Notizen im strukturierten Karteikasten

Karel Štípek

Einleitung

Die Idee ist nicht ganz neu. In PCNEWS Nr. 53 (Jahr 1997) habe ich als Beispiel eines Datenmodels einen Access-Karteikasten präsentiert. Allerdings damals ohne Beschreibung der konkreten technischen Lösung. Inzwischen habe ich das Programm erweitert und verbessert und möchte diesmal einen Blick in das Innere gewähren. Die Motivation ist eigentlich immer die gleiche, die Sie dem nächsten Absatz entnehmen können.

Kommt es Ihnen auch bekannt vor? Sie lösen irgendein kleines Problem und widmen sich dann beruhigt anderen Aufgaben. Wenn Sie dann nach längerer Zeit vor einem ähnlichen Problem stehen, können Sie sich an die Lösung nicht mehr erinnern. Und Sie müssen wieder nachdenken und probieren. Ebenso mühsam erinnern Sie sich, in welcher Zeitschrift Sie einen bestimmten Trick gefunden haben oder welchen Artikel Sie später lesen möchten. Im besten Fall machen Sie sich schriftliche Notizen auf Zetteln, die Sie aber nicht immer finden.

Notizen effizient ablegen

Das vorgestellte Programm kann Ordnung in verschiedene kleine Notizen bringen. Damit die Informationen übersichtlich abgelegt werden, ist es notwendig, sie hierarchisch zu organisieren. Jedes Thema kann Unterthemen enthalten, die mehrmals weiter gegliedert werden können. In diesem Programm sind bis zu fünf Hierarchieebenen möglich. Eine Notiz wird meistens dem Titel der letzten Ebene zugeordnet, Sie können aber auch zu jedem Zwischentitel einen Text speichern.

Wenn etwas gespeichert wird, muss das auch wieder gefunden werden. Dazu ist eine Möglichkeit der Volltextsuche notwendig.

Wichtig ist nicht nur der Inhalt selbst, sondern auch einige Metainformationen, wie beispielweise das Datum, wann die jeweilige Notiz erstellt, bzw. geändert wurde. Diese Datumsangaben werden sowohl für jeden Eintrag extra, als auch für die gesamte Tabelle angezeigt. Das Datum des letzten Zuwachses, bzw. der letzen Änderung der Tabelle hilft auch zur Unterscheidung der Version der Datenbank. Access ändert nämlich das Datum und die Zeit der Datei immer beim Schließen automatisch.

Bedienung des Programms

Benutzeroberfläche

Nachdem Sie die Datenbank `Karteikasten.mdb` mit Access (ab der Version 2000) geöffnet haben, erscheint das Hauptformular. Das folgende Bild stellt die Anzeige einer konkreten Notiz mit der Themenzuordnung dar. Unmittelbar nach dem Programmstart werden eigentlich nur die Comboboxen zur Tabellenauswahl und zur Anzeige des Haupttitels eingeblendet. Sie können erst dann weiter arbeiten, wenn Sie eine Tabelle ausgewählt oder neu erstellt haben.

Abbildung 1: Hauptformular des Programms

Die wichtigsten Steuerelemente

Alle Funktionen des Programms werden in diesem Formular implementiert. Die Namen der wichtigsten Steuerelemente, die im Programmcode häufig referenziert werden, sind in der folgenden Tabelle aufgelistet.

Name	Bedeutung
`cboTabellen`	Auswahl der Tabelle
`cbo1, cbo2, ... cbo5`	Titel und Untertitel
`txtText`	Anzeige der Notiz
`txtSuchen`	Eingabe des Suchbegriffs

Tabelle 1: Die wichtigsten Steuerelemente im Hauptformular

Themenbereich entspricht einer Tabelle

Auch wenn die Strukturierung der Titel auf fünf Ebenen möglich ist, ist es sinnvoll, die Notizen des gleichen Themenbereiches in einer getrennten Tabelle zu speichern. Die Datenmodellierung der Titelstruktur in einer einzigen Tabelle wird weiter unten erklärt. Eine vorhandene Tabelle kann in der obersten Combobox ausgewählt werden. Rechts davon werden der letzte Zuwachs, die letzte Änderung und die Anzahl der Notizen angezeigt.

Durch die Eingabe eines neuen Namen kann nach Abfrage eine neue Tabelle erstellt werden. Das Löschen und Umbenennen von Tabellen ist im Hauptformular direkt nicht implementiert und kann im Datenbankfenster (Anzeige mit der Taste F11) erfolgen.

Titel und Untertitel

Nach der Auswahl einer Tabelle werden alle Titel der höchsten Ebene in der ersten Combobox angezeigt. Nachdem Sie eine Zeile ausgewählt haben, wird automatisch die nächste Combobox mit den untergeordneten Titeln eingeblendet. Dieser Vorgang kann wiederholt werden, solange es weitere Untertitel gibt. Wenn Sie am Ende der Hierarchie gelangen und noch nicht die maximale Anzahl der Ebenen erreicht haben, wird auch die nächste leere Combobox angezeigt, damit die Struktur weiter verfeinert werden kann.

Wenn Sie im einer beliebigen Combobox einen neuen Titel eingeben, wird er ohne Abfrage auf die richtige Position in der Hierarchie eingefügt.

Eingabe von Notizen

Zu dem Titel einer beliebigen Ebene kann im großen Textfeld unten eine Notiz eingegeben werden. In den meisten Fällen werden Sie den Text erst am Ende eines Hierarchiezweiges, also zu dem Titel der niedrigsten Ebene, speichern. Die eventuell noch angezeigte leere Combobox wird ausgeblendet, wenn das Notizfeld aktiv wird.

