die letztlich auf Rechnern laufen kann, die selbst nicht Access installiert haben, eine sogenannte *Run time-Umgebung* – mit anderen Worten eine professionelle, verkaufbare Anwendung.

Im Rahmen diese Heftes können wir natürlich nur die Ansätze dazu entwickeln. Die Feinarbeit, die den Hauptteil der Arbeit ausmacht, musst Du schon selber erledigen.

Dieses Heft unterscheidet sich von anderen Büchern darin, dass es keinen Anspruch auf Vollständigkeit erhebt. Wir werden ganz gezielt Aufgaben durchgehen, die oft an eine kaufmännische Software gestellt werden; das kann eine CD-Verwaltung sein, aber auch ein Warenwirtschafts-System für einen Ladenverkauf, mit daran hängender Finanzbuchhaltung. Ich werde aber nicht alle bestehenden Möglichkeiten von Access beschreiben – und das nicht nur, weil dieses Heft zu klein ist. Access ist wie eine Sprache, die man ja auch nicht erst dann zu sprechen anfängt, wenn man alle Wörter kennt. Viele Dinge in Access ergeben sich von selbst, sobald Du einmal ein bestimmtes Rüstzeug beherrscht.

Ich verfahre in diesem Heft nach dem Prinzip: „Das Leben stellt die Aufgaben.“ Wenn ich weiss, was eine Schraube ist, und einen Schraubenzieher habe, muss ich nicht theoretisch wissen, was ein Linksgewinde ist. Passt eine Schraube nicht ins Gewinde, muss ich sie mir genauer ansehen – oder nachdenken, was nicht stimmt. Solange ich Lehrling bin, habe ich ausserdem meistens genug Zeit zum Fragen.

Der Rest ist ein bisschen Forscherdrang und die gut ausgebaute Access-Hilfe, die zu dem Empfänger Deiner Fragen werden soll. Später, beim Umgang mit Visual Basic, brauchst Du vielleicht noch eine Referenz-Liste für die einzelnen Befehle. Bis Du aber so weit bist, kannst Du bequem einen ganzen Supermarkt verwalten.

Der Autor

Ich heisse Rainer Schubert, 1951 geboren, und beschäftige mich seit 9 Jahren mit Software-Engineering und Dokumentation. Meine Kinder (6 und 9) spielen an meinem Rechner herum und malen Bilder, klicken sich durch Spiele und blättern sich durch „Wie funktioniert das?“. Ich versuche ihnen zu zeigen, dass der Rechner ein Hilfsmittel für die Menschen ist, nicht mehr und nicht weniger.

Access – was ist das?

Unsere Vorgangsweise sieht so aus: Wir erarbeiten eine Adress-Verwaltung für einen Verein, die alle Mitglieder aufnimmt, ihre Beiträge verwaltet und Veranstaltungen organisiert. Dabei halten wir die Datenbank so allgemein, dass sie auf möglichst viele verschiedene Vereine angewandt werden kann.

Dieses Fortgeschrittenen-Heft richtet sich an Dich, wenn Du grundsätzlich weisst, was mit einer Datenbank-Anwendung gemeint ist. Was Dir aber noch fehlt, ist die Kenntnis, wie eine Datenbank „rund“ wird, so dass ein Benutzer sie bequem bedienen kann.

Es folgt ein kurzer Überblick dazu, wie man Access einordnen kann. Ich weiss persönlich immer gern, womit ich es zu tun habe, wenn ich mich mit etwas Neuem beschäftige. Dieser Teil befasst sich allgemein mit Access – Du musst ihn also nicht unbedingt lesen, aber vielleicht hilft er Dir.

Access funktioniert nach dem Datenbanken-Modell. Und was ist das eigentlich? Datenbanken sind wie Karteikästen. Alle wesentlichen Informationen wie Anschrift und Telefonnummer einer Person (genannt: die Daten) werden an einem Ort gesammelt und gespeichert. Das Wort „Bank“ soll wohl einerseits ausdrücken, dass die Daten sicher an einem Ort liegen, wo niemand an sie herankommen kann, um sie zu verändern (verfälschen) oder zu entwenden (Sicherheit). Andererseits ist es der zentrale Ort der Aufbewahrung – man muss nicht weite Wege gehen, um Informationen zu sammeln. Im Englischen heisst Datenbank "database", womit die Basis der Daten bezeichnet wird.

Als Negativbeispiel sind Ämter zu nennen, in denen man zur Abwicklung eines Antrags an vielen Stellen vorstellig wird. Dieses nutzlose Umherlaufen ist auch für Daten in Computern wenig sinnvoll – weiss aber jeder, wo die Daten liegen, findet man sie schnell, und es gibt kein nutzloses Suchen.

Ausserdem steht das „moderne“ Konzept von Computern dahinter: Wenn es nur einen Ort gibt, an dem Informationen hinterlegt werden, kann nur jeweils einer auf diese Informationen zugreifen – was mit dem Wort *Redundanz* bezeichnet wird. Zusätzlich bekommt jeder eine Berechtigung für die Daten (Identifikation und Priorität), und jeder Besuch in der Datenbank kann protokolliert werden – wer wann welche Daten abgefragt hat. Es wird verhindert, was nicht nur in Computern schrecklich ist: Eine Abteilung schickt einen Brief an die alte, eine andere an die neue Adresse, weil die erste nicht wusste, dass sich die Adresse geändert hatte.

Wenn in Datenbanken durch einen Benutzer eine Adresse geändert wird, arbeitet der nächste Benutzer sofort mit der geänderten Information, er ist immer auf dem neuestem Stand.

Dieses Konzept wird auch die Grundlage unseres *Datenentwurfes* sein, wodurch die Datenbank schnell und übersichtlich wird. Ausserdem sollte die Möglichkeit gewährleistet sein, später andere Teile (Module) hinzuzufügen, ohne Veränderungen an bestehenden Tabellen vornehmen zu müssen.
Zu Access gehören zwei Teile. Da ist zum einem die Sprache, in der ich mich mit der Datenbank unterhalte. Sie heisst **SQL**, **S**tructured **Q**uery **L**anguage, auf deutsch strukturierte Abfragesprache. Abgefragt werden die Tabellen der relationalen Datenbank, die wir mit Access erstellen. In dem Wort „relational“ versteckt sich der ganze Sinn von Access. Wir werden im Laufe dieses Büchleins eine Adress-Verwaltung erstellen, wir legen also so etwas wie elektronische Karteikarten an. Die Anschriften können wir z.B. in den Briefköpfen eines Textverarbeitungsprogramms wie „Word“ einbinden.

Warum nicht gleich die Adressen in Word speichern? Weil wir zusätzlich zu den Adressen auch noch wissen wollen, ob ein bestimmtes Mitglied auch seine Beiträge bezahlt hat. In relationalen Datenbanken kann ich nach Belieben auf verschiedene Datenquellen zugreifen und sie miteinander verknüpfen, also Beziehungen oder Relationen herstellen. Wenn ich will, kann ich mir also alle Mitglieder ausdrucken lassen, die bis zum Jahresende noch nicht ihre Beiträge bezahlt haben, und ihnen dann Mahnungen schicken! In einer Textverarbeitung kann ich nur nach Adressen *oder* nach bezahlten Beiträgen suchen, *nicht* aber nach beiden gleichzeitig. Ich kann also nur jeweils eine variable Grösse (→ Glossar) abfragen.

Der andere Teil heisst **Visual Basic**. Das englische Wort „visual“, sichtbar, drückt aus, für was dieser Teil u.a. steht: die Präsentation. Visual Basic steuert den sichtbaren Bereich, wie die Eingabemasken und die Formulare, die wir erstellen. Es verbindet aber auch die verschiedenen Elemente, wenn Daten über Verknüpfungen zu Abfragen und Ausgaben herangezogen werden, und regelt die Automatisation im Hintergrund. Visual Basic ist eine „richtige“ Programmiersprache – „richtig“ meint hier auch die klassische Art, einzelne

Befehle einzutippen – wie das vor nicht einmal 10 Jahren noch üblich war!

Basic ist eine relativ einfache, leicht erlernbare Programmiersprache, die allerdings einen Nachteil hat: bei grossen und komplexen Anwendungen wird sie unübersichtlich. Diesen Nachteil versuchte Microsoft zu beheben, indem neue Sprachelemente hinzugefügt wurden, um Aufgaben, die man Prozeduren nennt, in einer eng umgrenzten Umgebung abzuwickeln. So erhielt Basic den Schliff moderner Hochsprachen wie C++, die man *prozedurale Programmiersprachen* nennt.

In Access hat man diese Teile zu **VBA** (**V**isual **B**asic for **A**pplications (Anwendungen)) zusammengefasst. Darin enthalten sind:

- die allgemeinen Visual Basic-Sprachelemente, die auch in Excel und der Makro-Programmierung von Word benutzt werden
- Access-spezifische Sprachelemente
- DAO (**D**ata **Access O**bjects (Daten-Zugriffs-Objekte)), die u.a. auf SQL aufgebaut sind.

Die Grundzüge dieser Programmierung sehen wir uns im letzten Teil dieses Heftes an.

Kurz dazu, was Access *nicht* kann. Auf niedriger Ebene *kann* Access zwar an der Hardware arbeiten – das Programm sollte aber dafür nicht benutzt werden. Es ist eher eine kaufmännische Programmier-Anwendung, die für technische Programmierung zuviel überflüssiges Werkzeug mitschleppt. Es gibt andere Programmiersprachen, die diesen Zweck besser erfüllen, wie etwa C oder C++.

Die *Hilfe* ist in Microsoft Access sehr gut ausgebaut. Sie ist deutsch abgefasst und bietet neben themenorientierten Demos ein Verzeichnis, das zu Beschreibungen fast aller auftauchender Begriffe führt. Darum habe ich dieses Buch so aufgebaut, dass Du die Themen, die nicht erschöpfend besprochen werden, in der *Hilfe* nachlesen kannst. Dies mag durch die vielfältigen Querverweise in der *Hilfe* oft langwierig sein, führt aber mit der Zeit zu einem geübten Umgang mit diesem Medium. Auch da heisst es manchmal, einfach das auszuprobieren, was die Hilfe anbietet. Ihre Anwendung kannst Du zu gegebener Zeit an einem Beispiel gut üben.

Das Glossar habe ich ausschliesslich auf Begriffe beschränkt, die nicht in der Hilfe vorkommen oder dort schlecht zu finden sind.

Zur Benutzung dieses Heftes

Ich habe eine Datenbank geschrieben, in deren Zentrum die Mitglieder eines Vereins stehen; wir laden sie zu Veranstaltungen ein und schicken ihnen Mahnungen, wenn sie nicht bezahlt haben. Diese Anwendung wird uns über das ganze Heft begleiten.

Meine Empfehlung ist, die Entwicklung dieser Datenbank in allen Einzelheiten auf Deinem Rechner nachzuvollziehen. Auf die Art und Weise kann ich Dir am besten zeigen, wie man meiner Meinung nach zu einer guten Datenbank kommt. Mein Ziel ist, dass Du Dich mit dem hier erlernten Rüstzeug an den Rechner setzen kannst, um Deine eigene Datenbank zu entwerfen.

Ein Wort zum Entwurf der Datenbank. Es mag lästig und überflüssig erscheinen, Dich vor dem Programmieren mit Papier und Bleistift hinzusetzen und Dich zu fragen, was Du eigentlich willst. Aber einerseits zwingt Dich dieser Schritt, Dir zu überlegen, welche Arbeiten das Programm eigentlich verrichten soll. Erst dann weisst Du, welche Daten Du überhaupt brauchst und in welchen Tabellen sie abgelegt werden. Denn es gibt bei professionellen Anwendungen einen Aspekt, der nicht ganz unwichtig ist: die Schnelligkeit Deines Programms. Diese wird durch die optimale Konfiguration (Entwurf) der Daten erreicht.

Ausserdem spart es Dir eine Menge Arbeit; denn wenn Du erst im Nachhinein feststellst, dass ein Datenfeld zu klein oder zu gross ist, benötigst Du wesentlich mehr Zeit, um die mit diesem Feld verbundenen Daten zu aktualisieren.

Zum anderen heisst professionell: das geschriebene Programm ist nicht für Dich gedacht, sondern für einen Benutzer, der von der Entstehung keine Ahnung hat. Der sitzt vor dem Bildschirm und weiss natürlich nicht, dass er bestimmte Tasten gerade nicht drücken soll. Das heisst, Du musst ihn mit einplanen, bei den Daten und bei der Bedienung.

Allgemein: Die einzelnen Schritte sind in diesem Heft zwar stark an der Datenbank „Verein“ orientiert, weil das die Beschreibung viel eindrücklicher macht. Trotzdem musst Du dieses Heft nicht durcharbeiten. Du kannst Dir auch eine eigene, ähnliche Datenbank anlegen und dann die Kapitel lesen, zu denen Du Unterstützung brauchst. Oder Du nimmst Dir eine der Beispiels-Datenbanken, die im Lieferumfang von Access enthalten sind, und änderst sie nach Deinen Bedürfnissen ab.

Schreibweisen

Bestimmte Teile des Textes werden durch besondere Schweibweisen hervorgehoben:

- **`Eingabe`** : für Eingaben Deinerseits
- **Alt+Tab** : Tastaturkürzel – hier zu beachten: die 1. Taste wird festgehalten, während die 2. gedrückt wird.
- **Zurück** bezeichnet Schaltflächen oder *Buttons*
- **Bearbeiten / Datensatz markieren** : Befehle, die aus der Menüleiste aufgerufen werden
- → weist darauf hin, dass ein Begriff im Glossar näher beschrieben ist. Steht kein Begriff dahinter, ist es ein Tip, den Du ausprobieren kannst. Am Ende steht dann ←

Was Du können solltest

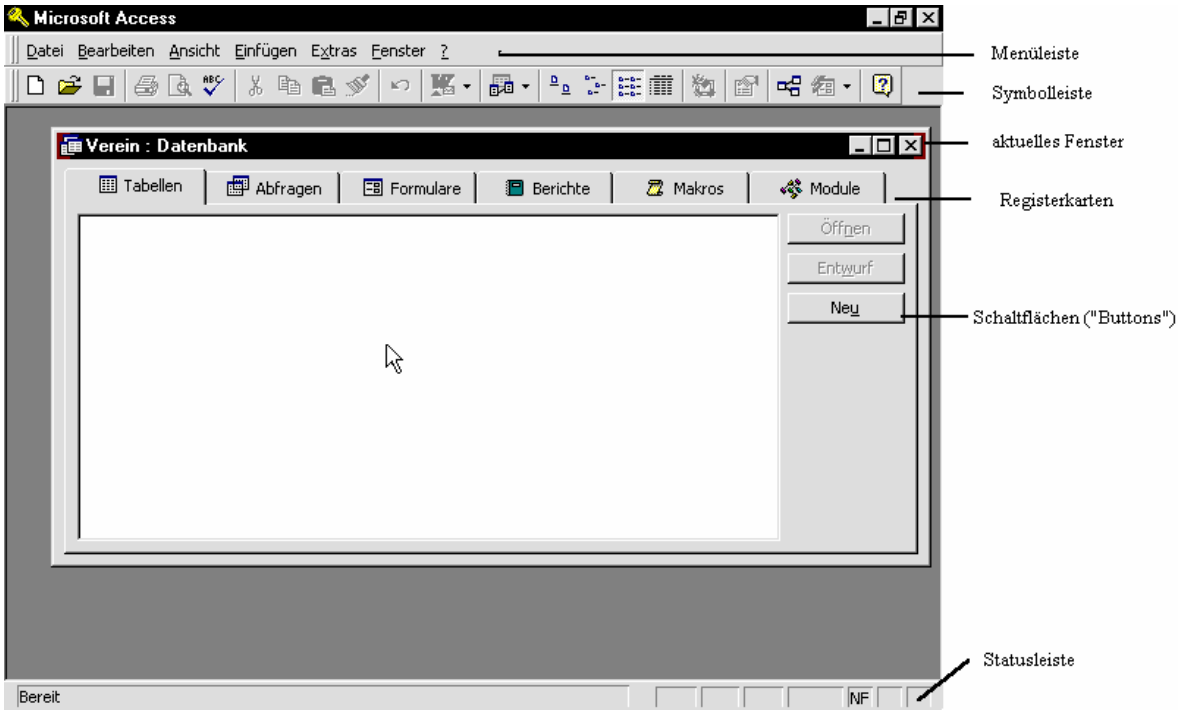
Grundlage für die Beschreibungen in diesem Heft ist die Version Access 97; wobei Leser, die mit Access 2.0, Access 95 oder 2000 arbeiten, die Unterschiede sicher schnell entdecken werden. Wo sie gravierend sind, beschreibe ich sie gesondert. Hauptsächlich sind dies der Umgang mit den verschiedenen Assistenten und die bei Access 95 neu eingeführte Programmiersprache Visual Basic, die früher Access Basic hiess. Für Access 2000 hat Microsoft die Oberfläche an die übrigen Office-Produkte angepasst.

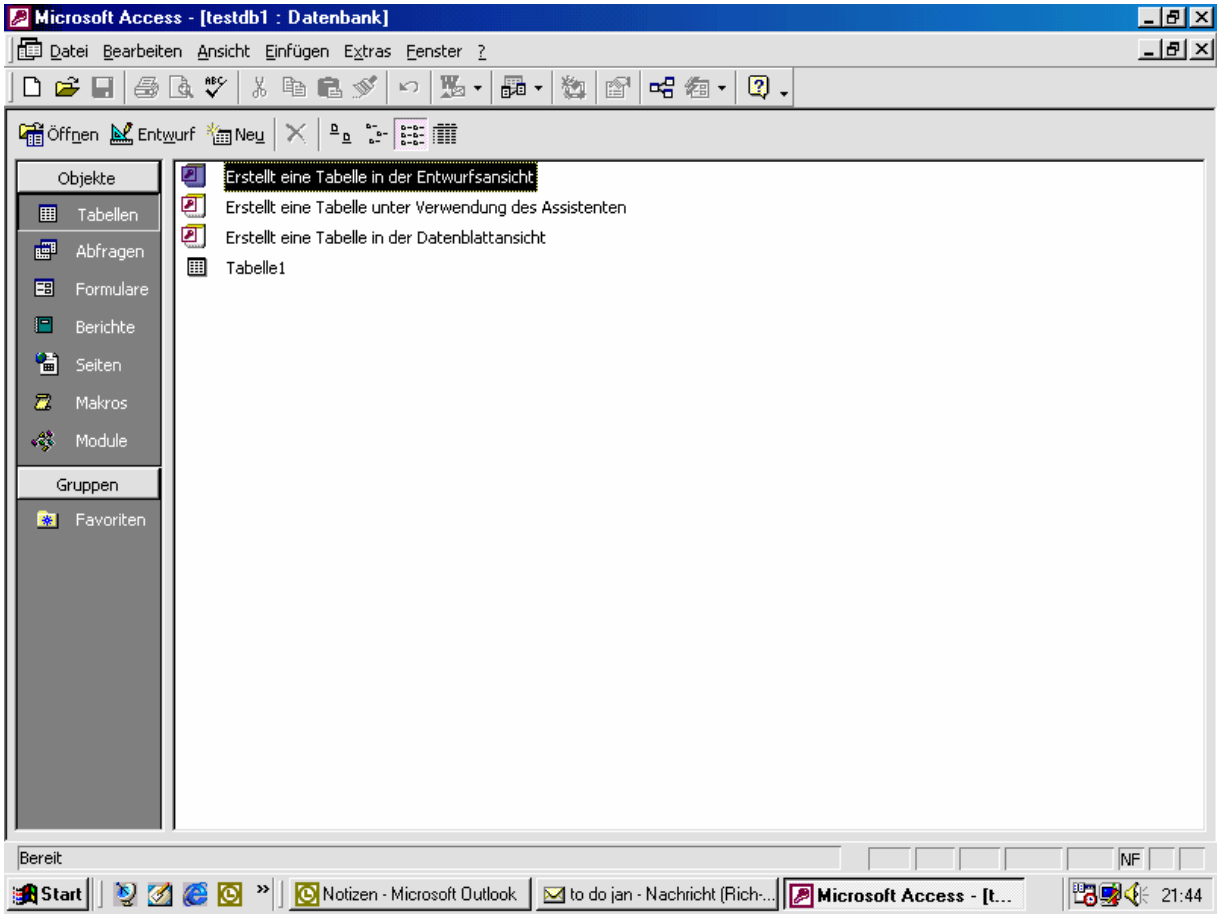
So wird neuerdings schärfer zwischen dem Entwurf von Abfragen etc. *mit* oder *ohne* Assistenten unterschieden. Die wesentlichen Neuerungen von Access 2000 spielen sich auf der Ebene von VBA ab, die einen klaren Schritt in Richtung objektorientierte Programmierumgebung bedeuten.

Kåre Thomsen hat für den *KnowWare* Verlag ein sinnvolles und gut aufgebautes Buch geschrieben. Es heisst „Start mit Access 7/97“. Darin hat er die Grundlagen aufgeführt, die Dir bei dem ersten Arbeiten mit Access begegnen, aufgeteilt in einfache Aufgaben. Das Heft ist zwar nicht Voraussetzung für den Umgang mit diesem Heft, aber es entspricht in ungefähr dem Wissensstand, den Du mitbringen solltest.

Mit anderen Worten solltest Du Dich im grundsätzlichen Umgang mit der Access-Oberfläche auskennen – Du solltest wissen, wie eine *Tabelle* erstellt oder eine einfache *Abfrage* entworfen wird und wie Du zu einem einfachen *Formular* kommst, als Eingabe-Maske und als Menü, von dem aus Du Funktionen steuerst. Das gleiche gilt für *Berichte*, nicht aber für *Makros*, die einen wesentlichen Teil dieses Heftes einnehmen.

Kurz – Du solltest wissen, was passiert, wenn Du auf die einzelnen Registerkarten im folgenden Fenster klickst.





So sieht die Sache in Access 2000 aus.

Was Du auch kennen solltest, ist das grundsätzliche Schema, mit dem Du in Access arbeitest:

- alle Daten kommen aus den *Tabellen* und gehen in sie ein
- eingegeben werden die Daten in den *Formularen*
- ausgegeben, zum Drucken oder Ansehen, werden sie in *Berichten* – manchmal auch in *Formularen* und *Abfragen*
- Berichte wiederum basieren auf *Abfragen*, die die eigentliche Verarbeitung darstellen. Die Verarbeitungen der Daten vom Formular über die Abfrage bis zur Ausgabe im Bericht wird von den *Makros* gesteuert.
- Spezielle, von Dir selbst erstellte Funktionen werden in den *Modulen* erfasst. Das Arbeiten mit *Modulen* erfordert Kenntnisse in VBA.

Schliesslich solltest Du noch wissen, wie man sich durch die Windows-Oberfläche bewegt. Nachfolgend habe ich Dir die Tastenkombinationen zusammengestellt, mit denen Du – neben der Mausbedienung – Deine Arbeiten ausführen kannst. Bei bestimmten, immer wieder auftauchenden Bedienungen arbeite ich manchmal lieber mit Tasten, weil die Maus mich durch die feinmotorischen Bewegungen oft verspannt macht.

F11	Wechsel ins Datenbankfenster
Strg+F6	Wechsel zwischen geöffneten Fenstern
F6	in der Entwurfsansicht von Tabellen und Abfragen Wechsel zwischen oberen und unterem Bereich
Tab	Bewegung durch die einzelnen Felder (Formulare, Berichte) oder Spalten (Datenblatt)
F2	ein Feld neu beschreiben (Editor)
Pos1	zum Anfang der Zeile gehen
Strg+C	markierten Inhalt kopieren
Strg+V	Inhalt einfügen
Strg+F	Datensatz suchen

Die Datenbank *Verein*

Programmierer haben ein Handikap: sie wissen zwar, wie Computer funktionieren, kennen aber meist die Arbeitsumgebung des Betriebes nicht, in dem ihr Programm später arbeiten soll. Nur selten entwerfen Programmierer Anwendungen für sich selbst als Benutzer.

Darum müssen wir ein Thema besprechen, über das sich Programmierer ungern Gedanken machen, weil es sie in ihrem Tatendrang hindert: das Thema des *Datenentwurfs*. Der Datenentwurf für Programme in Firmen fängt grundsätzlich mit einer Arbeitsanalyse an. Dabei wird festgestellt, welche Aufgaben verrichtet werden sollen und welche davon durch EDV unterstützt oder übernommen werden können. Programmierer setzen sich also mit den Zuständigen der Firmen zusammen, die die Anforderungen an das Programm beschreiben. Das ist oft sehr schwierig, weil eine Firma natürlich gewachsen ist und ständig neue Arbeitsabläufe entwickelt. Ausserdem wissen Firmen-Mitarbeiter meist viel über ihre eigene Arbeit, aber wenig davon, welchen Stellenwert diese für den übrigen Betrieb hat.

In dieser Phase muss ein Programmierer zuhören können; mit der Zeit entwickelt er meist auch den nötigen Instinkt, um zu wissen, wo er noch einmal nachfragen sollte, um einen Arbeitsablauf in allen Einzelheiten kennenzulernen.

Der Programmierer wertet die Anforderungen aus und dokumentiert auf dieser Grundlage, wie er/sie sich die Verwirklichung mithilfe von EDV vorstellt. Neben der Hardware-Umgebung, die die Verarbeitung übernimmt, beschreibt er die Struktur, in der die einzelnen Teile oder Objekte miteinander in Verbindung stehen. Ausserdem entwickelt er ein Konzept für die Gewährleistung der Datensicherheit, also wo und wie die Datensätze gespeichert werden.

Ist dies alles bekannt, übersetzt der Programmierer, in diesem Falle Du, es in eine Datenstruktur. Die erste Frage ist hier, welche Daten im Zentrum stehen – anders ausgedrückt: welches die *Stammdaten* sind. Du definierst die einzelnen Tabellen mit ihren Feldern, in denen die Daten zusammenfliessen und gespeichert werden.

Dann stellst Du fest, wo die Daten jeweils benötigt werden – Du definierst ihre Verarbeitung in den einzelnen Programmteilen. Das sind die Eingabe (in *Formularen*), die Verarbeitung (mit den *Abfragen*), und die Ausgabe in *Berichten*, im Ausdruck etc.

Wenn Du später professionell arbeitest, stellt dieser Datenentwurf die Basis für das *Pflichtenheft* dar, das von Dir und dem Benutzer erstellt wird, also dem Betrieb, der das Programm einsetzt. Darin wird genau beschrieben, was ein Programm leistet und wo die Verarbeitung stattfindet. Hier wäre der leichtfertige Hang von Programmierern anzumerken, ihrem Programm mehr zuzumuten, als ihre Programmierkenntnisse tatsächlich leisten können! Dieses Pflichtenheft gilt als Rechtsgrundlage zwischen dem Benutzer und dem Programmierer – es ist von beiden Seiten einklagbar und entspricht somit einem Vertrag, der eingehalten werden sollte.

Das mag Dir bislang nicht wichtig gewesen sein, weil Du erstmal das Handwerkszeug von Access kennengelernt hast. Spätestens dann, wenn eine Deiner Tabellen 20 Datenfelder aufweist, solltest Du Dir aber überlegen, ob das nicht vielleicht zu viele sind und ob einige sich auf weitere Tabellen verteilen lassen. Ein Grundsatz sollte lauten:

Jeder Vorgang in Deiner Anwendung sollte nur so viel Daten bewegen wie unbedingt notwendig.

