Piccoli problemi di archiviazione... risolti con VB

Esistono i grandi problemi di archiviazione ed esistono i piccoli problemi di archiviazione. I grandi si affrontano in genere con prodotti di tipo DBMS, che abitualmente servono per risolvere complesse problematiche di tipo gestionale, i piccoli problemi, relativi ad insiemi di elementi più limitati oppure meno complessi dal punto di vista relazionale, si affrontano anche con altri prodotti, meno impegnativi per lo sviluppatore

di Francesco Petroni

Per piccoli problemi di archiviazione intendiamo quelli che riguardano l'organizzazione di insiemi di elementi omogenei, che sempre più di frequente ci capita di raccogliere nei nostri voluminosi hard disk. Ad esempio insiemi di file con icone, insiemi di file con immagini (questo particolare argomento lo abbiamo sviscerato nel numero scorso), insiemi di file sonori, WAV o MIDI, animazioni, AVI o FLI, oppure insiemi di file testuali oppure ancora file unici, più tradizionali, comprendenti collezioni di record.

Ormai i nostri computer sono multimediali, per cui aumentano le tipologie di file da trattare, inoltre è sempre più frequente il caso in cui il materiale ci viene fornito in gran quantità, si pensi ai dischetti o addirittura ai CD, acclusi alle riviste.

Il nostro obiettivo è quello di fornire, ad un utente medio, e quindi non particolarmente esperto, una serie di spunti che lo stimolino innanzitutto a riordinare il proprio materiale e poi a costruirsi dei programmini, con uno degli strumenti software di cui dispone, per gestire questo materiale.

Un po' di DOS e un po' di Windows

Diamo innanzitutto per scontato il fatto che l'utente medio conosca il DOS e il File Manager di Windows. Sarebbe per costui un bene anche conoscere un po' di dBase, da usare come intermediario nel caso voglia dare una certa organizzazione strutturata al propio materiale.

In figura 1 vediamo alcuni comandi DOS utili quando si voglia costruire un file testuale, che comprenda l'elenco dei file con una certa estensione presenti nelle varie directory del disco rigido. In figura 2 invece controlliamo, attraverso l'Edit del DOS mostrato in una finestra Windows, il risultato di una tale operazione. I record del file prodotto contengono sia nomi di file, sia percorsi di directory, sia dati riassuntivi della directory, sia... righe vuote. Poi vedremo come interpretare tali record.

Sempre in figura 1 vediamo i comandi di dBase da usare quando si vogliano caricare in una struttura DBF i record del file testuale. Sono operazioni che ci serviranno in seguito.

Comandi DOS per elencare Files di pari suffisso	
`DIR`	Elenca tutti i file presenti nella directory
`DIR *.ICO`	Elenca tutti i files con desinenza ICO
`DIR *.ICO/O`	Elenca tutti i files in ordine alfabetico
`DIR *.ICO/S`	Elenca tutti i files in tutte le directories
`DIR *.ICO/S>ELENCO.TXT`	Genera un file testuale con l'elenco dei file *.ICO presenti in qualsiasi directory
Comandi dBASE per elencare convertire un File Testuale in un file DBF strutturato	
`CREATE ELENCO`	Crea la struttura di un file ELENCO.DBF in cui inserire per primo un campo di tipo Testo di lunghezza adatta a ricevere le righe di ELENCO.TXT e poi i campi strutturati desiderati
`APPEND FROM ELENCO.TXT SDF`	Riempi il CAMPO del file ELENCO.DBF con le righe del file ELENCO.TXT
`REPLACE ALL CAMPOx WITH SUBSTRING(CAMPO,X,Y)`	Trasferisce le porzioni di CAMPO, quello con la riga completa, in CAMPOx, quello strutturato
`DELETE FOR LEN(TRIM(CAMPO))=0`	Cancella i record in cui il CAMPO sia vuoto

Figura 1 - DOS e dBASE - Strumentazione. Il comando DIR del DOS permette numerose varianti che possono servire sia a ricercare i file desiderati in tutte le directory del disco rigido, sia a riversare il risultato della ricerca in un file testuale utilizzabile successivamente per vari scopi. Se si usa il dBASE è possibile convertire il file testuale in un file «strutturato» DBF facilmente riutilizzabile in differenti situazioni operative, ad esempio in un programma Visual Basic.

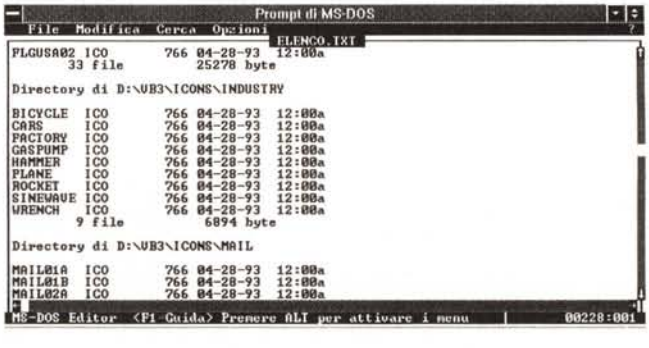


Figura 2 - Editor del DOS in una finestra Windows. Ecco come appare il file testuale ottenuto con uno dei comandi DIR mostrati in precedenza. Lo analizziamo con l'Edit del DOS, visto in una finestra grafica Windows. Se si vuole utilizzare tale file come Base Dati occorre ripulirlo delle righe vuote e delle righe non significative, come quelle che indicano il numero e l'occupazione del file.

```
Dim DB As Database, SN As Snapshot, D$, I$, N
Sub Form_Activate ()
    Set DB = OpenDatabase("D:\HHHH", False, False, "dBASE III;")
    S$ = "SELECT * FROM ICONE ORDER BY NOME"
    Set SN = DB.CreateSnapshot(S$)
    D$ = "D:\HHHH\ICONE\": I$ = ".ICO"
End Sub
Sub C1_Click ()
   Do While Not SN.EOF
       L1.AddItem SN("NOME"): SN.MoveNext
   Loop: N = L1.ListCount: T2.Text = N: P = 1
   Do While P <= N
       L1.ListIndex = P: T1.Text = L1.Text: T2.Text = P
       On Error Resume Next
       P1.Picture = LoadPicture(D$ + T1.Text + I$)
       P = P + 1
   Loop
End Sub
Sub L1_Click ()
    T1.Text = L1.Text
    T2.Text = L1.ListIndex
    On Error Resume Next
    V$ = D$ + T1.Text + I$
    P1.Picture = LoadPicture(V$)
End Sub
```

Figura 3 - MS Visual Basic 3.0 - Elenco delle Icone.
Costruito, con il sistema or ora mostrato, un file testuale con i nomi dei file ICO (quelli con le Icone di Windows) è abbastanza facile realizzare una microapplicazione VB che mostra, in una List Box, l'elenco delle icone e che riproduce, in una Picture Box, l'icona vera e propria, ovviamente quella selezionata nella lista. Nel testo descriviamo nel dettaglio il funzionamento del programma del quale, sullo sfondo della figura, vedete tutti i listati.