Die Änderung des Textes einer Notiz wird ohne Abfrage gespeichert, nachdem Sie das Textfeld verlassen haben. Wenn Sie Esc unmittelbar nach der Eingabe drücken, werden die letzten Änderungen rückgängig gemacht. Unter dem Notizfeld wird das Erstellungs-, bzw. auch das letzte Änderungsdatum angezeigt.

Text suchen

Im Programm ist eine Volltextsuche eingebaut. Wenn Sie den gesuchten Suchbegriff im weißen Textfeld eingeben und auf die Schaltfläche Suchen klicken, werden sowohl alle Titeln als auch alle Notizen der aktuellen Tabelle durchgesucht. Die Groß-, Kleinschreibung wird dabei ignoriert, genauso wie die Position des Suchbegriffs im Titel oder in der Notiz. Rechts oben können Sie die Anzahl der gefundenen Notizen sehen.

Der durchgeführte Suchvorgang reduziert die Anzeige der Titel und Notizen laut folgenden Regeln:

Wenn der gesuchte Text in einem Titel gefunden wird, werden auch alle untergeordneten Elemente (Untertitel und Notizen) angezeigt.

Wenn der gesuchte Text in einem Titel oder Notiz gefunden wird, wird auch die ganze zu ihm führende Titelhierarchie (alle Vorgänger) angezeigt.

Nachdem Sie den Inhalt des Suchfeldes löschen und auf Suchen klicken, stehen wieder alle Datensätze zur Verfügung.

Karel Štípek *kstipek@gmx.net*

Combobox-Kontextmenü

Für die effiziente Verwaltung von gespeicherten Daten sind noch einige weitere Funktionen notwendig, die aus einem Kontextmenü aufgerufen werden. Dieses Menü wird durch das Klicken mit der rechten Maustaste auf eine Combobox eingeblendet und enthält folgende Punkte:

Umbenennen

Die aktuelle Combobox wird weiß hinterlegt und Sie können den Text des Titels ändern. Die Änderung wird ohne Abfrage nach dem Verlassen der Combobox übernommen.

Löschen

Nach einer Sicherheitsabfrage wird der aktuelle Titel mit allen Untertiteln und Notizen entfernt.

Verschieben und Einfügen

ermöglichen das Umstrukturieren der Titelhierarchie.

Hierarchiestruktur ändern

Nachdem Sie vom Kontextmenü den Punkt ***Verschieben*** ausgewählt haben, erscheint in der Titelleiste des Hauptformulars die Information, dass der jeweilige Titel verschoben wird.

Sie können sich jetzt in der Struktur beliebig bewegen. Nachdem Sie in einer Combobox vom Kontextmenü den Punkt ***Einfügen*** auswählen, wird der verschobene Titel mit allen untergeordneten Titeln und Notizen nach einer Abfrage auf die neue Position verschoben.

Wenn Sie das Verschieben ohne Auswirkungen auf die aktuelle Struktur unterbrechen möchten, wählen Sie ***Einfügen*** auf der alten Position des verschiebenden Titels, der Vorgang wird ohne Abfrage beendet.

Notiz ausdrucken

Wenn eine Notiz gerade angezeigt wird, können Sie mit der Schaltfläche ***Drucken*** die Titelhierarchie und den Text der Notiz ausdrucken.

Beispieltabellen

Die mitgelieferte Datenbank enthält zwei Beispieltabellen:

- `Anleitung` - eine Kurzfassung der oben beschriebenen Funktionen
- `Computer Tipps & Tricks` - Auszug aus meiner aktuellen Sammlung

Speichern von Titeln und Notizen

Eine Tabelle reicht

Wie viele Tabellen braucht man eigentlich, um eine Struktur mit einer beliebigen Anzahl von Hierarchiestufen abzubilden? Eine Tabelle reicht. Um einen Titel, der physikalisch einen Datensatz darstellt, eindeutig zu identifizieren, muss man nur die Zuordnung zu dem Vorgänger kennen. Bei der Darstellung der nächsten Ebene werden die Datensätze gesucht, die den aktuellen Schlüssel als Vorgänger definiert haben. Die Einträge der ersten hierarchischen Ebene haben keinen Vorgänger, in dem dafür vorgesehenen Feld steht also Null.

Die fünf Ebenen der Titelhierarchie könnten auch mit 5 Tabellen, die in einer Beziehung 1:N stehen, abgebildet werden. Das eingesetzte Konzept hat einige Vor- und Nachteile. Es ist einfacher eine neue Tabelle anzulegen, dafür müssen der Lösch- und Suchvorgang mit rekursiven Prozeduren realisiert werden.

Struktur der Tabelle

Jede Notizentabelle hat folgende Struktur:

Feldname	Typ	Bedeutung
`lKey`	Autowert	Primärer Schlüssel
`lPrev`	Long Integer	Schlüssel des Vorgängers
`sTitel`	Text	Titel
`mText`	Memo	Text der Notiz
`dtmCreate`	Datum	Erstellungsdatum
`dtmUpdate`	Datum	Änderungsdatum
`ySel`	Ja/Nein	Angezeigt ?

Tabelle 2: Struktur der Notizentabelle

Beispiel des Tabelleninhaltes

Die technische Umsetzung der in der **Abbildung 1** angezeigten Struktur mit einigen "verwandten" Datensätzen können sie dem folgenden Bild entnehmen. Die relevanten Zeilen haben die primären Schlüssel 3, 249 und 325. (**Abbildung 2**)

Abbildung 2: Inhalt der angezeigten Tabelle

Ein aufmerksamer Leser stellt sich sicher die Frage: Warum sind die Felder `dtmCreate` und `dtmUpdate` bei einigen Datensätzen (offensichtlich ohne eine bestimmte Logik) leer ?

