# Inhaltsverzeichnis








# Einleitung

Die Datenbankprogrammierung ist ein wichtiges Thema. Vermutlich ist es für Visual-Basic-Programmierer sogar das wichtigste Thema, denn bei einer von Microsoft im Vorfeld der Einführung von Visual Basic 6.0 durchgeführten Umfrage setzten mehr als 90% der Befragten das Thema Datenbankzugriff an oberste Stelle ihrer Prioritätenliste. Es verwundert daher nicht, daß Microsoft bei Visual Basic den Datenbankzugriff von Version zu Version verbessert hat und es auch zum Schwerpunkt künftiger Entwicklungen machen wird. Doch was bedeutet Datenbankprogrammierung konkret? In den meisten Fällen versteht man darunter den Zugriff auf bereits vorhandene Datenbanken. Hier ein konkretes Beispiel von vielen aus der Praxis. Stellen Sie sich vor, eine Stadtbücherei hat sowohl ihren Buchbestand, die Adressen ihrer Entleiher als auch die aktuellen Entleihdaten (also wer zur Zeit welches Buch ausgeliehen hat und welche Bücher von wem vorbestellt sind) in einer Datenbank untergebracht. Dann muß es natürlich möglich sein, nach diesen Daten zu suchen, um z.B. beim Entleihen eines Buches nach Eingabe der Mitgliedsnummer den vollständigen Namen des Entleihers und vielleicht auch die Titel aller noch nicht zurückgegebenen Bücher zu sehen. Dies ist eine typische Datenbankabfrage nach dem Motto »Zeige mir alle Datensätze, die ein bestimmtes Kriterium erfüllen«. Aber Datenbankprogrammierung bedeutet noch (viel) mehr. Zum einen möchten die Besitzer der Daten sehr schnell mehr aus ihrem Datenbestand (der nicht selten das Kapital des Unternehmens darstellt; denken Sie an die vielen Direktversender oder an Headhunting-Agenturen, die ohne ihre Geschäftsdaten nicht funktionsfähig wären) herausholen. Unsere Stadtbücherei möchte z.B. wissen, welche Buchkategorie am häufigsten ausgeliehen wird, wann ein beliebtes Buch zum frühesten Termin vorbestellt werden kann oder in welchen Stadtteilen die »Ausleihfreudigkeit« am höchsten ist. Alles das sind Dinge,






die sich mit Datenbankprogrammierung ohne großen Aufwand erledigen lassen. Leider ist dabei auch schnell jene unsichtbare Grenze erreicht, die sich bei kritischer Betrachtung durchaus als Mißbrauch der Daten auslegen läßt. Würde die Stadtbücherei z.B. die Namen ihrer schulpflichtigen Klienten, die ihre Bücher vormittags ausleihen, an die Schulbehörde weiterleiten, die sie wiederum mit den (in einer anderen Datenbank gespeicherten) Stundenplänen vergleicht, um jene Schüler herauszufinden, die ihren persönlichen Spielraum bei der Gestaltung ihres Unterrichts über Gebühr ausnutzen, so wäre das zwar programmiertechnisch machbar und für einen Programmierer durchaus reizvoll, es wäre aber auch (vermutlich) ein Verstoß gegen die Datenschutzbestimmungen[1].

Datenbankprogrammierung bedeutet also in erster Linie das Abfragen der vorhandenen Daten. Zwar ist es kein Problem, eine Datenbank in Visual Basic (auch programmgesteuert) neu zu erstellen, doch wird diese Aufgabe in der Regel von dem Datenbanksystem übernommen. Datenbankprogrammierung beinhaltet aber auch andere wichtige Aufgaben, wie das sogenannte Einpflegen der Daten (also die Übernahme neuer Daten in die Datenbank, etwa wenn eine komplette Schulklasse aufgenommen werden soll, die Stadtbücherei aber nur eine unvollständige Liste der Namen und Anschriften erhält), die Durchführung von Sicherungskopien, das Anfertigen von Datenbankreports und andere Dinge. Kurzum, die Datenbankprogrammierung ist ein hochinteressanter und vielseitiger Bereich. Und da unsere (entstehende) Informationsgesellschaft immer mehr auf den »Rohstoff« Daten angewiesen ist, gewinnt dieser Bereich immer mehr an Bedeutung.