Figura 4 - MS Visual Basic 3.0 - Le Icone in Griglia - La Form.
*Un poco (solo un poco) più complesso è il programma che «scarica» l'elenco delle icone e le icone stesse in una griglia (nel mio PC i file *.ICO sono oltre 1.500). È molto interessante, oltre che utile, il fatto che la griglia possa essere usata come vero e proprio contenitore, dal quale è possibile «attingere» sia nomi che figure.*

Tipi di File con i Dati

Ripetiamo alcuni concetti di base a beneficio dei meno esperti.

Un'applicazione che serve a risolvere problemi di archiviazione utilizza uno o più file con i dati, esterni rispetto al file con il programma vero e proprio.

Esistono prodotti che utilizzano dei «flat file», ovvero dei file dati che vengono caricati in memoria all'inizio e salvati alla fine del lavoro. Questa tecnica del «tutto dentro tutto fuori» può essere praticata solo se il file è di piccole dimensioni e se è conveniente disporre di tutti i dati in memoria. È la tecnica usata nei fogli elettronici e nei word processor.

Quando il file è molto grande e/o quando interessa trattare pochi dati per volta (si pensi ad un magazzino che contiene 50.000 articoli che vengono movimentati in piccole quantità, ad esempio 300 al giorno) non si può, ed in ogni caso non conviene, usare tecniche «flat file».

In questo caso si usa un file dati esterno e ne va gestito l'accesso in modo che siano prelevati e trasferiti in memoria solo i record che servono al momento e in modo che vadano rimessi al loro posto quando siano stati aggiornati. Se l'applicazione è complessa non basta un solo file con i dati ma ne servono tanti, ognuno contenente una serie di record omogenei e collegati tra di loro con delle relazioni. Nel primo caso si tratta di file dati unici, semplici Liste, nel secondo caso si parla di Database Relazionali.

Figura 5 - MS Visual Basic 3.0 - Le Icone in Griglia - Il codice.
Il codice, meglio descritto nel testo, è limitato a tre subroutine. La Form_Activate che predispone la griglia, impostandone i titoli di riga e di colonna e dimensionando le rispettive altezza e larghezza, la C1_Click, che corrisponde al pulsante Esegui, che legge in modo sequenziale il file DBF e scarica nelle celle il nome come testo (G1.Text) e la immagine (G1.Picture) delle varie Icone. Facendo successivamente, in fase di utilizzo del programma, clic sulla griglia (G1.Click) vengono individuate la riga, la colonna e quindi il nome e l'immagine.

```
Dim DB As Database, SN As Snapshot

Sub Form_Activate ()
  Set DB = OpenDatabase("D:\HHHH", False, False, "dBASE III;")
  S$ = "SELECT * FROM ICONE ORDER BY NOME"
  Set SN = DB.CreateSnapshot(S$)
  G1.Row = 0: For C = 1 To 15
  G1.ColWidth(C) = 500: G1.Col = C: G1.Text = C: Next C
  G1.Col = 0: For R = 1 To 110
  G1.RowHeight(R) = 500: G1.Row = R: G1.Text = R: Next R
End Sub

Sub G1_Click ()
  L1.Caption = G1.Row: L2.Caption = G1.Col
  L3.Caption = G1.Col + (G1.Row - 1) * 15
  P1.Picture = G1.Picture: T1.Text = G1.Text
End Sub

Sub C1_Click ()
i = 0: SN.MoveFirst
Do While Not SN.EOF
  C = i Mod 15 + 1: R = i \ 15 + 1
  G1.Row = R: G1.Col = C
  P$ = "D:\HHHH\ICONE\" + SN("NOME") + ".ICO"
  T1.Text = P$
  On Error Resume Next
  G1.Text = P$: G1.Picture = LoadPicture(P$)
  SN.MoveNext : i = i + 1
Loop
End Sub
```

In questo caso si usa un file dati esterno e ne va gestito l'accesso in modo che siano prelevati e trasferiti in memoria solo i record che servono al momento e in modo che vadano rimessi al loro posto quando siano stati aggiornati. Se l'applicazione è complessa non basta un solo file con i dati ma ne servono tanti, ognuno contenente una serie di record omogenei e collegati tra di loro con delle relazioni. Nel primo caso si tratta di file dati unici, semplici Liste, nel secondo caso si parla di Database Relazionali.

Il nostro titolo, piccoli problemi di archiviazione, concerne proprio la gestione di semplici Archivi, contenenti quindi una semplice lista di dati omogenei (un Indirizzario, l'elenco degli Articoli di MC, l'elenco dei file ICO presenti in una certa subdirectory, ecc.).

Ora va deciso in quale formato conservare il file dati contenente la lista. Questo può essere un formato testuale oppure un formato strutturato. Un formato testuale contiene solo i dati e non la sua struttura e può essere, al limite, anche letto con il comando DOS TYPE. Un formato strutturato invece contiene anche la struttura, per cui esiste come file anche se è vuoto. La struttura è il contenitore i dati sono il contenuto.

I file testuali appartengono in generale a due tipi: il tipo «fixed lenght» e il tipo «comma delimited». Nel primo caso il record è costituito da una serie di caratteri, che possono essere interpretati solo se si conosce la Struttura del Re-

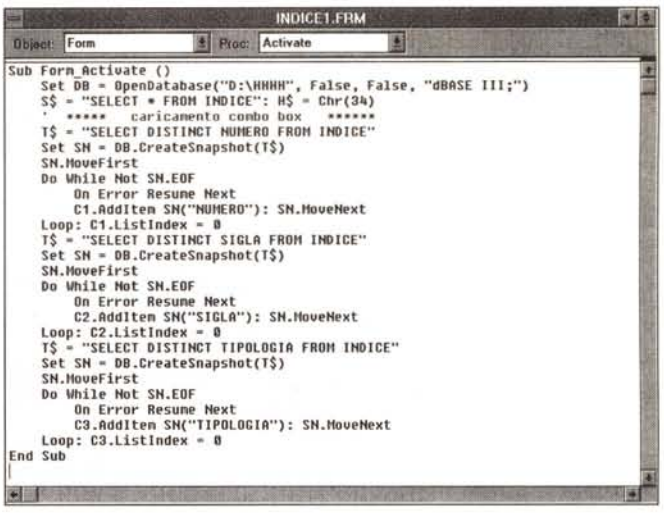


Figura 6 - MS Visual Basic 3.0 - Indice degli Articoli di MC - Il programma finito.
Un elenco, che avete tutti, perché era nel dischetto allegato al numero 146 di MC, è quello che contiene l'elenco degli articoli di MC, dal numero 1 al numero 146. Oltre al campo con il Titolo, c'è quello con il Numero di MC in cui l'articolo è stato pubblicato, poi sigla dell'Autore dell'articolo, Rubrica di riferimento e numero della pagina. Un piccolo archivio da consultare secondo varie chiavi di lettura.