Die Antwort ist überraschend: Sie sehen hier ein praktisches Beispiel einer Softwareentwicklung mit der (nicht besonders empfehlenswerter, trotzdem beliebter) "Schwalbennesttechnologie". Kurz gesagt – die Idee mit der Speicherung der Datumsangaben ist mir erst eingefallen, als der Karteikasten schon einige Zeit im Einsatz war.

Auswahl einer Tabelle

Alle in der Datenbank vorhandenen und nicht ausgeblendeten Tabellen sollen in der Combobox zur Auswahl angeboten werden. Ihre Namen werden der Systemtabelle `MSysObjects` entnommen. Der richtige Ausdruck, der die eingeblendeten Tabellen ausfiltert, wird in der Eigenschaft Datensatzherkunft der Combobox eingetragen.

```
SELECT name FROM msysobjects WHERE (type = 1) AND (Flags=0) ORDER BY
name;
```

Anmerkung: Um die Systemobjekte im Datenbankfenster sehen zu können, muss ihre Anzeige zuerst in ***Extras->Optionen->Ansicht*** aktiviert werden.

Zugriff auf Tabellen über eine Abfrage

Die Notizentabellen können beliebige Namen haben. Damit das Programm auf alle Tabellen gleich zugreifen kann, werden in den meisten Fällen die konkreten Namen durch die Abfrage `AktTab` abgeschirmt. Die Abfrage wird nach der Auswahl einer Tabelle dynamisch definiert und enthält alle Datensätze der ausgewählten Tabelle, bei denen das Feld `ySel` den Wert `True` (Ja) hat. Dieses Feld wird bei der Suche in gefundenen Datensätzen auf `True` gesetzt. Durch den Filterausdruck in der Abfrage `AktTab` wird damit die Anzeige der Suchergebnisse laut den o.g. Regeln durchgeführt.

Für die Suchfunktion (wird später erklärt) ist es einfacher den aktuellen Tabellennamen auch in der globalen Variablen `SelTabName` zu speichern als auf das Formularelement zu verweisen.

```
Private Sub cboTabellen_AfterUpdate()
...
g.SelTabName = Me.cboTabellen
Set qry1 = CurrentCurrentDB.QueryDefs("AktTab")
qry1.SQL = "SELECT * FROM [" & g.SelTabName &  "] WHERE ySel"
...
```

Neue Tabelle erstellen

Nach der Eingabe eines neuen Tabellennamen in der Combobox `cboTabellen` wird die Ereignis-Prozedur aufgerufen, die eine neue Tabelle (nach der Sicherheitsabfrage) erstellt. Sie wird nicht mit der DDL-Anweisung `CREATE TABLE` erstellt, sondern wird einfach die leere Tabelle `tblStruktur` unter dem neuen Namen gespeichert. Diese Tabelle `tblStruktur` wird als ***Ausgeblendet*** gekennzeichnet, damit sie in der Tabellenauswahl nicht erscheint.

```
Sub cboTabellen_NotInList(NewData As String, Response As Integer)
If MsgBox("Neu Tabelle '" & NewData & "' erstellen ?", _
   vbQuestion + vbYesNo) = vbYes Then
  DoCmd.RunSQL "SELECT tblStruktur.* " _
    "INTO  [" & NewData & "] FROM tblStruktur;"
```

```
    Response = acDataErrAdded
  Else
    SendKeys "{ESC}"
    Response = acDataErrDisplay
  End If
```

Globale Variablen und Hilfsfunktionen

Globale Variablen

Während der Arbeit mit dem Programm ist es notwendig, mehrere Informationen über den aktuellen Stand der Auswahl, bzw. Bearbeitung der Titel zu speichern. Alle dafür verwendeten globalen Variablen sind in einem gemeinsamen Modul "g" definiert. Die wichtigsten davon werden in der folgenden Tabelle aufgelistet:

Variable	Bedeutung
frm	Referenz auf das Hauptformular
SelTabName	Name der ausgewählten Tabelle
lSelectKey	zuletzt ausgewählter Datensatz
iSelectLevel	zuletzt ausgewählte Ebene
EditMode	Titelname wird geändert
OldTitel	der alte Titel
lOldKey	vorher ausgewählter Datensatz

Tabelle 3: Bedeutung der globalen Variablen

Hauptformular öffnen

Beim Öffnen des Hauptformulars werden die globale Referenz frm auf das Formular gesetzt und die Variablen lSelectKey und iSelectLevel initialisiert. Außerdem wird im Kontextmenü der Punkt ***Einfügen*** deaktiviert.

```
Sub Form_Open(Cancel As Integer)
  Set g.frm = Me
  g.lSelectKey = 0
  g.iSelectLevel = 0
  CommandBars("KarteiKasten-ComboBox").Controls("Einfügen").Enabled = _
    False
...
```

In einem Satz kleiner Hilfsprozeduren oder -funktionen werden die Vorgänge zusammengefasst, die auf mehreren Stellen des Programmcodes aufgerufen werden. Sie sind alle im Modul basEvents gespeichert.