## Visual Basic und die Datenbanken

Warum gerade Visual Basic? Warum werden die im letzten Abschnitt beschriebenen Aufgaben nicht von dem »Datenbankprogramm« erledigt? Nun, so ist es in der Praxis in vielen Fällen auch noch. Wenn Sie etwa eine Stadtbücherei betreten, werden Sie dort nicht automatisch (Windows-)PCs antreffen, sondern meistens sogenannte »EDV-Systeme«, bei denen die Datenbank und die Software, die auf die Datenbank zugreift, eine Einheit darstellen. Irgend jemand hat die Software in einer für das Datenbanksystem typischen Programmiersprache programmiert und das komplette System an die Stadtbücherei verkauft. Dies ist der traditionelle DV-Ansatz. Doch diese Ansätze befinden sich in der Auflösung. An Datenbankanwendungen werden Anforderungen gestellt, die sich mit traditionellen Systemen nicht mehr realisieren lassen. Die Stadtbücherei wird mit anderen Stadtbüchereien in

1 Das Beispiel ist natürlich konstruiert und nicht ganz ernst gemeint – es soll lediglich die grundsätzlichen Möglichkeiten andeuten.

einen Verbund eingegliedert, die Buchbestände sollen übergreifend abfragbar sein. In einer mobilen Bücherei sollen auch die Daten aller Hauptbüchereien abfragbar sein, die Datenbestände müssen auf die Notebook-Computer der mobilen Stationen repliziert werden. Das Ausleihen soll auch über das Internet möglich sein und vieles mehr. Dies sind Dinge, die sich mit traditionellen Datenbanksystemen im allgemeinen nicht oder nur mit hohem Aufwand realisieren lassen. Jetzt kommt Visual Basic als ein vielseitiges, modernes und leistungsfähiges Programmiersystem ins Spiel, das vor allem eine preiswerte Software-Entwicklung ermöglicht. Wer heute eine Datenbankanwendung (also ein Programm, das auf einer Datenbank basiert) programmiert, muß sich zwischen einer speziellen Lösung (bei der manchmal auch die Hardware schon vorgegeben ist) und einer universellen Lösung entscheiden. Beide Ansätze haben ihre spezifischen Vor- und Nachteile. Der Vorteil der ersten Lösung ist vor allem Stabilität, Robustheit und eine genaue Übereinstimmung mit dem Arbeitsablauf in dem Unternehmen (meist wurde dieser über die Jahre an die »Gewohnheiten« der EDV-Anlage angepaßt). Der Vorteil der zweiten Lösung ist die Flexibilität (und in der Regel auch der Preis), denn Visual Basic ist auf keine Datenbank fixiert, sondern dank standardisierter Schnittstellen (ODBC und OLE DB) weitestgehend offen. In der PC-Welt gibt es eine große Auswahl leistungsfähiger Datenbanksysteme: Oracle, Informix, Visual FoxPro, der SQL-Server von Microsoft und natürlich Microsoft Access seien hier als Beispiele genannt. Die meisten Datenbanksysteme bieten eine starke Datenbank, aber nur eine schwache Programmierungsumgebung (Visual FoxPro natürlich ausgenommen). Hier kommt Visual Basic als leistungsstarke Ergänzung ins Spiel, die entscheidende (Standort-)Vorteile besitzt: Es ist relativ leicht zu lernen, verhältnismäßig preiswert und inzwischen sehr weit verbreitet. Das ist der Grund, warum der Datenbankprogrammierung mit Visual Basic eine solch große Bedeutung zukommt, die in Zukunft vermutlich weiter steigen wird.

## Visual Basic oder Microsoft Access?

Dies ist eine wichtige Frage, die an dieser Stelle dennoch nur kurz abgehandelt werden soll. Auch das populäre Microsoft Access ist ein vollständiges Entwicklungssystem, mit dem sich komplette Datenbankanwendungen erstellen lassen. Allerdings kommt Microsoft Access aus der Ecke der Endanwender-Produkte. Der normale »Werdegang« eines Access-Entwicklers sieht so aus: erst Microsoft Access als reine Endanwendung kennenlernen und nach und nach in die Programmierung mit VBA (früher AccessBasic) einsteigen. Für viele kleinere und mittlere Anwendungen ist das auch eine optimale Strategie. Visual Basic ist dagegen ein universelles Entwicklungswerkzeug mit einer sehr guten Datenbankschnittstelle. Allerdings steht die Da-