Manchmal ist ein Verknüpfung von Daten durch *Tabellen* oder *Abfragen* eher angesagt und macht Dein Programm übersichtlicher, als wenn eine Tabelle ihren Rahmen durch allzu viele Daten sprengt.

Access bietet zwar freundlicherweise eine ganze Anzahl von Werkzeugen, mit denen Du den Datenaufbau nachträglich verbessern kannst. Du findest sie unter **Extras|Analyse**. Trotzdem solltest Du Dich nicht darauf verlassen, weil Du damit das Konzept aus der Hand gibst und wohlmöglich Deinen Datenaufbau hinterher nicht mehr verstehst.

Grundsätzlich gilt es Gleichgewicht zu halten zwischen möglichst kleinen Tabellen mit wenigen Datensätzen und undurchschaubar werdenden Anwendungen, die sich durch ein Übermass an Verknüpfungen auszeichnen. Access überwacht Verknüpfungen von Daten selbständig durch die *referentielle Integrität*, auf deren verschiedene Einstellungen wir später noch zu sprechen kommen.

Ein weiterer Schritt, der Dir unter Umständen sehr viel Arbeit ersparen kann, ist die Zusammenfassung bestimmter Aufgaben zu sogenannten Programmgruppen. So könnten z.B. die *Etiketten* eine solche Programmgruppe sein, die aus verschiedenen Teilen Deines Programms ausgedruckt werden sollen. Wenn Du ihr Aussehen bestimmst

und Dir Gedanken machst, wie sie aktiviert werden, solltest Du Dir auch gleich überlegen, an welchen Stellen Du sie noch anwendest. Dann kannst Du nämlich den Vorgang so einstellen, dass Du ihn nur einmal programmieren musst.

So ein Datenentwurf kann eine schwierige Sache sein, weil man praktisch schon vor dem Beginn wissen muss, wo man mit der Programmierung hin will und was das Programm leisten soll. Andernfalls kann es passieren, dass man sich unnötig viel Arbeit macht. Stellst Du z.B. während der Programmierung fest, dass der *Felddatentyp Zahl* für eine *Mitgliedsnummer* in der Datenbank ungeeignet ist, weil Deine Mitgliedsnummern auch Buchstaben enthalten, müsstest Du eventuell nicht nur den Tabellenentwurf ändern, sondern alle Stellen im Programm, an denen das Feld *Mitgliedsnummer* auftaucht, da andernfalls die *referentielle Integrität* verletzt würde. Ausserdem müssten alle vorhandenen Datensätze neu angepasst werden – und das wäre eine ganze Menge Arbeit. Nachträgliche Korrekturen bedeuten meist wesentlich mehr Mühe als ein anständiger Datenentwurf.

→ Vielleicht hast Du vom gigantischen Computer-„Gau“ im Wechsel zum Jahr 2000 gehört? Die Probleme der Grossrechner rühren daher, dass die Datumsfelder mit zwei Stellen belegt sind. Zu Sylvester 1999 haben die Rechner daher ein Problem, weil das System-Datum auf „00“ springt. Mit „00“ wissen die Rechner aber nichts anzufangen, weil sie nicht entscheiden können, ob damit 1900 oder 2000 gemeint ist. Zum 2.1.1900 muss keine Prämie für eine Auto-Versicherung gemahnt werden, weil es damals noch keine Autos gab – zum 2.1.2000 ist das aber durchaus möglich. Werden die Mahnbescheide aber nicht verschickt, können Verluste für die Versicherungen entstehen. Das ist der extremste Fall eines schlechten Datendesigns, das grosse Firmen im Augenblick Tausende von DM kostet – und das nur, weil die Programmierer in den 50-er und 60-er Jahren nicht damit rechneten, dass ihre Programme durch einen Jahrhundert- oder Jahrtausendwechsel einmal 4-stellige Datumsfelder benötigen würden. ←

Für eine übersichtliche Darstellung der Daten in relationalen Datenbanken hat sich eine bestimmte Methode als sehr gut herausgestellt: die *Normalisierung.* Dabei wird zwischen verschiedenen *Normalformen* unterschieden. In der ersten Form werden alle Datenfelder so aufgelistet, wie sie von der Benutzerseite her benötigt werden. Dann überträgst Du die Daten in mehreren Schritten in die Form, wie Du sie in Deiner Datenbank auf die unterschiedlichen, miteinander verknüpften Tabellen verteilst. Ich komme im Kapitel *Beziehungen* auf Seite 16 noch einmal darauf zurück und werde es da an einem Beispiel erläutern.

Für den Entwurf der Daten gibt es viele Möglichkeiten. Die erste und einfachste ist: Papier und Bleistift sowie – meist – ein Radiergummi.

Übersichtlicher und für die Nachwelt verständlicher sind besondere Programme, die sich ausschliesslich mit der Konfiguration oder dem Entwurf der Daten und deren Verarbeitung im Programm beschäftigen. Diese nennt man *Datenflusspläne* – was sehr in die Einzelheiten geht – und *Programmaublaufpläne*. Die notwendigen Symbole findest Du unter dem Menüpunkt **Autoformen** auch in Microsoft Graph.

Eine weitere Möglichkeit besteht in Grafikprogrammen, mit denen Du Entwürfe einfach und sauber erstellen kannst. Die einmal erzeugten Elemente kannst Du einzeln abspeichern und in anderen Datenentwürfen wiederverwenden.

Access bietet Dir ausserdem ein Werkzeug, mit dem Du der Nachwelt imponieren kannst: die *Selbstdokumentation* gibt Dir nach Fertigstellung der Anwendung die Möglichkeit, den Entwurf Deiner Tabellen z.B. mit allen *Feldern* und *Felddatentypen* ausdrucken zu lassen. Aber das lässt sich erst hinterher machen, wenn alles fertig ist.

Wenn Du hier einen Entwurf machst, tust Du es für Dich selbst und für Deinen eigenen Überblick. Lass Dich nicht entmutigen, wenn andere Programmier glauben, dass man das hinterher machen könnte. Ein guter Datenentwurf ist wichtig – daran erkennt man einen professionellen Programmierer.

Was ich hier beschrieben habe, sind natürlich nur Ansätze. Wenn die Aufgabe eines *Pflichtenheftes* an Dich gestellt wird, solltest Du Dich tunlichst vorher in dieses Metier einlesen, um keine Überraschungen zu erleben.

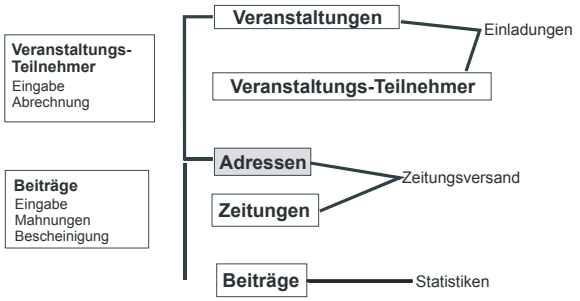
In unserem Fall mit der Datenbank *Verein* sind der Ausgangspunkt die Adressen der Mitglieder mit allen persönlichen Daten. Haben wir diese Daten für alle Mitgliedern eingegeben, können wir sie nach beliebigen Kriterien sortieren, etwa dem Namen oder der Postleitzahl. Wir können die entstandenen Listen auf dem Drucker ausgeben oder aus ihnen Serienbriefe in einer Textverarbeitung machen, wir können Statistiken erstellen oder sie in ein Tabellen-Kalkulations-Programm exportieren, wo sie berechnet oder anders weiterverarbeitet werden.

Unsere Datenbank „Verein“ soll folgende Arbeiten erledigen:

- Die Adressen aller Mitglieder
  - müssen vollständig aufgenommen werden
  - sollen schnell zu finden sein
  - werden nach diversen Gruppen sortiert, denen Serienbriefe zugeschickt werden oder nach denen Etiketten gedruckt werden
  - sollen einzeln in Briefköpfe von „Word“ exportiert werden
- Im Mitgliedsbeitrag ist eine Zeitung enthalten
  - diese wird per Post verschickt
- Mitglieder können an Veranstaltungen teilnehmen
  - dafür müssen sie registriert werden
  - sie entrichten eine Teilnahmegebühr
  - die Veranstaltungen werden abgerechnet
  - es wird eine Teilnehmerliste ausgedruckt
  - Mitglieder werden gezielt zu Veranstaltungen eingeladen, je nach früheren Besuchen
- Mitglieder bezahlen Beiträge
  - sie bekommen Mahnungen und Bescheinigungen über bezahlte Beiträge.

Der Datenentwurf sieht folgendermassen aus:

- Im Zentrum stehen die Mitglieder mit ihren Adressen, die in der *Tabelle „Adressen“* aufgeführt sind. Zur eindeutigen Erkennung jedes Datensatzes wird die Mitgliedsnummer mit einem *Primärschlüssel* versehen – jede Nummer kommt nur einmal vor.
- Mit der Kombination aus den Tabellen *„Adressen“*, *„Veranstaltungen“* und *„Veranstaltungs-Teilnehmer“* werden die Veranstaltungs-Teilnehmer erfasst.
- Aus den Tabellen *„Veranstaltungen“* und *„Veranstaltungs-Teilnehmer“* werden Einladungen erstellt, mit den Tabellen *„Adressen“* und *„Zeitung“* werden Zeitungen verwaltet, und über die Tabelle *„Beiträge“* werden BeitragsÜberwachung und Statistiken abgewickelt.



Die Grafik gibt natürlich nur einen groben Überblick; in einem *Pflichtenheft* müssten an dieser Stelle die einzelnen Funktionen sowie die *Beziehungen*, die die Tabellen zueinander aufweisen, genau ausgeführt werden. In unserem Fall werden die Beziehungen über die *Mitgliedsnummer* hergestellt, die als Primärschlüssel der Tabelle *„Adressen“* im Verhältnis 1:n zu *„Beiträge“*, *„Veranstaltungs-Teilnehmer“* und *„Zeitungen“* steht; auf einen Datensatz in *„Adressen“* können also mehrere Datensätze der anderen Tabellen kommen. Siehe auch die Illustration von *Beziehungen* auf Seite 19. Ausserdem müssten die genauen Felddefinitionen von Art und Grösse der Felder für jede dieser Tabellen beschrieben werden.

Hier ein kleiner Vorgeschmack davon:

- Die *Tabelle* **Adressen**
  - *Primärschlüssel* **`Mitgliedsnummer: (Datentyp)`** *Zahl*, (Grösse) *Long Integer*
  - Name: **`Text, 30`** (30 Buchstaben)
  - **`Eintrittsdatum: Datum`** (Datum, mittel)
  - etc. die anderen *Datenfelder*

Für die Verarbeitung muss aufgelistet werden:

- Das *Formular* „Adressen“ enthält folgende Eingabefelder:
  - **`Mitgliedsnummer`**
  - **`Name`**
  - **`Vorname`**
  - etc. alle Adress-relevanten Felder

Auch muss beschrieben werden, ob das Formular auf einer *Abfrage* basiert, oder ob ein Datensatz, d.h. eine Adresse, nur durch die Mitgliedsnummer identifiziert wird, wie das in unserem Beispiel der Fall sein wird.

Hier sei auf eine grundlegende Überlegung zur *Mitgliedsnummer* hingewiesen, die ein Programmierer mit den Anforderungen des Benutzers abgleichen muss:

Besteht die Mitgliedsnummer nur aus Zahlen, oder sind in ihr auch Zeichen oder Buchstaben enthalten?

Sind es nur Zahlen, wird die Bedienung im Programm für den Programmierer um ein Vielfaches einfacher; *Zahl* nimmt weniger Speicherplatz ein – was die Schnelligkeit erhöht – und lässt sich einfach berechnen.

In einem *Textfeld* lassen sich dagegen *Informationen* speichern. Du könntest mit einem Buchstaben vor der Nummer eine bestimmte Gruppierung für den Datensatz vornehmen. Setzt Du ein „i“ vor eine Nummer, z.B. *i0043*, kannst Du damit ausdrücken,dass der betreffende Interesse an dem Verein gezeigt hat –dafür das „i“) –, aber noch kein Mitglied werden will. Später kannst Du dann alle „i“-Leute erneut kontaktieren und fragen, ob sie nicht endlich Mitglieder werden wollen – z.B. in einem Serienbrief.

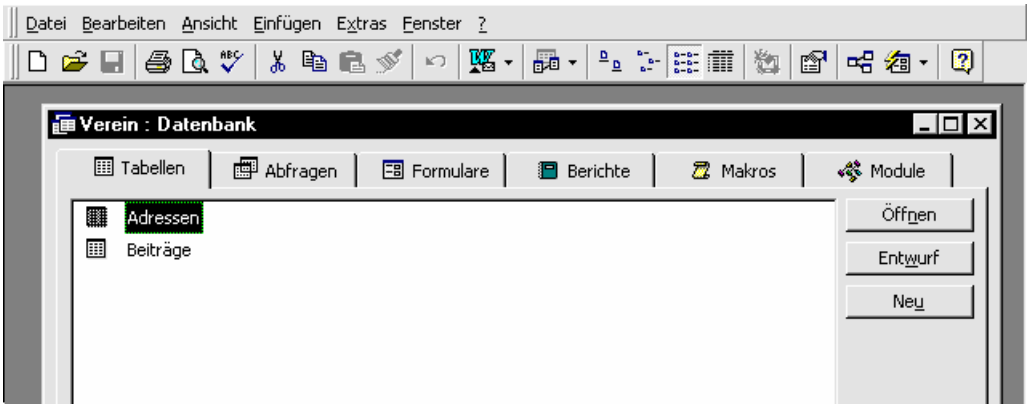
In diesem Fall wäre es vermutlich sinnvoller, wenn Du diese Information in einem speziellen *Feld* führtest, wie etwa „Status“, oder „Index“.

Zu überlegen wäre auch, ob die Mitgliedsnummer gleichzeitig *Primärschlüssel* ist, oder ob Du für den Primärschlüssel einen *Auto-Wert* nimmst, wobei Access für jeden Datensatz eine eigene, beliebige Nummer vergibt. Darauf kommen wir bei dem Entwurf der Tabellen zurück.

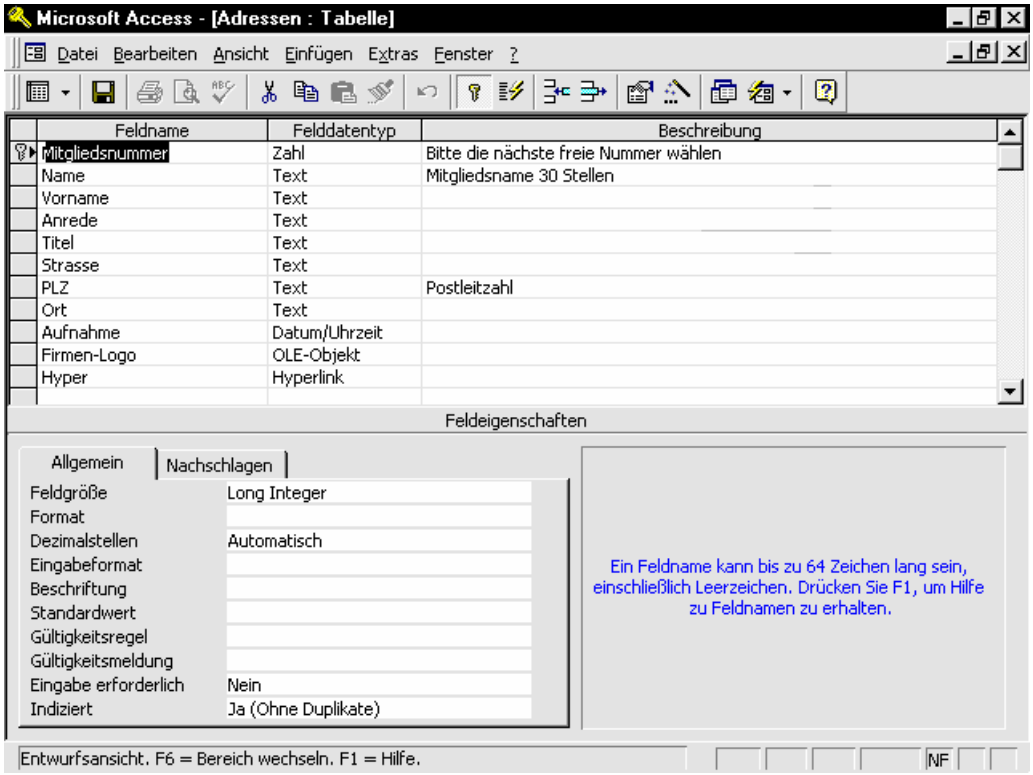
Diese Überlegungen zu Herkunft und Verbleib der Daten soll uns hier zunächst genügen. Das Nähere sehen wir uns beim Entwurf der Tabellen an.

Die Zentrale: Tabellen

Damit kommen wir endlich zur tatsächlichen Arbeit: wir definieren unsere Tabellen mit den dazugehörigen Feldern. Zunächst benötigen wir zwei Tabellen:



Die Tabelle „Adressen“ besitzt folgende Datenfelder, die Du entsprechend der Abbildung eingibst:



Erscheint bei Dir die Tabelle nur teilweise, klickst Du die Schaltfläche, um das Fenster in die *Vollbildansicht* zu holen.

Dem Feld *Mitgliedsnummer* habe ich die Grösse *Long Integer* zugeordnet, was den Zahlenbereich von –2.147.483.648 bis +2.147.483.647 umfasst. *Integer* nimmt zwar nur 2 Byte Speicherplatz ein, könnte aber aufgrund ihres Zahlenbereichs von –32.768 bis +32.767 eventuell zu klein werden.

Die anderen Felder sollten folgende Grössen haben – den *Felddatentyp* findest Du in der zugehörigen Spalte im oberen Bildschirm-Teil:

Name = **30**
Vorname = **25**
Anrede = **10**
Titel = **10**
Strasse = **35**
PLZ = **10** (muss Text sein, weil die Nummern evt.Länderkennzeichen enthalten!)
Ort = 35
Aufnahme = **Datum, mittel** (das Feld dient der Kenntnis, wann Du einen Datensatz aufgenommen hast). Du kannst die Jahreszahl 2-stellig eingeben (Definition im *Eingabeformat*), Windows speichert sie auf dem Rechner aber 4-stellig.

Zusätzlich nehmen wir noch auf:

Adresszusatz1 mit Text, = 40
Adresszusatz2, = 40

Dies für den Fall, dass eine Adresse genauere Informationen benötigt, wie etwa Stadtteile oder Distrikte, oder auch beschreibende Namen von Firmen.

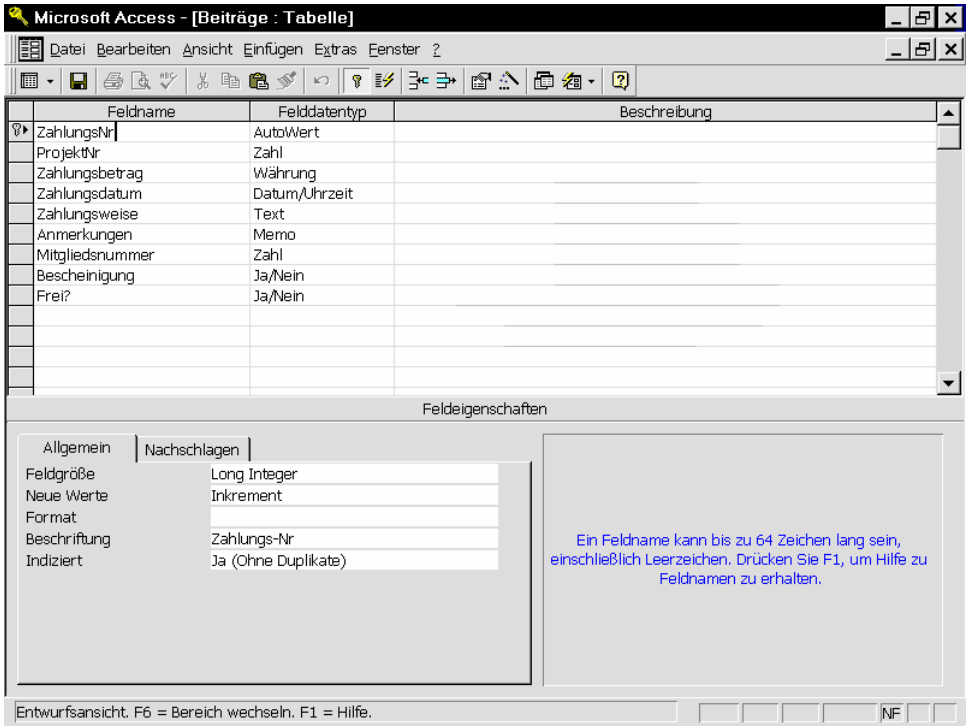
Mit der gezeigten Schaltfläche gehst Du in die *Datenblattansicht* und gibst einige Datensätze ein. Mit **Tab** oder **Pfeil rechts** bewegst Du Dich von Spalte zu Spalte. Das Ergebnis sollte etwa wie im Bild aussehen – aus Platzgründen habe ich die Spaltenbreiten verkleinert:

Adressen : Tabelle

Mitglieds	Name	Vorname	Anrede	Titel	Strasse	PLZ	Ort	Aufnahme
1	Schubert	Rainer	Herr		Glashüttenstr. 2	20357	Hamburg	03.01.98
2	Lichtenberg	Katrin	Frau	Dr.	Feldstr. 14b	28329	Bremen	30.12.97
3	Meyer	Eduard	Herr	Dipl.Ing.	Meyerhof 17	60117	Frankfurt	30.12.97
4	Meyer	Graziella	Frau		Meyerhof 17	60117	Frankfurt	30.12.97
5	Meier	Bärbel	Frau		Regenhain 1a	73148	Wrist	30.12.97
6	Langzahn	Susanne	Frau		Friederich-Eber	99096	Erfurt	30.12.97
7	Liebermann	Chantal	Frau		Altenbergerstr.	01277	Dresden	30.12.97
8	Rakow	Uwe	Herr		Maadesiel 157	26384	Wilhelmshaven	30.12.97

atensatz: 1 von 8

Du kannst natürlich andere Namen wählen. Mindestens ein Name, wie hier „Meyer“, sollte doppelt auftauchen – wir brauchen ihn später noch. Die Tabelle „Beiträge“ sieht folgendermassen aus:

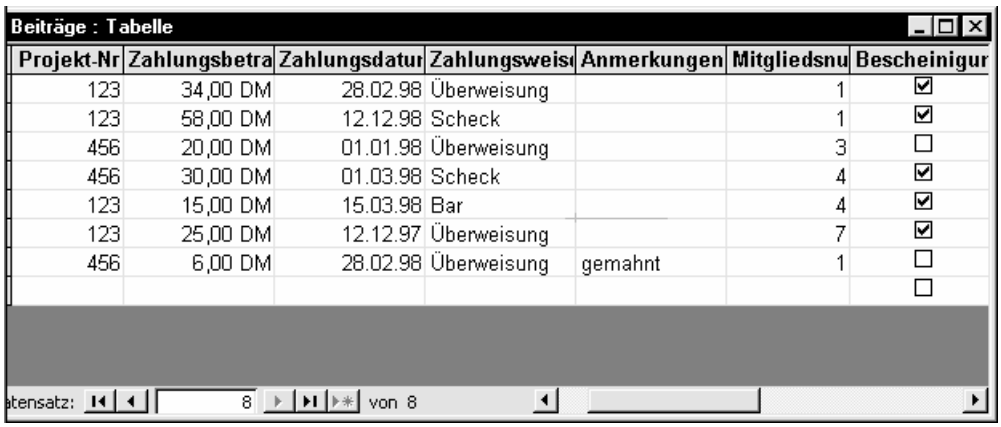


Die Felder haben folgende Grössen:

- **`ProjektNr = Long integer`**
- **`Zahlungsbetrag = Währung; Festkommazahl, Dezimalstellen = Automatisch`**
- **`Zahlungsdatum = Datum, kurz`**
- **`Zahlungsweise = 30`**
- **`Anmerkungen = Memo`**
  (die Grösse wird von Access vorgegeben)
- Die Mitgliedsnummer sollte die gleiche Grösse haben wie in der Tabelle „Adressen“. Über dieses Feld werden wir die *Beziehung* zur Tabelle „Adressen“ herstellen.
- **`Bescheinigung = Ja/Nein`** (Dies ist ein „Flag“-Feld, das keine einzustellende Grösse hat, weil nur zwischen den Aussagen **`Ja oder Nein`** unterschieden werden kann, d.h. die Fahne oder „flag“ ist gehisst oder nicht. Dahinter steht die kleinste Arbeitseinheit des Computers von 1 Bit.)

Gib auch hier einige Daten direkt in das *Datenblatt* der Tabelle ein. Die Mitgliedsnummer, die Du einträgst, muss in der Tabelle „Adressen“ vorhanden sein, sonst gibt es später Probleme.

So könnte es aussehen:



Felddatentypen

Im Folgenden sehen wir uns die Felddatentypen an, die in den beiden bestehenden Tabellen nicht auftauchen.

OLE Objekt:

OLE, Objekt linking and imbedding, bezeichnet ein verknüpftes oder eingebettetes Objekt etwa aus einer anderen Office-Anwendung wie Word, Excel etc. Hiermit können auch Grafiken eingebunden werden, wie z.B. ein Firmen-Logo.

Du kannst ein Feld mit dem Namen „Firmen-Logo“ in die Tabelle „Adressen“ aufnehmen. Bei den *Formularen* weiter unten können wir dann eine Grafik einbinden.

Hyperlink:

Eine Einheit aus Zahlen und/oder Text, die an einer bestimmten Adresse abgespeichert wird. Auf den Inhalt wird zugegriffen, sobald das Feld aktiviert wird. Praktiziert wird diese Methode z.B. in den Querverweisen der Hilfe-Texte.

Man unterscheidet zwischen Verweisen auf eine Internet-Adresse, also einem URL (Uniform Resource Locator), oder dem Pfad eines Netzwerk-Servers wie etwa UNC (→ Glossar).

Gib ein Feld mit dem Namen „Hyper“ in die Tabelle „Adressen“ ein. Die Bedienung sehen wir uns bei den *Formularen* an.

Nachschlage-Assistent

Aufgabe dieses Assistenten ist es, ein Nachschlage-Feld zu installieren. Auf diese Weise können Werte aus anderen Tabellen oder Abfragen übernommen werden. In der nachzuschlagenen Tabelle wird ein Primärschlüssel-Feld erstellt, das den Werten des Nachschlag-Feldes entspricht. Du kannst den gleichen Vorgang auch von Hand machen, indem Du zwei Tabellen zueinander in *Beziehung* stellst. Darauf kommen wir später noch zurück – nämlich im Kapitel *Makros/Dialogfenster einbinden* auf Seite 46.

Eine grundsätzliche Überlegung, die Du zu der Anwendung der verschiedenen *Felddatentypen* machen solltest, ist, welchen Typ Du wann benutzt.

