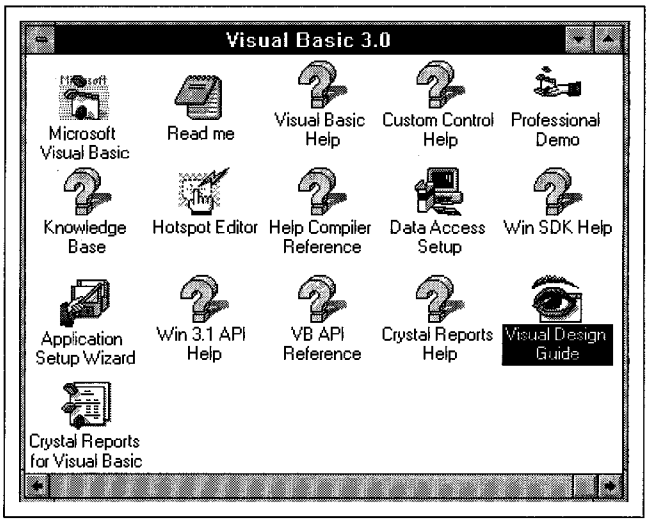


"A fast food job lasts from bun to bun,
But a programming job is never done."

John Shore, The Sachertorte Algorithm and other Antidotes to Computer Anxiety, New York 1986: Penguin Books, S. 168)

Programmieren für Juristen: Visual Basic – Lektion II

Maximilian Herberger

Diesmal: Datenbankprogrammierung

Zweckmäßig konzipierte juristische Anwendungen werden immer einen Datenbankschwerpunkt haben. Man muß dabei nicht nur an die "klassische" Urteilsdatenbank denken. Auch Lernprogramme beispielsweise werden den Lernstoff in Datenbankform ablegen. Deswegen widmet sich diese zweite Lektion nach der ersten elementaren Einführung in Visual Basic der Datenbankprogrammierung. Es wird sich dabei zeigen – dies sei vorweggenommen – daß hier eine der Stärken von Visual Basic Professional liegt.

"Recycling" von dBASE-Datenbanken

Zugrundegelegt wird für die erste Programmiersequenz eine im juristischen Umfeld immer noch häufig anzutreffende Situation: Vorhanden sind dBASE-Datenbanken, die man unter Windows mit einer ansprechenden Oberfläche versehen will. Außerdem möchte man die so entstehende Anwendung weitergeben, ohne daß der Nutzer seinerseits auch dBASE benötigt.

Die dBASE-Datenbank sei eine Urteilsdatenbank genannt URTEILE.DBE Die Leitsätze befinden sich in einem Memo-Feld, weswegen es auch noch eine Datei URTEILE.DBT gibt. Über das Feld mit der Gerichtsbezeichnung sei ein Index gelegt, der in der Datei GERICHTE.NDX enthalten ist.

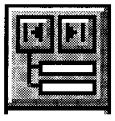


Abb. 1: Das "Werkzeug"-Symbol für die Data-Control

Datenbankzugriff über die Data-Control

Die Arbeit beginnt – wie in Folge 1 besprochen – mit einem leeren Formular. Darauf plazieren wir zunächst eine Textbox, in der die Gerichtsnamen erscheinen sollen. Für den Feldnamen sehen wir ein Label vor. Hinzu kommt die Data-Control *(vgl. Abb. 1)*. Mit diesen drei Elementen ist (nach einigen, sogleich zu erläuternden Property-Einstellungen) bereits eine rudimentäre Datenbank-Anwendung vorhanden.

Beginnen wir mit den Property-Einstellungen bei der Data-Control. Nach dem Beigeben eines sprechenden Textes für "Caption" (etwa "Blättern in der Datenbank") ist die Data-Control zunächst mit einer Datenbank zu verbinden *(vgl. Abb. 2)*. Zu diesem Zweck trägt man unter "Connect" den Typ der Datenbank ein.