Prozedur cboDisable

Diese Prozedur wird mit einem Parameter start aufgerufen. Alle Comboboxen, beginnend ab der Ebene start werden mit Null befüllt und ausgeblendet.

```
Sub CboDisable(start%)
Dim i%
  For i = start To MAXLEVEL
    g.frm("cbo" & CStr(i)) = Null
    g.frm("cbo" & CStr(i)).Visible = False
  Next i
End Sub
```

Prozedur cboInit

Die Prozedur cboInit aktualisiert den Inhalt der ersten Combobox und blendet alle nachfolgenden aus. Sie wird nach der Tabellenauswahl und vor der Anzeige der Suchergebnisse aufgerufen.

```
Sub CboInit()
  DoCmd.Requery "cbo1"
  With g.frm
    .cbo1.Visible = True
    .cbo1.SetFocus
    .cbo1 = Null
  End With
  CboDisable (2)
End Sub
```

Prozedur MemoUpdate

Alle Steuerelemente des Hauptformulars sind ungebunden und müssen aus diesem Grund explizit je nach Bedarf befüllt werden. Die Prozedur MemoUpdate liest für den letzten selektierten Schlüssel den Inhalt der Notiz aus und zeigt sie im Feld txtText an. Anschließend werden auch die Felder für die Anzeige des Datums des letzten Hinzufügen, bzw. Ändern der Notiz aktualisiert.

```
Sub MemoUpdate()
  With g.frm
    .txtText = DLookup("mText", "AktTab","lKey=" & g.lSelectKey)
    .txtCreate = DLookup("dtmCreate", "AktTab", "lKey=" & g.lSelectKey)
    .txtUpdate = DLookup("dtmUpdate", "AktTab", "lKey=" & g.lSelectKey)
  End With
End Sub
```

Prozedur MemoInit

Die Prozedur MemoInit befüllt die o.g. drei Felder mit Null-Werten und löscht damit die Anzeige der Notiz.

```
Public Sub MemoInit()
  With g.frm
    .txtText = Null
    .txtCreate = Null
    .txtUpdate = Null
  End With
End Sub
```

Modul modForms

Das Modul modForms enthält die vier einzeiligen Funktionen FormOpen(), FormClose(), WinMax() und WinRestore(). Sie haben dann ihre Berechtigung, wenn ein Formular oder Bericht keine weitere Funktionalität erfordert. Dann können nämlich ihre Aufrufe direkt im Eigenschaftsfenster eingetragen werden und für das jeweilige Objekt muss kein Klassenmodul existieren.

In unserem Programm werden zwei davon im Bericht rptKartei1 eingesetzt. Beachten Sie die letzte Zeile in diesem Bild, die ein Objekt ohne Klassenmodul definiert.

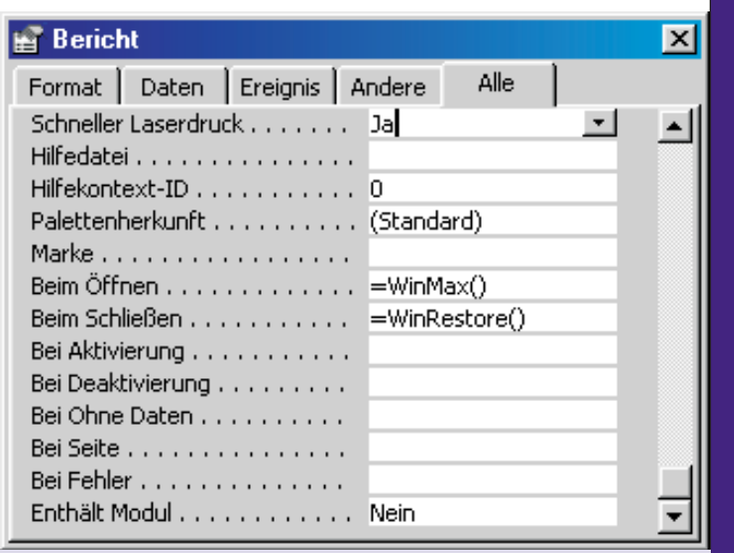

Abbildung 3: Eigenschaften des Berichts rptKartei1

Ereignisprozeduren der Comboboxen

Prozeduren, die im Zusammenhang mit der Auswahl des Titels, bzw. Untertitels einer bestimmten Ebene aufgerufen werden, stellen die Kernfunktionen des Programms dar.

Behandelte Ereignisse

Programmtechnisch sind alle fünf Comboboxen praktisch gleich. Die Ereignisprozeduren sind als allgemeine Funktionen im Modul basEvents implementiert. Die Ebene der jeweiligen Combobox wird im nummerischen Parameter iLevel (der Wert 1 bis 5) an die Funktionen übergeben. Die Referenz auf das Steuerelement wird über den Namen in der Auflistung Controls festgelegt. Die folgende Code-Zeile steht in jeder Funktion am Anfang und wird in den Listings nicht nochmals angeführt.

```
Set ctl = g.frm.Controls("cbo" + CStr(iLevel))
```

Die behandelten Ereignisse und die Namen der aufgerufen Funktionen sind in der folgenden Tabelle aufgelistet:

Ereignis	Funktion im Modul basEvents
Nach Aktualisierung	CboUpdate()
Bei Nicht in Liste	CboNotInList()
Bei Fokuserhalt	CboGotFocus()
Bei Fokusverlust	CboLostFocus()

Tabelle 4: Ereignis-Funktionen im Modul basEvents.