tenbank anders als bei Microsoft Access nicht im Mittelpunkt. Wer ein neues Formular hinzufügt, erhält damit keinerlei Datenbankanbindung. Diese muß erst Schritt für Schritt hinzugefügt werden. Dafür bietet Visual Basic eine Reihe von Leistungsmerkmalen, die es bei Microsoft Access nicht gibt: einen echten Compiler (der Maschinencode produziert), die Möglichkeit, COM-Komponenten erstellen zu können, eine um Add-Ins und Designer erweiterbare Entwicklungsumgebung (das gibt es erst bei Microsoft Access 2000) und vor allem ein Programmiermodell, das nicht automatisch datenbankzentriert ist. Vereinfacht läßt sich folgende Empfehlung geben: Wer nicht oder nur wenig programmiert, aber eine richtige Datenbankanwendung mit VBA-Programmierung erstellen möchte, ist bei Access besser aufgehoben. Wer dagegen in erster Linie an einem universellen Programmierwerkzeug interessiert ist, aber die Datenbankprogrammierung dennoch in den Mittelpunkt stellen möchte, findet mit Visual Basic das leistungsfähigere Werkzeug. Natürlich gibt es auch einen Bereich, in dem sich beide Produkte überschneiden.

## ADO oder DAO – keine leichte Entscheidung

Datenbankprogrammierer, die mit Visual Basic oder Microsoft Access arbeiten, stehen zur Zeit vor einer Entscheidung: ADO oder DAO? Beide Buchstabenkürzel stehen für eine Datenbankzugriffsmethode (am Ende des Buches werden Sie diese und viele andere Kürzel im Schlaf beherrschen), wobei ADO die moderne und DAO die alte Variante ist. Für DAO spricht, daß es sich in der Vergangenheit beim Zugriff auf Access-Datenbanken bewährt hat, es viele Programmierer beherrschen und ein Umstieg auf ADO eine mehr oder weniger umfangreiche Neuprogrammierung erforderlich macht – teilweise mit ungewissem Ausgang, weil noch nicht alle DAO-Merkmale unter ADO implementiert wurden. Für ADO spricht der Umstand, daß es die Zukunft in der Microsoft-Strategie ist. DAO dürfte keine wesentlichen Weiterentwicklungen mehr erfahren, Microsoft steckt die gesamte Entwicklungsarbeit in ADO. Visual Basic 6.0 ist mit seinen neuen Datenbanktools, wie dem Datenumgebungsdesigner, der beste Beweis. Visual InterDev, das Microsoft Entwicklungssystem für Webanwendungen, basiert vollständig auf ADO. Über kurz oder lang werden alle Programmierer auf ADO umsteigen müssen, wenngleich DAO auch von kommenden Visual Basic- und Access-Versionen unterstützt wird.

Da der Umstieg von DAO auf ADO noch eine Weile dauern kann, stellte sich natürlich auch für mich als Buchautor die Frage, welcher Datenbanktechnologie ich den Vorzug geben sollte. Auch wenn sich ADO und DAO in ein und derselben Prozedur gemeinsam einsetzen (allerdings nicht kombinieren – ein DAO-Recordset und ein ADO-Recordset sind inkompatibel)

lassen, mußte eine Entscheidung getroffen werden, denn ein Mix aus DAO und ADO würde nicht nur den Platzrahmen über Gebühr beanspruchen, es würde auch viele Leser irritieren. Nach reiflicher Überlegung habe ich daher entschieden, ADO in den Mittelpunkt zu stellen und DAO weitestgehend zu ignorieren. Auch wenn DAO, gerade wenn es um den Zugriff auf Access-Datenbanken geht, genau das ist, was ca. 70% aller Datenbankprogrammierer jemals benötigen werden, die Zukunft gehört ADO. Wer jetzt DAO lernt, wird mit an Sicherheit grenzender Wahrscheinlichkeit in den nächsten zwei bis drei Jahren wieder umlernen müssen. Zwar ähneln sich ADO und DAO in ihren Grundzügen, doch ist das sehr viel flexiblere und universellere Konzept von ADO gerade für erfahrene DAO-Programmierer eine echte Herausforderung. Diese vermissen ihre vertraute *FindNext*-Methode, die *Seek*-Methode und die *NoMatch*-Eigenschaft und wundern sich, wie sie um alles in der Welt eine ODBC-Tabelle an eine Access-Datenbank hängen oder auf Datenbanken über einen ISAM-Treiber zugreifen sollen. Wer dagegen bereits mit den RDO-Objekten unter der Enterprise Edition von Visual Basic 5.0 programmiert hat, wird bei ADO viele vertraute Konzepte wiederfinden.