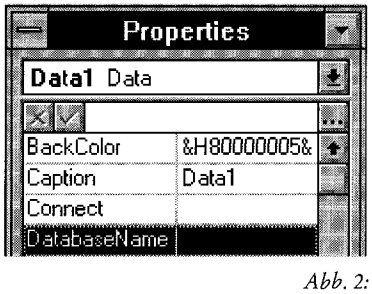
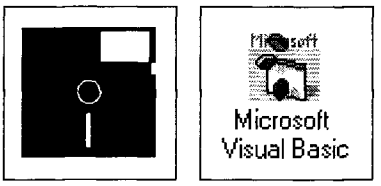


Abb. 2: Properties für das "Binden" der Data-Control

Für dBASE III+ lautet die Eintragung "dbase III;". (Im Falle einer Access-Datenbank – dazu im weiteren Verlauf des Kurses mehr – ist kein "Connect"-Eintrag erforderlich.) Unter "DatabaseName" ist hier im Falle einer dBASE-Datenbank (etwas kontraintuitiv) der Pfad einzugeben, in dem sich die Datenbank befindet. Der Datenbankname wird weiter unten in der Properties-Liste unter "RecordSource" eingegeben. Dabei kann man sich bereits des Auswahlknopfes (Pfeil nach unten) im Kopf der Properties-Liste bedienen. Auf Grund der Voreinstellungen unter "Connect" und "DatabaseName" werden die im angegebenen Verzeichnis vorhandenen dBASE-Datenbanken zur Auswahl angeboten.



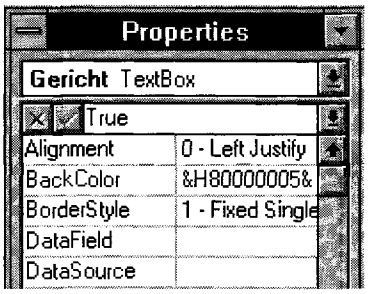


Abb. 3:
Verbindung Text-Feld – Datenbank-Feld

Die so an die Datenbank "gebundene" Data-Control würde beim Starten des Programms bereits auf die Datenbank zugreifen, auch Blättern könnte man in der Datenbank – nur sehen würde man davon nichts. Also bedarf es noch einer Anzeige. Diese soll im Textfeld "Gericht" stattfinden. Erreicht wird das mit den Eigenschaften "DataSource" und "DataField" *(vgl. Abb. 3)*.
In beiden Fällen kann man sich des "Auswahlpfeils" bedienen. Unter "DataSource" zeigt er "Data1" an, d.h. den Namen der auf dem Formular plazierten Data-Control. Unter "DataSource" erscheinen die Felder der verbundenen dBASE-Datenbank zur Auswahl. Wir wählen "Gericht".

"No installable ISAM"
Nun könnte man das Programm starten und auch in der Datenbank blättern, wenn da nicht möglicherweise die Fehlermeldung "No installable ISAM" zu erwarten wäre. (ISAM steht für "Index Sequential Access Method", eine Art des Datenbankzugriffs, die durch die jeweils eingesetzte DLL realisiert wird.) Wenn das beim Start des Programms aus Visual Basic heraus im "interpretierenden" Modus der Fall ist, sollte man die Datei VB.INI im Windows-Verzeichnis einer Inspektion unterziehen. Wenn dort unter der Rubrik "Installable ISAMs" der Eintrag für dBASE III (und die entsprechende DLL) fehlt, erscheint die erwähnte Fehlermeldung.
Der folgende Auszug aus der VB.INI zeigt, welche ISAM-Datenbank-Treiber installierbar sind:

```
[Installable ISAMs]
Btrieve=C:\WINDOWS\SYSTEM\btrvll0.dll
FoxPro 2.0=C:\WINDOWS\SYSTEM\xbsll0.dll
FoxPro 2.5=C:\WINDOWS\SYSTEM\xbsll0.dll
dBASE III=C:\WINDOWS\SYSTEM\xbs200.dll
dBASE IV=C:\WINDOWS\SYSTEM\xbs200.dll
Paradox 3.X=C:\WINDOWS\SYSTEM\pdxll0.dll

[dBase ISAM]
Deleted=On
```

Um die Fehlermeldung "No installable ISAM" beim Aufruf der kompilierten Programmversion zu vermeiden, muß man dafür sorgen, daß im Windows-Verzeichnis eine INI-Datei für das kompilierte Programm mit folgendem Inhalt vorhanden ist:

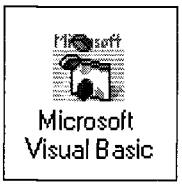
```
[Installable ISAMs]
dBASE III=C:\WINDOWS\SYSTEM\xbs200.dll
[dBASE ISAM]
Deleted=On
```

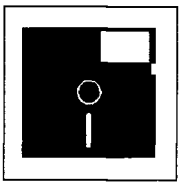
Einbeziehen der Index-Datei
Aus der dBASE-Umgebung fehlt jetzt noch die Einbeziehung der Index-Datei. Zu diesem Zweck ist in dem Verzeichnis, in dem sich die Datenbank befindet, eine Datei einzurichten, die (bis auf die Extension) namensgleich mit der .DBF-Datei ist und die Extension .INF erhält. In unserem Falle heißt diese Info-Datei also URTEILE.INF. Darin ist folgender Eintrag vorzunehmen:

```
[dbase]
NDX1=gerichte.ndx
```

Mit NDX2=, NDX3= usw. können weitere Index-Dateien in Bezug genommen werden.
Zu achten ist darauf, daß der Index aktuell sein muß. Bildet er einen früheren Zustand der Datenbank ab, so ergeben sich unerwartete Ergebnisse.

Memo-Felder: (Fast) Kein Problem
Wer sich aus alten dBASE-Zeiten noch an die mit Memo-Feldern verbundenen Unwägbarkeiten erinnert, wird sich gespannt an die nächste Übung machen: Das Hinzufügen eines Textfeldes, in dem die Leitsätze aus dem Memo-Feld der Urteilsdatenbank angezeigt werden sollen. Wir plazieren zu diesem Zweck ein weiteres Textfeld auf dem Formular, nennen





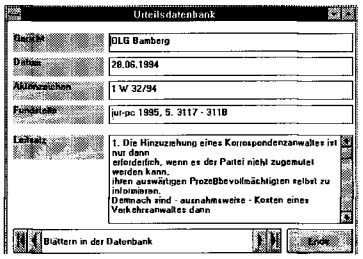


Abb. 4:
Das Basisformular "Urteilsdatenbank"

es Leitsatz (Property "Name"), setzen die Property "Text" auf leer, damit nicht "Text2" in dem Feld steht, fügen eine vertikale Scrollbar hinzu (Property "ScrollBars" auf "2 – vertical"), weisen der Property "Multiline" den Wert "true" zu, damit ein Zeilenumbruch stattfindet und binden dann wie im eben beschriebenen Fall des Textfeldes "Gericht" das Textfeld "Leitsatz" an das Datenfeld "Leitsatz": Nach dem Start des Programms erscheint der Leitsatztext im dafür vorgesehenen Feld *(vgl. Abb. 4)*. Das in Abb. 4 zu sehende Formular ist ergänzend zum bisher beschriebenen noch um einige Felder ergänzt worden. Das Hinzufügen dieser Felder erfordert keine zusätzlichen Kenntnisse über das bisher Dargestellte hinaus.
Hinter dem "Ende"-Button schließlich verbirgt sich für das Ereignis "Click" der Code für das Schließen der Datenbank und das Verlassen des Programms in folgender Form:

```
Sub Command1_Click ()
    Data1.Recordset.Close
    End
End Sub
```