Funktion CboUpdate

Nachdem der Titel einer bestimmten Ebene ausgewählt wurde, werden folgende Aktionen durchgeführt:

- Der Schlüssel des ausgewählten Datensatzes und die Ebene der aktuellen Combobox werden in globalen Variablen gespeichert.
- Die Notiz wird angezeigt (wenn vorhanden)
- Die folgende Combobox wird mit den Untertiteln des jeweiligen Titels befüllt und dadurch, dass der Fokus auf sie gesetzt wird, gleich heruntergeklappt.

```
Function CboUpdate(iLevel%)
...
If Not IsNull(ctl.Column(0)) Then
  g.lSelectKey = ctl.Column(0)
  g.iSelectLevel = iLevel
  MemoUpdate
  If (iLevel < MAXLEVEL) Then
    sql1 = "SELECT * FROM AktTab " & _
           "WHERE lPrev = " & g.lSelectKey & _
           " ORDER BY sTitel"
    Set NextCtl =
         g.frm("cbo" & CStr(iLevel + 1))
    NextCtl.Visible = True
    NextCtl.RowSource = sql1
    NextCtl = Null
    NextCtl.SetFocus
...
```

Funktion CboNotInList

Das Ereignis `Bei Nicht in Liste` kommt dann vor, wenn in die Combobox ein Text eingetragen wird, welches es unter den für diese Combobox richtigen Werten nicht gibt. In diesem Fall bedeutet das, dass ein neuer Titel erstellt wird.

Die Funktion `CboNotInList()` muss folgende Aufgaben erfüllen:

- Einen neuen Datensatz in die Tabelle einfügen und den neuen Titel darin speichern.
- Diesen Datensatz im `Feld lPrev` mit dem Vorgänger-Schlüssel versehen, damit er auf die richtige Position in der Struktur eingegliedert wird. Der Vorgänger-Schlüssel wird der Combobox der Ebene `iLevel-1` entnommen. Laut Definition am Anfang wird für den Titel der ersten Ebene im Feld `lPrev` `Null` eingetragen.

```
Function CboNotInList(iLevel%)
...
If Len(Trim(ctl.Text)) > 0 Then
...
  Set rec = CurrentDb.OpenRecordset ("SELECT * FROM AktTab")
  rec.AddNew
  rec.Fields("sTitel") = ctl.Text
  rec.Fields("ySel") = True
  If iLevel > 1 Then
    rec.Fields("lPrev") =
    g.frm("cbo" & CStr(iLevel - 1)).Column(0)
  Else
    rec.Fields("lPrev") = 0
...
```

Funktion CboGotFocus

Die Funktion `CboGotFocus()` wird aufgerufen, wenn die Combobox den Fokus erhält.

- Zuerst wird überprüft, ob die Combobox nicht leer ist. In diesem Fall werden der aktuelle Schlüssel des Titels und die aktuelle Ebene in globalen Variablen `lSelectKey` und `iSelectLevel` gespeichert und die Notiz (falls vorhanden) mit der Prozedur `MemoUpdate` angezeigt.
- Untergeordnete Comboboxen werden mit der Funktion `cboDisable()` ausgeblendet.
- Wenn der Titel leer ist, werden alle Funktionen des Kontextmenüs ausgeblendet, Es kann nur der Punkt ***Einfügen*** in Abhängigkeit vom Status des Verschiebevorgangs aktiv sein.
- Die Combobox wird heruntergeklappt.

```
Function CboGotFocus(iLevel%)
...
If Nz(ctl.Column(0), 0) > 0 Then
  g.lSelectKey = ctl.Column(0)
  g.iSelectLevel = iLevel
  MemoUpdate
End If

CboDisable (iLevel + 1)

popup = Len(Trim(Nz(ctl.Value, ""))) > 0
With CommandBars("KarteiKasten-ComboBox")
  .Controls("Umbenennen").Enabled = popup
  .Controls("Löschen").Enabled = popup
  .Controls("Verschieben").Enabled = popup
End With
SendKeys "%{DOWN}"
...
```

Funktion CboLostFocus

Die Funktion `CboLostFocus()`, die beim Verlassen der Combobox aufgerufen wird, hat nur eine Aufgabe - das eventuell aktive Umbenennen eines Titels (die Variable `EditMode` in von der Funktion `CboRename` auf `True` gesetzt) zu Ende zu bringen, also den neuen Titel in die Tabelle zu speichern.

Zuerst wird der Datensatz mit dem alten Titel (in der Variablen `OldKey` wird in der Funktion `CboRename()` der alte Titel abgelegt) in der Tabelle gesucht und der Titel je nach der neuen Eingabe geändert.

Am Ende der Funktion wird die Farbe der Combobox mit umbenannten Titel wiederhergestellt und die Variable `EditMode` auf `False` gesetzt.

```
Function CboLostFocus(iLevel%)
...
If EditMode Then
  Set rec = CurrentDb.OpenRecordset
            ("SELECT * FROM AktTab WHERE lKey=" & lOldKey)
  rec.Edit
  rec.Fields("sTitel") = ctl
  rec.Update
End If

ctl.LimitToList = True
ctl.BackColor = g.ColorNormal
EditMode = False
```

Funktionen des Kontextmenüs

Das Kontextmenü, welches jeder Combobox zugeordnet ist, hat vier Punkte. Die aufgerufenen Funktionen sind in der folgenden Tabelle zusammengefasst:

Menüpunkt	Funktion
Umbenennen	`CboRename()`
Löschen	`CboDelete()`
Verschieben	`CboMove()`
Einfügen	`CboPaste()`

Tabelle 5: Funktionen des Combobox-Kontextmenüs

Die Funktion `CboDelete()` erfordert einen rekursiven Vorgang und wird erst im nächsten Teil erklärt.

