

OPL Workshop 5 - Datenbanken: Suchen/Gehe zu

Ulrich Krinzner, 04/1999; Originalvorlage OPL Quelle: Dieter Gansberger

"Eine Datenbank ohne Suchfunktion ist ein gar nutzlos Ding!". So oder ähnlich würden wir heute wahrscheinlich unsere geliebten Klassiker Goethe und Schiller zitieren, wenn diese damals schon Datenbanken gekannt hätten. So wird der Satz wohl lediglich als "Volksmund" in die Geschichte eingehen. Wie auch immer - wir wollen uns dieser Weisheit beugen und unsere Datenbank vom letzten Workshop mit einer entsprechenden Routine (und anderem mehr) aufpäppeln.

Vieler Worte werden wir dabei nicht bedürfen, denn die Verfahrensweisen sind Ihnen ja inzwischen hinlänglich bekannt. Aber so ganz ohne Tips gehen Sie auch aus diesem Workshop nicht hinaus.

Der Quellcode, den Sie wieder am Ende des Textes finden, enthält nur die neuen Elemente und zwei Änderungen an bereits bestehenden Prozeduren. Die Methode, denn Code für den Serie 3a/c/mx darzustellen und die Variationen für den Siena (SN), WorkAbout (WA) und Serie5 (S5) hinter REM's anzugeben, haben wir beibehalten.

Wenn Sie die neuen Routinen dem alten Quelltext einfach hinzuzufügen und die Veränderungen vornehmen, sollte das neue Gesamtprogramm sofort ins Laufen bringen.

Änderungen

Es geht um lediglich zwei Änderungen, die erforderlich werden, weil das Menü um zwei Einträge erweitert werden soll. Neben der Suchroutine werden wir nämlich noch das Unterprogramm "Gehe zu" schreiben.

Zuerst korrigieren Sie in "Showmenu%:" die entsprechende mCARD-Zeile mit den im Programmlisting aufgeführten Begriffe und deren Tasten-Kürzel. Die Funktionsprinzipien haben wir bereits beschrieben.

Damit "DoMenu:(k%)" auch auf diese Änderungen reagieren kann, müssen danach die beiden Buchstaben "s" (Suchen) und "g" (Gehe zu) der Definition des Strings "menuops\$" beigefügt werden.

Suchen

Unsere erste neue Routine heißt "wahls%:". Sie könnte wesentlich kürzer ausfallen, wenn man auf die "Komfortfunktionen" verzichten würde. Gemeint sind das Anhalten der Anzeige, wenn der Bildschirm voll ist, und Fehlermeldungen bei leerer Datenbank. Aber ein wenig benutzerfreundlich soll unser Programm schon sein. Das ist auch schlicht der Grund dafür, weshalb die Routine etliche IF .. ELSE .. ENDIF Abfragen enthält.

Nehmen Sie den Quelltext zur Hand: Zuerst wird die Frage gestellt, ob die Datenbank überhaupt Daten enthält. Das geschieht durch "IF COUNT", was sich wie ein unvollendet gebliebener Fragesatz anhört. Sie hätten wahrscheinlich soetwas wie "IF COUNT<>0" erwartet. OK. Reden wir darüber!

IF - trickreich genutzt

"IF" wird üblicherweise herangezogen, wenn es darum geht, einen Entscheid zu treffen, um danach alternative Programmteile auszuführen. Diese Entscheidung trifft "IF" zwar - aber erst aufgrund einer vorangegangenen Vergleichsoperation, die die Antwort in Form einer Zahl liefert. Beispiel:

```
IF (a% > b%) .. ELSE .. ENDIF
```

Die intern vorgeschaltete Vergleichsoperation prüft, ob die Aussage (a% > b%) wahr oder falsch ist und liefert für die wahre Aussage eine "-1", ansonsten eine "0" als Ergebnis. "IF" zieht nun zu seiner Entscheidung nur noch diesen Zahlenwert heran. In der Praxis gelten bequemerweise alle von Null abweichenden Zahlenwerte (also nicht nur "-1") als logisches WAHR. Nur die Null erlaubt die Interpretation FALSCH.