Figura 7 - MS Visual Basic 3.0 - Indice degli Articoli di MC - L'attivazione della Form.
Questo programma può essere realizzato con la versione Pro del Microsoft Visual Basic 3.0, che permette un accesso diretto ai file, senza dover passare attraverso il Data Control. Da programma, ed in qualsiasi maniera, può essere confezionata una istruzione SQL, anche molto complessa, che esegue la ricerca dei dati desiderati. Questo metodo, meglio descritto nel testo, è utilissimo in attività di tipo interrogazione, molto meno in attività di tipo gestionale, per le quali VB è poco adatto.

cord, che indica nome del campo e sua posizione (dal carattere X, per Y caratteri) all'interno della struttura. Nel secondo caso i contenuti dei vari campi sono separati, all'interno del record, da appositi caratteri di separazione. In questo secondo caso è necessario conoscere a priori il nome del campo, ma non la sua lunghezza.

I formati strutturati, che come detto comprendono contenitore e contenuto, sono invece legati al particolare prodotto con il quale sono stati realizzati. Il formato può noto è il DBF, che è talmente diffuso che può essere letto e può essere scritto da moltissimi prodotti. Il formato più recente è l'MDB, producibile sia con Access che con Visual Basic.

La differenza fondamentale tra DBF e MDB consiste nel fatto che mentre ad un file DBF corrisponde una singola tabella, il file MDB invece è unico, comprende tutte le tabelle, tutti gli indici, tutte le relazioni e tutto il resto dell'applicazione.

Visual Basic 3.0 è in grado di leggere tutti questi tipi di file, testuale sequenziale, DBF e Access, più tutti gli altri tipi, compreso il suo tipo «ad accesso casuale», un po' superato dei tempi. Dal Visual Basic, oppure da qualsiasi altro prodotto che legga e scriva i vari formati, è possibile eseguire conversioni. Ad esempio nella figura 1 abbiamo visto alcuni dei comandi che servono per importare in dBase dei file testuali.

```
Dim DB As Database, SN As Snapshot, S$, C$, H$, T$
Sub C1_Click ()
    T$ = S$ + " WHERE NUMERO=" + H$ + C1.Text + H$
    T$ = T$ + " ORDER BY PAGINA"
    T1.Text = T$: L1.Clear
End Sub
Sub C2_Click ()
    T$ = S$ + " WHERE SIGLA=" + H$ + C2.Text + H$
    T$ = T$ + " ORDER BY NUMERO"
    T1.Text = T$: L1.Clear
End Sub
Sub C3_Click ()
    T$ = S$ + " WHERE TIPOLOGIA=" + H$ + C3.Text + H$
    T$ = T$ + " ORDER BY NUMERO"
    T1.Text = T$: L1.Clear
End Sub
Sub P1_Click ()
    Set SN = DB.CreateSnapshot(T$): SN.MoveFirst
    Do While Not SN.EOF
        N$ = SN("NUMERO") + "-": P$ = SN("PAGINA") + "-"
        G$ = SN("SIGLA") + "-": A$ = SN("ARGOMENTO")
        L1.AddItem N$ + P$ + G$ + A$: SN.MoveNext
    Loop
End Sub
```

Figura 8 - MS Visual Basic 3.0 - Indice degli Articoli di MC - Sotto le Combo Box.
L'aspetto più interessante, e più impegnativo per il programma, è costituito dalla modalità di caricamento delle tre List Box che contengono gli elenchi dei Numeri di MC, delle Sigle degli Autori e delle Rubriche. Le tre liste vengono caricate automaticamente dall'elenco degli articoli. Se ne occupano tre comandi SQL, che, con l'opzione DISTINCT, creano proprio degli elenchi senza ripetizione del campo (o dei campi) desiderati. Questo significa che il programma può funzionare anche con successive versioni dell'indice degli articoli.

Problemi di prestazioni

Il problema immediatamente successivo rispetto a quello della scelta del formato è quello che riguarda le prestazioni, ovvero, detto in parole povere, il tempo che ci vuole per eseguire una certa operazione, quale può essere un ordinamento, oppure una ricerca, oppure una selezione e così via.

Come noto anche agli utilizzatori meno esperti, in qualsiasi prodotto DBMS è possibile definire ed utilizzare degli indici, che servono per organizzare in maniera logica i dati della tabella, in modo che le ricerche rispetto a quell'indice siano molto più veloci.

Ebbene per i file sequenziali non è possibile stabire dei file indice. Per i file DBF è possibile crearli, mantenerli aggiornati ed utilizzarli.

Si tratta dei file NDX che vanno creati a parte ed aperti a parte, a cura dell'utilizzatore. I file MDB in formato Access sono omnicomprensivi, nel senso che è il creatore del database MDB che definisce la tabelle e all'interno di queste i campi indice e il campo chiave, che è quello con l'indice principale.

Visual Basic si comporta di conseguenza. Se si utilizzano file testuali sequenziali non c'è nessuna possibilità di sfruttare indici per velocizzare le operazioni. Se si utilizzano file DBF è possibile usare gli NDX, solo che, trattandosi di file esterni, questi vanno in un