Eigenschaft ActiveControl

Alle Funktionen, die die Comboboxen betreffen, sind allen gemeinsam. Es muss bei jedem Aufruf festgelegt werden, um welche Combobox es sich gerade handelt. In den Funktionen des Kontextmenüs wird eine andere Technik angewendet als in den Ereignisfunktionen (siehe oben). Die Referenz an die jeweilige Combobox wird mittels der Eigenschaft `Screen.ActiveControl` festgelegt. Alle Funktionen enthalten am Anfang folgende Zeile, die bei der weiteren Beschreibung nicht nochmals angeführt wird.

```
Set ctl = Screen.ActiveControl
```

Funktion CboRename

Die Funktion `CboRename()` initialisiert das Umbenennen eines Titels. Der Vorgang wird erst beim Verlassen der Combobox (in der Funktion `CboLostFocus()`) komplett abgeschlossen und die Änderung in der Tabelle gespeichert.

- Der bestehende Titel und Schlüssel des geänderten Datensatzes werden in globale Variablen gespeichert.
- Die Variable `EditMode` wird auf `True` gesetzt.
- Die Farbe der Combobox wird auf Weiß geändert.
- Die Eigenschaft der Combobox `LimitToList` muss auf `False` gesetzt werden, damit der geänderte Titel nicht als neuer Eintrag interpretiert wird.

```
Function CboRename()
...
If Not IsNull(ctl.Column(0)) Then
  OldTitel = ctl
  lOldKey = ctl.Column(0)
  EditMode = True
  ctl.BackColor = g.ColorEdit
  ctl.LimitToList = False
...
```

Funktion CboMove

Das Ändern der Titelhierarchie bedeutet einen bestimmten Titel mit allen Untertiteln und Notizen auf eine neue Position zu verschieben. Es ist ein zweiphasiger Vorgang der aus Verschieben und Einfügen besteht. Die erste Phase wird durch die Funktion `CboMove()` gestartet.

Die Funktionalität ist ähnlich wie bei `CboRename()`.

- Der bestehende Titel und Schlüssel des geänderten Datensatzes werden in globale Variablen gespeichert.
- In der Titelleiste des Hauptformulars wird die Information über den laufenden Vorgang angezeigt.
- Alle Menüpunkte des Kontextmenüs außer Einfügen werden deaktiviert.

```
Function CboMove()
...
If Not IsNull(ctl.Column(0)) Then
  OldTitel = ctl
  lOldKey = ctl.Column(0)
  iOldLevel = g.iSelectLevel
  g.frm.Caption = "'" & ctl.Value & "' WIRD VERSCHOBEN"
  With CommandBars("KarteiKasten-ComboBox")
    .Controls("Umbenennen").Enabled = False
...
```

Funktion CboPaste

Nachdem die neue Position für den verschobenen Titel gefunden wird, wird vom Menüpunkt ***Einfügen*** die Funktion `CboPaste()` aufgerufen. Beim Speichern der Änderungen muss unterschieden werden, ob der Titel als ein Haupt- oder ein Untertitel verschoben wird. Die Ebene `iLevel` der neuen Position wird von dem Namen der aktuellen Combobox abgeleitet.

```
Function CboPaste()
...
Set ctl = Screen.ActiveControl
iLevel = CInt(Right(ctl.Name, 1))
```

Wenn der Titel als Haupttitel eingefügt wird,

- wird in der Tabelle der Datensatz mit dem gespeicherten Schlüssel `OldKey` gefunden und der Wert des Vorgänger-Schlüssels `lPrev` auf 0 gesetzt
- wird der Inhalt der Combobox mit der Ebene `iOldLevel` (alte Position des Titels) und des aktuellen aktualisiert

```
If iLevel = 1 Then
  If MsgBox(...
    DoCmd.RunSQL "UPDATE AktTab " & _
          "SET lPrev= 0 " & _
          "WHERE lKey=" & lOldKey "
    g.frm("cbo" & CStr(iOldLevel)).Requery
    ctl.Requery
```

Wenn der Titel nicht auf die höchste Ebene verschoben wird, ist der Vorgang ähnlich - nur muss in das Feld `lPrev` der Schlüssel des Datensatzes aus der Combobox, welche um eine Ebene höher liegt, eingetragen werden.

```
If MsgBox(...
  DoCmd.RunSQL "UPDATE AktTab SET lPrev= " & g.frm("cbo" & _
  CStr(iLevel - 1)).Column(0) & " WHERE lKey=" & lOldKey
  ...
```

Rekursives Löschen

Rekursion

Das Speichern der gesamten Titelhierarchie mit Notizen in einer Tabelle bringt etwas mehr Aufwand bei der Lösch- und Suchfunktion, da hier rekursive Vorgänge eingesetzt werden müssen. Da die Rekursion eine eher weniger übliche Programmiertechnik ist, wird sie zuerst kurz theoretisch erläutert.

Ein bisschen Theorie

Eine rekursive Prozedur (oder Funktion) ist eine Prozedur die in ihrem Code entweder direkt oder indirekt (über eine andere Prozedur) sich selbst aufruft. Die Definition selbst klingt etwas schleierhaft und man hat das Gefühl, es kann sich um nichts anderes als eine endlose Schleife handeln.

Es ist aber trotzdem in manchen Fällen eine effiziente Programmiertechnik, die im Endeffekt auch übersichtlich ist. Beim Aufruf einer Prozedur werden die Rücksprungsadresse und alle lokalen Variablen im Stack gespeichert und nach dem Beenden der Prozedur wieder gelöscht.