Ja/Nein ist der einfachste Typ – er beansprucht wenig Speicherplatz, hat aber auch wenig Aussagewert, weil er nur eine „Ja“- *oder* „Nein“-Entscheidung enthält.

Zahl nimmt verhältnismässig wenig Speicherplatz ein. Dieses Feld lässt sich einfach verarbeiten, um Berechnungen anzustellen, oder wenn Du eine bestimmte Zahl suchen willst. Es ist dem *Textfeld* vorzuziehen, mit dem Verarbeitungen schwieriger, aber durchaus möglich sind. Hätten wir z.B. für die *Mitgliedsnummer* ein Textfeld gewählt, wäre die Suche nach der nächsten freien Nummer schwierig. Zur Darstellung eines Buchstaben benötigt der Computer 1 Byte.

AutoWert, der automatische Zähler, eignet sich besonders für den Primärschlüssel, falls kein anderer vorgesehen ist. Bei *Replikationen* (siehe Kapitel *Schluss* auf Seite 49) wird die normale Grösse von 4 Byte auf 16 Byte erhöht.

Memo sollte nur für Anmerkungen und dergleichen benutzt werden. Die Werte in diesem Feld können nicht berechnet oder anderweitig verarbeitet werden.

Beziehungen

Im *relationalen Datenbank*-Modell werden Beziehungen zwischen den Datenquellen hergestellt. Was heisst das?

Mitglieder bezahlen Beiträge. Jedes Mitglied wird über einen Datensatz in der Tabelle „Adressen“ identifiziert. Diese Mitglieder bezahlen monatlich oder jährlich ihre Beiträge, d.h. auf einen Datensatz in „Adressen“ kommen im Laufe der Zeit mehrere Datensätze in „Beiträge“. Dies äussert sich in der Beziehung 1:n, wobei „n“ eine beliebige Anzahl bedeutet.

Im Karteikarten-Modell bewahrt man die Adressen der Mitglieder meistens in einem Kasten auf und alle bezahlten Beiträge in einem Ordner bzw. einem weiteren Kasten. Bezahlt ein Mitglied seinen Beitrag, musst Du über den Namen die Mitgliedsnummer suchen, der Du den bezahlten Beitrag zuordnest.

Ähnlich macht man es in einer *relationalen* Datenbank. Du brauchst ein *Feld*, das in beiden Tabellen gleich ist. Mit seiner Hilfe wird eine *Beziehung* zwischen den beiden Tabellen hergestellt. Bezahlt ein Mitglied seinen Beitrag, sucht Du über den Namen die *Mitgliedsnummer* – der *Name* allein reicht nicht, weil Namen gleichlautend sein können; also keine eindeutige Kennung abgeben! Dann gibst Du den Beitrag ein, der über die *Mitgliedsnummer* eindeutig einem Mitglied zugeordnet wird. Das gleichlautende Feld

Mitgliedsnummer der zwei Tabellen „Adressen“ und „Beiträge“ nennt man das *Verknüpfungsfeld*.

An dieser Stelle möchte ich noch einmal kurz auf die *Normalformen* zurückkommen, mit denen Du Dich in relationalen Datenbanken sauber an eine übersichtliche Datenstruktur heranarbeiten kannst. Ausgangspunkt sind hier die Bedingungen der realen Welt, das heisst die Dinge, die ein späterer Benutzer Deines Programms braucht, um seine Arbeit zu erledigen – in der Fachsprache werden sie als *entities* bezeichnet. In unserem Fall wäre das z.B. die Bescheinigung über bezahlte Beiträge etwa für die Steuer, die zu Ende eines Jahres als Serienbrief versandt wird.

In der *unnormalisierten* Form der realen Welt sind für eine Bescheinigung folgende Daten notwendig:

- Name und komplette Adresse des Mitglieds
- Zahlungsbetrag
- Zahlungsdatum
- Zahlungsweise
- Bescheinigung (braucht das Mitglied eine oder nicht)
- sowie eventuell Anmerkungen.

Wie gesagt brauchst Du zur eindeutigen Identifizierung des Mitglieds noch eine Ordnungsnummer, in diesem Fall also die Mitgliedsnummer. Genauso benötigst Du die ZahlungsNr für die Beiträge.

- Mitgliedsnummer
- ZahlungsNr

Diese Daten nennt man in ungeordneter Form *unnormalisiert*.

Da die Beiträge sich aber *wiederholen*, da Du also mehrere Beiträgssätze für ein Mitglied haben wirst, trennst Du die Daten in zwei Abteilungen auf – zur Vereinfachung habe ich die Adressdaten wie Ort, PLZ etc. in ein Kästchen getan. Dies nennt man die *Überführung in die erste Normalform.*

Das könnte man etwa so illustrieren:

Gruppe

Name	Adresse	**Mitgliedsnummer**

Wiederholungsgruppe

ZahlungsNr	Zlgs-betrag	Zlgs-datum

Zlgs-weise	Bescheinigung	**Mitgliedsnummer**

Die *Mitgliedsnummer* ist in der Gruppe wie erwähnt *Primärschlüssel*, in der *Wiederholungsgruppe* – oder wie es in Access heisst: im *Detaildatensatz* – ist sie der *Fremdschlüssel*.

Wenn die Daten umfangreicher sind und die *Wiederholungsgruppe* in mehrere Tabellen aufgeteilt werden muss, kann die Überführung in weitere Normalformen notwendig werden.

So verfährst Du mit allen Prozessen Deines Benutzers und fügst die neu hinzukommenden Daten den vorhandenen Tabellen zu oder bildest neue Tabellen, falls das erforderlich ist.

Dies ist im Rahmen dieses Heftes eine sehr einfache Darstellung; sie soll Dir eine Vorstellung davon geben, wie Du die Übersicht über Deine Daten behalten kannst. Wenn Deine Datenbanken grösser werden, wirst Du automatisch zu solchen Methoden greifen – spätestens dann, wenn Du – leider – Nächte durchsessen hast, um Deine Datenstruktur nachträglich zu ändern. Ich habe solche Nächte schon hinter mir.

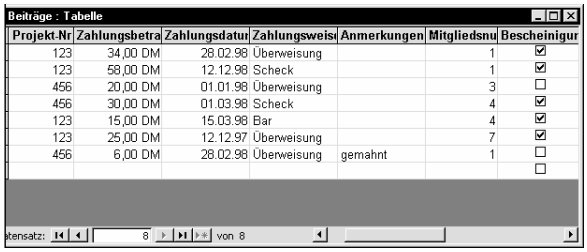
Doch jetzt genug des Moralisierens!!

Exkursion: die Hilfe

Sehen wir uns als Exkursion den Gebrauch der Hilfe näher an. Ich finde, dass sie sehr sinnvoll und gut durchdacht aufgebaut ist. Es gibt nicht nur verschiedene Arten und Weisen, sich einem gesuchten Begriff zu nähern, sondern Du begegnest auch unterschiedlichen Schwierigkeitsgraden, je nachdem, wie weit Du Dich in ein bestimmtes Gebiet hineinbegibst.

Microsoft Access-Hilfe

Ich gehe davon aus, dass Du mit der üblichen Hilfsfunktion unter Windows vertraut bist. Es gibt aber eine zusätzliche Form der Access-Hilfe. Diese Hilfe kann über **F1** auch unmittelbar aus Access aufgerufen werden. In der Version 2.0 gab es sie noch nicht. Probiere selber die verschiedenen Hilfemethoden aus und wähle die, die Dir am meisten zusagt.



Man kann die Microsoft Access-Hilfe als Kombination der beiden anderen Hilfeformen bezeichnen. Auf der einen Seite ist sie eine Direkthilfe.Sie geht aber auch gezielt zu den Suchbegriffen der „grossen“ Hilfe. Wenn Du diese Hilfe aufrufst, registriert sie automatisch, wo Du Dich gerade befindest. Sie bietet Dir mit anderen Worten eine Hilfestellung für den aktuellen Arbeitsbereich an, ob es sich nun um *Tabellen*, *Makros*, *Abfragen* oder um anderes handelt, und beachtet dabei auch, ob Du aktuell in der *Entwurfs*- oder der *Formularansicht* arbeitest. Der Nachteil dieser Direkthilfe ist, dass ihre Antworten sehr allgemein gehalten sind und Du Dich durch etliche Querverweise durchfragen musst. Ihr Vorteil ist andererseits, dass es manchmal schwierig ist, eine gezielte Frage an das Hilfe-System zu stellen. Darum könnte man die *Microsoft Access-Hilfe* als eine Agentur bezeichnen, die zwischen Deinen Problemen und der riesigen Auswahl an Hilfethemen vermittelt. Sie schafft mit anderen Worten eine Vorauswahl, indem sie sich fragt, welche Probleme Du als Benutzer in einem bestimmten Augenblick haben könntest, und versucht eine Lösung zu finden, indem sie in den Hilfethemen nach einer passenden Antwort sucht.

Microsoft im Web

Microsoft *ist* nicht nur modern –Microsoft *macht* modern. Das Unternehmen bestimmt den Massstab von „modern“. Logischerweise findest Du auch Hilfe im Internet. Eine Anwendung wie Access ist riesengross, so dass sich stetig neue Problemlösungen ergeben, die unmittelbar von Microsoft veröffentlicht oder aber auch in Foren von Anwendern diskutiert werden. Ich finde es manchmal ganz angenehm zu sehen, dass es andere gibt, die die gleichen Probleme haben wie ich. Wenn ich dann noch Antworten auf meine Fragen bekomme – was will ich mehr!

Voraussetzung ist, dass Du Dich bei einem „Provider“ anmeldest, der Dir den Zugang ins Internet zur Verfügung stellt – etwa CompuServe, T-Online, Microsoft Netzwerk, AOL oder andere. Softwareseitig sind alle Vorbedingungen geschaffen, um eine Verbindung herzustellen. Dazu musst Du den Internet Explorer installiert haben, über den Du den Setup-Assistenten startest. In diesem Heft ist leider nicht genug Platz, um näher auf diese Aufgabe einzugehen.

Aus dem Internet kannst Du Dir unter Umständen auch Tools (Werkzeuge) herunterladen, die Dir die Arbeit erleichern oder die Microsoft inzwischen neu entwickelt hat – wie z.B. das „patch“, mit dem Du das Euro-Zeichen einbinden kannst.

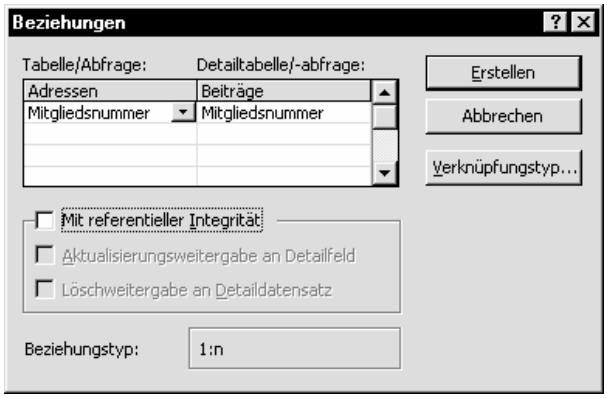
So benutzt Du die Hilfe

Was die verschiedenen Formen der Hilfe und ihre Nützlichkeit angeht, musst Du einfach üben. Begegnet Dir ein Wort, das Du nicht verstehst, schau nach. Findest Du es nicht in der Hilfe, sieh im Glossar am Schluss dieses Heftes nach. Manchmal hilft auch ein Lexikon, weil man bestimmte Dinge, für die es im Deutschen unmittelbar kein Wort gibt, besser mit Fremdwörtern ausdrücken kann. „Erst-Schlüssel" hat z.B. nicht die gleiche Bedeutung wie das Wort „Primärschlüssel", das sich im Englischen über einen langen Zeitraum hin entwickelt hat – auch wenn es lateinischen Ursprungs ist.

Noch einmal: Beziehungen

Kommen wir zu den *Beziehungen* zurück. Für das Bearbeiten von Beziehungen bietet Access ein eigenes Fenster an, das Du über den Menüpunkt **Extras | Beziehungen** aufrufst. Werden hier die Tabellen nicht angezeigt, holst Du das über den Menüpunkt **Beziehungen | Tabellen anzeigen** nach.

Du markierst die entsprechende Tabelle und klickst auf **Hinzufügen**, so dass die Tabellen *Adressen* und *Beiträge* im Fenster *Beziehungen* erscheinen. Dann klickst Du in der Tabelle *Adressen* auf *Mitgliedsnummer*, hältst die Maustaste gedrückt und ziehst den Cursor auf die *Mitgliedsnummer* in der Tabelle *Beiträge*. Access erstellt eine Verbindungslinie zwischen beiden Tabellen. Lässt Du die Maustaste los, siehst Du dies Fenster:



Du kannst das Fenster auch öffnen, indem Du auf die Verbindungslinie der beiden Tabellen doppelklickst, die Du im nächsten Bild siehst.

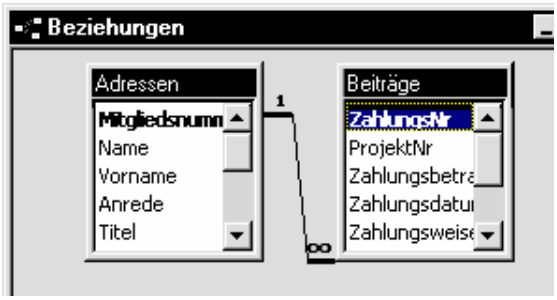
Nun klickst Du auf referentielle Integrität und die damit verbundenen Optionsfelder Aktualisierungsweitergabe an Detailfeld sowie Löschweitergabe an Detaildatensatz.

→ Klickst Du auf ? in diesem Fenster, aktivierst Du die *Direkthilfe*. Ziehe nun den zum Fragezeichen gewordenen Mauszeiger auf *referentielle Integrität*. Ich finde die Erklärung sehr hilfreich – ich könnte es selber nicht besser erklären! Abschliessend gibt es bei » einen Verweis auf die „grosse“ Hilfe, wo das Thema noch einmal ausführlich beschrieben wird.
Entsprechend verfährst Du mit den beiden anderen erwähnten Optionsfeldern und dem *Beschreibungsfeld* „Beziehungstyp“. ←

Referentielle Integrität ist eine echte Arbeitserleichterung von Access, die das Umgehen mit verknüpften Daten entscheidend vereinfacht. Natürlich zwingt Dir die Integrität ein Korsett auf, das bestimmte Veränderungen an den Daten nicht erlaubt. Später wirst Du von Fall zu Fall entscheiden müssen, ob Du diese Hilfe von Access in Anspruch nimmst oder die Relationen von Hand programmierst.

Nun klickst Du auf **Verknüpfungstyp...**, um festzulegen, in welcher Weise die Daten der beiden Tabellen bearbeitet werden sollen. Was die Verknüpfungstypen betrifft, empfehle ich Dir ebenfalls die *Direkthilfe*, um den Sachverhalt verständlich zu machen. „1“ ist die Standardeinstellung, wonach bei einer Abfrage nur die Datensätze gewählt werden, die in beiden Tabellen vorkommen – also keine leeren, d.h. in der Detail-Tabelle nicht vorkommende Datensätze.

Ein Klick auf **Erstellen** im letzten Bild ergibt folgende Beziehung:



Vergiss nicht, vollzogene Änderungen umgehend zu speichern – sonst können Deine Daten durch einen etwaigen Systemabsturz oder sonstigen Stillstand verloren gehen. Immerhin arbeitest Du mit dem Betriebssystem „Windows“, das nach den Qualitätsmassstäben von vor 20 Jahren niemals auf den Markt gekommen wäre; man hätte sich schlicht geschämt, ein so unfertiges Produkt anzubieten. Heute ist das anders, wir alle sind zu unbezahlten, dafür aber selbst bezahlenden Betatestern geworden. Leider sind die denkbaren Alternativen noch

schlechter oder einfach schlecht vermarktet (Linux und andere Unix-Derivate).

Darum: immer wieder zwischendurch speichern – und nicht erst dann, wenn Du Deine Arbeit abgeschlossen hast!

Für das weitere Arbeiten benötigen wir noch folgende Tabellen mit den den entsprechenden Feldern:

Eine Tabelle „Veranstaltungs-Teilnehmer“:

- **laufende Nummer = AutoWert**
- **Mitgliedsnummer = Zahl; Long Integer**
- **Veranstaltungs-Nr = Text; 10**
- **Bezahlt = Währung**
- **Zahlungsweise = Text; 20**
- **Anzahlung = Währung**
- **Wann? = Datum**
- **Abgesagt = Ja/Nein**
- **Mitglied? = Ja/Nein**
- **Bemerkungen = Text; 10**

Diese Tabelle ist über *Mitgliedsnummer* mit der Tabelle „Adressen“ verknüpft; Beziehung: n:1; Referentielle Integrität: ja; Aktualisierungsweitergabe: ja; Löschweitergabe: ja sowie über *Veranstaltungs-Nr* mit der Tabelle *Veranstaltungen* verknüpft ; n:1; ja; ja; ja.

Eine Tabelle *Veranstaltungen*:

- **Veranstaltungs-Nr = Text; 10; Primärschlüssel**
- **Veranstaltungs-Name = Text; 50**
- **Veranstaltungs-Art = Text; 50**
- **Dauer von = Datum**
- **Dauer bis = Datum**
- **Max Teilnehmer = Zahl; Byte**
- **Bemerkungen = Memo**
- **Kosten = Währung**

Diese Tabelle ist mit der Tabelle *Veranstaltungs-Teilnehmer* verknüpft, s.o.

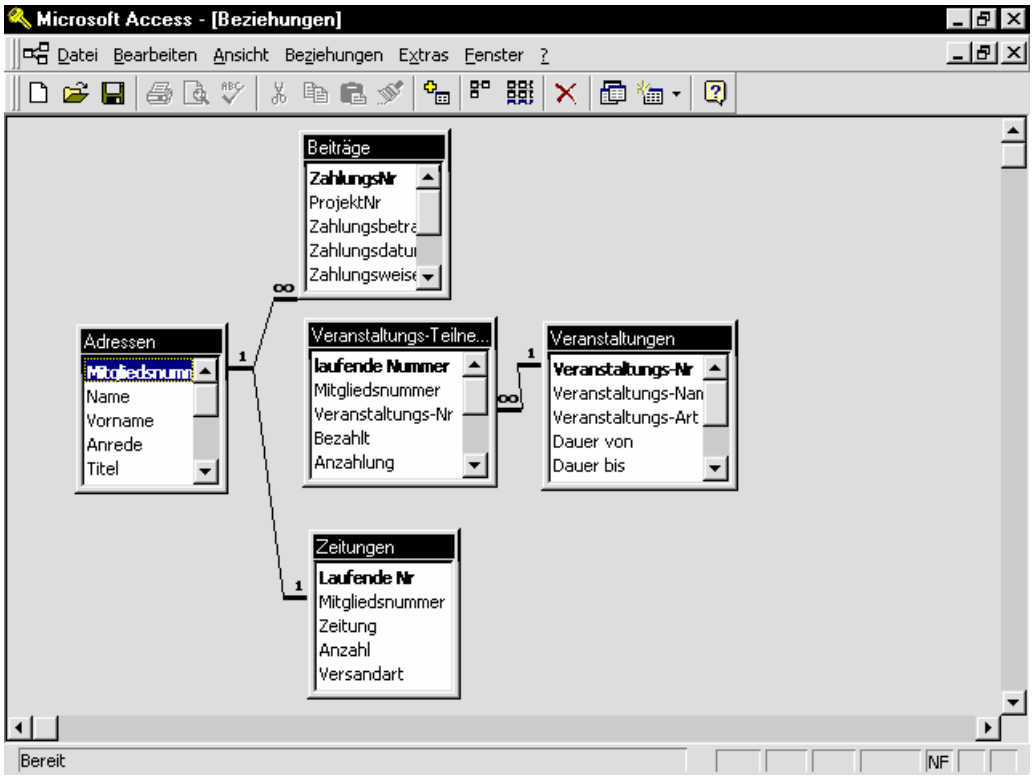
Eine Tabelle *Zeitungen*:

- **Laufende Nr = AutoWert**
- **Mitgliedsnummer = Zahl; Integer**
- **Zeitung = Ja/Nein**
- **Anzahl = Zahl; Byte**
- **Versandart = Text; 6**

Diese Tabelle ist über *Mitgliedsnummer* mit *Adressen* verknüpft; **n:1; ja; ja; ja**.

Zunächst einmal legst Du die Tabellen nur an – ihre Daten werden wir später mit Hilfe der entsprechenden Formulare eingeben.

Das Ergebnis ist das folgende Fenster:



Abfragen

Abfragen sind der Bereich in Access, in dem die eigentliche Arbeit verrichtet wird. Ausserdem sind die Abfragen sozusagen Mainzelmännchen, die der Benutzer nicht sieht, weil sie ihr Treiben im Hintergrund verrichten – im Gegensatz zu den *Formularen* und *Berichten*, mit denen der Benutzer ständig zu tun hat.

Du hast sicher schon einfache Abfragen erstellt und weisst vermutlich auch schon, was eine *Auswahlabfrage* ist. Es sollte Dir auch bekannt sein, dass die Auswahlabfrage die Grundlage für andere Abfragetypen wie *Kreuztabellenabfragen* oder *Löschabfragen* darstellt.

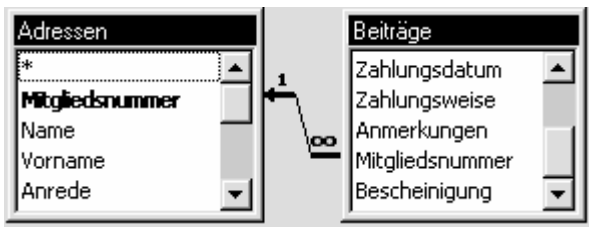
In diesem Kapitel wollen wir uns so eine Umwandlung erarbeiten. Ausserdem wollen wir über Abfragen Berechnungen durchführen und die Anzahl von ausgesuchten Datensätzen angeben. Zu diesem Zweck werden wir viel mit der gezeigten Schaltfläche arbeiten, dem *Ausdrucks-Editor*.

Beziehungen in Abfragen

Abfragen in Access sind die *Auswertungen* des Programms. Sie beziehen ihre Daten aus verschiedenen Quellen. Am naheliegensten sind die Tabellen – dort werden die Daten gespeichert. Neben den Tabellen kannst Du auch Abfragen heranziehen oder sogar beide kombinieren.

Darum wollen wir uns noch einmal den *Beziehungen* der Datenquellen widmen – diesmal aber vor allem insoweit sie eine Abfrage betreffen. Denn hier müssen es nicht dieselben Beziehungen sein, die wir vorher für die Tabellen festgelegt haben.

Also erstellst Du zunächst einmal eine neue *Auswahlabfrage* in der *Entwurfsansicht*. Du fügst die Tabellen „Adressen“ und „Beiträge“ hinzu, markierst alle Felder beider Tabellen und ziehst sie in den QBE-Bereich (→ Glossar) der Abfrage. Das kannst Du übrigens auch über die mit einem * versehene Zeile bewerkstelligen.

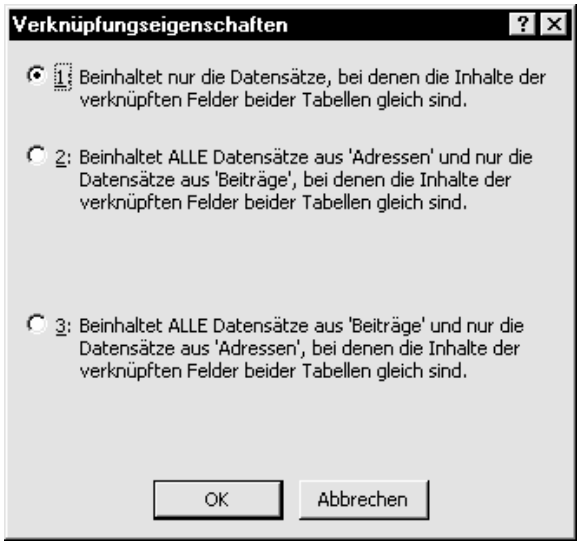


Das Ergebnis der Aktion ist, dass im QBE-Bereich der Tabellen-Name mit einem * steht, was besagt, dass alle Felder der Tabelle in die Abfrage einbezogen sind. Das ist zwar schnell gemacht, dafür kannst Du in diesem Fall aber die Felder nicht einzeln bearbeiten – etwa nach einem Feld sortieren oder *Kriterien* bestimmen. Darum solltest Du die Methode über das Markieren benutzen, da wir die einzelnen Felder benötigen.

→ Probiere beide Methoden aus und sieh Dir die Unterschiede an! ←

Speichere die Abfrage als **`Beiträge eingeben`** und führe sie durch, indem Du die Schaltfläche anklickst. Hast Du in die Tabelle „Beiträge“ Daten eingegeben, erscheinen jetzt im *Datenblatt* alle Beiträge mit den kompletten Adressen der Mitglieder.

Du kehrst in die *Entwurfsansicht* zurück und doppelklickst auf die Verbindungslinie zwischen den Tabellen, was Dich ins nächste Fenster bringt:



Typ „1“, die Exklusionsverknüpfung, ist voreingestellt, weil sie am häufigsten vorkommt. Eine genauere Beschreibung findest Du unter *Tabellen* auf Seite 13.

Hier aktivierst Du Typ „2“ und klickst auf **OK**. Wie Du siehst, ändert sich der Pfeil zwischen den beiden Tabellen. Gehst Du wieder in die *Datenblattansicht* der Abfrage, siehst Du, dass alle Datensätze aus „Adressen“ angezeigt werden.

Ebenso verfährst Du mit dem Typ „3“, der „Rechts-Exklusionsverknüpfung“.

Ein weiterer Verknüpfungstyp ist die *Reflexivverknüpfung*, die Du anwenden kannst, um in einer Abfrage eine Tabelle zweimal abzufragen. So

kannst Du z.B. eine Tabelle gleichzeitig nach zwei *Feldern* abfragen, deren Werte gleich sind.

Für diesen Zweck fügst Du in der *Entwurfsansicht* von „Beiträge eingeben“ die Tabelle „Beiträge“ ein zweites mal hinzu – Access gibt ihr den Namen „Beiträge_1“. Eine Verbindung zwischen beiden Tabellen erstellst Du über die Felder „ZahlungsNr“ und „Mitgliedsnummer“ und ziehst wie im Bild **Beiträge eingeben: Auswahlabfrage** die folgenden 4 Felder in den Entwurfsbereich:

Wenn Du die Abfrage ausführst, erhältst Du alle Mitglieder, die Datensätze mit gleicher *Mitgliedsnummer* und *ZahlungsNr* haben.