Als Anwender einer so zentralen Technologie wie es die Datenbanktechnologie nun einmal ist, gerät man schnell in die typische »Microsoft-Falle«. Microsoft (der Hersteller von Windows, Visual Basic, Office und vielen anderen schönen Produkten) basiert seine gesamte *Universal Data Strategy* (UDA) auf OLE DB und ADO, einem überaus leistungsfähigen Konzept. Dagegen gibt es grundsätzlich nichts einzuwenden. Im Gegenteil, ADO verspricht eine Menge und kann davon bereits jetzt einiges halten (ADO 1.0 wurde ursprünglich mit dem Internet Explorer 4 eingeführt). ADO bedeutet maximale Flexibilität bei maximaler Performance und vieles mehr. Das Problem dabei ist nur, daß ADO eine Komplexität einführt, von der viele Visual-Basic-Programmierer nicht oder nur indirekt profitieren. Wer nur eine Adreßverwaltung, eine Lagerverwaltung, eine normale kaufmännische Anwendung oder eine Webdatenbank programmiert, kommt mit einer Datenbankfunktionalität aus, die in eine 400 Kbyte große DLL gepackt werden kann. Für diese Anwender sind eine flexible Auswahl einer Datenquelle, Skalierbarkeit auf SQL-Server, asynchrone Abfragen, persistente Datensätze und verbindungslose Recordsets einfach kein Thema. Dennoch zahlen Sie den »Preis« (in Form von Lernaufwand), den auch jene Programmierer zahlen müssen, die in Unternehmen Gigabyte große Datenbanken mit Tausenden von Anwendern konzipieren, die auf Multiprozessorensystemen laufen und Tag und Nacht im Einsatz sind. Das ist in etwa so, als würde Microsoft einen Hochleistungs-LKW mit 24 Gängen, einem supermodernen Motor, geringem Benzinverbrauch, automatischer Be- und Entladung und anderen High-End-Eigenschaften zu einem konkurrenzlosen Preis (oder gar kosten-

los) anbieten. Für Inhaber von Speditionsfirmen wäre es eine tolle Sache (sofern die Wartungskosten nicht exorbitant sind), wer dagegen nur seinen Umzug machen oder ein paar Gegenstände um den Häuserblock fahren will, muß erst einmal einen Speziallehrgang für Fernfahrprofis besuchen[1]. Bezogen auf die Datenbankprogrammierung sind die ADOs der hochmoderne Truck, die DAOs der Kleintransporter, der stärker auf die Bedürfnisse der klassischen PC-Datenbankprogrammierer ausgerichtet ist.

Eine der wenigen Alternativen zur Jet-Engine und den damit einhergehenden DAO- und ADO-Schnittstellen ist *CodeBase* von der kanadischen Firma *Sequiter* (*www.sequiter.com*), das hohe Performance bei geringem Platzbedarf verspricht und sich im Grunde als »ideale Lösung« darstellt. Hier gibt es keine Objekte (wenngleich ein Zugriff alternativ über ODBC und OLE DB und damit über DAO und ADO möglich ist), statt dessen sieht die Programmierung so aus, wie es einige ältere Datenbankprogrammierer noch aus der xBase- und Clipper-Ära gewohnt sind. Allerdings ist CodeBase nicht ganz billig und erfordert letztendlich eine Programmierung, die mit der modernen Visual-Basic-Programmierung nicht viel gemeinsam hat. Wer sich auf eine Speziallösung einläßt, hat es später einmal schwer, diese auf ein anderes System zu übertragen.

Das bekannte Fazit lautet wieder einmal: Der Fortschritt ist nicht aufzuhalten und wer nicht technologisch auf ein Abstellgleis fahren will, muß sich anpassen. Die Radikallösung, nämlich die völlig Abkehr von einer Microsoft-Datenbankschnittstelle, kommt bei Visual Basic für die meisten Programmierer nicht in Frage. Zum einen gibt es kaum Alternativen zur Jet-Engine, DAO und ADO, zum anderen koppelt man sich ebenfalls vom Zug der Entwicklung ab und kann dann gleich bei Visual Basic 3.0 bleiben. Die letzte Alternative wäre der Einsatz eines ganz anderen Entwicklungssystems, doch soll diese Variante aus begreiflichen Gründen in diesem Buch nicht näher diskutiert werden[2], zumal dort die gleiche Problematik, diesmal unter anderem Vorzeichen, wieder auftauchen dürfte.

Die ganze Betrachtung soll Sie als hoffnungsfrohen Neuling bei der Datenbankprogrammierung natürlich nicht entmutigen oder gar abschrecken. Sie soll lediglich die Entscheidung pro ADO ein wenig illustrieren und deutlich machen, daß es im Leben im allgemeinen und in der Software-Entwicklung

---

1 Ganz so kraß ist ADO natürlich nicht, es sind auch mit ADO kleine und überschaubare Programme möglich – die Beispielprogramme in diesem Buch beweisen es.

2 Nicht, daß dies ein Tabuthema wäre, doch muß eine solche Entscheidung jeder nach sorgfältiger Abwägung der Vor- und Nachteile für sich selber treffen. Eine allgemeine Empfehlung ist praktisch unmöglich.

im speziellen nur selten eindeutige Verhältnisse gibt. Die Wirklichkeit ist äußerst vielschichtig.