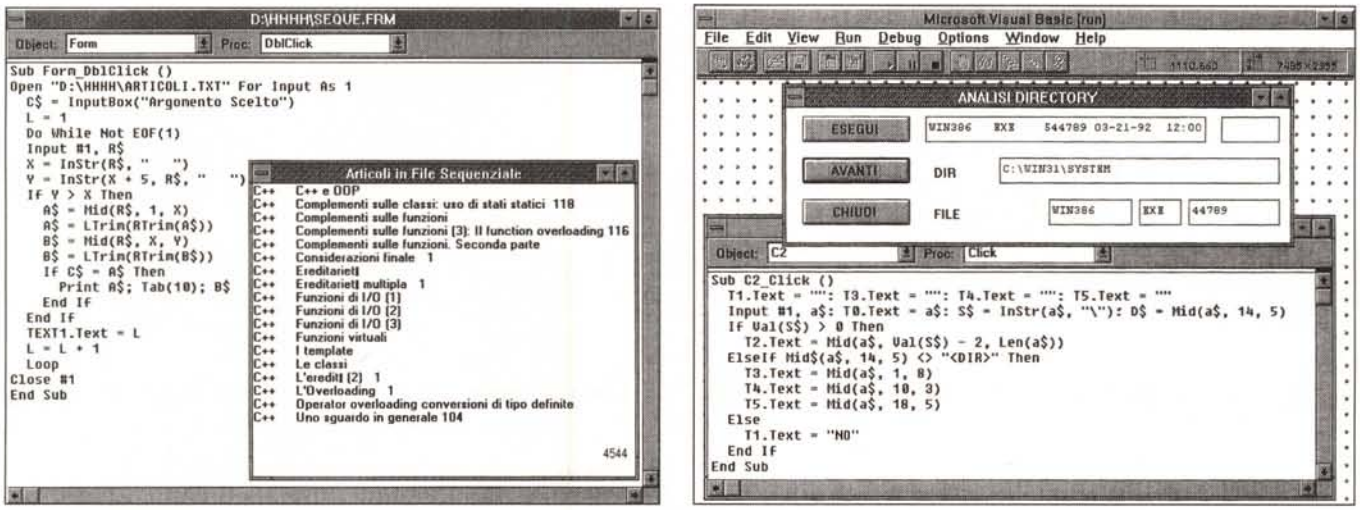


Figura 9 - MS Visual Basic 3.0 - Per chi ha la versione standard e legge... testuale.
Il catalogo degli articoli di MC è disponibile sia in formato DBF che in un formato testuale un po' sporco, nel senso che non è né fixed lenght, né comma delimited. È possibile tentare una sua interpretazione con le istruzioni «classiche» di lettura di file sequenziali. Tali istruzioni, presenti anche nelle primissime versioni del Basic, prescindono dal Data Control e anche dalle istruzioni dirette fino ad ora utilizzate ed erano disponibili anche nelle prime versioni del VB. In pratica i record del file vengono letti come «stringone» di dati che poi vanno in qualche maniera interpretate e fatte a pezzi con le funzioni di manipolazione della stringa.

Figura 10 - MS Visual Basic 3.0 - Per un proprio File Manager.
*Supponiamo di aver eseguito il comando DIR *.*/S>DATI.TXT. Ci ritroviamo un file testuale, un po' pasticciato, che però possiamo scorrere alla ricerca di ciò che ci interessa. Occorre individuare i file e le directory e scartare quello che non serve. Sapendo come è fatto il file è facile individuare, con funzioni di sottostringa, file, loro caratteristiche e directory di appartenenza.*

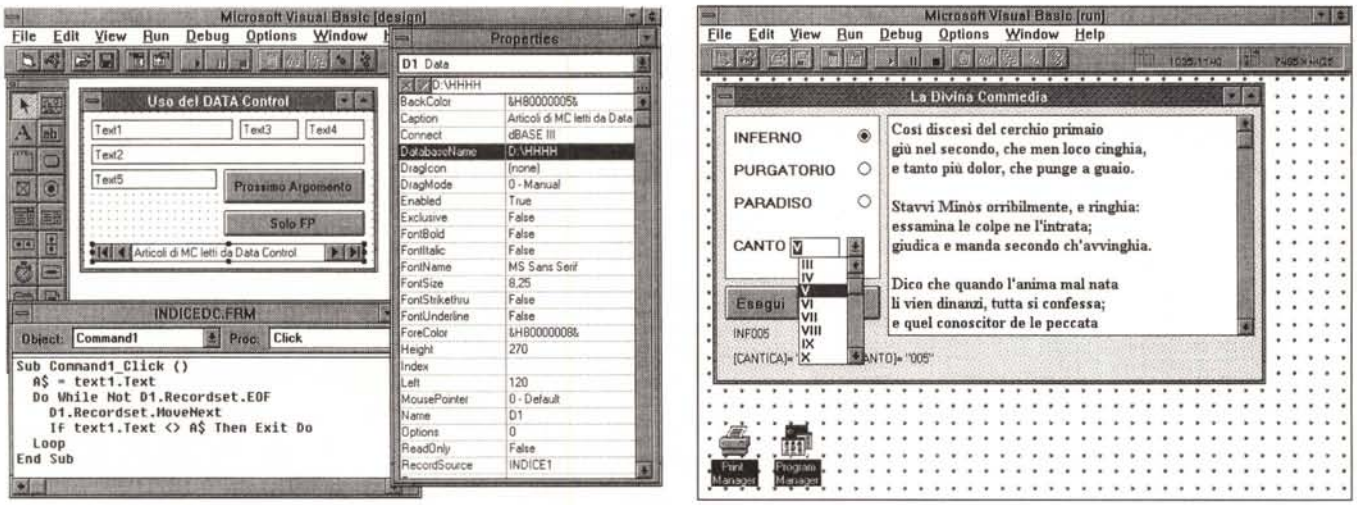


Figura 11 - MS Visual Basic 3.0 - Per chi ha la versione standard e legge... DBF.
La modalità di accesso ai dati disponibile nella versione Standard (quella non professionale) di Visual Basic sono limitate all'uso del Data Control. È possibile utilizzare indirettamente il Data Control lavorando direttamente sul suo RecordSource, che può essere variato al volo, e sul suo RecordSet, che può essere usato per spostarsi avanti ed indietro. Se del caso il Data Control può essere reso invisibile.

Figura 12 - MS Visual Basic 3.0 - Ricerca nella Divina Commedia.
Anche la Divina Commedia può essere trattata (o maltrattata) come un file. Supponiamo quindi di aver creato una struttura che comprenda Cantica (Inferno, Purgatorio o Paradiso), Canto (in totale sono 100) e singolo Verso, magari con un numero Progressivo. Si possono ipotizzare alcuni tipi di ricerca. La prima sarà per Cantica e Canti, ad esempio il Quinto Canto dell'Inferno (quello dedicato ai Lussuriosi). Oppure ricerche sequenziali, ad esempio cercare i versi in cui appare un dato nome. Nella nostra Form ipotizziamo tre Pulsanti di Opzione per selezionare la Cantica e una List Box per scegliere il Canto.

certo senso dichiarati e quindi aperti autonomamente. Se si utilizzano file MDB il problema non sussiste nel senso che se l'indice c'è (perché è stato creato) viene utilizzato quando si fanno operazioni su quel campo, se non c'è l'operazione diventa sequenziale, e quindi lenta.

La lentezza dipende comunque dalle dimensioni della tabella e dalla velocità della macchina.

I discorsi generali fatti fino ad adesso riguardano il file con i dati, ora vediamo come sia possibile, con un prodotto «all purpose», come il Visual Basic, utilizzare questi file.