Name	Ort	Zahlungs-Nr	Mitgliedsnummer
Schubert	Hamburg	1	1
Meyer	Frankfurt	3	3
Meyer	Frankfurt	4	4

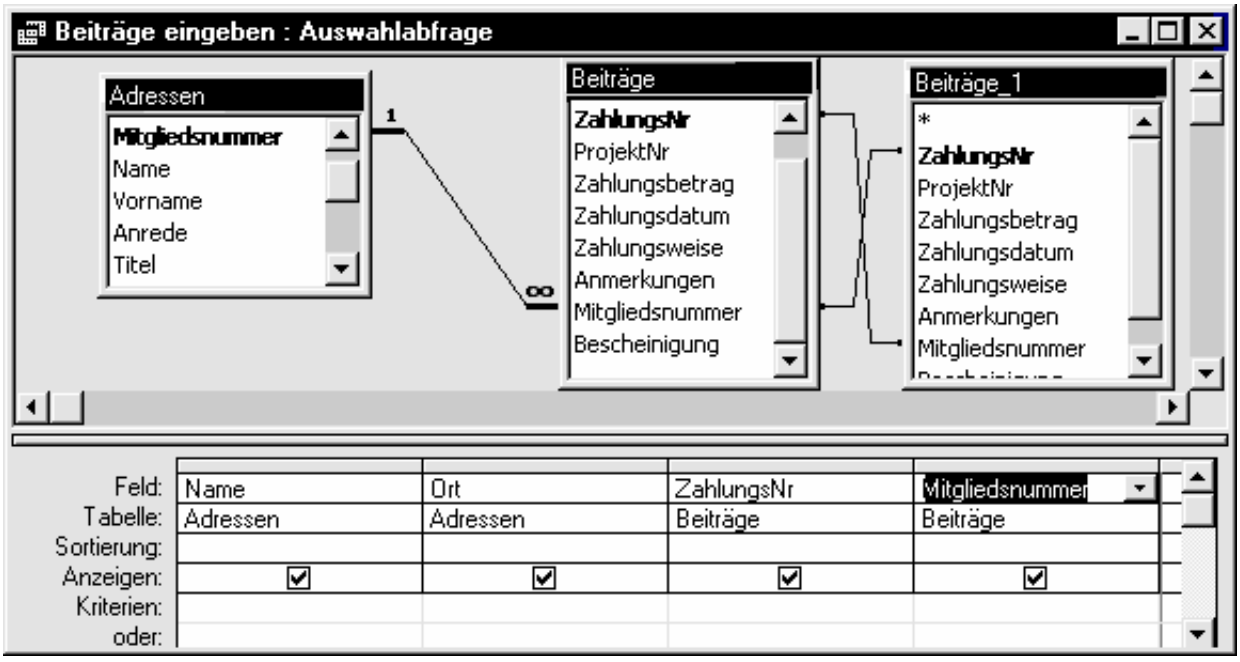
Steinige mich nicht für dieses etwas gezwungene Beispiel! Es geht schlicht darum, Dir zu zeigen, was eine Reflexivverknüpfung ist. Die Abfrage hat augenblicklich keinen praktischen Wert.

Du kannst die Abfrage in den Kriterien etwa des Feldes „Name“ beliebig einschränken, indem Du in der Zeile „Kriterien” z.B. **`Meyer`** eingibst. In diesem Fall werden nur die Meyers angezeigt, die die gleiche „ZahlungsNr“ und „Mitgliedsnummer“ haben.

Ausserdem muss es sich hier nicht unbedingt um Tabellen handeln – die Daten können auch aus einer Abfrage oder einer mit einer Tabelle verknüpften Abfrage stammen. Es kommt ganz darauf an, wie Du die Daten zusammenstellen willst. Der Vorgang ist in jedem Fall derselbe: Du fügst eine Tabelle oder Abfrage in der Entwurfsansicht hinzu und verknüpfst sie miteinander über Felder Deiner Wahl.

Die Sache kann z.B. folgenden Sinn haben : Du hast eine Abfrage erstellt, von der Du weisst, dass Du sie immer wieder benutzen wirst. Um sie aber nicht jedesmal abwandeln zu müssen, wenn Du unterschiedliche Werte abfragen willst, nimmst Du diese Abfrage als Basis für eine neue Abfrage. Nur in dieser letzteren Abfrage gibst Du dann die gewünschten *Kriterien* ein.

→ Erzeuge nun eine Abfrage auf der Basis von „Beiträge eingeben“, die Dir die *Zahlungsbeträge* grösser als DM 10 (>10) ausgibt.←



Aggregatfunktionen und andere Funktionen

Access bietet eine Anzahl sogenannter *Aggregatfunktionen* an, durch die Du verschiedene Informationen über die Datensätze einer Tabelle oder einer Abfrage erfährst. Probieren wir das einmal aus: Du erstellst eine neue Abfrage, die auf den Tabellen „Adressen“ und „Beiträge“ basiert. Die bei dem Hinzufügen der Tabellen vorgegebene *Exklusions-Verknüpfung* Typ 1 – beschrieben unter *Beziehungen in Abfragen* – kann beibehalten werden. Eventuell kannst Du die bestehende Abfrage „Beiträge eingeben“ kopieren und benutzen. Das machst Du so:

Du kopierst sie aus dem *Datenbankfenster* in die Zwischenablage und fügst sie mit neuem Namen wieder ein.

Du ziehst das Feld „Name“ in den Entwurfsbereich und klickst auf Σ, um die Zeile *Funktion* einzufügen. Dann gehst Du mit der Maus in diese Zeile und klickst auf ▼, um Dir die Liste der Funktionen anzeigen zu lassen, die Du zur Berechnung der Datensätze heranziehen kannst.

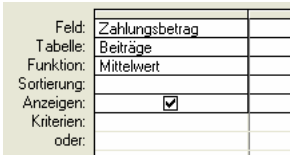
Formel	Ergebnis	Datentypen
Anzahl	Anzahl der Werte (keine Null-Werte)	Zahl, AutoWert, Datum/Zeit, Währung, Text, Memo, OLE-Objekte
ErsterWert	Wert im ersten Satz	Zahl, AutoWert, Datum/Zeit, Währung, Text, Memo, OLE-Objekte
LetzterWert	Wert im letzten Satz	Zahl, AutoWert, Datum/Zeit, Währung, Text, Memo, OLE-Objekte
Max	Höchster Wert	Zahl, AutoWert, Datum/Zeit, Währung, Text, Memo
Min	Niedrigster Wert	Zahl, AutoWert, Datum/Zeit, Währung, Text, Memo
Mittelwert	Mittelwert	Zahl, AutoWert, Datum/Zeit, Währung
StdAbw	Standardabweichung	Zahl, AutoWert, Datum/Zeit, Währung
Summe	Summe aller Werte	Zahl, AutoWert, Datum/Zeit, Währung
Varianz	Varianz der Werte eines Feldes	Zahl, AutoWert, Datum/Zeit, Währung
Auswahl	Zweck	
Gruppierung	Gruppen definieren	
Ausdruck	Für ein berechnetes Feld	
Bedingung	Feld ausblenden	

Du wählst den Punkt „Anzahl“, um Dir die Anzahl der Datensätze ausgeben zu lassen, die Nachnamen von Mitgliedern enthalten, und führst die Abfrage aus.

Das Ergebnis kannst Du Dir u.a. in einem Feld eines Formulars anzeigen lassen. Eine Möglichkeit besteht z.B. in einem ungebundenen *Kombinationsfeld*, bei dem Du in der *Datensatzherkunft* – im Eigenschaften-Fenster des Kombinationsfelds – auf das Ergebnis der Abfrage verweist. Mehr dazu im entsprechenden Abschnitt auf Seite 32.

Nun ziehst Du die Felder *Mitgliedsnummer*, *Name* und *Zahlungsbetrag* in den *Entwurfsbereich* und lässt den Mittelwert von „Zahlungsbetrag“ errechnen. Denke daran, dass der Mittelwert *nicht* für *alle* Zahlungen berechnet wird, sondern jeweils nur für *ein* Mitglied. Du musst also mehrere Beiträge für die Mitglieder eingegeben haben, aus denen dann der Mittelwert errechnet wird.

Willst Du den Mittelwert *aller* Mitgliedsbeiträge berechnen, musst Du allein das Feld *Zahlungsbetrag* in den Entwurfsbereich ziehen und die *Funktion* auf *Mittelwert* stellen.



→ Probiere die anderen *Aggregat*-Funktionen ebenfalls aus. ←

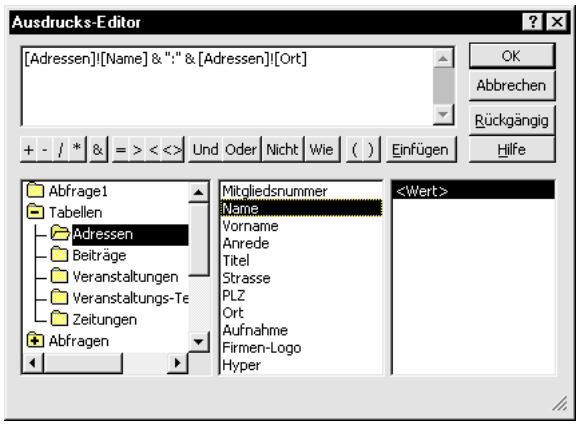
Berechnungen in Feldern

In Access kannst Du Berechnungen auf vielfältige Art und Weise in Abfragen durchführen lassen. So ist es möglich, in der Abfrage „Beiträge eingeben“ den *Zahlungsbetrag* mit einem bestimmten Wert zu multiplizieren, um z.B. eine Umrechnung von „DM“ auf „Euro“ zu durchzuführen. Ebenso kannst Du z.B. zwei *Textfelder* zusammenfügen, um Artikelgruppen mit den dazugehörigen Artikeln in einer Liste anzuzeigen. Natürlich kann Access auch Quadratwurzeln aus Zahlen ziehen oder mit Tangens (*tan*) einen Kreiswinkel berechnen.

Nehmen wir nun an, dass Du *Name* und *Ort* der Tabelle „Adressen“ in einem Feld zusammenfassen willst. Das erfordert folgende Schritte:

- Zunächst erstellst Du eine neue *Auswahlabfrage* auf der Basis der Tabelle „Adressen“, ohne Felder in den *Entwurfsbereich* zu ziehen.
- Nun gehst Du in die Zeile „Feld“ und klickst auf *Aufbauen*.

- Hier *kannst* Du frei Eingaben machen, wobei die *Felder*, die in die Berechnung genommen werden, in eckige Klammern `[ ]` gesetzt werden müssen. Um Schreibfehler (SyntaxFehler) zu vermeiden, schlage ich vor, die Felder aus den vorgegeben Listen einzufügen. Die Syntax, also die Formulierung und Rechtschreibung, der *VBA*-Befehle erfordert korrekte Eingaben – der Computer versteht nur „richtig“ oder „falsch“, halbrichtig, wie im mündlichen Verkehr, gibt es leider nicht.



- Doppelklickst Du auf **Tabellen** und klickst dann auf **Adressen**, erscheinen im Fenster daneben die Felder aus *Adressen*. Hier klickst Du auf `[Name]`, dann in der Symbolleiste darüber auf `[Einfügen]`. Die Felddefinition erscheint im darüberstehenden Fenster.
- Nun klickst Du auf den *Operator* **„&“**, der die *Textfelder* verbindet oder *verkettet*.

➞ In der *Hilfe* unter dem Zeichen erfährst Du, was der Operator ”&” sonst noch verbinden kann!

←

- Zwischen Namen und Ort setzen wir als Trennzeichen einen Doppelpunkt und tragen ihn von Hand ein. Willst Du wissen, was ein Syntaxfehler ist, lässt Du die Anführungszeichen wie im Bild weg.
- Wie im Bild fügst Du einen weiteren Operator ein, …
- verfährst ebenso und ergänzt das Feld `[Ort]Ö`
- und klickst auf **OK**, um den *Ausdruck* in die Zeile „Feld“ einzufügen.
- Nun führst Du die Abfrage mit einem Klick auf die Schaltfläche aus. Die zu erwartende Fehlermeldung erscheint. Nach dem Schliessen des Fensters steht der Cursor an der Stelle, wo der Hase im Pfeffer liegt – was für den Hasen nicht gerade angenehm ist!

- Also setzt Du die Anführungszeichen ein. Das kannst Du auch unmittelbar im Feld machen, ohne den *Ausdrucks-Editor* aufzurufen – Du gehst mit der Maus auf die bezeichnete Stelle und gibst die Zeichen ein.

➞ Merke Dir: Wenn ein Syntax-Fehler vorliegt, hast Du meistens eine Kleinigkeit vergessen oder ein bisschen zu viel getan – etwa ein *Leerzeichen* gesetzt. Siehe z.B. `[Adressen]![Ort]`: hier darf sich zwischen den Klammern kein Leerzeichen befinden.

In der Anfangszeit der Programmiersprachen waren Fehlermeldungen des Computers eine Seltenheit. Wenn ein Programm nicht lief, musste der Programmierer den kompletten Programm-Code durchlesen und den Fehler suchen. Das war oft mühseliger, als ein Programm zu schreiben. Heute steht der Cursor da, wo Du einen Fehler gemacht hast! ←

- Wenn Du in die Nachbarspalte gehst, stellst Du fest, dass Access automatisch die Bezeichnung „Ausdruck1“ hinzugefügt hat. Diese Bezeichnung gibt in der *Datenblattansicht* die Überschrift für das Feld ab. Statt dessen solltest Du einen einleuchtenden Namen angeben, etwa „Kombination“ o.ä.
- Führe die Abfrage noch einmal aus.

Setzt Du selber Deine Eingaben ein, kannst Du die Tabellen-Bezeichnung weglassen, weil Du als Basis für die Abfrage die Tabelle „Adressen“ angegeben hast. Der Ausdruck sieht dann so aus: **`Kombination: [Name] & ´´:´´ & [Ort]`**. *Kriterien:* Für dieses Feld „Ausdruck1“ bzw. „Kombination“ kannst Du wie bei anderen Feldern beliebig *Kriterien* angeben, also z.B. nur nach bestimmten Namen suchen lassen, eine *Sortierung* angeben oder *Berechnungen* anstellen, soweit das sinnvoll ist. Du kannst es wie ein normales Feld behandeln.

Nehmen wir an, dass Du Deine DM-Beträge in den Euro umrechnen willst. Also musst Du eine Berechnung für das Feld „Zahlungsbetrag“ der Tabelle „Beiträge“ durchführen und die Änderungen in die Tabelle zurückschreiben. Der Einfachheit halber lege ich einen Kurs von DM 2.- für 1 Euro zu Grunde.

- Du erstellst eine *Auswahlabfrage*, die ihre Werte aus der Tabelle „Beiträge“ bezieht, und ziehst die Felder „Zahlungsbetrag“ und „ProjektNr“ in den *Entwurfsbereich*.
- Dann erzeugst Du ein neues Feld mit folgendem *Ausdruck*:

`Ausdr1:[Beiträge]![Zahlungsbetrag]/`

und benennst den „Ausdruck1“ in **`„Euro“`** um (s.o.).

- Führe die Abfrage aus. Im *Datenblatt* der Abfrage erscheint in der Spalte „Euro“ der halbierte Betrag der bislang eingegebenen Beträge.
- Speichere die Abfrage als **`[Euro-Auswahl]`**.

Zahlungsbetrag	Projekt-Nr	Euro
34,00	123	17
58,00	123	29
20,00	456	10
30,00	456	15
15,00	123	7,5
25,00	123	12,5
6,00	456	3

Du hast eine Berechnung durchgeführt. Diese Berechnung wird aber erst richtig sinnvoll, wenn Du damit die Daten aus Deiner Tabelle „Beiträge“ auf die Euro-Werte aktualisieren kannst. Dazu musst Du Deine *Auswahlabfrage* in eine *Aktualisierungsabfrage* umändern.

Du kopierst den *Ausdruck* von Zeile *Feld* der Zelle **`„Euro“[Beiträge]![Zahlungsbetrag]/2`** in die Zwischenablage.

Dann entfernst Du die Felder *Euro* und *Projekt-Nr* aus dem *Entwurfsbereich* – und zwar so: Du ziehst den Mauscursor auf den schmalen Balken über der Zeile „Feld“, so dass er sich in einen senkrechten Pfeil verwandelt. Klickst Du diesen Pfeil an, wird die Zelle des Feldes markiert. Du drückst **Entf**, und die Spalte ist gelöscht.

Jetzt klickst Du auf den Pfeil im Button , dann auf *Aktualisierungsabfrage*.

In den Zellen der Felder entsteht eine neue Zeile *Aktualisieren*. Aufgabe dieser Zeile ist es, das Feld in der zugehörigen Tabelle um den Wert zu ändern, also zu aktualisieren, den Du hier eingibst.

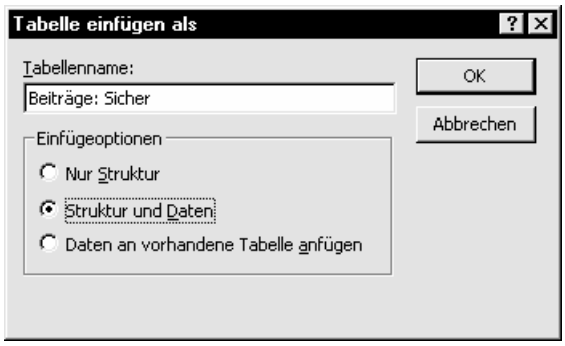
Wir haben in der Abfrage „Euro-Auswahl“ erfolgreich den Zahlungsbeitrag halbiert, was einer Kursumrechnung von 2 : 1 entsprechen würde. Darum nehmen wir den gleichen Ausdruck vom Feld „Euro“, der noch in der Zwischenablage ist, und fügen ihn in die Zeile „Aktualisieren“ ein.

„Euro“ und der „:“ müssen aus dem Ausdruck gelöscht werden, da sie nicht zur Aktualisierung gehören.

Du speicherst die Abfrage als **`[Euro]`** ab. Damit hast Du ein neues Symbol in der Liste Deiner Abfragen.

Schau Dir die Werte für das Feld *Zahlungsbetrag* in der Tabelle *Beiträge* noch einmal an, um nach dem Ausführen der Aktualisierungsabfrage eine Vergleichsmöglichkeit zu haben. (*Datenblatt* von *Beiträge*).

Um die alten Werte zu bewahren, kopierst Du die Tabelle *Beiträge* im *Datenblatt-Fenster* in die Zwischenablage und fügste sie wieder ein als **`Beiträge: Sicher]`**



Du klickst auf die Schaltfläche in der *Entwurfsansicht* oder wählst **Öffnen** im *Datenbank-Fenster*, um die Abfrage auszuführen.

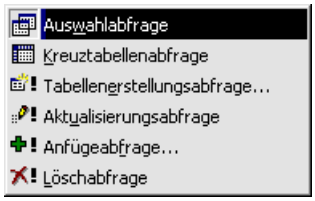
Öffnest Du die Tabelle „Beiträge“erneut in der *Datenblattansicht*, stellst Du fest, dass alle Werte in der Spalte „Zahlungsbetrag“ halbiert wurden.

Der Zweck dieser Abfrage besteht also darin, alle Werte in einer Tabelle auf einen Streich zu aktualisieren, damit Du das nicht von Hand machen musst.

Du kannst die Auswahl der Datensätze in der Zeile *Kriterien* auch einschränken, falls die Abfrage nur bestimmte Datensätze bearbeiten soll: Du gibst z.B. `> 10` als *Kriterium* für das Feld „Zahlungsbetrag“ ein. Siehst Du Dir anschliessend die Werte der Tabelle erneut an, hat die Abfrage nur die Werte von „Zahlungsbetrag“ geändert, die grösser waren als 10.

Andere Abfragearten

Für die meisten anderen Abfragearten gilt eine ähnliche Vorgangsweise: Du erstellst eine *Auswahlabfrage* und wandelst sie in den gewünschten Abfragetyp um. Es gibt folgende Möglichkeiten:



- Die *Kreuztabellenabfrage* stellt die Ausgabe von Daten in einer übersichtlichen, tabellarischen Form dar. Diesen Typ benutzt Du, wenn Du mit grossen Datenmengen arbeitest und Vergleiche unter den Daten anstellen oder Tendenzen aus ihnen ablesen willst – was in den einfachen *Datenblättern* oft unübersichtlich ist. Die *Kreuztabellenabfrage* kannst Du auch gesondert mit einem *Assistenten* erstellen.
- Die *Tabellenerstellungsabfrage* erzeugt mit den Daten der Abfrage eine neue Tabelle. Diese Abfrage benutzt Du, wenn Du alle oder nur bestimmte Daten aus einer Tabelle in eine neue Tabelle auslagern willst, um sie dort zu speichern oder weiterzubearbeiten. Sie wird meist bei der Automatisation mit *Makros* oder *Modulen* verwendet.
- Die *Anfügeabfrage* wird zur Erweiterung einer Tabelle durch neue Daten benutzt. Dabei kann sich die Ziel-Tabelle in einer lokalen oder aber externen Datenbank befinden.
- Die *Löschabfrage* löscht die in der Auswahlabfrage bezeichneten Datensätze aus der zugrundeliegenden Tabelle. Sie ist praktisch für die Datenpflege von Tabellen: hierdurch kannst Du Datensätze entfernen, die nicht mehr benötigt werden.

Zusätzlich gibt es folgende Abfragetypen:

- Die *Inkonsistenzsuche* sucht in *verknüpften* Tabellen nach Datensätzen, die nur in einer Tabelle vorhanden sind, nicht aber in den anderen.
- Die *Duplikatssuche* benutzt Du, um nach doppelt vorkommenden Datensätzen zu suchen. Gefundene Doppelsätzer kannst Du löschen oder weiterbearbeiten, z.B. wenn es sich um die Mitglieder handelt, die an einem bestimmten Ort wohnen.

→ Alle Abfragetypen sind ausführlich in der *Hilfe* beschrieben. ←

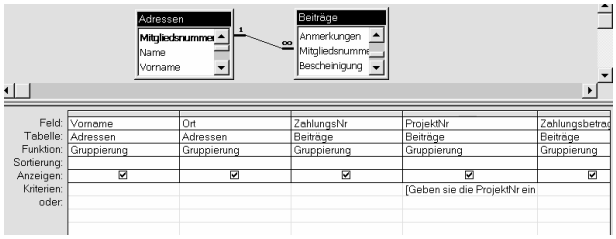
Parameterabfragen

Den Typ der *Parameterabfrage* wollen wir gesondert behandeln – in Verbindung mit aussenstehenden Benutzern ist er sehr nützlich. Es geht hier darum, dass die Abfrage beim Öffnen einen Parameter abfragt, den ein Benutzer eingeben muss, um die Abfrage auszuführen. Auf diese Art und Weise lässt sich das *Kriterium* einer Abfrage variabel je nach dem aktuellen Bedarf des Benutzers einstellen. So kann er z.B. alle Beiträge ausgeben lassen, die für ein bestimmtes Projekt geleistet wurden.

Wir wollen eine solche Abfrage herstellen. Du kopierst die Abfrage **`[Beiträge eingeben]`** in die Zwischenablage und fügst sie als **`[Beiträge eingeben-test]`** wieder ein. Dann öffnest Du die neue Abfrage in der *Entwurfsansicht*.

Willst Du alle Zahlungen zu einem bestimmten Projekt abfragen, passt Du die Abfrage folgendermassen an:

- Du gehst ins Feld **`[ProjektNr]`** und gibst in der Zeile *Kriterien* einen Text ein, der beim Öffnen der Abfrage eingeblendet wird – beispielsweise:
  `„Geben Sie die Projektnummer ein“`
  Achte darauf, dass nach dem Satzende *kein Satzzeichen erscheint* – andernfalls akzeptiert Access die Eingabe nicht. Das einzige zugelassene Satzzeichen ist ein *Doppelpunkt*, der in der Klammer immer als letztes stehen muss.



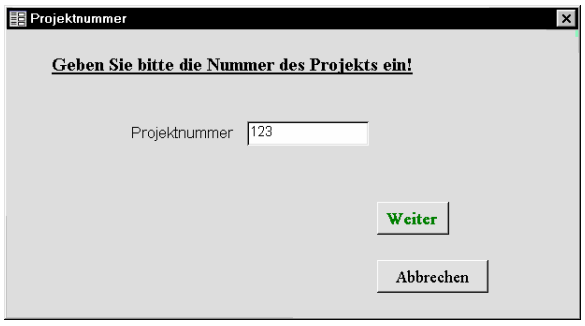
- Du speicherst die Abfrage. Dann aktivierst Du sie, gibst eine für Deine Datensätze vergebene Projektnummer ein und klickst auf **OK**. Das Ergebnis ist, dass die Abfrage alle Daten mit der entsprechenden ProjektNr auflistet.

Du kannst natürlich beliebig viele Felder in den *Entwurfsbereich* ziehen und ebenso Felder Deiner Wahl abfragen. Des weiteren kannst Du den **`[Ort]`** oder **`[Namen]`** abfragen sowie für mehrere Felder hintereinander Parameter vergeben, also z.B. erst den **`[Ort]`**, dann das **`[Projekt]`** und dann den **`[Namen]`**.

Mit dem Operator **`Zwischen [Bitte PLZ-Bereich eingeben] Und [Bereich-Ende]`** kannst Du zusätzlich einen Bereich abfragen, den der Benutzer eingeben soll.

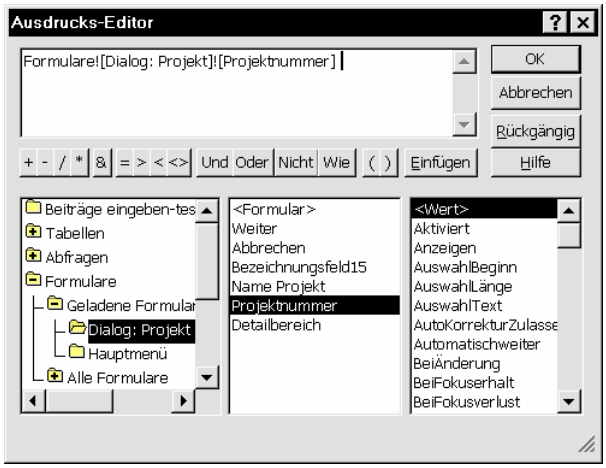
Diesen Vorgang kannst Du noch professioneller gestalten, indem Du den Dialog mit dem Benutzer über ein Formular führst.

Dazu erstellst Du ein *ungebundenes* Formular, also eines, das an keine Datenquelle gebunden ist. Es könnte z.B. so aussehen:



Das Feld **`Projektnummer`** ist ein *ungebundenes Textfeld*, das Du mit der *Toolbox* erstellst. Der Text darüber entstammt einem *Bezeichnungsfeld*. Die Schaltflächen baust Du ein; was ihre Funktionalität angeht, musst Du noch bis zu den *Makros* warten. Du speicherst das Formular unter dem Namen **`[Dialog: Projekt]`**.Dann öffnest Du es aus dem *Datenbankfenster,* gibst im dafür vorgesehenen Feld eine ProjektNr ein und speicherst den Datensatz über **Datensätze | Datensatz speichern**.

Nun gehst Du zur Abfrage und gibst im Feld **`[ProjektNr]`** bei *Kriterien* eine Bedingung ein:



Du speicherst die Abfrage und führst sie aus. Das Formular bleibt im Hintergrund geöffnet, damit die Abfrage den Feldwert übernehmen kann.

Schreibst Du später ein Makro, das das Formular als Dialogfenster öffnet und das Ergebnis der Abfrage in einem Bericht ausgibt, sieht das ganz ordentlich aus. Du kannst den Bericht natürlich auch jetzt schon erstellen.