## An wen richtet sich das Buch?

Das Buch richtet sich an erfahrene Visual-Basic-Programmierer, die aber bislang mit der Datenbankprogrammierung entweder keinen Kontakt hatten oder die neuen ADO-Objekte noch nicht kennen. Gute Grundkenntnisse in der Visual-Basic- bzw. VBA-Programmierung werden vorausgesetzt. Einen Eindruck soll der Titel nämlich nicht erwecken: Dies ist **kein** Buch für Programmiereinsteiger. Wer also gerade erst mit der Programmierung begonnen hat oder vielleicht sogar erst davor steht, sollte sich zunächst einmal in Ruhe mit den Programmiergrundlagen beschäftigen (dafür empfehle ich ganz uneigennützig mein Buch »Jetzt lerne ich Visual Basic«, ebenfalls erschienen bei Markt&Technik – selbstverständlich gibt es noch viele andere sehr gute Bücher zu diesem Thema, wobei auch das Handbuch gelobt werden muß). Das Buch setzt voraus, daß Sie wissen, wie Visual Basic »funktioniert«, die wichtigsten Sprachelemente von VBA kennen und vor allem mit dem Prinzip der Objekte (etwa der Möglichkeit, über den Menübefehl EXTRAS/VERWEISE eine Typenbibliothek einbinden zu können) vertraut sind. Wer diese Grundkenntnisse nicht besitzt, wird manches in dem Buch nur schwer nachvollziehen können. Hier ein kleiner Schnelltest nach dem Motto »Wie gut kenne ich Visual Basic?«. Welches Objekt wird durch den folgenden Ausdruck angesprochen:

```
Anwendung.FensterListe("Hauptfenster")._
UnterfensterListe("Unterfenster1").Titel?
```

Klar, dieser Ausdruck spricht (vermutlich) ein *Unterfenster*-Objekt an, genauer dessen *Titel*-Eigenschaft[1].

Eines muß man als angehender Datenbankprogrammierer wissen: **Es gibt keine einfache Datenbankprogrammierung**. Visual Basic 6.0 bietet bereits eine Menge Komfort, kommende Versionen von Visual Basic werden diesen Komfort mit Sicherheit weiter steigern. Doch von diesem Komfort profitieren in erster Linie erfahrene Datenbankprogrammierer, die sich viele Routineschritte sparen können. Wer nicht weiß, wozu die neuen Tools da sind, und die einzelnen Begriffe nicht einordnen kann, wird den neuen Komfort auch nicht nutzen können. Um es einmal etwas harsch zu formulieren: Wer nicht weiß, was ein Recordset oder eine ODBC-Datenquelle ist,

1 Hundertprozentig läßt sich das aus dem Ausdruck allerdings nicht ableiten. Es ist lediglich eine vernünftige Annahme, daß eine Auflistung mit dem Namen `UnterfensterListe` auch Unterfenster-Objekte enthält.

kann zwar den Datenumgebungsdesigner nach Anleitung bedienen und eine Datenbankanbindung herstellen. Er oder sie wird jedoch vermutlich nicht verstehen, welche Arbeit ihm oder ihr der Designer tatsächlich abnimmt und welche Nachteile durch den Komfort erkauft werden. (Komfort zieht – das ist eine Programmiererweisheit – stets gewisse Nachteile bei Performance und Flexibilität nach sich – beim Datenumgebungsdesigner sind es allerdings erstaunlich wenige Nachteile.)

Diese Warnung soll natürlich niemanden entmutigen. Im Gegenteil, verstehen Sie sie bitte als Herausforderung. Das vorliegende Buch soll Ihnen die erforderlichen Grundlagen vermitteln. Erwarten Sie bitte nur nicht, daß dies a) im Handumdrehen b) an einem Wochenende oder c) in 14 Tagen geschieht. Um ein guter Datenbankprogrammierer zu werden, können schon »ein paar Monate« vergehen.

Als Belohnung winkt ein universelles Grundwissen, das Sie auf vielfältige Weise einsetzen können. Es ist ein Wissen, das sich so schnell nicht ändern wird. Auch wenn es nicht den Anschein haben mag, so hat sich, was die Datenbankwelt angeht, in den letzten 10 bis 20 Jahren nicht allzu viel geändert.