La gestione delle Icone

Ad ogni applicazione Windows, anzi ad ogni oggetto Windows, è possibile associare una Icona. Lo stesso Visual Basic, quando lo installate, ne riversa diverse centinaia nel vostro disco rigido. L'Icona consiste in un file, estensione

```
Dim DB As database, SN As snapshot, C(34), CA, CN, H, AC

Sub Form_Activate ()
  C(1) = "I": C(2) = "II": C(3) = "III": C(4) = "IV": C(5) = "V"
  C(6) = "VI": C(7) = "VII": C(8) = "VIII": C(9) = "IX": C(10) = "X"
  C(11) = "XI": C(12) = "XII": C(13) = "XIII": C(14) = "XIV"
  C(15) = "XV": C(16) = "XVI": C(17) = "XVII": C(18) = "XVIII"
  C(19) = "XIX": C(20) = "XX": C(21) = "XXI": C(22) = "XXII"
  C(23) = "XXIII": C(24) = "XXIV": C(25) = "XXV": C(26) = "XXVI"
  C(27) = "XXVII": C(28) = "XXVIII": C(29) = "XXIX": C(30) = "XXX"
  C(31) = "XXXI": C(32) = "XXXII": C(33) = "XXXIII": C(34) = "XXXIV"
  For I = 1 To 34: CM.AddItem C(I): Next I
  01 = 1: CM.ListIndex = 0: H = Chr(34): AC = Chr(13) + Chr(10)
  Set DB = OpenDatabase("D:\HHHH", False, False, "dBASE III;")
  S$ = "SELECT * FROM DIVCOM"
  Set SN = DB.CreateSnapshot(S$)
End Sub

Sub CM_Click ()
  If 01 Then CA = "INF"
  If 02 Then CA = "PUR"
  If 03 Then CA = "PAR"
  CN = Format(CM.ListIndex + 1, "000"): L1.Caption = CA + CN
End Sub

Sub PU_Click ()
  T1.Text = "": T2 = ""
  CR = "[CANTICA]= " + H + CA + H + " AND [CANTO]= " + H + CN + H
  L0.Caption = CR
  SN.FindFirst CR: SN.MoveNext
  Do While 1
    T2 = T2 + SN("testo") + AC
    If SN("verso") Mod 3 = 0 Then T2 = T2 + AC
    SN.MoveNext : If SN("canto") <> CN Then Exit Do
  Loop
  T1.Text = T2
End Sub

Sub RI_Click ()
  K = UCase(InputBox("Chiave da Ricercare"))
  SN.MoveFirst
  Do While Not SN.EOF
    T = SN("testo")
    If InStr(UCase(SN("testo")), K) > 0 Then
      M = "Cantica " + SN("CANTICA") + AC
      M = M + "Canto  " + SN("CANTO") + AC
      M = M + "Verso  " + Str(SN("VERSO"))
      MsgBox M: Exit Do
    End If
    SN.MoveNext
  Loop
End Sub
```

Figura 13 - MS Visual Basic 3.0 - La Divina Commedia - Il Codice. Qui vediamo tutti i «pezzetti» del programma: la dichiarazione iniziale del Database e delle Variabili. Poi il caricamento, all'attivazione della Form, della lista dei Canti e l'apertura del file. Poi la costruzione del criterio, agendo prima sulle Option Box poi sulla Lista dei Canti, attraverso il quale si esegue l'istruzione FindFirst Criterio. Poi viene percorsa tutta la tabella a parità di criterio. L'ultimo pezzo riguarda la ricerca di un testo «qualsiasi» in tutta la tabella.

ICO, lungo 766 byte. Se provate a copiarle, anche a mano, tutte in una directory, potrete facilmente costruire un file con l'elenco delle Icone e poi un'applicazione VB che le elenchi e le visualizzi.

Allora, dopo aver riversato tutti i file ICO nella stessa directory, eseguiamo il comando DOS:

```
DIR *.ICO/O>ICONE.TXT
```

che come detto crea un file testuale, ICONE.TXT, comprendente il risultato del comando DIR ovvero l'elenco ordinato dei file con le icone.

Ora in dBASE creiamo una struttura ricevente:

```
CREATE ICONE
```

che comprenda un solo campo, di lunghezza otto caratteri. Per alimentarlo:

```
USE ICONE
APPEND FROM ICONE.TXT SDF
```

I vari comandi devono ovviamente tener conto delle directory in cui stanno le varie cose.

A questo punto abbiamo un file ICONE.DBF già ordinato che contiene l'elenco delle nostre icone.

La prima applicazione VB è costituita da una Form che comprende una List Box, che viene alimentata all'attivazione della Form stessa, con i nomi del file.

Poi al clic sulla Lista, è possibile, tramite le caratteristiche della Lista stessa, individuare su quale elemento (nome e progressivo) è stato fatto il clic e conseguentemente è possibile caricare l'immagine dell'icona nella Picture Box. Il tutto è documentato nell'unica figura 3.

Le istruzioni obbligatorie iniziali sono:

```
Dim DB As Database,
TB As Table, DS Ad Dynaset,
SN As Snapshot
Set DB = OpenDatabase
("XXXX",,"dBASE III";")
S = <istruzione SQL>
Set SN = DB.CreateSnapshot(S)
```

Innanzitutto vanno dichiarati gli oggetti che possono essere:

il database. Se si usa il formato DBF è il nome della directory con i file DBF

nessuna, una o più tabelle del database, aggiornabili (table)

nessuna, una o più viste logiche aggiornabili (dynaset)

nessuna, una o più viste logiche non aggiornabili (snapshot)

(per vista logica si intende una tabella virtuale che presenti solo i campi e i record desiderati).

Poi si apre il DB, seguendo la sintassi che prevede la dichiarazione del tipo di file. Nel comando visto sopra «XXXX» indica il percorso della directory in cui sono messi i file DBF.

È opportuno memorizzare l'istruzione SQL, che genera la vista logica, in una variabile (al limite per confezionarla al volo durante il programma). Per eseguire per la prima volta o rieseguire l'istruzione SQL, qualora nel corso del programma venga modificata, il comando è:

```
Set SN = DB.CreateSnapshot(S)
```

Abbiamo usato lo Snapshot, invece della Table, per poter far mettere in ordine i dati (che nel file DBF potrebbero non essere in ordine) dal comando SQL.

Abbiamo usato lo Snapshot invece del Dynaset per evitare di poter, per errore, aggiornare i dati. In pratica ci basta leggerli.

Questo particolare accesso diretto al file, con queste tre o quattro istruzioni, è possibile solo con il Visual Basic 3.0, Professional Edition. Se si dispone della Standard Edition occorre utilizzare il Data Control che, come vedremo, permette di eseguire, in altro modo, le stesse cose. Per scorrere i dati la struttura di programmazione è la seguente:

```
Do While Not DB.Eof
      ...
      DB.MoveNext
      MovePrevious,
      MoveFirst,
      MoveLast)
      Loop
```

Il passo immediatamente successivo è quello di vedere più Icone per volta. Si può usare una Griglia che oltretutto permette non solo di visualizzare le Icone, ma anche di memorizzare al suo interno il nome dell'icona stessa. Vediamo in figura 4 il risultato del nostro programma, e nella successiva, i tre listati che producono quel po po di output ben descritti nelle didascalie.

Gli articoli di MC

Al numero 146 di MC era allegato un file DBF contenente l'intero elenco degli Articoli di MC e comprendente il Titolo dell'articolo, il Numero di MC, la

Rubrica, l'Autore (in sigla), il numero della pagina in cui l'articolo è stato pubblicato. Cinque campi per oltre 4.000 record. Una lista, chiamiamola così, semplice, in quanto ha solo cinque campi, ma abbastanza lunga da richiedere un programmino che ne agevoli la consultazione.

Vediamo subito, in figura 6, il risultato finale della nostra applicazione VB. Vogliamo cercare gli articoli:

o di un dato Numero di MC,
o di un dato Autore,
o di un dato Argomento.