Das erspart uns letztendlich die Vergleichsoperation bei COUNT, wenn es um die Prüfung des Füllzustandes der Datenbank geht. So findet das schlichte "IF COUNT" schließlich seine Erklärung. (Zur Erinnerung: COUNT ist eine Funktion, die bei Aufruf die aktuelle Datensatzanzahl zurückgibt/enthält)

Hat nun "COUNT" den Wert Null, enthält die Datenbank keine Datensätze. Unsere Suchroutine schreibt in diesem Falle die geeignete Fehlermeldung auf den Bildschirm.

Beschaffungsmaßnahme

Bewahrt die Datenbank mindestens einen Datensatz auf, ist auch die Benutzung der Suchfunktion möglich. Vor den eigentlichen Suchvorgang hat die Routine aber erst einmal die Beschaffung des Suchbegriffes gestellt:

Vorsorglich werden alte Werte aus den Stringvariablen "name\$" und "tel\$" gelöscht ("delkndva:"). Es folgt das Aufblenden der Eingabemaske für den Suchbegriff ("ediSuch%:(title\$)"). Der Suchbegriff soll am Ende in der globalen Variablen "name\$" landen. Die wird schon mal mit dem Vorgabewert "" besetzt. Das ist das Wildcard-Zeichen für die Suche und Auflistung aller Datenbankinhalte. Durch diese Aktion wird lediglich die Faulheit des Nutzers unterstützt, der jetzt nur noch <Enter> drücken muß, um eine Listung aller Datensätze am Bildschirm zu bekommen.

In diesem und in allen Fällen, in denen in der Eingabemaske die <Enter>-Taste betätigt wird, ist der Rückgabewert der Routine "13" (für <Enter>) und der in das Eingabefeld eingetragene Begriff wird in die Variable "name\$" übernommen - so war das auch geplant. Abbruch der Eingaberoutine mit <Esc> liefert eine Null zurück, die Variable "name\$" behält den Wert "". Das hat aber überhaupt keine Folgen, da letztendlich die Suchroutine wegen der Null sowieso abgebrochen wird.

Da als Rückgabewert das Ergebnis von DIALOG ohne weitere Auswertung benutzt wird, kann man hier im Quelltext wieder sparen, indem man "RETURN" und "DIALOG" direkt verknüpft. (Die andere Schreibweise wäre z.B.: d%=DIALOG : RETURN d%)

Finden

Zurückgekehrt in die Prozedur "wahls%", führt also ein Rückgabewert von Null zum sang- und klanglosen Abbrechen der Suche. Aber alle anderen Werte veranlassen den Suchvorgang mit dem in "name\$" eingetragenen Begriff.

Bevor es im Programmtext weitergeht, hier einige allgemeine Bemerkungen zu "FIND", dem Befehl, mit dessen Hilfe die Suche vorgenommen wird. Der sucht nach Aufruf unspezifisch in allen Feldern der Datenbank. Für Mini-Datenbanken wie unsere ist das sicherlich kein Handicap. Für gehobeneren Anwendung muß man sich als Programmierer mit dem Befehl "FINDFIELD" auseinandersetzen und auch wissen, daß es einige Fehler in der Implementierung gibt, d.h. der Befehl reagiert unter bestimmten Bedingungen anders oder nur eingeschränkt im Vergleich zu seinem eigentlich geplanten Anwendungszweck.

"FIND" arbeitet sich bis zur ersten Fundstelle durch und macht den Datensatz mit dem gefundenen Suchbegriff zum aktuellen. Der Suchvorgang ist damit also erst einmal beendet. Bei größeren Datenansammlungen ist aber mit mehreren Fundstellen zu rechnen. Also veranlasst man die Wiederaufnahme der Suche, indem man den "Zeiger" auf den Folgedatensatz positioniert ("NEXT") und erneut ein "FIND" veranlaßt. Das macht man solange, bis die Datenbank vollständig durchkämmt ist. Ob dieser Fall eingetreten ist, kann mit dem Befehl "EOF" (end of file) getestet werden. Wenn das Ende des Datenbankbestandes erreicht ist, liefert EOF den Wert "-1" (also WAHR) zurück, ansonsten Null (also FALSCH).