Zuletzt sei noch etwas über den Gebrauch von *Abfragen* in Verbindung mit *Formularen* gesagt. Du kannst Formulare statisch auf Tabellen basieren lassen. Soll Dein Formular aber den aktuellen Stand einer Tabelle darstellen, ist es manchmal eher angebracht, eine Abfrage als Datenquelle für ein Formular zu nehmen. Die Arbeitsweise zum Erstellen des Formulars ist dabei die gleiche.

Das wäre ein erster Einblick in die Arbeitsweise mit Abfragen. Es lohnt sich, Abfragen nach Deiner Wahl auf der Basis der hier erlangten Kenntnisse zu erstellen. Führe Berechnungen durch und probiere die anderen Aggregatfunktionen aus. Benutze die unterschiedlichen Abfragetypen und sieh Dir an, was mit den Daten passiert. Hast Du Angst, Deine Datenbank zu zerschiessen, speicherst Du sie vorher unter einem anderen Namen ab und führst Deine Experimente in der Test-Datenbank durch.

Es gilt die alte Weisheit: *Versuch macht klug* – auch wenn dieser Spruch all die Schwierigkeiten beinhaltet, die dabei auftauchen und so frustrierend sein können. Lernen wirst Du leider nur, wenn Du diese Schwierigkeiten so selbstständig wie möglich bewältigt hast. Einen Fehler macht man meist nur ein Mal. Was letztlich den Meister ausmacht, ist die Tatsache, dass man sich dabei nicht entmutigen lässt.

Formulare

Ich gehe davon aus, dass Du grundsätzlich weisst, wie man Formulare erstellt – mit oder ohne Assistenten. Hier wollen wir uns mit einigen Sonderfällen beschäftigen, die in komfortablen Anwendungen von Nutzen sein können. Dazu gehören hauptsächlich der Entwurf von Haupt- und Unterformularen sowie von Pop Up-Formularen, die vielfach als Dialog-Fenster für Eingaben seitens der Benutzer dienen.

Haupt- und Unterformulare

Formulare können auf drei Ebenen ineinander verschachtelt sein. Du kannst also ein Formular, das ein Unterformular besitzt, in ein anderes einbinden. Natürlich kannst Du mehrere Unterformulare in einem Formular unterbringen sowie Formulare auf mehrere Seiten verteilen. Wir wollen uns aber nur mit einer Ebene beschäftigen – weitere Verschachtelungen bauen strukturell darauf auf.

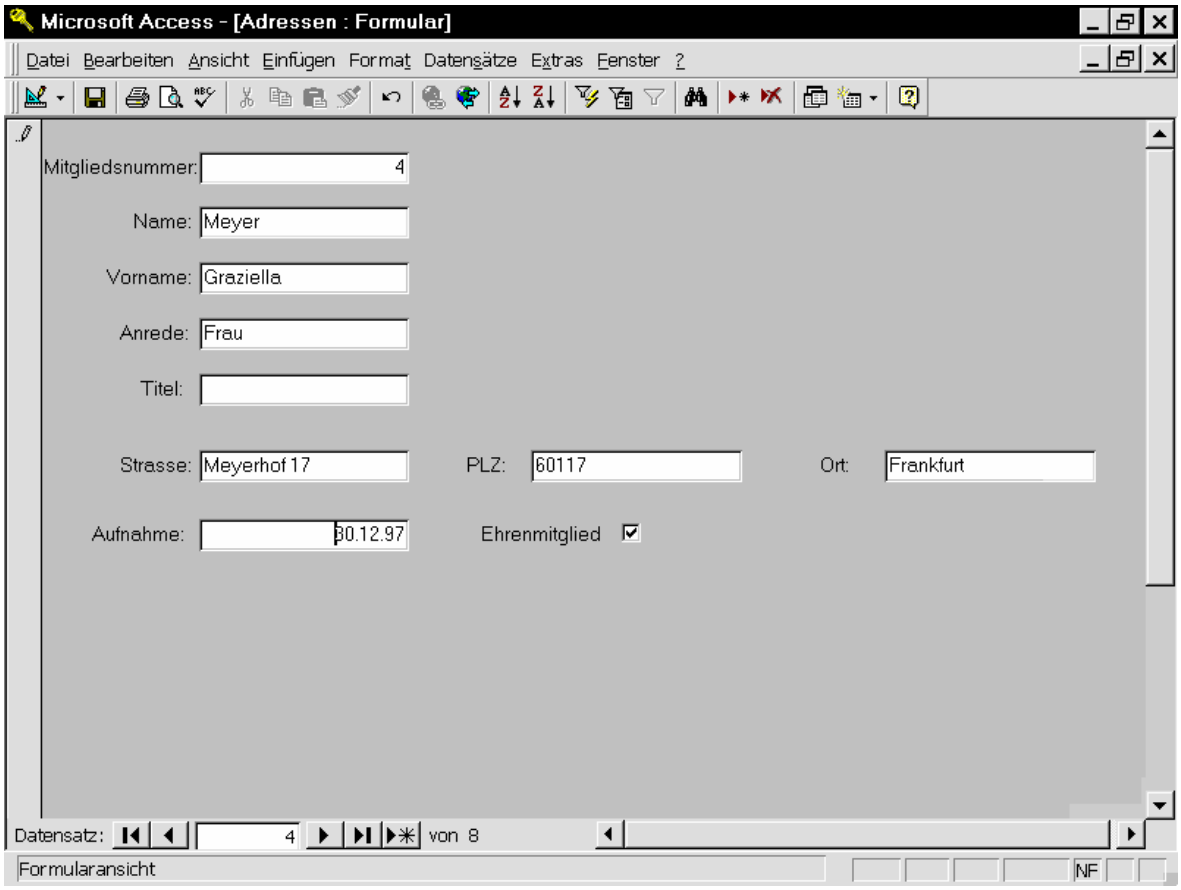
Du entwirfst ein Formular, das an die Daten der Tabelle „Adressen“ gebunden ist. Es sollte wie im Bild aussehen. In der *Entwurfsansicht* des Formulars klickst Du auf Feldliste und ziehst die *Felder* in das Formular.

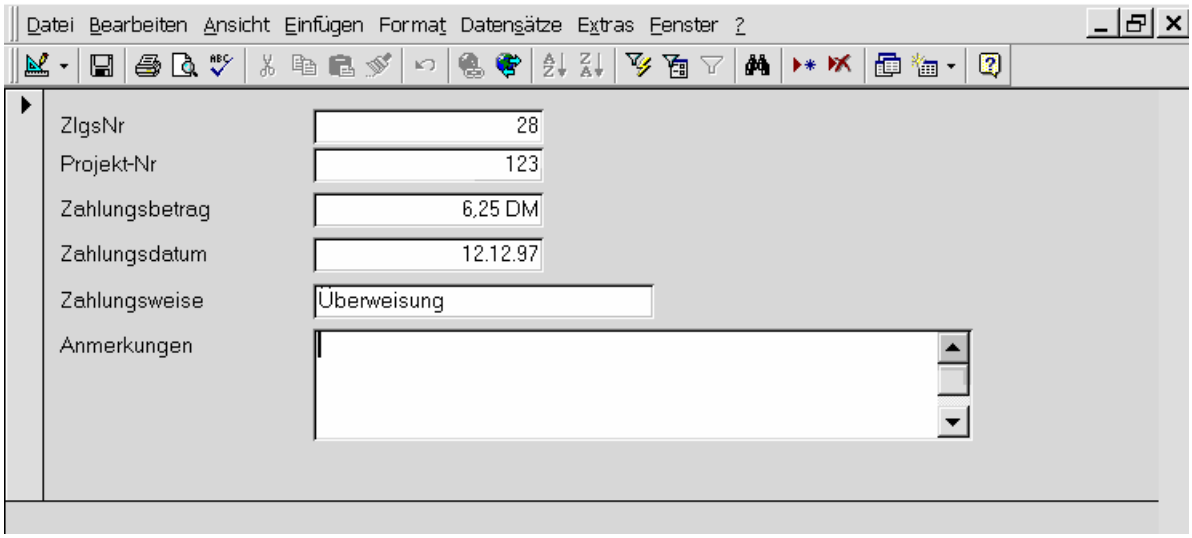
Baust Du das Formular mit einem Assistenten auf, änderst Du die Verteilung der *Steuerelemente* entsprechend der Abbildung.

Du speicherst das Formular unter dem Namen **`[Adressen]`**. Die *Eigenschaften* des Formulars sollten auf folgende Werte eingestellt sein:

- Unter *Registerkarte* **Format**:
  - *Standardansicht*: **`Einzelnes Formular`**
  - *Zugelassene Ansichten*: **`Beide`**
- Unter **Daten**:
  - *Datenherkunft*: **`Adressen`**
- Unter **Andere**:
  - *Zyklus*: **`Aktueller Datensatz`**

Hast Du Fragen zu den einzelnen Eigenschaften, ziehst Du die *Direkthilfe* auf die Zeile und liest Dir die Erklärungen durch, die meist durchaus verständlich sind.
Für die restlichen *Eigenschaften* kannst Du die Grundeinstellungen von Access beibehalten.





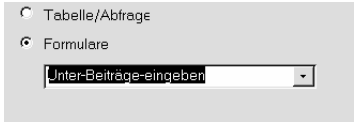
Ein zweites Formular sollte an die Tabelle „Beiträge“ gebunden sein.
Du speicherst das Formular unter dem Namen **`[Unter-Beiträge-eingeben]`** und stellst es auf folgende *Eigenschaften* ein:

- Unter der *Registerkarte* **Format**:
  - *Beschriftung*: **`Unter-Beiträge-eingeben`**
  - *Standardansicht*: Datenblatt
- Unter **Daten**:
  - *Datenherkunft*: **`Beiträge`**
  - *Sortiert nach*: **`Beiträge.Zahlungsdatum`**
- Unter **Andere**:
  - *Zyklus*: **`Alle Datensätze`**

Für die anderen Eigenschaften lässt Du die Grundeinstellungen bestehen. Daten brauchst Du hier nicht einzugeben, da Du das vermutlich schon bei den *Tabellen* gemacht hast. Solltest Du es dort versäumt haben, kannst Du es später im neuen Formular nachholen.

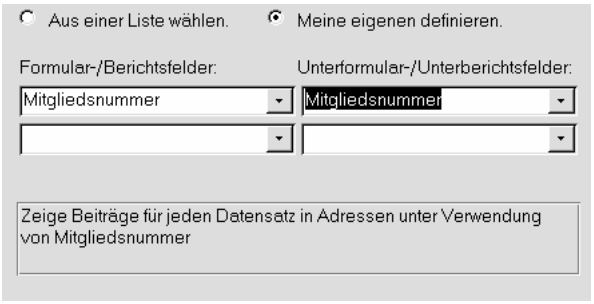
Öffne jetzt das Formular *Adressen* in der *Entwurfsansicht* und füge unterhalb der Adressdaten mit dem *Assistenten* aus der *Toolbox* ein *Unter-formular* ein. Ziehe es beim Einfügen nicht ganz an die Unterkante des Hauptformulars, da wir den Platz später benötigen.

Im 1. Fenster des Assistenten wählst Du die Einstellungen im Bild:



Im nächsten Fenster musst Du dem Assistenten mitteilen, über welche *Felder* die beiden Formulare miteinander verbunden sind.

Wir übernehmen den Vorschlag von Access, also die *Mitgliedsnummer*;

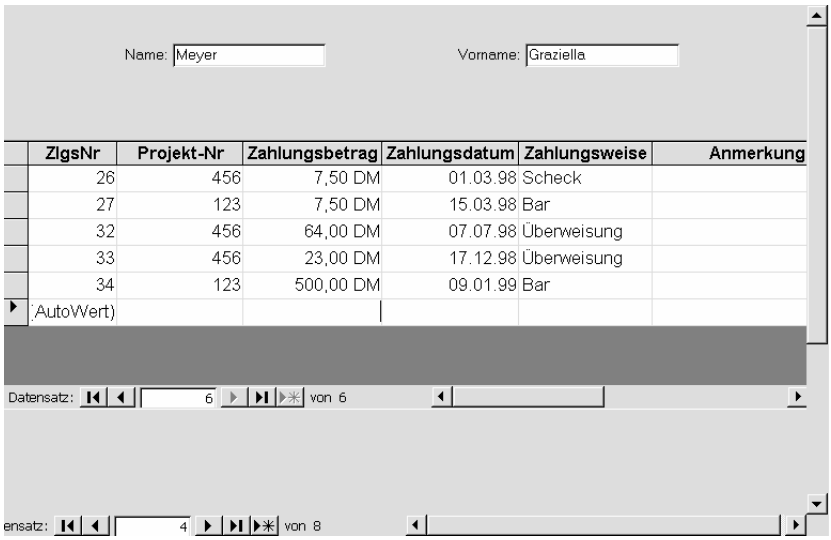


das Programm erkennt somit, dass die Tabellen „Adressen“ und „Beiträge“, die die Datenquelle für die Formulare sind, über diese beiden Felder im Verhältnis 1:n verbunden sind, und legt die gleiche Verbindung für die Formulare zu Grunde. Das finde ich ziemlich intelligent!

Im letzten Fenster des Assistenten gibst Du den Namen für das Formular an – das wäre hier **`[Unter-Beiträge-eingeben]`**. Dann speicherst Du das Hauptformular unter **`[Beiträge eingeben]`** und gehst in die *Formularansicht*.

Bei mir ist für das Unterformular zu wenig Raum. Ausserdem brauchen wir in diesem Formular nicht alle Adress-Daten des Mitglieds, weil es hier vornehmlich um die Beiträge geht – die Adress-Daten dienen nur zur Orientierung. Also löschst Du alle *Steuerelemente* ausser **`[Name]`** und **`[Vorname]`** und schaffst so genug Platz für das Unterformular, damit 5 – 6 Beitrags-Datensätze angezeigt werden können.

So kann es aussehen:
Hier wird die Eingabe von Datensätzen komfortabel: Du suchst einen Namen, und sofort erscheinen darunter die Beiträge, die ein Mitglied bezahlt hat. Später bei den *Makros* werden wir uns die Sache noch einfacher machen – alle Belange der Mitglieder werden aus einer einzigen Maske gesteuert. In diesem Fall sucht man nicht mehr nach sinngebenden Formularen, sondern klickt auf den geforderten Arbeitsbereich.



Gestaltung individueller Formulare

Formulare werden in ihrer Funktion von der *Steuelementen* bestimmt, die Du jeweils in sie einfügst. Eines haben wir schon besprochen das *Unterformular*. Klickst Du die gezeigte Schaltfläche, öffnet sich die Toolbox. Hier eine kurze Beschreibung aller *Steuerelemente*:

Bezeichnungsfeld	Dieses Feld hat rein informativen Charakter. Ein Beispiel wäre das Bezeichnungsfeld für Datenfelder in Formularen. Daten können hier nicht eingegeben werden.
Textfeld	Eingabefeld vom Format *Text*. Benutzer können Daten eingeben. Das Feld *Name* ist z.B. ein *Textfeld*.
Optionsgruppe	stellt eine Gruppe von Werten dar, die ein Benutzer auswählen kann. In einer solchen Gruppe können *Kontrollkästchen*, *Optionsfelder* und *Umschaltflächen* zusammengefasst werden – von denen jeweils nur eine Möglichkeit ausgewählt werden kann.
Umschaltfläche	erlaubt die Entscheidung zwischen verschiedenen *Ja/Nein*-Werten.
Optionsfeld	Ähnlich wie bei *Umschaltflächen* kann hier zwischen verschiedenen *Ja/Nein*-Werten gewählt werden, die zu *Optionsgruppen* zusammengefasst werden.
Kontrollkästchen	Wie *Optionsfelder* zur Entscheidung zwischen *Ja/Nein*-Werten genutzt. Innerhalb einer Gruppe (*Optionsgruppe*) können jedoch gleichzeitig mehrere Kontrollkästchen ausgewählt sein.
Listenfeld	Ein Benutzer kann aus einer Liste von Werten, die wie beim *Kombinationsfeld* unter dem Feld angezeigt wird, einen Wert auswählen. Die Eingabe der Werte entstammen einer festen Werteliste, wie Du sie bei der Anrede gebrauchen würdest; es werden selten oder nie neue Anredeformen hinzukommen. Die Werte können aber auch einer Tabelle oder Abfrage entstammen und dementsprechend ständig aktualisiert werden.
Kombinationsfeld	Ähnlich wie das *Listenfeld,* wobei es zusätzlich die Eigenschaften eines *Textfeldes* hat. Weiter unten werden wir noch darauf zu sprechen kommen.
Befehlsschaltfläche	Durch das Drücken (Klicken) dieser „Schaltfläche“ werden einer oder mehrere Befehle ausgeführt. Bei den *Makros* entwerfen wir solche Schaltflächen.
Bild	fügt eine Grafik aus einer Datei in das Formular ein.
Gebundene und ungebundene Objektfelder	*Gebundene* Objektfelder beziehen sich auf in Tabellen gespeicherte Objekte, *ungebundene* sind starr auf bestimmte Formulare beschränkte Objekte. Weiter unten im Kapitel *Ausdrücke benutzen* werden wir damit Berechnungen in Formularen durchführen.
Seitenwechsel	Dieses Steuerelement führt einen Seitenwechsel herbei, falls das Formular aus mehreren Seiten besteht.
Register-Steuerelement	erstellt ein mehrseitiges Formular mit Register-Steuerelementen, wie sie z.B. im *Datenbank-Fenster* erscheinen. Die einzelnen Registerkarten verweisen auf verschiedene Seiten des Formulars.
Unterformular /-bericht	fügt ein Unterformular in das bestehende Formular ein. Diese Option haben wir vorhin benutzt, um das Unterformular einzubinden.
Linie, Rechteck	fügt die grafischen Elemente *Linie* und *Rechteck* ein, um Formulare schöner oder übersichtlicher zu gestalten.
Weitere Steuerelemente	enthält neben den schon beschriebenen weitere nützliche Elemente, die Du in einem Formular verwenden kannst.

→ Im Rahmen dieses Heftes ist es unmöglich, auf alle Steuerelemente näher einzugehen. Doch es gibt zweierlei Dinge, die Du ausprobieren solltest. Das wäre zum einen die Direkthilfe, die Du auf die einzelnen Elemente der *Toolbox* ziehen kannst. Die andere Möglichkeit besteht darin, die verschiedenen Steuerelemente der Reihe nach zu installieren und festzustellen, was passiert. Speichere das Formular am besten vorher ab. Wenn Dir irgend etwas nicht gefällt, kannst Du immer noch zu Deinem alten Formular zurückkehren. ←

Wir wollen uns auf ein *Kombinationsfeld* konzentrieren, das als Suchfunktion in Formularen dienen kann. Des weiteren sehen wir uns *Ausdrücke* an, um Informationen in einem Feld aufzubereiten, sowie *Optionsgruppen*.

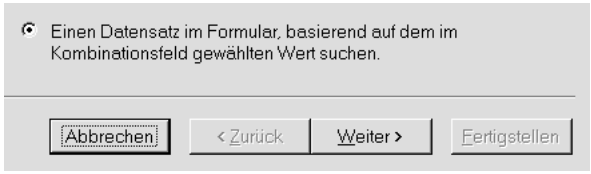
Kombinationsfeld

Kombinationsfelder erlauben dem Benutzer, sich aus einer Liste Werte auszuwählen, damit der ausgesuchte Wert im Datensatz gespeichert wird. Es ist aber auch möglich, ein Kombinationsfeld so einzurichten, dass Du es als Suchfeld benutzen kannst.

Du öffnest das Formular „Adressen“ in der *Entwurfsansicht* und aktivierst die *Toolbox*.

Dann schaltest Du den *Steuerelement-Assistenten* ein, klickst auf **Kombinationsfeld** und positionierst es rechts oben im Formular.

Im 1. Fenster des Assistenten klickst Du auf die Option, die angibt, dass das Kombinationsfeld einen Datensatz suchen soll.



Im 2. Dialogfenster übernimmst Du **Mitgliedsnummer**, **Name** und **Vorname** ins Fenster *Ausgewählte Felder*.

Im 3. Fenster blendest Du wie empfohlen die *Schlüsselspalte* aus, also die Mitgliedsnummer. Die Spaltenbreite stellst Du durch Ziehen an den Spaltenbegrenzern nach Deinen Bedürfnissen ein.

Im letzten Fenster gibst Du als Bezeichnung für das Kombinationsfeld `[Name suchen]` ein und klickst auf **Fertigstellen**.

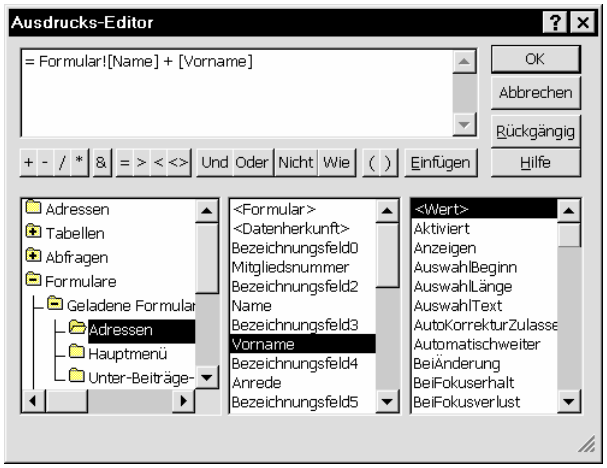
Du speicherst das Formular und wechselst in die *Formularansicht*. Wählst Du nun im Kombinationsfeld einen Namen, stellt Access das Formular auf den entsprechenden Datensatz.

Ausdrücke benutzen

Es kann vorkommen, dass Du dem Benutzer in einem Feld einen Wert ausgeben willst, den Du aus verschiedenen Feldern berechnet hast.

- Du öffnest das Formular, in dem Du dieses Feld unterbringen möchtest, in der *Entwurfsansicht*. In unserem Fall schlage ich „Adressen“ vor.
- Aktiviere die *Toolbox* und füge das Feld ein, das den berechneten Ausdruck enthalten soll. Dies wird in den meisten Fällen ein *Textfeld* sein. Das Feld ist ungebunden, d.h. nicht mit einem Datenfeld verknüpft, wie etwa die Felder, die Du aus der *Feldliste* in das Formular gezogen hast.
- Du kannst das Feld unmittelbar anklicken und den Ausdruck eingeben. Jeder Ausdruck fängt mit einem Gleichheitszeichen an, gefolgt vom Ausdruck selber. Du kannst Dich aber auch des *Ausdrucks-Editors* bedienen, der Dir die Arbeit erleichtert.
- Dazu rufst Du mit der Schaltfläche die *Eigenschaften* des Feldes auf, wählst unter der *Registerkarte* „Daten“ den *Steuerelementinhalt* und klickst auf ...

- Nehmen wir an, Du willst Namen und Vornamen des Mitglieds in ein und demselben Feld anzeigen. Also gibst Du folgenden Ausdruck in das Editor-Fenster ein: `=Formular![Name]+[Vorname]`, klickst die Werte in den Fenstern an und fügst sie mit **Einfügen** in das Fenster ein.
- Bewegst Du Dich von links nach rechts durch die Fenster, stellst Du fest, dass eine Auswahl im linken Fenster eine Liste von Möglichkeiten im nebenstehenden erzeugt.
- Dein Ausdruck sollte letztendlich wie im Bild aussehen.
- Du speicherst Dein Formular und wechselst in die *Formularansicht*. Dein ungebundenes Textfeld enthält die Werte der beiden Felder. Diese ändern sich jeweils von einem Datensatz zum nächsten.



Natürlich kannst Du in dem Textfeld auch kompliziertere Berechnungen anstellen. Die einfachsten *Operatoren* sind unmittelbar unter dem Fenster angeordnet. Willst Du komplizierte Berechnungen ausführen, musst Du auf die in Access eingebauten *Funktionen* zugreifen.

In unserem Formular „Adressen“ gibt es das Feld „Aufnahme“. Wir wollen erreichen, dass bei jeder Neuaufnahme eines Datensatzes das Tagesdatum vorgegeben wird.

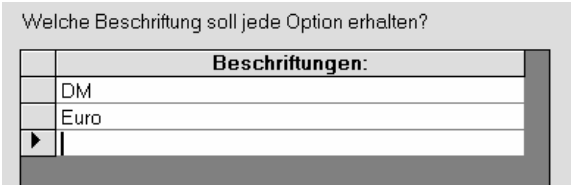
- In der *Entwurfsansicht* klickst Du auf das Feld und gehst bei den Eigenschaften zur Zeile *Standardwert*.
- Dann öffnest Du den Editor, wählst unter **Eingebaute Funktionen | Datum/Uhrzeit | Datum** und fügst das Ergebnis mit der Schaltfläche **Einfügen** ein. Vor den Ausdruck musst Du noch ein Gleichheitszeichen **=** setzen. Drückst Du auf **OK**, wird der Ausdruck in die Zeile *Standardwert* geschrieben.
- Du schliesst das Eigenschaften-Fenster, speicherst das Formular, gehst in die *Formularansicht* und gibst einen neuen Datensatz ein. Du wirst feststellen, dass das Feld „Aufnahme“ mit dem Tagesdatum vorbelegt ist.

→ Gehe auch die anderen *eingebauten Funktionen* durch, um Dir ein Bild davon zu machen, was dort auf Dich zukommt. ←

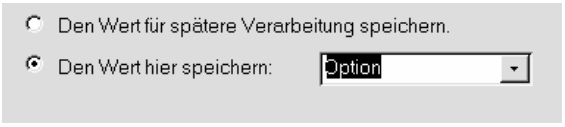
Optionsgruppen

Wir wollen ein Pop Up-Formular entwerfen, das aus einer *Optionsgruppe* besteht. Wir werden es später anwenden, um bei der Eingabe des Mitgliedsbeitrag zu überprüfen, ob es sich um einen Betrag in DM oder Euro handelt.

- Du öffnest die Tabelle *Beiträge* in der *Entwurfsansicht* und fügst ein neues Feld ein, das Du **`[Option]`** nennst, mit dem *Felddatentyp* **`Zahl`** und der *Feldgrösse* **`Byte`**. Es werden maximal Zahlen zwischen 1 und 10 vorkommen. Du speicherst die Tabelle und schliesst sie.
- Nun entwirfst Du ein neues Formular, dessen Daten aus der Tabelle „Beiträge“ stammen.
- Über die *Toolbox* fügst Du mit Hilfe des *Steuerelement-Assistenten* eine *Optionsgruppe* ein.
- Als Beschriftung für die Optionen im 1. Fenster des Assistenten gibst Du die beiden Namen [DM] und [Euro] ein.



- Im 2. Fenster legst Du die Standardauswahl auf „DM“ fest – bist Du der Meinung, dass der Euro sich schon durchgesetzt hat, nimmst Du diesen Wert. Trifft später ein Benutzer keine Auswahl in diesem Fenster, speichert Access den Standardwert.
- Die Wertvorgabe für die Option im 3. Fenster kannst Du bestehen lassen. Um zu entscheiden, ob ein eingegebener Mitgliedsbeitrag in Euro oder DM verrechnet werden soll, werden wir diesen Optionswert später abfragen.