Vogliamo che, eseguita la selezione usando una delle tre liste sulla sinistra, sulla listona a destra appaiano tutti gli articoli desiderati. Rinunciamo, per non complicare il programma (che comunque potete facilmente complicare voi), ad eseguire selezioni combinando i tre criteri, e a usare differenti organizzazioni in uscita, quindi produciamo sempre elenchi con Numero, Pagina, Sigla e Titolo.

L'istruzione SQL da eseguire è composta da due pezzi, il primo è fisso ed è memorizzato nella variabile S$, il secondo pezzo, T$, viene definito «al volo» quando si opera una selezione in una delle Liste. In pratica viene dinamicamente costruita pezzo per pezzo ed eseguita al clic sul pulsante Esegui.

L'aspetto più interessante del programma consiste nel fatto che i contenuti delle tre liste vengono caricati al volo, ciascuno con una specifica istruzione SQL. Si perde un po' di tempo, ma il caricamento viene eseguito una volta sola, al lancio dell'applicazione, anzi all'attivazione della Form (fig. 8).

Il comando SQL:

```
SELECT DISTINCT SIGLA FROM
INDICE
```

crea un elenco delle sigle presenti nel campo SIGLA senza ripetizioni. È proprio quello che serve per alimentare la List Box.

Problemi di versioni

Con le prime due versioni di VB, e con la dotazione standard di Controls, si potevano gestire solo file di tipo Sequenziale e Random. Con la versione 3.0 Standard si può utilizzare anche il Data Control. Con la versione 3.0 Professional sono possibili gli accessi ora visti, ed in più è possibile una serie di comandi di creazione del database, che non vediamo.

Quello che è opportuno vedere sono i comandi di accesso al file Sequenziale. La struttura:

```
Open "XXXX" For Input As 1
(1 è il canale)
Do While Not EOF(1)
        ....
        Input #1,C1$,C2$...Cx$
        ...
Loop
Close #1
```

C1$,C2$, ecc. sono i campi del record. Se si sta leggendo un «comma delimited» saranno tanti quanti sono i campi. Se si sta leggendo un «fixed lenght» allora basta uno solo, come nel nostro caso, R$, che poi va trattato con funzioni di sottostringa.

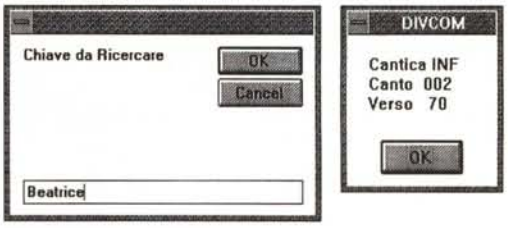


Figura 14 - MS Visual Basic 3.0 - La Divina Commedia - Ricerca.
L'operazione più frequente quando si utilizza un file è la ricerca di un record, e, nel caso della Divina Commedia vista come archivio, di un Verso. Se la ricerca riguarda un campo chiave ed è una ricerca «secca», i tempi necessari per eseguire la ricerca sono comunque immediati. Se il campo non è un campo chiave oppure se si esegue una ricerca per «sottostringa», come nel nostro caso in cui cerchiamo il primo verso in cui è presente, in qualsiasi posizione, una certa parola, la ricerca è sequenziale, quindi lenta, quindi tanto più lenta quanto più è grande l'archivio.

Nel disco con l'archivio degli articoli di MC c'è anche la versione «text» del file. Purtroppo è un po' sporca nel senso che non è né a formato fisso, né delimitato da virgole.

Proponiamo (in figura 9) la lettura del file, riga per riga, e una ricerca, con funzioni di sottostringa, che permetta di individuare la posizione degli spazi che isolano dal resto il campo Argomento. Notare con l'occasione il solito, e noiosissimo, problema del set di caratteri, che va affrontato (non lo facciamo) per far leggere bene anche le lettere accentate.

In figura 10 vi proponiamo un altro programmino, di cui però abbiamo documentato solo le parti principali, che «scandaglia» il file testuale prodotto con il comando DOS.

```
DIR *.*/S>DATI.TXT
```

che produce un file con l'elenco, organizzato per Directory, di tutto il contenuto del disco su cui viene eseguito. Al resto pensateci voi.

Il Data Control

Se, per accedere ai dati, si vuole utilizzare il Data Control occorre innanzitutto piazzarlo nella Form e poi definire alcune sue caratteristiche obbligatorie:

Connect	tipo di connessione, nel nostro caso dBASE
DatabaseName	nome della directory con i file DBF
RecordSource	nome del file DBF

Nel caso di accesso a file MDB:

Connect	si omette
DatabaseName	nome del file MDB
RecordSource	nome della Tabella desiderata

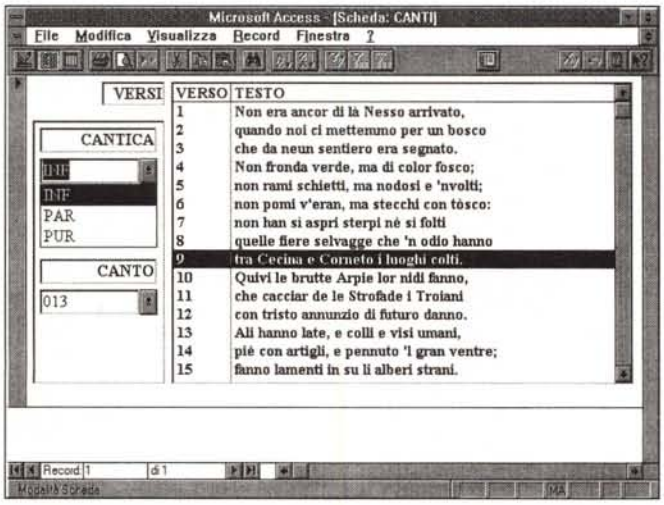


Figura 15 - MS Visual Basic 3.0 - La Divina Commedia in Access.
MS Visual Basic può utilizzare qualsiasi formato di file, o sequenziale, che viene letto direttamente grazie ai comandi di lettura, o MDB, ovvero Access, che è il suo formato interno, oppure DBF, dBase, che possono essere interpretati tramite driver ISAM. Sono utilizzabili anche Paradox, bTrieve, e, tramite i driver ODBC, qualsiasi altro formato. Il discorso va allargato agli indici, che non esistono per i sequenziali, che sono «incorporati» negli MDB di Access, esterni e quindi gestibili solo con ulteriori comandi, per i DBF. Ora realizziamo la stessa applicazione Divina Commedia con Access 2.0.