Bei den rekursiven Aufrufen der gleichen Prozedur funktioniert das genauso wie bei der Verschachtelung von verschiedenen Prozeduren. Bei jedem Aufruf werden neue lokale Variablen gespeichert, es bleiben also während der Rekursion mehrere Zustände der gleichen Prozedur erhalten, das Programm verschachtelt sich, bis die Endbedingung erfüllt wird. Dann werden die "offenen" Aufrufe der Prozedur wieder nacheinander beendet.

Zwei Bedingungen müssen erfüllt werden, damit die Rekursion eingesetzt werden kann:

- Das Problem kann in einfachere Unterprobleme aufgeteilt werden.
- Nach einer endlichen Anzahl von Aufteilungen kann das Unterproblem direkt (ohne den rekursiven Prozeduraufruf) gelöst werden.

Funktion CboDelete

Beim Löschen eines Titel mit all seinen Untertiteln und Notizen kommt die Rekursion zum erstenmal zum Tragen. Sie wird in der Funktion `DelAkt()` eingesetzt, die nach der Sicherheitsabfrage aus der Kontextmenü-Funktion `cboDelete()` aufgerufen wird.

Nachdem der ganze Zweig der Struktur gelöscht wurde, werden noch alle untergeordneten Comboboxen ausgeblendet und der Inhalt der Combobox, wo der gelöschte Titel vorher war, wird aktualisiert.

```
Function CboDelete()
...
If MsgBox(...
  MemoInit
  DelAkt (ctl.Column(0))
  CboDisable (iLevel + 1)
  ctl = Null
  ctl.Requery
  ctl.Dropdown
...
```

Funktion DelAkt

Die Funktion `DelAkt()` führt den eigentlichen Löschvorgang des aktuellen und aller untergeordneten Elemente durch. Der Parameter `DelKey` legt fest, welcher Datensatz bereits als der aktuelle betrachtet wird. Die rekursive Technik ist notwendig, weil die Anzahl der notwendigen Aktionen in voraus nicht bekannt ist.

Jeder Aufruf besteht aus zwei Teilen.

- Die Funktion ruft sich selbst (das ist die Rekursion) für alle Untertitel auf. Die Untertitel sind diejenigen Datensätze, die im Feld `lPrev` den aktuellen Schlüssel `DelKey` eingetragen haben.
- Wenn es keine Untertitel mehr gibt (Bedingung für das Beenden der rekursiven Aufrufe), wird der aktuelle Eintrag mit dem Schlüssel `DelKey` in der Tabelle gelöscht.

```
Sub DelAkt(DelKey&)
...
Set rec = CurrentDb.OpenRecordset _
          ("SELECT * FROM AktTab WHERE lPrev = " & DelKey)
Do While Not rec.EOF
  DelAkt (rec.Fields("lKey"))
  rec.MoveNext
Loop
DoCmd.RunSQL "DELETE * FROM AktTab WHERE lKey = " & DelKey
```

Rekursive Volltextsuche

Auch wenn die Arbeit mit den Titeln und Untertiteln bequem und übersichtlich ist, erweist sich eine Volltextsuche in der gesamten Tabelle als durchaus nützlich.

Die beim Suchvorgang eingesetzten Funktionen greifen auf die Tabelle nicht über die Abfrage `AktTab` zu, sondern direkt über den Tabellennamen. Dieser steht in der globalen Variablen `SelTabName` zur Verfügung. Der Grund dafür ist der, dass die Abfrage `AktTab` nur die selektierten Datensätze (`ySel=True`) enthält und dadurch es nicht möglich wäre, das notwendige mehrstufige Verfahren einzusetzen.

Drei Schritte zum Ergebnis

Damit die am Anfang des Artikels vorgestellten Suchergebnisse implementiert werden, sind folgende logische Schritte notwendig.

- Alle Datensätze, die im Titel (der beliebigen Ebene) oder in der Notiz den gesuchten Text enthalten, werden markiert (das Feld `ySel` auf `True` gesetzt).
- Alle Untertitel der im ersten Schritt markierten Titel werden markiert. Dieser Vorgang wird genauso wie das Löschen mit einer rekursiven Prozedur realisiert.
- Alle Vorgänger der im ersten Schritt markierten Titel werden markiert.

Nach diesen drei Schritten werden nur die markierten Datensätze der Tabelle angezeigt.

1.Schritt - Text suchen

In der Ereignisprozedur `cmdSuchen_Click` sind die drei o.g. Schritte gut nachvollziehbar.

Wenn der Suchbegriff leer ist, werden im ersten Schritt alle Einträge aktiviert, sonst werden alle Titel und Notizen durchsucht.

```
Sub cmdSuchen_Click()
...
If IsNull(Me.txtSuchen) Then
  s = "UPDATE [" & g.SelTabName & "] SET ySel=True;"
  DoCmd.RunSQL s
Else
  s = "UPDATE [" + g.SelTabName
  s = s + "] SET ySel=(InStr(sTitel,'" + Me.txtSuchen + "')>0"
  s = s + " OR InStr(mText,'" + Me.txtSuchen + "')>0);"
  DoCmd.RunSQL s
...
```

2. Schritt - Nachfolger markieren

Zuerst wird ein Recordset `selrec` geöffnet, welches alle im ersten Schritt markierten Datensätze enthält. Im 2. Schritt wird für diese Datensätze die rekursive Prozedur `MarkFoll()` aufgerufen.

```
...
Set selrec = CurrentDb.OpenRecordset _
  ("Select * from [" + g.SelTabName + "] WHERE ySel", DB_OPEN_SNAPSHOT)
If selrec.EOF Then
  MsgBox _
    "Gesuchte Zeichenfolge '" + Me.txtSuchen + "' wurde nicht gefunden"
  Exit Sub
End If
Do While Not selrec.EOF
  MarkFoll (selrec!lKey)
  selrec.MoveNext
Loop
...
```