Sie erkennen bestimmt spätestens jetzt, daß man den Suchvorgang sinnvollerweise in eine Schleife packt, die am ersten Datensatz startet und erst nach dem letzten Datensatz aufhört. Genau das haben wir getan. Vor der DO/UNTIL-Schleife wird der interne Zeiger noch auf den ersten Datensatz positioniert und ab geht's.

f%=FIND(name\$)

Der in "name\$" gespeicherte Suchbegriff wird dem Suchbefehl mitgegeben. Ist dieser gefunden, wird die zugehörige Datensatznummer der Integervariablen f% zugeordnet. In unserem Falle erfolgt dann die Ausgabe auf den Bildschirm, aber nur, wenn f% nicht Null ist. Läßt man hier die IF-Abfrage weg, funktioniert das Programm zwar immer noch richtig, gibt aber eine "unsauber" aussehende Zeile mit der Datensatznummer Null und leeren Feldern aus, wenn kein Eintrag zum Suchbegriff gefunden wurde.

Nach der Bildschirmausgabe wird der Zähler n% um eins heraufgesetzt. Der Zählerstand wird sofort in der folgenden Zeile ausgewertet. Die Rechenoperation bewirkt, daß die Bildschirmausgabe nach der Ausgabe von 10 Zeilen (6 Zeilen beim WA) angehalten wird. Erst ein Tastendruck löscht den Bildschirm, baut die Überschrift neu auf und erlaubt erneut die Darstellung von 10 Zeilen. Auf diese Art wird das Durchscrollen der Bildschirmdarstellung vermieden.

Nach der Bildschirmausgabe wird der innere Zeiger der Datenbank mit "NEXT" um eine Position weitergesetzt. Durch den erneuten Schleifedurchlauf beginnt die Suchaktion ab der neuen Position von vorn, bis am Ende die Abbruchbedingung "EOF" erreicht ist.

Wenn bei der Suche kein passender Eintrag gefunden wird, bleibt f% immer Null. Dadurch wird aber auch die Zählvariable n% nie erhöht und bleibt ebenfalls Null. Dieser Umstand wird nun hinter der Schleife benutzt, um

die passende Meldung ".. nicht gefunden" abzugeben. (Zur Erinnerung, wieso f% und n% hier den Start-Wert Null haben: lokale Variablen werden beim jedem Aufruf der Prozedur, in der sie stehen, neu auf den Wert Null initialisiert)

FIND richtig benutzen

Üblicherweise sucht man nach bekannten Begriffen, darf sich dabei aber nicht verschreiben. Kennt man bei großem Datenbankumfang nicht mehr alle Einträge genau oder man sucht nach mehreren gemeinsamen Namensstämmen (wie: alle Schmidts mit "tt" und "dt"), lassen sich Wildcards einsetzen. Das sind das Fragezeichen ("?"), das für einen einzelnen unbekannten Buchstaben steht und der Stern ("*"), der für eine Buchstabengruppe beliebiger Länge stehen kann.

Da die Suche nicht auf bestimmte Felder begrenzt ist (Name oder Telefonnummer), können Sie genauso gut nach Telefonnummern suchen lassen. Beachten Sie bei eigenen OPL-Experimenten: es handelt sich nicht um Zahlen, sondern immer um Strings - sowohl bei der Eingabe als auch bei der Suche!

Groß- und Kleinschreibung werden durch "FIND" nicht berücksichtigt.

Gehe zu

Diese kleine Routine soll helfen, Datensätze gezielt anzusteuern, um Veränderungen vornehmen zu können. Dazu läßt man zuerst die Suchroutine laufen und erfährt dabei die Nummer des Datensatzes, den man weiterbearbeiten will. Das Aufrufen der Routine und das Eintragen der Datensatznummer in die entsprechende Suchmaske führen schließlich zum Ziel.

Die meisten Befehle aus den Routinen "wahlg%" und "ediGoto&:(title\$)" können Sie mit dem hier erworbenen Wissen selber interpretieren. Auf einen Fallstrick wollen wir Sie allerdings noch aufmerksam machen: Beachten Sie, daß bei der Abfrage der Datensatznummer mit Long-Integer Zahlen gearbeitet wird und daher auch die Variable "d&" in "wahlg%" das Long-Integer-Format haben muß!

Das Feld, das die Eingabe der Datensatznummer erlaubt, wird mit dem dLONG-Befehl nach folgendem Schema aufgebaut:

dLONG eingabevariable&,"text_links",min.wert&,max.wert&

Gratulation!