- Im 4. Fenster veranlasst Du den Assistenten, den Wert im Feld *Option* zu speichern. Wähle das Feld aus der Scroll-Liste aus. Access speichert den oben festgelegten Optionswert, also die „1“ oder die „2“. Wenn wir später einen Geldbetrag ausgeben oder zu einer Berechnung herbeiziehen, müssen wir den dazugehörigen Optionswert abfragen, um die Währung zu bestimmen .
- Zur Einstellung des Aussehens der Optionsgruppe lässt Du die voreingestellten Werte *Optionsfelder* und *graviert* bestehen.
- Als Beschriftung gibst Du die Bezeichnung **`[DM-Euro]`** an und klickst auf **Fertigstellen**.
- Du speicherst das Formular unter dem Namen **`[Dialog: DM-Euro]`** und wechselst in die *Formularansicht*.

PopUp-Formulare

Um ein Formular zu einem PopUp-Formular bzw. Dialogfenster aufzubereiten, musst Du folgende Schritte vollziehen:

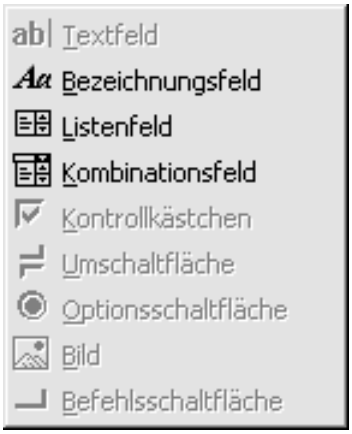
- Öffne das *Eigenschaften*-Fenster des Formulars **Dialog DM – Euro**.
- Unter der Registerkarte **Format** setzt Du
  - *Beschriftung* auf **`[Eingabe Währung]`**
  - *Bildlaufleisten* auf **`[Nein]`**
  - *Datensatzmarkierer* auf **`[Nein]`**
  - *Navigationsschaltflächen* auf **`[Nein]`** (Der aktuelle Datensatz wird später vom Makro eingestellt.)
  - *Trennlinie* auf **`[Nein]`**
  - *Automatisch zentrieren* auf **`[Ja]`**
  - *Rahmenart* auf **`[Dialog]`**
  - *MinMaxSchaltfläche* auf **`[Keine]`**
- Unter der Registerkarte **Andere** setzt Du
  - *PopUp* auf **`[Ja]`**
  - *Gebunden* auf **`[Ja]`**

→ Ziehst Du die *Direkthilfe* auf eine Zeile, erhältst Du die entsprechende Erklärung. *Gebunden* heisst, dass das Formular erst beendet sein muss, bevor jegliche Verarbeitung fortgeführt werden kann. ←

- Bei den anderen Eigenschaften kannst Du die Voreinstellungen übernehmen.
- Speichere das Formular und öffne es in der *Formularsicht*. Es sollte wie im Bild aussehen. Bei mir ist die Optionsgruppe *vertieft* dargestellt.

Ändern von Steuerelementen

Einige Steuerelemente kannst Du nachträglich verändern, auch wenn Du sie schon in Deinem Formular installiert hast.



Du markierst das gewünschte Element, indem Du es anklickst. Im Menü **Format | Ändern zu** siehst Du die Liste der Möglichkeiten, in welche Du das markierte Element abwandeln kannst.

So sieht die Liste aus, wenn das ausgewählte Feld ein *Textfeld* ist.

Hast Du einen anderen Typ von Steuerelement markiert, ändert sich diese Liste so, dass die jeweils entsprechenden Elemente hervorgehoben werden.

Ich hoffe, ich konnte Dir einen kleinen, sinnvollen Ausschnitt aus der breiten Palette von unterschiedlichen Formularen und Formulareinstellungen präsentieren, so dass Du nun frei weiterarbeiten kannst.

Makros

Wir kommen zu dem Teil von Access, der eine von Dir geschriebene Anwendung gewissermassen zu einem *Programm* werden lässt, das sich an eine bestimmte Benutzergruppe richtet. In unserem Fall, der Datenbank *Verein*, wäre das wahrscheinlich das Büro eines Vereins, das die Belange der Mitglieder und die Organisation des Vereins verwaltet.

Ein Benutzer sollte nicht vor dem *Datenbank-Fenster* sitzen und mühselig nach *Formulare* und *Abfragen* suchen – mit progammiertechnischen Aufgaben hat er eigentlich nichts zu tun.

Für ihn ist es sinnvoller, wenn er aus der Oberfläche von *Windows* auf ein Symbol klickt – übrigens kannst Du das Symbol auch durch Klicken auf die rechte Maustaste *umbenennen*! – und in ein *Hauptmenü* kommt, aus dem sich die jeweils gewünschte Arbeitsaufgabe für den Computer aktivieren lässt.

Die Vorleistung, die Du als Programmierer dafür erbringen musst, nennt sich *Automation.* Sie wird im ersten Schritt durch die **Makros** bewerkstelligt.

Du kennst Makros vielleicht aus anderen *Office*-Anwendungen wie *Word* oder *Excel*. Im Unterschied zu diesen Programmen kannst Du in Access keine Makros aufnehmen – dafür ist der Umgang mit der Erstellung von Makros viel ausgefeilter.

Du hast vielleicht schon einmal eine Schaltfläche in einem *Formular* mit dem Assistenten entworfen. Dies ist möglich, weil die grundsätzlichen Automationen in Access, wie etwa *Formular öffnen*, *Datensatz speichern* oder *Telefonnummern wählen,* schon vorgefertigt sind. Diese Automationen nennt man *Ereignisprozeduren*.

Mit diesem Angebot an Makros kommst Du jedoch nicht weit, und bald beginnst Du damit, eigene zu schreiben. „Schreiben“ ist vielleicht nicht ganz richtig –Wörter oder Befehle gibst Du auch hier nicht ein, aber mit den Makros dringst Du eine Ebene tiefer in Access ein. Das eigentliche Schreiben wird erst bei den *Modulen* erforderlich.

Willst Du ein Makro erstellen, klickst Du im *Datenbank-Fenster* auf die Registerkarte *Makros* und dann auf **Neu**.

Daraufhin erscheinen folgenden neue Symbole in der *Symbolleiste*:

- Bei Makros unterscheidet man zwischen den Oberbegriffen, die im *Datenbank-Fenster* erscheinen, und den darunter geordneten Namen der einzelnen Makros. Werden mehrere Makros unter einem Oberbegriff gefasst, solltest Du die Spalte *Makronamen* einblenden.
- *Bedingungen* ist ein weitere Spalte, die Du einblenden kannst. Hier definierst Du die Bedingungen, unter denen ein Makro ausgeführt wird. Das entspricht den *Kriterien* bei Abfragen.
- *Einzelschritt*. Führt ein Makro seine Aufgaben nicht korrekt durch, kannst Du es schrittweise ausführen lassen, um zu sehen, welche Auswirkungen die einzelnen Makro-Schritte haben – dabei entspricht ein Makro-Schritt einer Zeile im Makro).

Bevor wir anfangen, ein Wort zum *Datenentwurf*. Bei grösseren Anwendungen kann die Ansammlung von Makros leicht unübersichtlich werden. Makros sind eher fehleranfällig als andere Teile von Access. Taucht irgendwo ein Fehler auf, musst Du ihn schnell finden können. Darum solltest Du versuchen, eine gewisse Struktur bei der Anordnung der Makros einzuhalten und möglichst einleuchtende Namen zuzuteilen, wie etwa `[Formular Adressen öffnen]` o.ä.

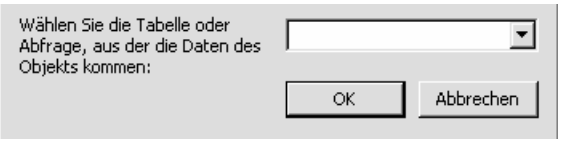
Zur Struktur ist zu sagen, dass Access, zumindest augenblicklich, keine ausgesprochen Objektorientierte Programmieranwendung ist. Wäre das der Fall, solltest Du Makros auch nach Objekten orientieren, auch wenn das nur eine Krücke ist. Ein Objekt der Datenbank *Verein* könnte z.B. *Beiträge* sein – hier würden alle Arbeiten zusammengefasst, die mit den Beiträgen zu tun haben. Eine objektorientierte Arbeitsweise wäre besser, weil natürlicher – welcher normale Mensch denkt in der Struktur von Dateien und Verzeichnissen? Wollen wir eine Party organisieren, interessiert uns die Grösse der Etiketten auf den Weinflaschen weniger als die Frage, ob genug Wein vorhanden ist.

Microsoft tut sich in dieser neuen Technologie etwas schwer. Vielleicht liegt das daran, dass Bill Gates so viel Geld mit dem Inbegriff der Dateiverzeichnisstruktur MS-DOS verdient hat, dass er sich von dieser Denkweise nicht mehr lösen kann.

Ich fasse die Makros nach ihrer Funktionsweise zusammen, etwa *Formulare öffnen*, *Abfragen öffnen* oder *Daten bewegen*. Das ist leichter, weil die Namen die *Aktionen* wiedergeben, die das Makro ausführt. Später kann man die Makros in aller Ruhe in eine objektorientierte Struktur umsetzen.

Wir fangen mit dem *Hauptmenü* an. Du kannst es grossenteils mit den vorgefertigten Routinen oder Ereignisprozeduren erledigen, die der Assistent von Access anbietet. Wir wollen das Menü aber von Hand programmieren.
Bei Makros fängt man grundsätzlich „unten“ an: zuerst das Makro, das die Arbeit ausführt, und dann die *Befehlsschaltfläche*, die das Makro startet.

Hast Du noch kein Hauptmenü, erzeugst Du ein neues Formular in der *Entwurfsansicht.* Dieses Formular ist an keine Daten aus einer Datenquelle gebunden. Die Zeile:



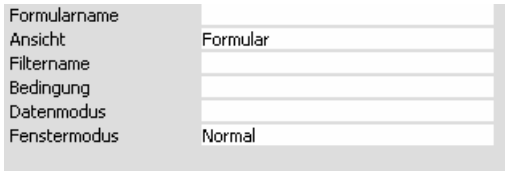
bleibt leer. Du speicherst das Formular unter dem Namen **`Hauptmenü`**.

In unserem Beispiel wollen wir das Formular *Beiträge eingeben* über eine Schaltfläche öffnen.

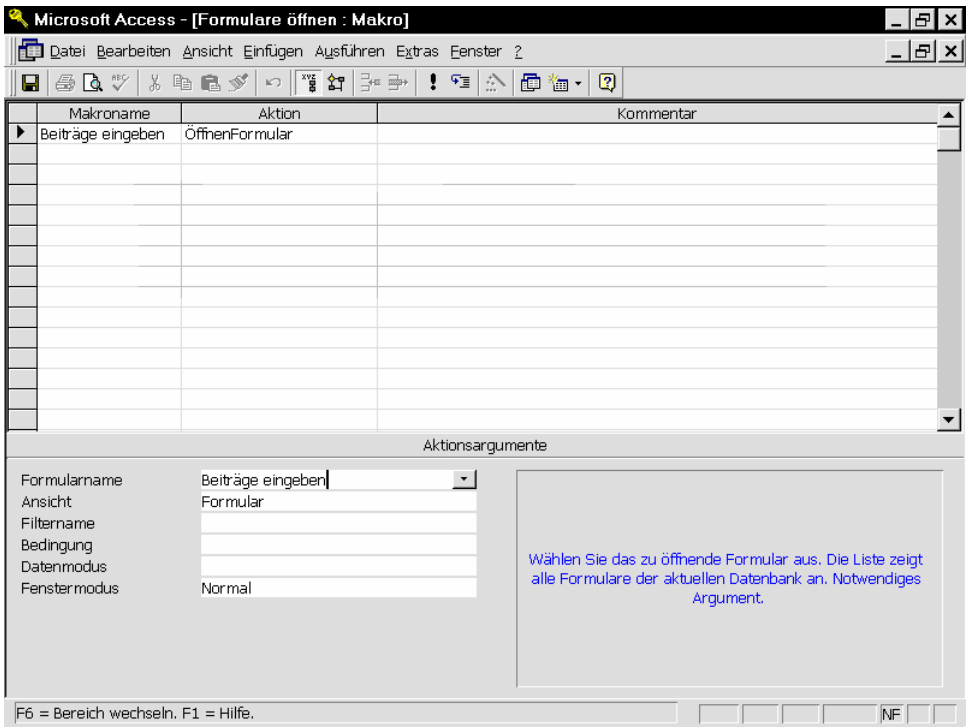
- Du gehst in das *Datenbank-Fenster*, klickst auf die *Registerkarte Makros* und dann auf **Neu**.
- Mit der Schaltfläche  fügst Du die Spalte *Makronamen* hinzu und schreibst hier **`Beiträge eingeben`**.
- Du speicherst das Makro unter dem Namen **`Formulare öffnen`**. 

- Dann gehst Du in die Spalte Aktion und klickst  , wodurch Dir in einem Scroll-Fenster alle Aktionen angezeigt werden, die ein Makro ausführen kann.

- Hier wählst Du **Öffnen Formular**, worauf sich in der unteren Bildschirmhälfte das folgende Fenster öffnet:



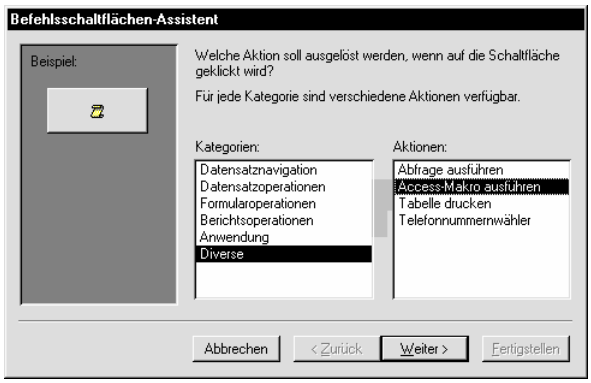
- Gehe dort in die Zeile **Formularname**. Achte auf das Fenster daneben, wo Access in blauer Schrift eine kurze Erläuterung dazu bietet, was Du in dieser Zeile eingeben sollst.
- Klickst Du auf  , öffnet Access ein Fenster, das alle Formularnamen enthält, die Du bislang erstellt hast. Hier wählst Du **Beiträge eingeben**.
- In der Zeile **Ansicht** gibt das Programm wahrscheinlich die Auswahl „Formular“ vor, die Du bestehen lässt, weil Du das Formular zur Eingabe von Daten öffnen möchtest – und dafür eignet sich die *Formularansicht* am besten.



- Die Zeilen *Filtername*, *Bedingung* und *Datenmodus* kannst Du fürs erste freilassen. Lies Dir jedoch die Kommentare im Nachbarfenster durch, um Dir ein Bild davon zu machen, wie Du die Daten und das Aussehen des Formulars an dieser Stelle beeinflussen kannst.
- In der Zeile *Fenstermodus* lässt Du die Voreinstellung „normal“ bestehen.
- Schliesslich solltest Du das Makro noch einmal speichern.

Damit hast Du Dein erstes Makro geschrieben. Was noch fehlt, ist der Punkt, wo es aufgerufen wird. Die einfachste Art ist, auf  zu klicken, womit das Makro mit allen seinen Aktionen ausgeführt wird. Praktischerweise sollte es aber als Ereignis hinter einer Schaltfläche liegen, um von da aus gestartet zu werden. Also gehst Du in das Formular „Hauptmenü“, das vermutlich noch offen ist – andernfalls öffnest Du es in der *Entwurfsansicht*.

- Du erstellst mit dem Assistenten eine neue Schaltfläche und wählst im oberen Fenster die markierten Teile.



- Im nächsten Fenster wählst Du unter den von Dir erstellten Makros dasjenige aus, das die Schaltfläche starten soll. Vorläufig sollte hier nur das Makro *Formulare öffnen.Beiträge eingeben* stehen. Der Oberbegriff des Makros *Formulare öffnen* steht zwar auch da, hat aber im Augenblick keine Bedeutung.

- Nach einem Klick auf **Weiter** kommst Du in das nächste Fenster. Hier klickst Du auf **Text** und gibst in der Zeile daneben die Bezeichnung ein, die Deine Schaltfläche bekommen soll – in diesem Fall **`Beiträge eingeben`**. Du kannst die Bezeichnung jederzeit wieder ändern.
- Du klickst auf **Weiter** und gibst der Schaltfläche einen Namen. Das ist wichtig, damit Du diese Schaltfläche mit einem Makro ansteuern kannst. Du nennst sie **`Beiträge eingeben`**.
- Endlich klickst Du auf **Fertigstellen –** womit Du ins Formular „Hauptmenü“ zurückkehrst, wo Du Deine Schaltfläche findest. Du kannst sie auf gewohnte Weise an einen frei gewählten Ort verschieben.

Apropos *Verschieben* – das hört sich wirklich gut an. Ich habe noch auf der Programmiersprache „Cobol“ gelernt. Da war das Positionieren noch ein echtes Abenteuer. Nicht, dass ich mich danach zurücksehne! Der Bildschirm war in Koordinaten aufgeteilt. Um etwas zu positionieren, musste man die Koordinaten eingeben und beschreiben, was dort eingesetzt werden sollte. Da konnte ein Hauptmenü leicht einen ganzen Tag beanspruchen. Darum liebe ich es, ein Objekt mit der Maus zu verschieben!

Das Aussehen der Schaltfläche kannst Du nach Belieben abwandeln. Wenn die Schaltfläche aktiv ist, Du also auf ihr stehst, klickst Du auf *Eigenschaften* und nimmst unter der Registerkarte *Format* Deine Einstellungen vor.

Gehst Du auf die Karte *Ereignis*, findest Du Dein Makro in der Zeile *Beim Klicken* aufgeführt. Wir werden später weitere Situationen kennenlernen, durch die ein Makro ausgelöst werden kann.

Soweit so gut. Du weisst jetzt, wie man ein Makro anfertigt und wie man es für sich arbeiten lässt. Eigentlich ist das die Lebensaufgabe von Computern – sie sollen uns Arbeit abnehmen. Sie sind die modernen Sklaven, ohne die Menschenpeinigung, dafür leider etwas zickig.

Entspanne Dich, der Einstieg ist geschafft!

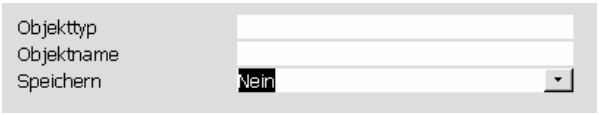
Sehen wir uns jetzt noch einmal das Makro an, mit dem das Formular geöffnet wurde. Wir wollen einige Verbesserungen vornehmen. Du öffnest *Formulare öffnen.Beitäge eingeben* in der *Entwurfsansicht*.

Willst Du sicherstellen, dass das Formular immer in der *Vollbildansicht* dargestellt wird, wenn Du es mit dem Makro öffnest, setzt Du eine weitere Zeile in das Makro ein.

Du gehst in der zweiten Zeile in die Spalte *Aktion* und wählst mit  die notwendige Aktion aus, die *Maximieren* heisst. Der Makroname bleibt gleich. Das Makro führt nacheinander alle Zeilen mit den *Aktionen* aus, die unter diesem Namen stehen. Ein neuer *Makroname* ist gleichbedeutend mit einem neuen Makro.)
Danach solltest Du das Makro speichern.

Irgendwann will der Benutzer natürlich wieder in das Hauptmenü zurück. Also musst Du in das Formular *Beiträge eingeben* eine Schaltfläche einbauen, die das bewerkstelligt.

- Der erste Schritt besteht wiederum darin, ein Makro zu schreiben. Also erzeugst Du ein neues Makro.
- Dann blendest Du die Spalte *Makroname* ein und schreibst **`Formulare schliessen`**.
- Die benötigte *Aktion* lautet *Schliessen*.



- Die Aktionsargumente unten lässt Du wie im Bild leer. Das hat einen einfachen Grund: wir wollen auf dieses Makro öfters zugreifen und führen deshalb nicht näher aus, welches Formular geschlossen werden soll. Access hat die Eigenschaft, dass es nach dem Schliessen eines Formulars immer zu dem Formular zurückkehrt, von wo es aufgerufen wurde. Das Argument *Speichern* setzt Du auf **Nein**, weil sich das Speichern nur auf Änderungen an dem Formular selbst bezieht und nicht auf die Datensätze, die Du eingegeben hast.
- Schreibe in die Spalte *Kommentare* eine kurze Bemerkung, damit Du Dich später erinnerst, warum Du das Makro so geplant hattest. Diese kleine Gedächtnisstütze kann sehr sinnvoll sein. Der Kommentar könnte wie im nachfolgenden Bild aussehen.

Makroname	Aktion	Kommentar
Formulare schließen	Schließen	Für Formulare, die zum aufrufenden Formular zurückkehren

- Danach kannst Du, wie oben schon beschrieben, die Schaltfläche mit dem Assistenten in das Formular einbauen und sie mit dem gerade entworfenen Makro verbinden. Du nennst sie **`Zurück`** oder **`Hauptmenü`**. Die Schaltfläche sollte natürlich im Formular **`Beiträge eingeben`** liegen und nicht im Unterformular.

Weiter im Hauptmenü: Du hast jetzt den Weg zu den *Beiträgen* und wieder zurück ins *Hauptmenü*. Entsprechend gehst Du vor, um in das Formular „Adressen“ zu kommen. Folgende Schritte sind notwendig:

- Du schreibst ein Makro namens **`Formulare öffnen.Adressen eingeben`**, das das Formular *Adressen* öffnet und es zur *Vollbildansicht* maximiert. Vergiss nicht, es zu speichern!
- Dann setzt Du eine Schaltfläche namens **`Adressen eingeben`** in das Hauptmenü, die das Makro auslöst.
- Anschliessend baust Du eine Schaltfläche in das Formular „Adressen“ ein, mit der Du in das Hauptmenü zurückkommst. Du brauchst kein eigenes Makro zu schreiben, sondern benutzt das Makro *Formulare schliessen.Formulare schliessen*. Dieser Name hört sich etwas komisch an, ist aber durchaus sinnvoll, wie Du noch sehen wirst.

Fertig! Der zukünftige Benutzer kann alle Aufgaben von einer zentralen Stelle steuern.

Gehen wir jetzt eine weitere Stufe zurück! Der Benutzer hat mit dem *Datenbankfenster* nichts am Hut – es interessiert ihn nicht und verwirrt höchstens. Darum soll er, sobald er die Datenbank *Verein* geöffnet hat, unmittelbar ins Hauptmenü kommen. Das Makro für diese Automatisation hat einen feststehenden Begriff, den die Freunde von MS-DOS noch gut kennen: *AutoExec*.

- Du erzeugst ein neues Makro und speicherst es unter dem Namen **`AutoExec`** ab.
- Dann setzt Du dieselben Aktionen ein wie hier im Bild:

Aktion	Kommentar
Maximieren	Wird automatisch ausgeführt, wenn Du die Datenbank öffnest.
Echo	Fixiert den Bildschirm während der Ausführung des Makros.
Sanduhr	Zeigt eine Sanduhr während der Ausführung des Makros an.
ÖffnenFormular	Öffnet das Formular "Hauptmenü".
Maximieren	

Somit gibt es einen Weg *in* Deine Datenbank. Was fehlt, ist eine Methode, um sie wieder zu verlassen. Das erreichst Du mit dem folgenden Makro mit der dazugehörigen Schaltfläche in Deinem *Hauptmenü*:

Makroname	Aktion	Kommentar
Datenbank schließen	Verlassen	Verlassen von Datenbank, wie auch Access
Formulare schließen	Schließen	Für Formulare, die zum aufrufenden Formular zurückkehren

Wie Du siehst, habe ich das Makro in die erste Zeile eingefügt.

→ Über die *Hilfe* erhältst Du Erläuterungen zur Funktionsweise derEntwurfsmaske. Klickst Du mit dem *Direkthilfe*-Zeiger auf die Spalte *Aktion*, öffnet sich ein kleines Fenster. Klickst Du in diesem Fenster auf , kommst Du in ein neues Fenster, das Dir den Umgang mit *Aktionen* erklärt. Du kannst auch in der **Hilfe | Inhalt und Index** den Suchbegriff *Aktion* eingeben. Im 2.Fenster klickst Du auf *Aktion*, dann unten auf *Anzeigen* und im folgenden Fenster auf *Aktionen, nach Aufgaben gruppiert*. Das Fenster, das sich dann öffnet, gibt Auskunft über alle *Aktionen*, die Du in Access unmittelbar auswählen kannst. Du tust gut daran, Dir über **Optionen | Thema** den Text ausdrucken zu lassen, da Du diese Anweisungen noch öfters gebrauchen wirst! ←

Ich habe die Schaltfläche **`Arbeit beenden?`** genannt, damit der Endbenutzer sie nicht unvorsichtig anklickt und so die Datenbank versehentlich schliesst. Am besten formatierst Du die Schrift rot oder legst ein Bild auf die Schaltfläche. In meiner Wahlheimat Hamburg würde man wahrscheinlich „Hand ab!!“ oder ähnliches daraufschreiben. Ja, die Menschen hier sind sehr freundlich!

Wird diese Schaltfläche aktiviert, schliesst Access die Datenbank inklusive der gesamten Access-Oberfläche. Sollten zu diesem Zeitpunkt noch *Formulare, Tabellen* oder *Abfragen* offenstehen, in die Datensätze eingegeben wurden – und diese Fenster werden ja durch den Klick auf die Schaltfläche ebenfalls geschlossen –, ist es unwahrscheinlich, dass sie verloren gehen, weil Access *normalerweise* die Datenänderungen speichert. Allerdings würde ich es nicht darauf ankommen lassen, sondern die Datenbank so konzipieren, dass der Benutzer ein Formular nach dem anderen schliessen muss, bevor er das *Hauptmenü* verlassen kann.

→Programmierer haben eine Verantwortung gegenüber den Benutzern, für die sie ihr Programm schreiben! Du solltest Dir stets vergegenwärtigen, dass später vermutlich jemand vor dem Programm sitzt, der von den Hintergründen keine Ahnung hat und wohl kaum weiss, welche Tasten *nicht* gedrückt werden dürfen. ←

Mit Rücksicht darauf wollen wir einen ersten Schritt tun und dem Benutzer die aufwendige Suche nach der nächsten freien Mitgliedsnummer in unserer Datenbank *Verein* ersparen. Auf dem Wege dahin werden wir eine weitere Sicherung einbauen, die sicherstellt, dass eine unfreiwillige zufällige Änderung einer Mitgliedsnummer nicht möglich ist. Wir wollen die Mitgliedsnummer also so gestalten, dass sie tabu und für einen Benutzer nicht zugänglich ist!

Das erfordert die Erstellung eines neuen Makro. Um uns einem objektorientierten Konzept anzunähern, nennen wir es **`Mitglieder`** und fügen hier alles ein, was Informationen über die Mitglieder des Vereins betrifft.