## Wie ist das Buch aufgebaut?

Das Buch ist in zwölf hoffentlich übersichtliche Kapitel aufgeteilt, wobei jedes Kapitel einen anderen Aspekt der Datenbankprogrammierung mit Visual Basic behandelt. Es beginnt ganz einfach mit dem allgemeinen Aufbau einer Datenbank und endet bei fortgeschrittenen Themen, wie dem Anfertigen von Datenbankreports. Ein warnender Hinweis aber vorweg. Eine »Linearität« bei der Vorgehensweise wäre (wie in jedem Buch) wünschenswert, läßt sich aber leider bei diesem Thema nicht realisieren. Sie finden daher, vor allem in der Mitte des Buches, häufiger Themen, die erst in den folgenden Kapiteln ausführlicher erklärt werden. So ist es sinnvoll, im Kapitel über das ADO-Datensteuerelement (Kapitel 4) Codebeispiele mit den ADO-Objekten vergleichend gegenüberzustellen, die aber erst in Kapitel 5 vorgestellt werden. Sowohl in Kapitel 4 als auch in Kapitel 5 finden Sie kleine SQL-Beispiele, wobei SQL aber erst in Kapitel 7 eingeführt wird. Diese Struktur geht konform mit dem Aufbau eines Internet-Dokuments, wo Sie über Querverweise Ihre eigene Lesereihefolge bestimmen können. Während Sie im Internet nur mit der Maus klicken, müssen Sie bei diesem Buch ein wenig blättern. Der rote Faden ist jedoch klar: Mit jedem Abschnitt lernen Sie etwas mehr über die Datenbankprogrammierung, so daß Sie am Ende des Buches mit allen wichtigen Aspekten vertraut sind.

## Was lesen Sie in diesem Buch?

In diesem Buch erhalten Sie einen Überblick über die Datenbankprogrammierung mit Visual Basic und den *Active Data Objects* (ADO). Diese Kenntnisse können Sie problemlos auch mit VBA und Office 97 bzw. vor allem Office 2000 einsetzen, wenngleich auf diesen Aspekt in diesem Buch nur am Rande eingegangen wird.

### Kapitel 1: Das kleine Einmaleins der Visual-Basic-Datenbanken

Sie lernen in diesem Kapitel erst einmal die wichtigsten Grundbegriffe aus der Sicht eines Visual-Basic-Programmierers kennen und erfahren, welche Form der Datenbankschnittstelle Microsoft für Visual Basic 6.0 gewählt hat.

### Kapitel 2: Datenbankdesign für Visual-Basic-Programmierer

Eine Datenbank besteht aus Tabellen, Feldern, Relationen, Abfragen, Schlüsseln und einigem mehr. Wie diese Begriffe zusammenhängen, wird am Beispiel der Buchdatenbank *Fuhrpark.mdb* erklärt.

### Kapitel 3: Die erste Datenbank im »Eigenbau«

In diesem Kapitel legen Sie mit Hilfe des Visual Data Managers, einem kleinen Visual-Basic-Hilfsprogramm, Ihre erste Datenbank an, die in den folgenden Kapiteln als Grundlage für alle Beispiele verwendet wird.

### Kapitel 4: Das ADO-Datensteuerelement und die gebundenen Steuerelemente

In diesem Kapitel lernen Sie das ADO-Datensteuerelement und die gebundenen Steuerelemente kennen, mit denen sich in wenigen Schritten »Eingabemasken« für das Ansehen und Bearbeiten von Datenbankinhalten erstellen lassen.

### Kapitel 5: Die ADO-Objekte stellen sich vor

In diesem Kapitel geht es an den Kern der Datenbankprogrammierung mit Visual Basic. Sie lernen die *Active Data Objects* (ADO) im Detail kennen, auf denen die gesamte Datenbankprogrammierung basiert.

### Kapitel 6: Der Datenumgebungsdesigner

Mit Hilfe des Datenumgebungsdesigners lassen sich *Connection*- und *Command*-Objekte einmal anlegen, so daß sie bei künftigen Projekten nur noch in Form einer Designerdatei eingefügt werden müssen. Der Umgang

mit dem Datenumgebungsdesigner, eine der wichtigsten Neuerungen bei Visual Basic 6.0, steht in diesem Kapitel im Vordergrund.

**Kapitel 7: Das ABC der Datenbanksprache SQL**

In diesem Kapitel lernen Sie das ABC der Datenbanksprache SQL kennen. Die Beispiele werden auf die in Kapitel 3 vorgestellte Fuhrpark-Datenbank angewendet.

**Kapitel 8: Der Umgang mit dem Recordset-Objekt**

Im Mittelpunkt aller Datenbankzugriffe mit ADO steht das *Recordset*-Objekt. Möchten Sie Datensätze löschen, aktualisieren oder nach ihnen suchen? Das *Recordset*-Objekt ist der Schüssel dazu.