"Recordset" ist die über die Data-Control mit Namen Data1 angesprochene dBASE-Datenbank. Eine Sequenz wie "Data1.Recordset.Close" ist zu lesen wie: *"Nimm den über die Data-Control mit Namen Data1 angebundenen Recordset (= dBASE-Datenbank) und schließe ihn"*.
Beim Schließen der Datenbank werden noch zum Vollzug anstehende Änderungen zurückgeschrieben (d.h. die im aktuellen Datensatz), außerdem wird der Index aktualisiert.
Ein kleines Problem wird der dBASE-erfahrene Programmierer in Abb. 4 sofort erkennen: Die "harten" Zeilenenden aus dem Inhalt des Memo-Feldes bleiben erhalten. Will man dieses unschöne Aussehen vermeiden, muß man den Inhalt des Memo-Feldes vor der Ausgabe in das Textfeld bearbeiten. Wie ein solcher Zugriff aussehen kann, läßt sich (leider) erst im weiteren Kursverlauf didaktisch sinnvoll beschreiben.
Der Zugriff auf Memo-Felder bringt u.U. noch ein weiteres Problem mit sich. Bekanntlich sind in nicht reorganisierten .DBT-Dateien bei mehrfach veränderten Memo-Feldern alle Änderungszustände gespeichert. Der hier beschriebene Zugriff fördert u.U. nicht den letzten Änderungsstand zutage. Man sollte deshalb die .DBT-Dateien reorganisieren, um dieses – wohl als Visual Basic-Bug einzustufende – Problem zu vermeiden.

Abb. 5:
"Werkzeugsymbol" für Label

Ändern von Datensätzen
Die im Formular angezeigten Daten können bearbeitet werden. Ergänzungen und Änderungen sind damit bereits jetzt möglich. Gespeichert werden derartige Überarbeitungen automatisch beim Wechsel von Datensatz zu Datensatz, ohne daß es dazu noch weiterer Vorkehrungen bedarf. Auch beim Verlassen des Programms mit dem oben für den "Ende"-Button angegebenen Code werden noch anstehende Änderungen abgespeichert. Die geänderte Datenbank ist übrigens "rückwärtskompatibel": Nach Änderungen unter Visual Basic konnte sie in dBASE weiterbearbeitet werden. Trotzdem sollte man sich darauf nie verlassen und eine "Nicht Visual Basic"-Datenbank vor der Bearbeitung mit Visual Basic anderwärts sichern.
Möglicherweise will man bei der Änderbarkeit der Daten differenzieren und nicht alle zur Änderung freigeben. Ein derartiger Änderungsschutz ist leicht dadurch einzurichten, daß man für die Anzeige statt der Textbox ein Label *(vgl. Abb. 5)* wählt.
Dieses kann über die Eigenschaften "DataSource" und "DataField" genau so wie eine Textbox an die Datenbank gebunden werden. Ein denkbarer Anwendungsfall wäre etwa die Anzeige einer ID-Nummer für das Dokument, die im Regelfall änderungsstabil sein soll.

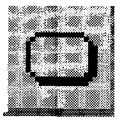
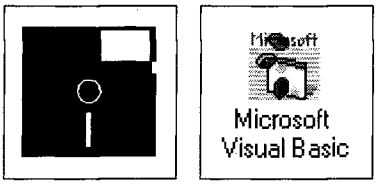


Abb. 6:
"Werkzeugsymbol" für Befehlsbutton

Neue Datensätze hinzufügen
Ein Datenbankprogramm wäre unvollständig ohne die Möglichkeit, neue Datensätze zu erfassen. Diese Erweiterung ist leicht zu bewerkstelligen. Wir fügen einen Befehlsbutton *(vgl. Abb. 6)* "Neuer Datensatz" mit folgendem Code für das Ereignis "Click" hinzu:

```
Sub Command2_Click ()
    Data1.Recordset.AddNew
End Sub
```



Das ist alles. Abgespeichert wird der neue Datensatz beim Blättern zu einem anderen Datensatz oder beim Verlassen des Programms mit "Ende". Diese einfache Ergänzungsmöglichkeit beruht darauf, daß die Textfelder an Datenbankfelder "gebunden" sind, neue Inhalte der Textfelder mithin zu einer neuen Belegung der Datenbankfelder führen.
Nach den mittlerweile angebrachten Verfeinerungen hat das Formular das aus Abb. 7 ersichtliche Aussehen.
Mit der ID-Nummer wird das Darstellen eines nicht für die Veränderung freigegebenen Feldes aus der Datenbank demonstriert, der Knopf "Neuer Datensatz" erlaubt das Hinzufügen von Datensätzen. (Und das Memo-Feld ist durch Editieren "händisch" um die störenden Zeilenumbrüche bereinigt worden.)