Prozedur MarkFoll

Die Prozedur `MarkFoll()` ist von der Arbeitsweise her der Prozedur `DelAkt()` ähnlich. Sie markiert alle Datensätze, die den Parameter `PrevKey` als Vorgänger enthalten und ruft dann für alle Nachfolger sich selbst wieder auf.

```
Sub MarkFoll(PrevKey&)
...
Set follrec = CurrentDb.OpenRecordset _
    ("Select * from [" + g.SelTabName + "]" + _
    " WHERE lPrev=" + CStr(PrevKey),...
Do While Not follrec.EOF
  follrec.Edit
  follrec!ySel = True
  follrec.Update
  MarkFoll (follrec!lKey)
  follrec.MoveNext
Loop
```

3. Schritt - Vorgänger markieren

Im dritten Schritt werden die im 1. Schritt markierten Datensätze (`Recordset selrec`) noch einmal bearbeitet. Für jeden Datensatz dieses Recordsets werden alle Vorgänger gesucht und markiert. Für diesen Schritt ist keine Rekursion notwendig

```
...
selrec.MoveFirst
Do While Not selrec.EOF
  PrevKey = selrec!lPrev
  Do While PrevKey > 0
    Set markrec =
    CurrentDb.OpenRecordset
    ("Select * from [" + g.SelTabName + "]" + _
    " WHERE lKey=" + CStr(PrevKey),...
    ...
    markrec.Edit
    markrec!ySel = True
    markrec.Update
    PrevKey = markrec!lPrev
  Loop
  selrec.MoveNext
Loop
...
```

Notiz drucken

Für den Ausdruck einer Notiz mit der entsprechenden Titelhierarchie wurde eine einfache Lösung angewendet. Die Datenfelder im Bericht `rptKartei1` referenzieren direkt die entsprechenden Steuerelemente im Hauptformular.

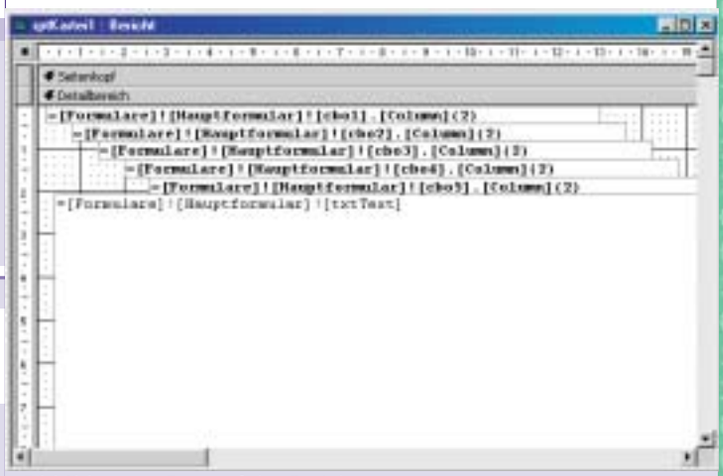

Abbildung 4: Entwurf des Berichts

Die Ereignisprozedur der Schaltfläche ***Drucken*** fragt die Existenz einer Notiz ab und ruft den Bericht auf.

```
Private Sub cmdDrucken_Click()
  If Len(Nz(Me.txtText, "")) = 0 Then
    MsgBox "Keine Notiz vorhanden"
  Else
    DoCmd.OpenReport "rptKartei1", acViewPreview
  End If
End Sub
```

Last but not least

Wenn man ein Notebook ohne angeschlossene externe Maus hat (z.B. auf den Knien im Zug) kommt man darauf, dass es auch nicht sinnlos ist, einige Funktionen mit der Tastatur durchführen zu können. Ich speichere zu jeder Datenbank im Makro `AutoKeys` (in Access 97 muss dieses Makro `Tastaturbelegung` heißen) folgenden Tastenkombinationen:

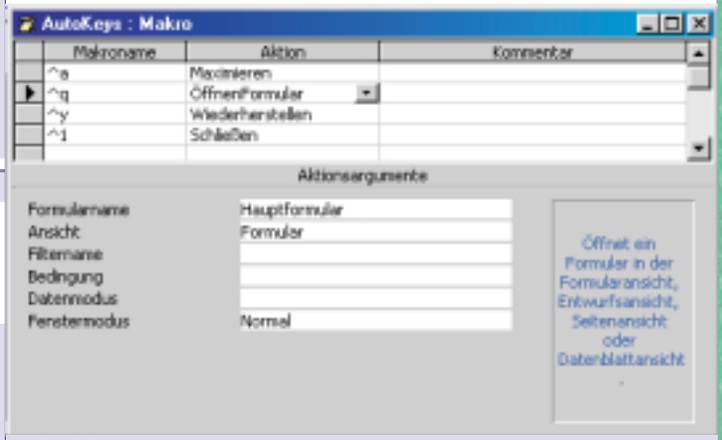

Abbildung 5: Inhalt des Makros AutoKeys

Die Funktionen, die man so schnell mit Tasten aufrufen kann, sind besonders bei der Entwicklung sehr nützlich.

Schlusswort

Das Programm hat sich trotz seiner einfachen Konstruktion in der Praxis sehr gut bewährt. Eine übersichtliche Informationsspeicherung ist in der komplexen EDV-Landschaft einfach unentbehrlich.