**Kapitel 9: Die Dateneingabe in der Praxis**

Neben einer Datenbank und den ADO-Objekten muß eine Datenbankanwendung auch Eingabedialogfelder enthalten. Dazu gehört z.B. das Suchen in einer Datenbank, das mit Hilfe von Eingabeformularen und per SQL oder mit der *Find*-Methode des *Recordset*-Objekts *Datensätze* erledigt wird.

**Kapitel 10: ADO für (etwas) Fortgeschrittene**

In diesem Kapitel geht es um die etwas fortgeschritteneren Themen der Programmierung mit ADO. Dazu gehört unter anderem das Einlesen einer Datensatzgruppe in ein Feld, das Abspeichern einer Datensatzgruppe oder das Einlesen einer Tabellenstruktur.

**Kapitel 11: Anfertigen von Datenbankreports**

Irgendwann (meistens ziemlich schnell) möchte man die Daten nicht nur am Bildschirm sehen, sondern Schwarz auf Weiß in der Hand halten. Wie sich Datenbankreports sozusagen in Schönschrift mit dem Visual-Basic-Datenbankreportmodul anfertigen lassen, verrät dieses Kapitel.

**Kapitel 12: Eine Datenbankanwendung in der Gesamtübersicht**

Im letzten Kapitel des Buches wird die Fuhrpark-Anwendung, die sich wie ein roter Faden durch das Buch zieht, noch einmal im Gesamtüberblick vorgestellt. Dabei werden die einzelnen Schritte zusammengefaßt, die zum Erstellen einer »vollständigen« Datenbankanwendung notwendig sind.

Nicht weniger informativ sind wie immer die Anhänge.

### Anhang A: Datenbanken von A bis Z

In der Datenbankprogrammierung wimmelt es von Abkürzungen und Begriffen. In diesem Anhang finden Sie eine fast vollständige Übersicht.

### Anhang B: ADO-Referenz

In diesem Anhang werden die ADO-Objekte mit ihren Eigenschaften, Methoden und Ereignissen zusammengefaßt.

### Anhang C: Die Data Access Objects (DAO)

In diesem Anhang werden die Data Access Objects (DAOs) vorgestellt, die speziell dem Zugriff auf die Jet-Engine und Access-Datenbanken dienen. Anhand kleiner Beispiele werden die wichtigsten Datenbankoperationen veranschaulicht. Außerdem finden Sie hier eine direkte Gegenüberstellung zu den ADO-Objekten.

### Anhang D: Die Antworten zu Fragen

Am Ende eines jeden Kapitels werden eine Reihe von leichten und weniger leichten Fragen gestellt, deren Antworten Sie in diesem Anhang finden.

### Anhang E: Weiterführende Informationen

Dieses Buch kann nur die Grundlagen behandeln. Wer tiefer in die Datenbankprogrammierung mit Visual Basic, ADO oder SQL Server einsteigen will, findet in diesem Anhang ein paar Tips zu »weiterführender Literatur«.

## Welche Visual-Basic-Version wird vorausgesetzt?

Dieses Buch setzt Visual Basic ab Version 6.0 voraus[1]. Die meisten Beispiele (aber nicht alle) lassen sich auch mit dem kostenlos erhältlichen »Ablaufmodell« von Visual Basic 6.0 umsetzen[2]. Sie finden das Ablaufmodell bei verschiedenen Gelegenheiten (z.B. auf Zeitschriften-CDs), aber nicht als Download im Internet. Wer sich ernsthaft mit der Datenbankprogrammierung beschäftigen möchte, sollte sich die Profi-Edition von Visual Basic 6.0 zulegen. Die Enterprise-Edition ist nicht zwingend erforderlich, da die zusätzlichen Features, wie etwa der SQL-Server 6.5, der T-SQL-Debugger und andere Tools, für das Lernen von Visual Basic und bei »normalen« Da-

1 Sollte es, wenn Sie dieses Buch lesen, bereits ein »Visual Basic 7« geben, dürfte es bezüglich der elementaren ADO-Objekte und des Datenumgebungsdesigners keine Unterschiede geben.