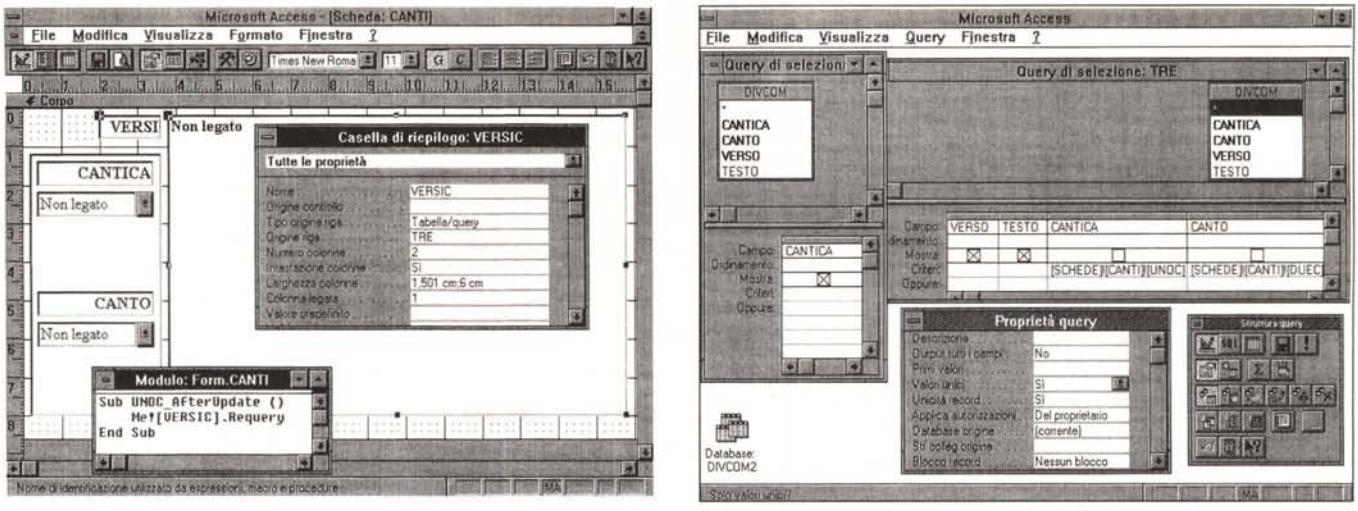


Figure 16,17 - MS Access 2.0 - La Divina Commedia in Access - Dinamicità.
Facciamo riferimento alla Scheda mostrata in figura 15 per documentare come si possa legare il contenuto della lista con i Versi, quella sulla destra, alle scelte, di Cantica e di Canto, eseguite sulle due liste di sinistra. Il meccanismo consiste nel legare la List.Box di destra (che abbiamo chiamato VERSIC) ad una Query le cui regole di selezione siano riferite ai valori impostati nelle due liste di sinistra (che si chiamano UNOC e DUEC). Nella prima figura vediamo come la lista, ovvero la casella di riepilogo, sia originata dalla Query TRE. Nella successiva vediamo la Query Tre «in persona», con i due criteri che vanno a pescare i campi UNOC e DUEC proprio dalla Scheda. Infine va legata l'azione Requery dell'oggetto Versic all'evento After-Update del campo Unoc e Duec.

Se è presente nella Form un Data Control così impostato è possibile collegargli alcuni tipi di Control, ad esempio dei Text.Box, le cui caratteristiche obbligatorie sono:

DataSource — che sarà il nome del Data Control (es. Data1)

DataField — che è il nome del campo il cui contenuto occuperà la Box.

Impostate queste proprietà obbligatorie si può lanciare l'applicazione per vedere se il collegamento «funge» e quindi se effettivamente facendo clic sulle frecce del Control ci si sposta su e giù nella tabella.

Un utilizzo più spregiudicato del Data Control è quello che prevede la sua... soppressione. No, in realtà in certi casi si possono impostare al volo le caratteristiche del Data Control. Sopponendo che si chiami D1:

```
S$="Select * From Indice Order By Argomento"
D1.RecordSource=S$
D1.Refresh
```

Oppure per muoversi tra i record:

```
D1.RecordSet.EOF
D1.RecordSet.MoveNext
```

La Divina Commedia

Anche la Divina Commedia può essere trattata come un file. Volendo strutturare questo file per farlo sembrare un archivio avremo un campo testuale «verso», con il verso, poi necessariamente un campo Cantica (Inferno, Purgatorio o Paradiso), un campo Canto (in totale i canti sono 100).

Potrebbe essere utile un campo con un numero progressivo che indica la posizione del verso, nel canto e nella cantica. Ordinando per Cantica, Canto e Progressivo, ma visualizzando solo i Versi, avremo la nostra Divina Commedia nell'ordine corretto.

Forse sarebbe meglio usare un Word Processor (i versi sono circa 14.000) per impaginare «in bella» l'opera. Volendo assolutamente costruirci un'applicazione Visual Basic si può pensare ad un semplice esercizio di ricerca di Cantica+Canto e di ricerca di Parola nei versi.

In figura 12 la Form con il risultato desiderato. Sulla destra abbiamo una grande Text.Box, con la proprietà MultiLine messa in On, nella quale, con una semplice procedura di ricerca, viene scaricato l'intero canto.

Sulla sinistra un blocchetto con le tre Option.Box che servono per scegliere la Cantica. Più in basso una Combo.Box (caricata all'inizio) con l'elenco dei Canti, in un elegante formato «Numero Romano» (fatto a mano).

Si sceglie la Cantica, si sceglie il Canto e si fa clic su Esegui. Viene creato un Criterio CR (sulla base delle impostazioni precedenti), con il quale viene eseguita una ricerca:

```
SN.FindFirst CR
```

che «becca» il primo Verso della Cantica e Canto desiderato.

Il ciclo, eseguito fino a «rottura» del campo, serve per caricare tutto il canto nella Text.Box.

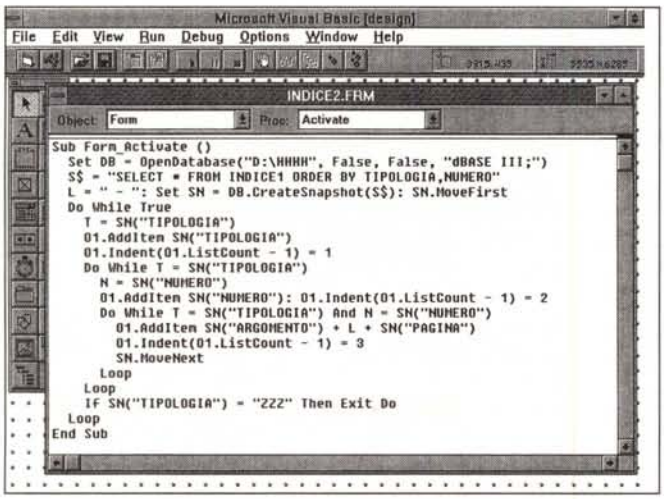
Due considerazioni. La prima è che occorre, riga dopo riga, inserire i salti riga (carattere Ascii 13 e carattere Ascii 10). La seconda è che con questo metodo risulta impossibile risalire alla riga del Canto. Avremmo potuto caricare i versi in una List.Box, che dispone di un suo contatore interno che a questo punto sarebbe stato un contatore gratuito.

In figura 13 vediamo i listati, commentati in didascalia. Nella figura 12 è presente, anche se si vede poco, un Pulsante che serve per cercare un Verso di cui sia nota una parola. In realtà si tratta di una ricerca eseguita tramite la funzione di sottostringa, che risponde (fig.14) Cantica, Canto e Verso e che si ferma solo alla prima ricorrenza.