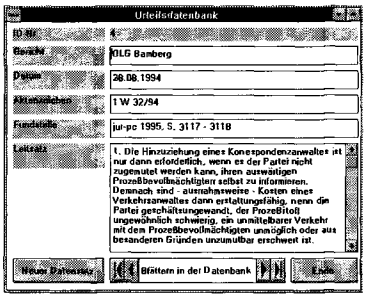


Abb. 7:
Formular mit Knopf für neue Datensätze

Volltext-Anbindung
Um die aus Visual Basic heraus gegebenen Integrationsmöglichkeiten an einem ersten datenbankbezogenen Beispiel zu demonstrieren sei angenommen, daß die Urteilsdatenbank in einem Feld (genannt "Datei") den Namen der Datei mit dem Volltext des Urteils mitführt. Dieser Volltext soll mit Hilfe des Programms "NOTEPAD" angezeigt werden.
Zum genannten Zweck muß ein neuer Befehlsbutton eingerichtet werden, nennen wir ihn "Volltext", dem folgender Programmcode zu hinterlegen ist:

```
Sub Command3_Click ()
    datei = Datal.Recordset.Fields("Datei")
    i = Shell("notepad " + datei, 3)
End Sub
```

Erläuterungsbedürftig ist zunächst die Sequenz

```
Datal.Recordset.Fields("Datei")
```

Diese Sequenz liefert aus dem durch die Data-Control mit Namen Data1 angebundenen Recordset (= dBASE-Datei) den Inhalt des Feldes "Datei" (in dem der Name der Datei mit dem Urteilsvolltext samt Pfad steht). Dieser Eintrag wird der Variablen *Datei* zugewiesen.
Die dann folgende Zeile

```
i = Shell("notepad " + datei, 3)
```

ruft die Funktion "Shell" auf, die ihrerseits den Aufruf eines externen Programms bewerkstelligt. Der erste übergebene Parameter ist der Name des externen Programms (hier: Notepad) zusammen mit dem vorher ermittelten Dateinamen. Der zweite Parameter (hier: 3) bestimmt, wie das Fenster aussehen soll, in dem das aufgerufene Programm abläuft. "3" steht für "maximized with focus". All das kann man übrigens, wie bereits früher erwähnt, der Hilfe entnehmen.

"Logische Teildatenbanken"
Der Benutzer von Datenbanken mit heterogenem Material wird meist den naheliegenden Wunsch haben, gezielt auf eine Untermenge zuzugreifen. Es handelt sich dabei (einem terminologischen Vorschlag von *Berkemann* folgend) um "logische Teildatenbanken". In der dBASE-Terminologie würde man von "Filtern" sprechen. Wir wollen hier, bezogen auf das Merkmal "Gericht", einen solchen Filter implementieren. Zu diesem Zweck plazieren wir eine sogenannte Combo-Box auf dem Formular *(vgl. Abb. 8)*, und zwar vor dem Textfeld "Gericht" (anstelle des bisher dort vorhandenen Labels).
Eine Combo-Box stellt eine aufklappbare Liste zur Verfügung. Damit diese Liste einen Inhalt hat, muß sie aufgefüllt werden. Der geeignete Moment dafür ist das Laden des Formulars. Anders ausgedrückt: Man muß das Ereignis "Form Load" ausnützen. Nach Doppelklick auf das Formular öffnet sich das entsprechende Code-Fenster, das man gemäß Abb. 9 ausfüllen kann.
In der Combo-Box stehen jetzt beispielhaft die Auswahlmöglichkeiten "BGH", "OLG", "LG" und "AG" zur Verfügung. Aktiviert der Benutzer eine dieser Eintragungen in der Absicht, die logische Teildatenbank "BGH" etc. zu bilden, so muß in der Konsequenz ein Ausblenden der nicht zu dieser Untermenge gehörenden Datensätze stattfinden. Die ad-

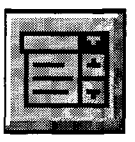


Abb. 8:
Werkzeug-Symbol für die Combo-Box

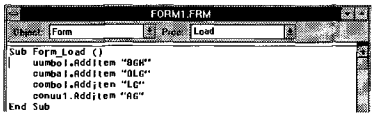
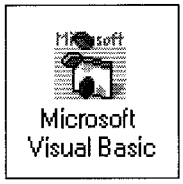
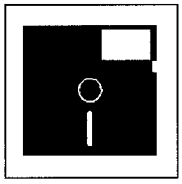