2 Ich habe damit viele Beispielprogramme des Buches entwickelt.

tenbankanwendungen keine Rolle spielten dürften. Wer nur Visual Basic 5.0 besitzt, kann einen Teil der Beispiele umsetzen, muß dabei aber immer bedenken, daß es den Datenumgebungsdesigner, der in vielen Beispielen vorausgesetzt wird, bei Version 5.0 nicht gibt. Wie sich der Datenumgebungsdesigner durch entsprechende ADO-Befehle ersetzen läßt, wird in Kapitel 5 beschrieben. Visual Basic 5.0-Programmierer benötigen die ADO-Objekte, die sich im Rahmen des *Microsoft Data Access Components-Pack* (MSDAC) unter *http://www.microsoft.com/data/mdac2.htm* herunter laden lassen. (Laden Sie nicht das SDK, sondern lediglich das »Microsoft Data Access Components 2.0 Typical Setup« – der genaue Wortlaut kann variieren – mit ca. 6 Mbyte Umfang herunter.) Wer noch unter Visual Basic 5.0 programmiert und dennoch nicht auf ein ADO-Datensteuerelement verzichten möchte, kann es als Freeware von der Firma *IsgSoft* unter *http://www.isgsoft.com/products/ISGData/* herunterladen. Eine Zusammenstellung wichtiger Ressourcen für angehende Datenbankprogrammierer finden Sie in Anhang E.

## Gibt es diesmal keine Buch-CD-ROM?

Autor und Verlag haben sich entschlossen, diesem Buch keine CD-ROM beizulegen, da es abgesehen von den relativ kleinen Beispielprogrammen und der Fuhrpark-Datenbank nicht allzu viel beizulegen gäbe. Ich gehe davon aus, daß jeder Leser über Visual Basic 6.0 (oder Version 5.0) verfügt. Die Beispielprogramme des Buches sind so gehalten, daß sich durch das Abtippen und Ausprobieren ein Lerneffekt ergibt, der gerade bei der Datenbankprogrammierung von großer Bedeutung ist. Auch die Datenbank *Fuhrpark.mdb*, die in Kapitel 3 vorgestellt wird, sollten Sie Schritt für Schritt umsetzen und nicht einfach von einer CD-ROM kopieren (die in diesem Fall nicht existiert). Wer sich diese Arbeit dennoch ersparen möchte oder partout nicht weiterkommt, kann sich die Datenbank *Fuhrpark.mdb* sowie einige der Beispielprogramme des Buches von meiner Webseite herunterladen. Ich möchte jedoch an dieser Stelle jeden ermutigen, es zunächst auf eigene Faust zu probieren. Auch, wenn ein Beispiel auf Anhieb nicht funktionieren sollte (das kommt schon einmal vor), der Lerneffekt ist einfach größer.

## Brauche ich einen Internet-Anschluß?

Grundsätzlich natürlich nicht, doch Sie werden schnell feststellen, daß der Internet-Anschluß viele praktische Vorteile bietet. Sie können sich über die neuesten Entwicklungen bei Microsoft informieren, mit anderen Entwicklern Tips und Erfahrungen in verschiedenen Newsgroups austauschen, die

Webseite des Autors besuchen und sich den Straßenbahnfahrplan der Fidschi-Inseln ausdrucken lassen. Selbst wer keinen Internet-Anschluß besitzt, kann in unzähligen Internet-Cafés (z.B. in einem Kaufhaus) etwa die Beispielprogramme des Buches herunterladen oder mehr über die ADO-Programmierung lesen. Ein Internet-Anschluß lohnt sich in jedem Fall und sollte für Visual-Basic-Entwickler selbstverständlich sein.

## Kontakt mit dem Autor

Wie immer freue ich mich über Zuschriften, Kommentare, Anregungen (darüber besonders), aber natürlich auch über Kritik, Hinweise auf Fehler, Geldspenden usw. Ich gehe davon aus, daß nicht alles auf Anhieb klar sein wird und sich sicherlich auch das eine oder andere »Fehlerchen« eingeschlichen haben wird. Aktuelle Fehlerkorrekturen sowie die Beispielprogramme und die Datenbank des Buches finden Sie auf meiner Webseite mit der Adresse:

*www.activetraining.de/jldb.htm*

Die Endung »jldb« steht übrigens für »Jetzt lerne ich Datenbankprogrammierung«, was das Erinnern der URL ein wenig erleichtern dürfte. Meine E-Mail-Adresse lautet:

*vba@activetraining.de*

Am Ende dieser Einleitung möchte ich Ihnen viel Spaß und Erfolg beim Einstieg in die Datenbankprogrammierung mit Visual Basic und den *Active Data Objects* (ADOs) wünschen. Es wird nicht immer ganz leicht sein, doch es lohnt sich in jedem Fall, denn die Datenbankprogrammierung unter Visual Basic bzw. Office wird auch im Jahr 2000 (also im nächsten Jahrtausend) zu den mit Abstand wichtigsten Themen in der Windows-Welt gehören.

Viel Spaß beim Entdecken der vielen Möglichkeiten wünscht Ihnen

*Peter Monadjemi*