Sì... ma gli indici

La soluzione proposta non è brillante dal punto di vista prestazionale, perché tutte le ricerche sono... banalmente sequenziali, quindi lente. Se si lavora con file DBF sarebbe possibile (ma non lo facciamo) sfruttare anche gli indici NDX e quindi utilizzare delle istruzioni veloci che usano gli indici, paragonabili alla SEEK o alla FIND del linguaggio dBase. Il nostro FindNext Criteria, usato prima, è invece equivalente al lento LOCATE.

Trasportiamo il tutto in Access. Definiamo il Database Divina Commedia e al suo interno la Tabella Divina Commedia, nella cui struttura impostiamo come



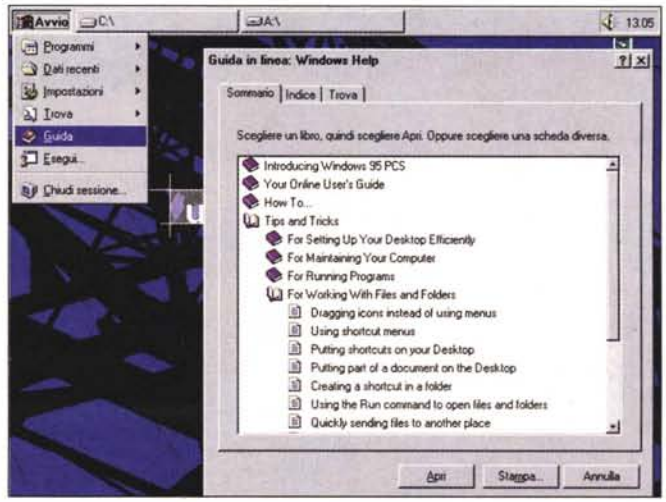


Figura 18 - MS Visual Basic 3.0 - Outline - Costruzione.
Innanzitutto occorre mettere l'elenco in ordine di Rubrica, poi, a parità di rubrica, in ordine di Numero e infine in ordine di Articolo. Per costruire, con Visual Basic 3.0 Pro (che dispone del Control Outline), la struttura occorre eseguire un ciclo sull'elenco, poi a parità di Rubrica un ciclo sui Numeri e poi a parità di Rubrica e Numero un ciclo sugli articoli. Il metodo con il quale si assegna il livello all'elemento della lista è Indent.

Figura 19 - Windows '95 - Outline negli Help.
Personalmente sono favorevolissimo al concetto di Outline, che mi sembra un modo brillante per organizzare e conseguentemente per ricercare i dati. Quindi mi ha fatto molto piacere vedere che in Windows '95 anche gli Help, che a loro volta sono da considerare delle basi di dati testuali (anzi ipertestuali) possono essere organizzati in forma strutturata.

Chiave la concatenazione dei campi Cantica, Canto e progressivo del Verso. Questa chiave fa vedere i dati comunque in ordine, anche se qualche sconsiderato li avesse inseriti in... ordine sparso, e soprattutto rende veloci le ricerche che seguano la regola dell'indice (fig.15). In questa figura vediamo una Form di funzionamento apparentemente analogo a quella realizzata in VB. La differenza sta nel fatto che la Lista sulla destra è alimentata da una Query che viene rieseguita, ogni volta che si cambiano Cantica o Canto, sulla base di cantica e canto prescelte. In pratica viene eseguito un ping-pong tra la Scheda, in cui si scelgono Cantica e Canto, una Query che ha due criteri presi dai due campi della Scheda, e l'ultima Lista sulla scheda, che viene alimentata proprio dalla Query. Per creare questo meccanismo occorre innanzitutto definire o ricordarsi i nomi dei vari oggetti coinvolti, nel nostro caso:
– la Scheda di chiama CANTI,
– il campo Lista dei Cantici si chiama UNOC ed è alimentato da una Query, di nome UNO, di solo Valori Unici (come visto nel caso VB),
– il campo Lista dei Canti si chiama DUEC ed è alimentato come il primo da una Query, di nome DUE,
– il campo Lista dei Versi si chiama VERSIC ed è alimentato dalla Query TRE.

Queste cose le potete vedere (in parte) nella figura 16.

Nella figura 17 invece vediamo un fotomontaggio riguardante la Query UNO (con la specifica Valori Unici) e la TRE. La cosa più importante è la sintassi di quest'ultima che permette di inserire come Criterio della Query TRE i campi UNOC e DUEC della scheda CANTI:

```
[SCHEDE]![CANTI]![UNOC]
```

che significa il campo UNOC della scheda CANTI della classe SCHEDE.

L'aver inserito questi campi rende la Query TRE del tutto asservita alla scheda CANTI.

Per poter far funzionare il tutto occorre infine inserire, e lo facciamo sotto l'evento After-Update, dopo l'aggiornamento, sia del campo UNOC che del campo DUEC, l'istruzione che esegue la Requery della Query e quindi provvede a riaggiornare il campo VERSIC (figura 16, di nuovo).

Nella figura 17 diamo un rapido sguardo all'ambiente Query dell'Access in cui abbiamo inserito un criterio «parametrizzato» che permette di trovare tutti i versi in cui appaia una data parola. La sintassi è la seguente:

```
Like "*"&[CHIAVE]&"*"
```

comporta il fatto che viene chiesta una chiave, alla quale si può rispondere, ad esempio BEATRICE, il criterio diventa *BEATRICE*, con i caratteri jolly aggiunti dalla funzione di concatenamento delle stringhe.

Outline

Un tipo speciale di Lista è quella di tipo Outline, cui siamo molto abituati se usiamo il File Manager di Windows, nella sua parte di sinistra. Ebbene anche il Visual Basic 3.0 (ma solo la versione Pro) permette di creare delle Liste tipo Outline che si caricano alla stessa maniera di quelle normali, con il metodo ADDITEM, con la differenza che in questo caso occorre anche dare un valore di livello al singolo elemento, con il metodo INDENT.

Il nostro scopo è quello di creare una lista di Numeri di MC (usiamo il file già usato prima), poi clic sul numero e appare la lista delle Rubriche, poi clic sulla Rubrica ed appare la lista degli Articoli, e basta. La parte più cospicua del programma è quella che serve per caricare automaticamente la struttura (la vediamo in figura 18) e non la commentiamo più di quanto non sia stato fatto nella didascalia.

Conclusioni

Questo sistema di Outline rende possibile la creazione di liste strutturate, anche di liste molto lunghe, che sono facilmente percorribili in quanto sono organizzate gerarchicamente.

Personalmente sono convinto della efficacia di tale sistema di presentazione dei dati. Altro sistema che mi è sempre piaciuto è quello rappresentato dall'Help di Windows, vero e proprio gestore di ipertesti.

Potete immaginare la mia felicità quando ho constatato che in Windows '95 il nuovo gestore degli Help è in grado di utilizzare Sommari in formato Outline (fig. 19). Prima o poi dovremo parlarne diffusamente. MC