Sie haben nun eine Datenbank, mit der man sogar richtig richtig arbeiten kann. Mit dem erworbenen Wissen sind Sie aber auch in der Lage, eine ganz persönliche Datenbank zu erstellen.

Die zukünftigen Programmierer unter Ihnen tauchen jetzt sicherlich zu weiteren Höhenflügen in die Handbücher ab. Alle anderen: Wir treffen uns beim nächsten Mal wieder hier - bitte erscheinen Sie zahlreich!

```
PROC showmen%:
    ...
    mCARD "Kunden", "Neu", %n, "Aendern", %a, "Suchen", %s, "Gehe zu", %g, "Loeschen", %l
    ...
ENDP

PROC DoMenu: (k%)
    ...
    menuops$="enagls"
    ...
ENDP
```

```

PROC wahl$:
    REM Eintrag Suchen
    LOCAL f%, erg%, n%
    IF COUNT      REM es gibt Eintraege!
        delkndva:
        IF ediSuch%:("Suche nach ..")
            FIRST
            CLS
            PRINT "Index-Name-Telefon:" : PRINT
            DO
                f%=FIND(name$)
                IF f%
                    PRINT f%," - ";A.name$;" - ",A.tel$
                    n%=n% + 1
                    IF n%=10*(n%/10)      REM [WA: IF n%=6*(n%/6)]
                        PRINT : PRINT "Taste fuer weiter!"
                        GET : CLS
                        PRINT "Index-Name-Telefon:" : PRINT
                    ENDIF
                ENDIF
            NEXT
        UNTIL EOF
    IF n%=0
        PRINT " .. nichts gefunden!"
    ENDIF
    PRINT : PRINT "Suche beendet. Taste druecken."
    GET

    ELSE
        REM Abbruch der Suche ohne Meldung
    ENDIF
ELSE
    CLS
    PRINT "Keine Daten! Taste."
    GET
ENDIF
FIRST
showknd:
ENDP

PROC ediSuch%:(title$)
    name$="*"
    dINIT title$ REM [wg. Schriftgröße am WA: dINITS title$]
    dEDIT name$,"Kundenname:"
    dBUTTONS "Abbruch",27,"Fertig",13
    RETURN DIALOG
ENDP

PROC wahlg%:
    REM Gehe zu (Datensatz)
    LOCAL f%, ds&, n%
    IF COUNT
        ds&=ediGoto%:("Gehe zu Datensatz")
        POSITION ds&
    ELSE
        CLS : PRINT "Kein Daten! Taste." : GET
    ENDIF
    showknd:
ENDP

PROC ediGoto%:(title$)
    REM Datensatznummer f. "wahlg%:" bereitstellen
    LOCAL eingabe&
    eingabe&=1
    dINIT title$      REM [wg. Schriftgröße am WA: dINITS title$]
    dLONG eingabe&,"Datensatz-Nr.",1,COUNT
    dBUTTONS "Fertig",13
    DIALOG
    RETURN eingabe&
ENDP

```

```

PROC wahlx%: REM Programm beenden  REM [S5: PROC wahle%:]
      USE A : CLOSE : STOP
ENDP

```

K U N D E N D A T E N

Kundenname: an
 @ Telefon: 2341

Datei Kunden

Neu ⌘N
 Aendern ⌘A
 Suchen ⌘S
 Gehe zu ⌘G
 Loeschen ⌘L

Datensatz 1/13
 <- Zurueck/Vor -> od. Menue-Taste

K U N D E N D A T E N

Kundenname: anton
 @ Telefon: 2341

Gehe zu Datensatz

Datensatz-Nr.? 1

Fertig
 Enter

Datensatz 1/13
 <- Zurueck/Vor -> od. Menue-Taste

K U N D E N D A T E N

Kundenname: anton
 @ Tele

Suche nach ..

Kundenname: *

Abbruch Fertig
 Esc Enter

Datensatz

<- Zurueck/Vor -> od. Menue-Taste

Index-Name-Telefon:

4	- Donald	- 4123
16	- Daisy	- 4411
17	- Daniel	- 4444
23	- Dagobert	- 4222

Suche beendet. Taste druecken.