- Als ersten Makronamen in dieses neue Makro gibst Du **`Neuer Datensatz`** ein.
- Die erste *Aktion*, die das Makro ausführen soll, heisst wieder *Echo.* Zunächst setzt Du bei den *Aktionsargumenten* das *Echo* auf **Ja**, damit Du die Verarbeitung des Makros auf dem Bildschirm verfolgen kannst. Später kannst Du es dann auf **Nein** setzen – den Endbenutzer interessiert die Verarbeitung vermutlich nicht. Die Aktionsargumente erreichst Du auch mit **F6**, nach oben erneut **F6**.
- Ist das Echo ausgeschaltet, empfiehlt es sich, dem Benutzer die *Sanduhr* anzuzeigen, damit er weiss, dass der Computer arbeitet. Willst Du Dir die Arbeit erleichtern, kannst Du diese zwei Zeilen aus dem *AutoExec*-Makro in die Zwischenablage kopieren und sie an der richtigen Stelle bei *Mitglieder* einfügen. Faulheit ist die Antriebskraft für Genialität!
- Im nächsten Schritt musst Du herausfinden, welches die letzte benutzte Mitgliedsnummer ist. Dafür benötigst Du eine *Abfrage,* die alle Mitgliedsnummern auflistet. Gehst Du in der Abfrage zum letzten Datensatz, addierst hier **`1`** und machst diese Zahl zur neuen Mitgliedsnummer, kann diese Nummer ins *Adress*-Formular eingetragen werden, ohne dass der Benutzer etwas dazu tun muss.
- Also entwirfst Du eine neue Abfrage mit der Tabelle *Adressen* als Datenquelle. Du ziehst die *Mitgliedsnummer* in den *Entwurfsbereich* und speicherst die Abfrage unter dem Namen **`Neue MitgliedsNr`**.
- Willst Du die Nummer um einen Zähler hochrechnen, benötigst Du ein berechnetes Feld *Mitgliedsnummer + 1*. Du gibst den Namen in die 2. Spalte in der Zeile *Feld* ein; ich habe das Feld „Nr+1“ genannt. Dahinter musst Du einen **`„:“`** setzen. Dann klickst Du auf *Aufbauen*. Du fügst entsprechend das Feld *Mitgliedsnummer* ein und addierst es um 1 Zähler. Wird die Abfrage ausgeführt, enthält das Feld „Nr+1“ die addierte Zahl. Du speicherst



die Abfrage und schliesst sie.

- Zurück zum Makro. In der nächsten Zeile musst Du die Abfrage öffnen:
  Öffnen Abfrage mit den Argumenten wie in : **`Neue MitgliedsNr;Datenblatt;Bearbeiten`**
  Diese Schreibweise – die Argumente werden durch Semikolon getrennt, hinter dem letzten Argument steht keines – werde ich in der Folge beibehalten. Die Argument-Bezeichnungen werde ich nicht nennen – sie folgen immer der Reihe nach, von oben nach unten. Bleibt ein Argument leer, steht ein Semikolon ohne Text dahinter, also **`;`**
- Die nächsten Schritte vollziehen sich so, wie Du es auch von Hand machen würdest. Im letzten Datensatz der Abfrage steht die Zahl, die Du brauchst, um die neue Mitgliedsnummer zu festzulegen. Also gehst Du zu diesem Steuerelement der Abfrage, dann zum letzten Datensatz, kopierst die Zahl in die Zwischenablage, wechselst zum Steuerelement *Mitgliedsnummer*, gehst dort zu einem neuen Datensatz, fügst die Zahl aus der Zwischenablage ein und speicherst den Datensatz. Wenn Du die Abfrage geschlossen hast, befindet sich in der Tabelle *Adressen* ein neuer Datensatz, der als einzige Information die neue Mitgliedsnummer enthält. Alles klar?

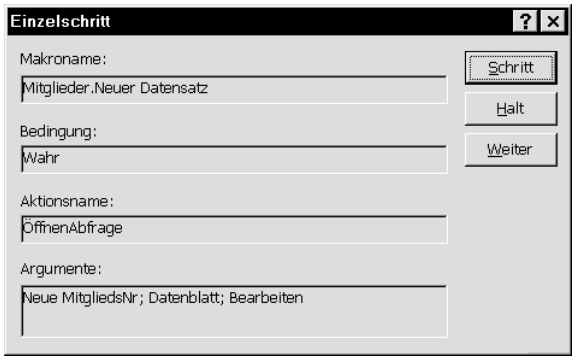
Hier die einzelnen Schritte mit ihren Argumenten:

- *GeheZuSteuerelement*: **`Nr+1`**
- *GeheZuDatensatz*: **`;;Letzter;`**
- *AusführenBefehl*: **`Kopieren`**
- *GeheZuSteuerelement*: **`Mitgliedsnummer`**
- *GeheZuDatensatz*: **`;;Neuer;`**
- *AusführenBefehl*: **`Einfügen`**
- *AusführenBefehl*: **`DatensatzSpeichern`**
  Access schreibt den Datensatz zurück in die der Abfrage zugrunde liegende *Tabelle*, in diesem Fall also die Tabelle „Adressen“.
- *Schliessen*: **`;;Nein`**
  Diese Aktion schliesst Objekte. Das zuletzt geöffnete Objekt war die Abfrage, mit der wir einen neuen Adress-Datensatz erstellt haben. Darum können wir die ersten beiden Argumente *Objekttyp* und *Objektname* leer lassen. Die Abfrage selbst braucht nicht gespeichert zu werden, weil wir an dem Entwurf durch das Makro nichts geändert haben, 3. Argument also **Nein**.

- Jetzt brauchst Du eigentlich nur noch das Formular „Adressen“ zu öffnen und zum letzten Datensatz zu gehen – und der Benutzer kann seine Daten eingeben.

Die weiteren Schritte folgen hier: Wir haben bereits ein Makro, das auf das Öffnen von Formularen spezialisiert ist, also lassen wir unser Makro dort weiterlaufen. Du springst sozusagen von einem zum nächsten Makro. Der letzte Schritt im Makro sieht so aus:

- *AusführenMakro*: das 1.Argument ist das Makro, das aufgerufen wird. Da wir das noch nicht haben, muss das Argument vorerst leer bleiben. Die beiden anderen kannst Du ebenfalls frei lassen.
- Es folgt die nächste Etappe: Du öffnest das Makro *Formulare öffnen* in der *Entwurfsansicht*, legst einen neuen Makronamen **`Adresse neu`** an und trägst folgende Aktionen in das Makro ein:
  - *ÖffnenFormular*: **`Adressen;Formular;;;Normal`**
  - *Maximieren*
  - *GeheZuSteuerelement*: **`Name`**
    Das Makro öffnet unmittelbar das Feld „Name“, damit der Benutzer die gewünschten Daten eingeben kann.
  - *GeheZuDatensatz*: **`;;Letzter;`**
    Die Argumente Objekttyp und -name brauchst Du nicht anzugeben, weil das aktive Objekt das geöffnete Formular ist.
- Das wär’s! Du musst nur noch einmal zum aufrufenden Makro *Mitglieder.Neuer Datensatz* zurückkehren und hier den Namen des Makro eingeben, das *AusführenMakro* aufrufen soll. Im Argument *Makroname* kannst Du Dir aus der Liste das Makro *Formulare öffnen.Adresse neu* auswählen. Die Argumente *Wiederholungen* und *Wiederholbedingungen* lässt Du leer.
- Nun gehst Du ins *Hauptmenü* und baust  eine Schaltfläche **Neue Adresse** ein, die das Makro startet. Sieh Dir sicherheitshalber die Abarbeitung des Makros im *Einzelschritt*-Modus an.



- In diesem Fenster werden für den jeweiligen Schritt des Makro alle aktuellen Kriterien angezeigt, wie etwa der Makroname, die Bedingung, unter der das Makro ausgeführt wird – dazu kommen wir noch – und der Aktionsname mit seinen Argumenten. Ist Dein Makro fehlerhaft, kannst Du über diesen Modus feststellen, wie es in allen Einzelheiten abläuft. Die Schaltflächen helfen Dir: **Schritt** geht zur nächsten Aktion, **Halt** stoppt das Makro, **Weiter** lässt es bis zum Ende durchlaufen.

Ein Wort noch zum Aufrufen von Makros: wie Du in den *Argumenten* des Befehls gesehen hast, kannst Du unter *Wiederholungen* angeben, wie oft das Makro wiederholt werden soll. Im nächsten Argument kannst Du Bedingungen angeben, unter denen eine Wiederholung des Makros ausgeführt wird.

→ Ziehe die *Direkthilfe* auf das Argument **`[Wiederholbedingungen]`** und lies Dir den Text durch. ←

Eine Anwendung in unserem Fall könnte z.B. sein, die neue Mitgliedsnummer nicht schlicht aufzuaddieren, sondern eine Routine zu schreiben, die die nächste *freie* Nummer sucht. Das kann sinnvoll sein, wenn in der Datenbank viele Datensätze gelöscht worden sind, was freie Nummern bedeutet. Du könntest über das Makro die Mitgliedsnummern so lange durchsuchen, bis es einen Unterschied zwischen zwei Nummern >1 findet, worauf es dann die dazwischenliegende Nummer für den neuen Datensatz verwenden.

Du kannst Makros – ähnlich wie auch Formulare, siehe Seite 28 – auch untereinander verschachteln. Mehr darüber findest Du in der *Hilfe*-Beschreibung. Verschachtelung ist in 3 Ebenen möglich. Du kannst in einem Makro also ein anderes aufrufen, von da aus ein weiteres und von dieser Ebene noch eins, bevor die Ausführung zu dem ursprünglichen Makro zurückkehrt. Aber Vorsicht: das Hin- und Herspringen birgt Fehlerquellen in sich, weil viele *Objekte,* also Tabellen, Abfragen etc, gleichzeitig geöffnet sind. Das kann zu dem führen, was Grossrechner-Programmierer der frühen *Basic*-Programme als Spaghetti-Programmierung bezeichnet haben: unübersichtlich ineinander verwurstelte Gebilde.

Sinnvoll angewendet kann die Makro-Verteilung aber auch zur Übersichtlichkeit Deines Programms beitragen, wie ich das schon beim *Datenentwurf* beschrieben habe. Das mag für Dich zum gegenwärtigen Zeitpunkt unwesentlich erscheinen, weil Deine Programme noch klein sind. Ein Objektorientiertes Datenmodell ist aber nicht etwa ein blosses Wort – sein Ziel sind „gefühlte“ Programme. In solchen intuitiven Datenmodellen bist Du mit Deinem Entwurf näher an der realen Welt Deines Benutzers – bzw. Deines Kunden, wenn Du professionell arbeitest.

Wir hatten beabsichtigt, den Zugriff auf die *Mitgliedsnummer* durch Benutzer zu verhindern. Den für einen Benutzer notwendigen Schritt, eine neue Nummer an ein neues Mitglied zu vergeben, haben wir bereits automatisiert. Diese Routine wird durch das obige Makro ohne Eingriff des Benutzers vom Hauptmenü aus abgewickelt.

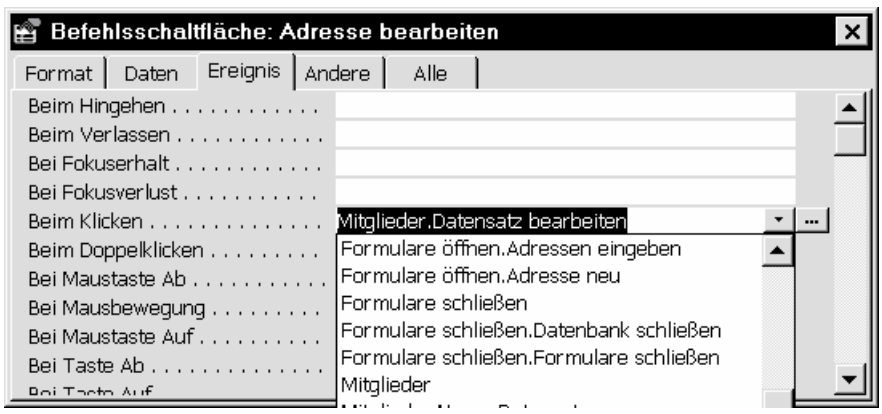
Ein Benutzer muss aber einen Datensatz auch ändern können. Also müssen wir auch dafür eine Routine erzeugen. Dafür benötigen wir im Hauptmenü eine Schaltfläche, durch die Du in das Formular „Adressen“ kommst, um die dortigen Datensätze zu bearbeiten.

Das erfordert folgende Schritte:

- Wir wandeln die Schaltfläche **Adressen eingeben** im Hauptmenü mit dem daran hängenden Makro ab, weil wir sie in dieser Form nicht mehr brauchen.
- Du öffnest das Makro *Formulare öffnen.Adressen eingeben* in der *Entwurfsansicht*.
- In der ersten Zeile *ÖffnenFormular* änderst Du das Argument *Datenmodus* in **`Bearbeiten`**.
- Die zweite Zeile *Maximieren* bleibt. Mit der Schaltfläche setzt Du eine neue Zeile ein.
- Der Benutzer soll zum *Feld* „Name“ gebracht werden, wo er mit Hilfe einer Suchfunktion ein bestimmtes Mitglied suchen kann. Die Suchfunktion entspricht einer vorgegebenen AccessRoutine, die mit der Schaltfläche aus der Menüleiste aufgerufen werden kann.
  Entsprechend dem Bild fügst Du die *Aktionen* mit folgenden Argumenten ein:

Adressen eingeben	ÖffnenFormular
	Maximieren
	GeheZuSteuerelement
	AusführenBefehl

  - *GeheZuSteuerelement*: Name
  - *AusführenBefehl*: Suchen
- Die Routine soll der Übersichtlichkeit halber aus dem Makro „Mitglieder“ aufgerufen werden. Also erstellst Du ein neues Makro namens **`Mitglieder.Datensatz bearbeiten`**. In seiner einzigen Zeile ruft es das Makro *Formulare öffnen.Adressen eingeben* auf. Wie das geht, weisst Du vermutlich noch – genau so wie beim Makro *Mitglieder.Neuer Datensatz*.
- Jetzt musst Du das Aussehen der Schaltfläche abändern sowie das Makro, das hinter der Schaltfläche liegt.
  Dazu öffnest Du das Formular *Hauptmenü* in der *Entwurfsansicht,* gehst zur Schaltfläche **Adressen eingeben** und klickst auf , um die *Eigenschaften* der Schaltfläche zu ändern. Unter der Registerkarte **Format/Beschriftung** gibst Du **`Adresse bearbeiten`** ein. Unter der Registerkarte **Ereignis/Beim Klicken** wählst Du das Makro *Mitglieder.Datensatz bearbeiten*, das durch einen Klick auf die Schaltfläche aufgerufen wird.



- Dann positionierst Du die Schaltfläche und passt ihre Grösse an, indem Du mit der Maus an den Zugpunkten ziehst.
- Du speicherst das Formular und die Makros und probiere Deine neue Schaltfläche aus. Ziemlich komfortabel, nicht wahr?

Da wir davon ausgehen, dass die Nummer nicht geändert werden darf, sperrst Du jetzt den Zugriff auf die Mitgliedsnummer seitens des Benutzers. Dafür öffnest Du das Formular *Adressen* in der *Entwurfsansicht*, klickst auf das Feld *Mitgliedsnummer* und öffnest die *Eigenschaften* des Feldes.

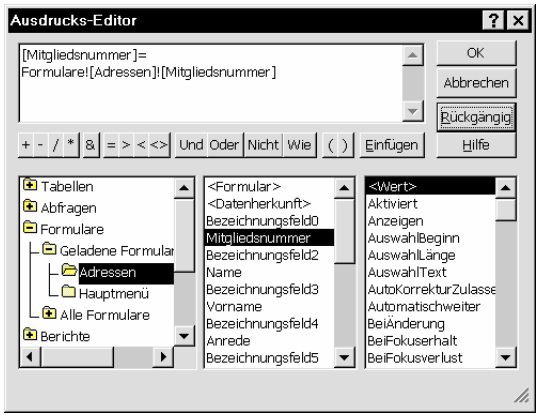
Unter der *Registerkarte* **Daten** stellst Du *Aktiviert* auf **`Nein`** und *Gesperrt* auf **`Ja`**. Ausserdem kannst Du unter **Andere** den Punkt **In Reihenfolge** auf **`Nein`** setzen, worauf das Feld nicht mehr durch die **TAB**-Taste erreicht wird.

Damit hast Du in Deinem Programm eine mögliche Fehlerquelle weniger. Willst Du Dein Programm anderen Benutzern zur Verfügung stellen, wird die Arbeit neben der Stammdaten-Verarbeitung vor allem in der Automatisation bestehen – und dazu gehört die Überprüfung und Vermeidung von Falscheingaben. Zunächst mag das nebensächlich erscheinen, letztendlich ist es aber der Hauptteil Deiner Arbeit!

Filter und Bedingungen

Wir wollen die Eingabe der Beiträge seitens der Mitglieder komfortabel gestalten. Einige Mitglieder brauchen keine Beiträge zu bezahlen. Sie sollen vor der Eingabe ausgefiltert werden.

- Du entwirfst ein neues Makro *Beiträge* und gibst ihm den Namen **`Beiträge eingeben`**.
- Dann aktivierst Du das vorhandene Makro *Formulare öffnen.Beiträge eingeben* mit folgendem Befehl:
  Ausführen: **`Formulare Öffnen.Beiträge eingeben;;`**
  Dies ist die einzige Zeile des Makros – also kannst Du es jetzt speichern.
- Anschliessend öffnest Du das Makro *Formulare Öffnen. Beiträge eingeben* in der *Entwurfsansicht*.
- Wir gehen davon aus, dass der Benutzer hauptsächlich im Formular „Adressen“ arbeitet, das ja die wesentlichen Informationen über die Mitglieder enthält. Von hier aus soll er auch in das Formular „Beiträge eingeben“ gelangen, ohne erst den Datensatz eines Mitglieds suchen zu müssen. Also musst Du in der ersten *Aktion* des Makros *ÖffnenFormular* den aktuellen Datensatz aus dem Formular „Adressen“ übergeben. Dafür eignet sich das Argument *Bedingung* der Aktion am besten. Wir wollen dies Argument wie folgt abändern:
- Du klickst auf die *Editor*-Schaltfläche 
  oder in der Symbolleiste auf  . Es geht darum, den aktuellen Wert der *Mitgliedsnummer* aus dem Formular „Adressen“ an das Formular „Beiträge eingeben“ zu übergeben. Die Syntax heisst **`[Mitgliedsnummer]`** (aus dem zu öffnenden Formular)
  `= Formulare![Adressen]![Mitgliedsnummer]`
  Nur die erste Mitgliedsnummer muss eingegeben werden, alles andere holst Du Dir durch Klicken. Du beginnst im linken Fenster. Steht **`Adressen`** offen, wählst Du **`Geladene Formulare`**, andernfalls **`Alle Formulare`**.
  Im mittleren Fenster gehst Du wie im nachfolgenden Bild zu **`Mitgliedsnummer`**. Im rechten Fenster wählst Du **`<Wert>`** aus. Mit **Einfügen** überträgst Du die Auswahl in das kleine Fenster, mit **OK** ins Argument.



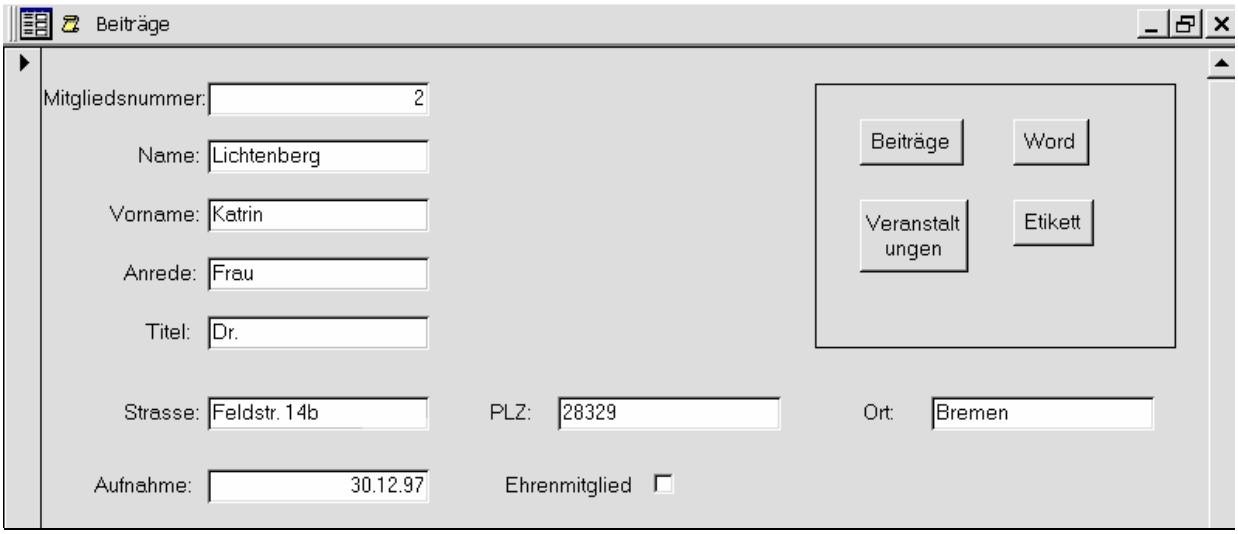
Die Syntax der Befehlssprache von Access erfordert, wie oben schon beschrieben, absolut eindeutige Schreibweisen. So ist es wichtig, dass Du das Steuerelement „Mitgliedsnummer“ in eckige Klammern **`[]`** setzt, weil Access es andernfalls als reinen *Text* interpretieren würde. Ebenso darf vor und hinter dem Gleichheitszeichen **`=`** keine Leerstelle sein. An dieser Stelle ist das Programm zwar grosszügig – es korrigiert eine Fehleingabe automatisch beim Speichern des Makro. Probiere es aus! Dennoch solltest Du Dir die richtigen Schreibweisen angewöhnen. Du kannst nicht davon ausgehen, dass der Kompilierer – also der Bestandteil des Programms, der Deine Eingaben in Programmiercode verwandelt – grundsätzlich einen Fehler erkennt. Bist Du Dir also nicht sicher, ob Du alles richtig geschrieben oder eingegeben hast, hilft es oft, die Eingabe sofort zu speichern, um zu sehen, was passiert.

- Die zweite Zeile *Maximieren* kann so bleiben. In der dritten fügst Du dann die Aktion *GeheZuSteuerelement* ein. Der Benutzer soll gleich an das Unterformular verwiesen werden, um dort die Beiträge einzugeben. Das Argument lautet:
  GeheZuSteuerelement: **`Unter-Beiträge-eingeben`**
- In diesem Zusammenhang kannst Du im Formular *Beiträge eingeben* die Felder *Mitgliedsnummer, Name* und *Vorname* für den Fokus sperren – was bedeutet, dass Benutzer diese Felder weder mit der Maus noch der Tastatur erreichen kann, da er ja an dieser Stelle nichts daran ändern soll. Die Vorgangssweise habe ich weiter oben beschrieben: die Eigenschaft *Aktiviert* muss auf **`[Nein]`**, die Eigenschaft *Gesperrt* auf **`[Ja]`** gesetzt werden.

- Was fehlt, ist der Startpunkt für das Makro – etwa eine Schaltfläche; im Formular erstellst Du dafür neben den Datenfeldern einen eigenen Breich, von dem aus der Benutzer die für dieses Formular notwendigen Aufgaben erreicht.

Das könnte etwa aussehen wie im Bild.

Hier haben wir auch eine eigene formularbezogene angebracht Menüleiste, was sehr komfortabel ist. Nachfolgend sehen wir uns an, wie wir so ein Ergebnis erreichen.



Entwurf eigener Menüleisten

Hier hast Du zwei Möglichkeiten – Du kannst:

- eine vorgegebene Menüleiste abändern
- eine neue entwerfen, die nach Bedarf in Formularen eingeblendet wird.

Ersteres hat den Nachteil, dass die Änderungen im Menü global auftreten, also z.B. in allen Formularen – ansonsten sind die Methoden gleichwertig. Wir wollen eine neue Leiste entwerfen, aus der wir in die Eingabemaske für die Beiträge gelangen.

- Also öffnest Du das Formular „Adressen“ in der *Entwurfsansicht*.
- Dann wählst Du unter Ansicht | Symbolleisten den Punkt Anpassen..., klickst auf Neu und gibst den Namen **`[Mitglieder]`** ein.
- Die nächsten Schritte schlägst Du in der *Hilfe* nach, wo sie der Reihe nach aufgeführt werden. Du hältst das Hilfefenster geöffnet und führst parallel in Access die relevanten Schritte aus.
- In der *Index*-Hilfe gibst Du **`Menüleiste`** ein und wählst **Menüs/Erstellen von Menüleisten-/Anzeigen**, im nächsten Fenster *Erstellen einer benutzerdefinierten Menüleiste in der aktuellen Datenbank*.

Erstellen einer benutzerdefinierten Menüleiste für die aktuelle Datenbank

1 Zeigen Sie im Menü **Ansicht** auf **Symbolleisten**, und klicken Sie dann auf **Anpassen**.

2 Klicken Sie in der Registerkarte **Symbolleisten** auf **Neu**.

3 Geben Sie im Feld **Symbolleistenname** den gewünschten Namen ein, und klicken Sie dann auf **OK**.

4 Klicken Sie in der Registerkarte **Symbolleisten** auf **Eigenschaften**.

5 Klicken Sie im Listenfeld **Typ** auf **Menüleiste**.

6 Stellen Sie bei Bedarf weitere Eigenschaften ein, und klicken Sie dann auf **Schließen**.

Die neue Menüleiste befindet sich jetzt hinter dem Dialogfeld **Anpassen**. Zum Anzeigen der Menüleiste schieben Sie das Dialogfeld **Anpassen** zur Seite. Anschließend fahren Sie mit Schritt 7 fort.

7 Zum Vervollständigen der Menüleiste fügen Sie benutzerdefinierte Menüs hinzu. Zum Anzeigen weiterer Informationen klicken Sie auf ».

Anmerkungen

- Microsoft Access unterstützt nach wie vor Menüleistenmakros für Anwendungen, die mit einer früheren Version von Microsoft Access erstellt worden sind. Zum Anzeigen weiterer Informationen klicken Sie auf ».
- Sie können die benutzerdefinierte Menüleiste in ein Formular oder in einen Bericht einbinden. Zum Anzeigen weiterer Informationen klicken Sie auf ». Andererseits können Sie die Leiste auch als globale Menüleiste definieren. Zum Anzeigen weiterer Informationen klicken Sie auf ».

- Du fährst bei Punkt 4 fort. Bist Du zu Punkt 7 gelangt, klickst Du auf worauf-hin Du ins nächste Hilfefenster kommst: *Hinzufügen eines benutzerdefinierten Menüs zu einer Menü- oder Symbolleiste.*.
- Bei Schritt 6 gibst Du **`Beiträge`** ein.
- Nach Schritt 7 klickst Du erneut auf die Schaltfläche . Du kommst in das Hilfefenster *Hinzufügen eines Befehls zu einem Menü*.
- Unter Punkt 4 wählst Du *Hinzufügen eines Befehls, der ein Makro ausführt*.