Abb. 9:
Auffüllen der Combo-Box beim Laden des Formulars





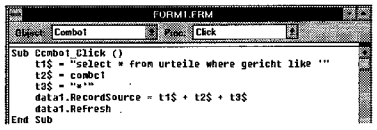


Abb. 10:
Code für einen Datenbank-Filter

äquate Schaltstelle für diesen Vorgang ist das Ereignis "Click" auf die entsprechende Eintragung der Combo-Box. Wir hinterlegen also dieses Ereignis mit dem aus Abb. 10 ersichtlichen Code.
Der Code zu "Combo1 – Click" generiert einen SQL-Befehl. "SQL" steht für *"Structured Query Language"*. Es handelt sich dabei um einen de-facto-Standard zur Abfrage von Datenbanken mit 'geschichteter' Tabellenstruktur. Visual Basic implementiert diese Suchsprache. Einzelheiten findet man in der Hilfe über die Eingabe "SQL".
Der unter "Combo1 – Click" generierte SQL-Befehl lautet beispielsweise für die Auswahl "OLG"

```
select * from urteile where gericht = 'OLG*'
```

Dieser Befehl bedeutet in (annähernd) umgangssprachlicher Übersetzung:
"Wähle alle Datensätze aus der Datenbank 'Urteile' aus, die der Bedingung genügen, daß das Feld 'Gericht' mit der Zeichenfolge 'OLG' beginnt."
Entscheidend für die Umsetzung dieser Direktive ist dann die Zeile, in der der so gebildete SQL-String "Data1.RecordSource" zugewiesen wird. Erinnern wir uns: "RecordSource" ist eine Eigenschaft ("Property") der Data-Control "Data1". Wir hatten zu Beginn an dieser Stelle den Namen der dBASE-Datenbank eingegeben (also: Urteile). Nun akzeptiert Visual Basic an dieser Stelle nicht nur Datenbanknamen, sondern auch SQL-Befehle. Das kommentierte Code-Stück weist zur Laufzeit einen entsprechenden SQL-Befehl zu. Damit er wirksam werden kann, eröffnet "Data1.Refresh" die Datenbank neu, und zwar unter Beachtung des vorher darüber gelegten SQL-Filters.

Datenbank-Suche:
SQL-Variante
Was eben zur SQL-Syntax gesagt wurde, erlaubt auch eine spezielle ("SQL") Suchimplementierung. "Suche" ist danach nichts anderes als die Überlagerung der Datenbank durch ein in der SQL-Sprache ausgedrücktes Kriterium. In anderer Formulierung: Als Suchergebnis erscheint eine Teildatenbank. Methodisch ist diese Sicht der Dinge vollkommen adäquat: Jede Suche hat als Ergebnis eine Teildatenbank.

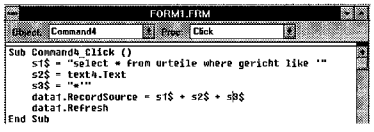


Abb. 11:
Suche per SQL-Filter

Wegen der Ähnlichkeit zum vorstehend behandelten Punkt "SQL-Filter" gestaltet sich die Implementierung der gleichartig strukturierten Suche sehr ähnlich. Wir legen zu diesem Zweck einen Befehlsbutton "Suche Gericht" an, der den Eintrag aus einem neuen Textfeld zur Grundlage einer SQL-Suche macht. Es geschieht dies gemäß dem aus Abb. 11 ersichtlichen Code, der angesichts der vorher im Detail erläuterten SQL-Filtermethode keiner gesonderten Erläuterung mehr bedarf.
So methodisch adäquat der eben behandelte Suchweg ist, so sehr ist er doch auch zeitkritisch, weil die Datenbank unter Berücksichtigung des Filters neu eröffnet (und durchlaufen) werden muß. Möglicherweise (dies ist jeweils zu erproben) schneiden im Vergleich dazu andere Such- und Darstellungsmethoden unter Geschwindigkeitsaspekten besser ab.