Für den Punkt 5 wählst Du dann in der Spalte *Befehle* das Makro *Beiträge.Beiträge bearbeiten*, wie hier im Bild:

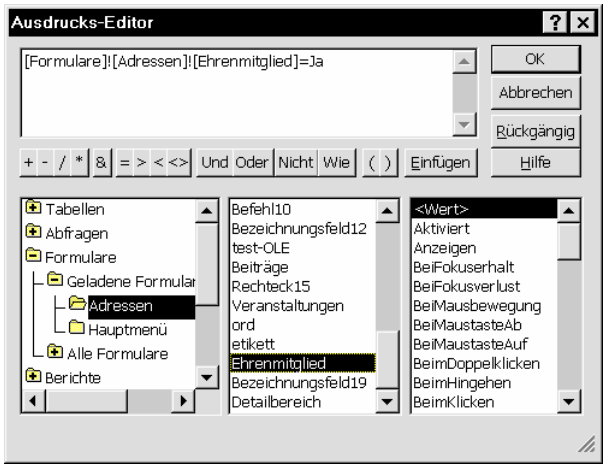
Du ziehst es auf den Menüpunkt *Beiträge* und lässt es dort los.

- Dann öffnest Du das *Eigenschaften*-Fenster des Formulars „Adressen“ und stellst unter der Registerkarte *Andere* die Eigenschaft *Menüleiste* auf *Mitglieder* ein.
- Anschliessend wechselst Du in die *Formular-ansicht*, ziehst Deine neue Menüleiste an die Stelle, wo sonst im Formular *Adressen* die Menüleiste steht, und lässt sie los. Nun sieht Dein Formular so aus wie im Bild auf Seite 44.
- Du speicherst das Formular, wählst einen Datensatz und klickst auf das Makro-Symbol in Deiner neuen Menüleiste. Wie Du siehst, hat das denselben Effekt wie ein Klick auf eine im Formular integrierte Schaltfläche.

Wir wollen in unser Makro eine zusätzliche Änderung einbauen. Da einige Mitglieder von der Beitragszahlung befreit sind, soll das Programm eine Meldung ausgeben, falls das angesprochene Mitglied zu dieser Gruppe gehört. Dazu sind folgende Schritte notwendig:

- Du öffnest das Makro *Formulare öffnen.Beiträge eingeben* in der Entwurfsansicht.
- Als erste Zeile im Makro fügst Du folgende Aktion ein:
  `Meldung: Mitglied ist von der Beitragszahlung befreit!;Ja;Ohne Symbol;Ehrenmitglied`
- Die Aktion gibt ein PopUp-Fenster aus, das durch Klick auf OK abgearbeitet werden muss. Dieses Fenster soll erscheinen, falls das aktuelle Mitglied im Formular „Adressen“ ein Ehrenmitglied und folglich von der Beitragszahlung befreit ist.
  Also musst Du die Spalte Bedingungen einblenden. Die Aktion Meldung soll nur unter der oben beschriebenen Bedingung ausgeführt werden.
  Um die Bedingung zu bearbeiten, klickst Du auf den Editor in der Symbolleiste. Das bedingte Ausführen eines Befehls wird in Programmiersprachen als *if-Schleife* bezeichnet und hat einen Wahr- und einen Falsch-Ausgang. Der Vorgang hier – in den Makros – ist als eine einfache, aber sehr brauchbare Methode anzusehen.
  Das Makro fragt den Wert der Bedingung ab. Trifft er zu, ist er also **`Wahr`**, wird die dahinterstehende Aktion ausgeführt sowie alle weiteren, die in der Spalte Bedingungen mit **`„...“`** gekennzeichnet sind. Ist der Wert dagegen **`Falsch`**, werden die entsprechenden Zeilen ausgelassen, und das Makro geht zu den nachfolgenden Aktionen über.
  Ist das Formular „Adressen“ geöffnet, kannst Du den Abfragewert *Ehrenmitglied: Ja/Nein* ins Editoren-Fenster laden. Das in Programmiersprachen gebräuchliche *if* wird in diesem Fall weggelassen.

- Mit einem Klick auf **OK** sollte der Ausdruck in der Spalte Bedingungen erscheinen.



Das ganze hört sich schwieriger an, als es ist. Hoffentlich habe ich es halbwegs verständlich beschrieben – wenn nicht, kann ich Dir die Hilfe vorschlagen, die recht verständlich ist. Du ziehst die Direkthilfe in die Spalte Bedingung und hangelst Dich durch die verschiedenen Hilfeangebote. Die Hilfe bietet u.a. auch Beispiele dafür, in welchen Situationen Du Bedingungen einsetzen kannst.

- Du benötigst eine weitere *Aktion* in Deinem Makro. Falls die Bedingung zutrifft, musst Du nach der Meldung dem Makro mitteilen, was es tun soll. Tatsächlich soll es zurück in die Adress-Maske gehen, wo der Benutzer dann ein anderes Mitglied bearbeiten kann. Das erreichst Du durch die folgende Aktion:
  GeheZuSteuelement: **`Name`** (Der Benutzer kann sich hier ein neues Mitglied suchen)
  StopMakro: (Hält das Makro an).
- Vergib interessehalber für ein oder mehrere Mitglieder den Status [Ehrenmitglied] und probiere das Makro bei diesen und bei anderen aus.

Dialogfenster einbinden

Bei den *Formularen* haben wir ein Formular entworfen, das sich „Dialog DM – Euro“ nannte. Wir haben es als Pop Up-Formular gespeichert, und ich hatte angedeutet, dass wir es dazu benutzen wollten, die Eingabe des Mitgliedsbeitrag auf die Währung hin zu überprüfen.

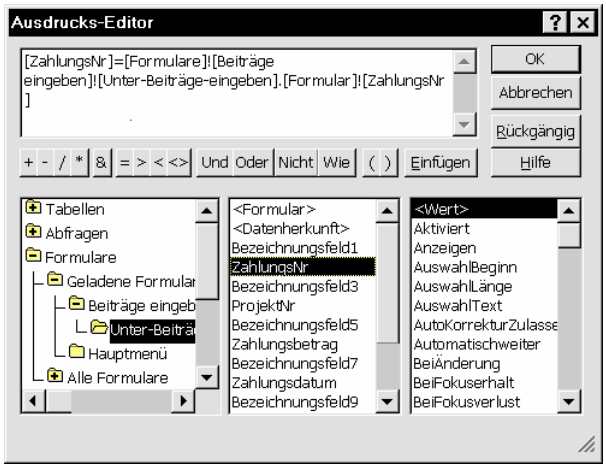
Dieses Dialogfenster soll immer dann eingeblendet werden, wenn der Benutzer in der Beitragsmaske einen Geldbetrag eingegeben hat und das Feld verlassen will. Den Wert, den der Benutzer im Formular einstellt, speichern wir im Feld *Option* der Tabelle *Beiträge*, auf der das Formular basiert – so können wir ihn später in die gewünschte Währung umrechnen oder beide Beträge gleichzeitig anzeigen.

Beim Aufruf des Formulars *Dialog: DM-Euro* müssen wir wissen, welcher Datensatz im Formular „Beiträge eingeben“ aktuell ist. Auf eben diesen Datensatz stellen wir das Formular ein. Für den aktuell eingegebenen Betrag im Feld *Zahlungsbetrag* im Unterformular *Unter-Beiträge-eingeben* speichern wir den Optionswert aus der *Optionsgruppe*. Diesen können wir dann nach Bedarf abfragen.

Du öffnest das Makro *Beiträge* in der *Entwurfsansicht* und fügst ein neues Makro mit dem Namen **`DM-Euro`** ein. Das Makro muss folgende Aktionen abwickeln:

- *AusführenBefehl*: *DatensatzSpeichern*. Der Datensatz aus dem aufrufenden Formular *Beiträge eingeben* wird gespeichert. Dies ist notwendig, weil die nächste Aktion sonst nicht den aktuellen Datensatz findet.
- *ÖffnenFormular*: **`Dialog: DM-Euro;Formular;; [ZahlungsNr]=[Formulare]![Beiträge eingeben]![Unter-Beiträge-eingeben].[Formular]![ZahlungsNr];;Dialog`**
  Zu dem *Aktionsargument* „*Bedingung*“ ist zu sagen, dass Access das Formular *Dialog: DM-Euro* auf den Datensatz des Formulars *Beiträge eingeben* einstellt, ohne dass das Steuerelement *ZahlungsNr* überhaupt im Formular enthalten ist. Das liegt daran, dass beide Formulare auf der gleichen Tabelle basieren. Du kannst Dir dieses Argument in der beschriebenen Methode mit dem Ausdrucks-Editor  erstellen.
  Willst Du im 1. Fenster auf „geladene Formulare“ doppelklicken, muss das entsprechende Formular in der *Formularansicht* geöffnet sein. Denke daran, dass das Feld *ZahlungsNr* sich im

Unterformular *Unter-Beiträge-eingeben* befindet. Das Bild zeigt, welchen Weg Du gehst.



Das *Aktionsargument* **`Fenstermodus=Dialog`** bewirkt, dass das Formular, das durch die Aktion aufgerufen wird, grundsätzlich auf die *Eigenschaften* **`Pop Up`** und **`Gebunden`** eingestellt wird, ohne Rücksicht darauf, ob das Formular als Pop Up-Formular gespeichert wurde.

Das Makro hätten wir – jetzt kommt der Aufruf. Das Makro soll gestartet werden, nachdem der Benutzer den Geldbetrag eingegeben hat. Also öffnest Du das Formular *Unter-Beiträge-eingeben* in der *Entwurfsansicht* und dann für das Feld *Zahlungsbetrag* mit dem Button  das *Eigenschaftenfenster*. Unter der Registerkarte *Ereignis* findest Du die verschiedenen Bedingungen, unter denen Du ein Makro für das Feld starten kannst. Uns interessieren die Momente des *Verlassens* – also *Nach Aktualisierung*, *Bei Verlassen*, *Bei Fokusverlust*, vielleicht *Bei Änderung*. Ziehst Du die *Direkthilfe* auf die verschiedenen Zeilen, kannst Du leichter beurteilen, was am besten geeignet ist. Ich plädiere für *Bei Verlassen*. Du stellst die Eigenschaft auf das Makro *Beiträge.DM-Euro* ein.

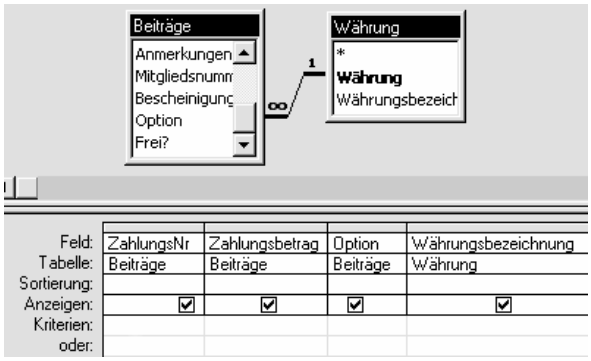
Das wäre das Makro, das uns *ins* Formular bringt – fehlt noch eines, um es wieder zu verlassen. Dafür werden folgende Aktionen benötigt:

- *AusführenBefehl*: **`DatensatzSpeichern`**. Der Befehl speichert den aktuellen Datensatz im Formular *Dialog: DM-Euro*. Damit wird die Optionskennziffer aus der *Optionsgruppe* in die Tabelle *Beiträge* zurückgeschrieben – wodurch sich der vom Benutzer eingegebene Zahlungsbetrag auf seine Währung hin überprüfen lässt.
- *Schliessen*: **`;;Nein`**
  Die Aktionsargumente *Objekttyp* und *Objektname* können leer bleiben, weil das Formular *Dialog: DM-Euro* im Vordergrund steht. Dahinter liegt *Beiträge eingeben*, weil von dort das Makro aufgerufen wurde. Wird das vorderste Fenster geschlossen, kehrt das Makro ins darunterliegende Formular zurück, was es in diesem Fall auch soll. Der Benutzer kann anschliessend weitere Beiträge eingeben.

Eine Bemerkung zur Währung: Bislang hast Du nur eine Optionskennziffer. Um die Währungsbezeichnung zu erhalten, erzeugst Du eine neue Tabelle mit folgenden Werten:

Währung	Währungsbezeichnung
1	DM
2	Euro
3	$
4	£
5	Yen
1	

Dabei enthält das Feld „Währung“ die gleichen Optionswerte, wie sie auch in der Tabelle *Beiträge* vorkommen. Das Feld ist in der Tabelle „Währung“ *Primärschlüssel* und *1:n* über das Feld „Option“ mit der Tabelle „Beiträge“ verknüpft. In einer Abfrage, die beide Tabellen einbezieht, kannst Du also die Währungsbezeichnung ausgeben lassen, wenn Du aus „Beiträge“ die Optionswerte abfragst und aus „Währung“ die entsprechende *Währungsbezeichnung* hinzufügen lässt.



Die ganz einfache Version wäre, wenn Du das Feld *Option* der Tabelle *Beiträge* als *Nachschlagfeld* erstellst, das sich die Werte aus der Tabelle *Währung* holt. Der Benutzer kann sich dann aus der Liste, die das Nachschlagfeld zur Verfügung stellt, die entsprechende Währungsbezeichnung auswählen. Der Nachteil ist, dass Du bei jeder Ausgabe in mehreren Währungen die Beträge umrechnen musst.

Makros in VBA konvertieren

Makros sind die eine Möglichkeit in Access, Arbeitsautomatisationen zu programmieren. Damit kannst Du gut und schnell Erfolge erlangen.

Schwierig wird die Sache aber, wenn Dein Programm umfangreiche Berechnungen anstellen muss, die variable Grössen enthalten. Du kannst mit Makros schlecht temporäre Daten aufbewahren, um sie zu Berechnungen heranzuziehen oder aktuelle Werte zu speichern. Dazu musst Du auf VBA umsteigen – hier kannst Du nämlich globale Variablen definieren, denen Du aktuelle Werte zuweist.

Damit Du Deine Makros in diese Anwendungen einbauen kannst, besteht in Access die Möglichkeit, Makros in VBA zu konvertieren. Dies machst Du im Datenbankfenster unter **Extras | Makro | Makros zu Visual Basic konvertieren.** Dazu muss das Makro markiert sein. Im folgenden Fenster solltest Du beide Optionen anklicken, damit die Funktionen korrekt ausgeführt werden.

Aber Vorsicht! Konvertierte Makros können in VBA leicht unübersichtlich sein, weil Makros nach einem anderen Schema arbeiten. Ich kann Dir an dieser Stelle nur eines empfehlen: probiere selber aus, wann Du eine Anwendung mit Makros erstellst oder wo Du besser gleich zu VBA greifst.

Eine Bemerkung zur Schnelligkeit: die Anwendung wird sichtbar schneller, wenn alle Makros zu VBA konvertiert wurden.

Import / Export

Access bietet auch Internet-Möglickeiten an. Unter dem Menüpunkt **Datei | Im HTML-Format speichern** findest Du die Möglichkeit, eine Datenbank komplett oder teilweise im Internet zu veröffentlichen. Falls Du den Web Publishing Assistenten auf Deinem Rechner installiert hast, kannst Du sie sogar anderen Benutzer zur Verfügung stellen.

Klickst Du auf den oben beschriebenen Menüpunkt, startet Access den *Assistenten zur Veröffentlichung im Netz*, der Dich durch die Aufgaben geleitet.

Access speichert die ausgewählten Datenbank-Objekte im HTML-Format (→ Glossar) in einem speziellen Verzeichnis, das Du angibst. Das Ergebnis sind mehrere Web-Seiten, die Deine Datenbank-Objekte wie *Formulare, Abfragen* und *Berichte* verwalten.

Schluss

Du wunderst Dich an dieser Stelle vielleicht, dass die Datenbank *Verein*, die Du zur Übung parallel zu dem Heft entwickeln solltest, bei weitem nicht den Umfang hat, den ich anfänglich beim *Datenentwurf* beschrieben habe. Das liegt zum Teil am Umfang dieses Heftes – aber es hat auch damit zu tun, dass ich Dir eine Idee geben wollte, wie eine ansatzweise komplexe Datenstruktur aussieht und wie Du damit umgehen kannst. Trotzdem würde es mich freuen, wenn Du es als Deine Aufgabe ansiehst, den Datenentwurf fertigzustellen – damit hättest Du auch gleich ein Übungsfeld. Wenn Deine Datenbank „Verein" alle anvisierten Funktionen erfüllt, so dass ein echter Verein damit arbeiten könnte, kannst Du, glaube ich, mit Fug und Recht behaupten, dass Access Dir gut vertraut ist. Wenn Du dazu Anschauungsmaterial benötigst, hat der *Know Ware* Verlag eine Internet-Seite, von der Du Dir die Datenbank herunterladen kannst. Die dortige Datenbank enthält alles, was wir in diesem Heft entwickelt haben – und noch ein bisschen mehr. Die Adresse lautet

`www.knowware.de/access/`

Ein Wort noch zur sogenannten *Run time-Version*, die ich auf Seite 2 erwähnt habe: Run time, also ablauffähiges Programm, ist unabhängig von der Access-Plattform. Im Umwandlungsvorgang wird Deine Datenbankdatei, die, wie Du vielleicht im Explorer festgestellt hast, mit der Dateierweiterung *.MDB bezeichnet wird, in eine EXE-Datei verwandelt. Dabei wird das Entwicklungswerkzeug von Access aus der Datenbank entfernt – was bleibt, ist die reine Anwendung, so wie Du sie geschrieben hast. Diesen Prozess, der in allen Programmiersprachen und –anwendungen existiert, nennt man *Kompilieren*. Die Datenbank wird so wesentlich kleiner und kann unkompliziert auf andere Betriebssystem-Plattformen portiert werden, wie z.B. Unix-Umgebungen. Änderungen an der Anwendung selber kannst Du dann nur noch über Umwege vornehmen.

Bis zur Version Access 2.0 war dieses Tool zur Umwandlung noch standardmässig in der Software enthalten. Heute musst Du eine *Professional Edition* erwerben, die ungefähr das Doppelte der Standardversion kostet. Darin enthalten ist aber auch eine gute *Replikations*verwaltung. Diese gewährleistet, dass alle Kopien oder Replikate, die Du von der Datenbank angefertigt hast, um sie mit der Originaldatenbank wieder zusammenzuführen, ohne Konflikte wieder miteinander abgeglichen werden. Replikate eignen sich z.B. für Aussendienstmitarbeiter, die auf Notebooks arbeiten und abends in der Firma vor Ort oder über Datenfernübertragung (DFÜ) ihre neuen oder geänderten Daten mit der Zentrale abgleichen.

Ich hätte Dir gern noch all die anderen Features des Programms beschrieben, doch dann hätte ich wohl weitere 500 Seiten schreiben müssen. Ich fand es eher sinnvoll, die behandelten Teile ausführlicher zu beschreiben, anstatt viele Dinge anzureissen und Dich dann mit den Einzelheiten, die oft das schwierigste sind, alleine zu lassen. Ich hoffe, dass ich Dir auf diese Art und Weise etwas helfen konnte. Vielleicht hast Du Vertrauen in den Umgang mit Access gewonnen und hast nun genug Mut und Motivation, selbständig weiterzuarbeiten.

Weisst Du gar nicht weiter, willst Du professionelle Anwendungen schreiben, oder verlangt der Betrieb, in dem Du arbeitest, etwas, das über Deine Kenntnisse hinausgeht, kannst Du die Software-Firma konsultieren, für die ich arbeite, oder auch mich persönlich. Du wirst verstehen, dass ich einen Unterschied machen muss zwischen persönlichen und professionellen Anfragen. Unter folgender e-mail-Adresse bin ich jederzeit für Dich erreichbar:

RainerSchubert@t-online.de

Ich wünsche Dir viel Spass – den braucht man – und Erfolg beim Umgang mit Access.

Glossar

Add-In: Hilfswerkzeuge für das Programm. So sind Assistenten und Editoren Add-Ins. Add-Ins können aus mehreren Visual Basic-Prozeduren und Dialogfeldern bestehen.

ADT: Acess **D**eveloper´s **T**oolkit: Professionelles Erweiterungspaket von Access. Damit kannst Du u.a. ablauffähige Programme erstellen, die selbständig arbeiten und von der Access-Oberfläche unabhängig sind. War bis Access 2.0 noch in der normalen Version enthalten, jetzt ist der Erwerb ziemlich teuer.

Anwendung: Eine mit einem *Programm* erstellte Arbeitseinheit. Als Beispiel gilt die Datenbank „Verein“, mit der die organisatorischen Aufgaben in einem Verein abgewickelt werden.

API: Application **P**rogramming **I**nterface; Programmierschnittstelle für Entwickler von Anwendungen.

AutoExec: Ein Makro, das bei dem Start einer Access-Anwendung automatisch ausgeführt wird. Ein Benutzer kann gezielt zu einem Haupt-Formular geführt werden.

Benutzerkonto: speichert die Zugriffsberechtigungen eines Benutzers. Kann mit Passwort gesichert werden.

Bit: Grundsätzliche Entscheidungs-Einheit im Computer. Danach fliesst der Strom auf dem „Ja“- oder „Nein“- Zweig weiter bis zur nächsten Entscheidung.

Byte: $2^3 = 8$ *Bit*. Mit 1 Byte kann man eine Zahl oder einen Buchstaben eindeutig identifizieren (aus 255 Möglichkeiten).

Client/Server: Allgemeines Kommunikations-Konzept zwischen mehreren Computern. Der Benutzer arbeitet selbst am Client-Rechner, bezieht seine Daten aber vom Server-Rechner, wo er sie auch speichert. Die Daten befinden so immer auf dem aktuellen Stand. (→ relationale Datenbank)

Daten: Werte, die z.B. in ein *Feld* eingegeben werden. Das *Feld* stellt das leere Blatt Papier dar, auf das Du einen Brief schreibst. Das, was Du schreibst, sind dann die *Daten,* also der Inhalt. Ohne das Blatt Papier könntest Du aber keinen Brief schreiben!

Datensatz: 1 Mitglied mit all seinen Informationen ist 1 Datensatz. Ein Datensatz besteht aus verschiedenen Feldern.

DDE: Dynamic **D**ata **E**xchange: Kommunikationsprotokoll, mit dem Daten zwischen verschiedenen Anwendungen und Programmen ausgetauscht werden können.

DLL: Dynamic **L**ink **L**ibrary: dynamische Bibliotheken, auf die aus verschiedenen Programmen zugegriffen werden kann. Hier sind z.B. die API-Funktionen gespeichert.

Dynaset: nennt man das Ergebnis einer *Abfrage*. Hier geänderte Daten werden in der zugehörigen Tabelle abgespeichert.

Eigenschaften: Die Datenbank-Objekte wie *Formulare, Abfragen* etc. besitzen spezielle Eigenschaften, die in dem *Eigenschaften-Fenster* eingestellt werden.

Ereignis: Datenbank-Objekte reagieren auf Ereignisse, wie z.B. auf Klick, der dann eine bestimmte Aktion ausführt.

Exportieren / Importieren: Kopieren von Daten in / aus andere/n Anwendungen oder Datenbanken.

Feld: Der „Name“ des Mitglieds ist z.B. ein Feld. Da ein Name aus Buchstaben besteht, haben wir es als Textfeld, seine maximale Grösse mit 35 Zeichen festgelegt. Andere Felder sind z.B. numerisch, sie enthalten dann ausschliesslich Zahlen. Die einzelnen Felder mit ihren Daten bilden einen Datensatz.

Feldbegrenzer: Umfassen ein Feld in seiner Grösse. An seinen Grenzen ändert sich der Mauszeiger, so dass Du seine Grösse ändern kannst.

Grundeinstellung: Bestimmte Grössen können in Access vorgegeben sein, wie die Standard Feldgrösse für ein Textfeld. Diese Einstellungen kannst Du unter **EXTRAS|OPTIONEN** ändern oder im Datenbank-Objekt selber (*Standardwert*).

Filter: Benutzerdefinierte oder vorgegebene Einschränkungen einer Datenauswahl.

FTP: File **T**ransfer **P**rotocol, bezeichnet eine Übereinkunft für das Lesen von Daten auf FTP-Rechnern im *Internet*.

Funktion: eine VBA-Prozedur, die nach ihrer Ausführung einen Wert zurückgibt. Mit Access werden eine Reihe von Funktionen mitgeliefert wie Datum(), Du kannst aber auch selbst Funktionen schreiben.

HTML: Hyper **T**ext **M**arkup **L**anguage: Kodierungsformat für das Internet. Access-Daten können

in diesem Format gespeichert und dann im Web veröffentlicht werden.

HTTP: das Protokoll zur Übertragung von Daten im Internet.

Jet Database Engine: eine Art Datenmanager in Access, der im Hintergrund den Zugriff auf die Daten der Datenbank regelt.

Markieren: Hervorheben, für eine Aktion aussuchen.

Modem: MOdulator / DEModulator: Gerät, das die Daten aus dem Rechner zum Versenden über die Telefonleitung in Töne verwandelt. Beim Empfänger geschieht der umgekehrte Prozess – die Töne werden in Daten zurückgewandelt (demoduliert). ISDN-Leitungen können die digitalen Daten des Rechners direkt transportieren. Eine Modulation wird hier nicht benötigt.

ODBC: Open **D**ata**b**ase **C**onnectivity: ein Protokoll, das u.a. von Microsoft unterstützt wird. Es hilft Dir, Daten von externen Datenbanken in Deine Anwendung einzubinden.

Office: = Büro, hier Software Paket, mit dem die Aufgaben von kleinen bis zu grossen Büros abgewickelt werden können.

OLE: Object **L**inking and **E**mbedding, bedeutet die Methode, Objekte aus anderen Office-Anwendungen (Word, Excel) zu verbinden (verknüpfen) oder einzubetten. Beim Verknüpfen wird nur ein Verweis auf die Daten erstellt, während beim Einbetten die Daten selbst gespeichert werden. Änderungen an der Datenquelle werden nur bei der *Verknüpfung* berücksichtigt und können aktualisiert werden.

Pivot-Tabellen können grosse Datenmengen übersichtlich darstellen. Dies wird durch *pivot* = engl. für *sich drehen* erreicht, was hier soviel heisst wie, dass Zeilen und Spalten einer Tabelle leicht ausgetauscht, also gedreht werden können. So können die Daten übersichtlicher dargestellt werden.

Programm: Software, mit der man eine *Anwendung* erzeugt. So gesehen ein Werkzeug aus der Werkzeugkiste *Computer*.

QBE: Query **B**y **E**xample beschreibt die Methode, Abfragen grafisch zu erstellen. Die untere Hälfte des *Entwurfs-Fensters* für die Abfrage nennt man den QBE-Bereich.

Routine: Immer wieder auftauchende Arbeiten, die der Computer übernehmen soll.

Steuerelement: sind die Elemente in *Formularen* und *Berichten*, die z.B. die Daten aus einem Tabellenfeld darstellen. Sie können aber auch das Ergebnis einer Berechnung oder eine Grafik ausgeben.

UNC: Universal **N**aming **C**onvention: Standardformat für die Eingabe eines Pfads, der sich auf einen Netzwerk-Dateiserver bezieht.

variable Grösse: Der Inhalt von Feldern, der in verschiedenen Datensätzen verschiedene Werte annehmen kann.

VBA: Visual **B**asic für **A**pplikationen (Anwendungen): Programmiersprache für Office-Programme

Verknüpfung: Verbindung von zwei Datenquellen (z.B. *Tabellen*), die zu einer gemeinsamen Auswertung herangezogen werden (z.B. *Abfrage* oder *Bericht*).

Zugriffsrechte: siehe *Benutzerkonto*