Il Data Control di Visual Basic 3.0
primi approcci

Ricordate? Avevamo un appuntamento.
Nel numero scorso di MC nel presentare la prova del Visual Basic 3.0 vi abbiamo avvertito del fatto che in un successivo articolo, questo, avremmo approfondito uno degli aspetti più interessanti di questa nuova versione di VB, la gestione dei Database

di Francesco Petroni

Si tratta di un tema vasto per due motivi.

Il primo è che comunque l'argomento DB è enorme per conto suo, il secondo è che Visual Basic dispone, nei confronti della gestione dei file con i dati, di svariati strumenti operativi (control, istruzioni, funzioni) e alcuni di questi in certi casi si sovrappongono.

In questo articolo iniziamo a trattare il tema. Iniziamo nel senso che non potremo mai arrivare ad un punto in cui potremo dire «questo è tutto».

Il taglio dell'articolo è molto pratico. Insomma vi proponiamo una serie di esercizi che sarebbe meglio per voi, per una migliore comprensione dei vari comandi, rieseguire passo passo.

Si tratta di esercizi «sintetici» nel senso che vengono evitate tutte le parti non strettamente inerenti la funzionalità che si sta esaminando e questo riduce le operazioni a pochissime e le istruzioni, quando occorre scriverle, a poche righe.

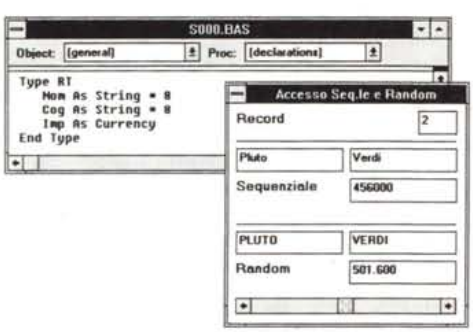


Figura 1 - Visual Basic 3.0 - Ripasso dei metodi tradizionali - Form. Visual Basic non rinnega la sua discendenza diretta dai vari Basic, GWBasic, Quick Basic, ecc. Dai suoi avi ha ereditato le classiche istruzioni di accesso ai file sequenziali e Random. In questa e nella successiva figura riassumiamo i due metodi. Qui vediamo la dichiarazione della variabile RT che identifica la struttura del record del file Random e la Form che visualizza i dati.

Cosa faremo

Dapprima riepilogheremo i metodi tradizionali di accesso ai file, realizzabili anche con Visual Basic versioni 1.0 e 2.0.

Poi passeremo al nuovo Control DataControl (solo VB 3.0) con il quale «aggrediremo», nel senso che lo accerchieremo e attaccheremo da tutte le parti, un Database dBase III. Dapprima un solo archivio DBF, poi un insieme di tre archivi.

Dopo una pausa di riflessione, nella quale tireremo le somme e cercheremo di schematizzare il rapporto Visual Basic e DB, passeremo ad Access, al formato Access che è il formato «nativo» per i Database di Visual Basic.

Alla fine una serie di piccoli «trucchi» utili quando si realizzano delle Form che dialogano con Database.

L'accesso ai Dati. Ripasso delle origini

Questo è un file testuale FixedLenght:

```
Pippo Rossi 345000
Pluto Verdi 456000
PaperinoBlu 567000
```

Questo è un file testuale Delimited:

```
"Pippo","Rossi",345000
"Pluto","Verdi",456000
"Paperino","Blu",567000
```

Visual Basic legge e scrive file sequenziali di questi due tipi. Il migliore dei due è il secondo nel senso che è più facile individuarne la struttura, sia per noi che per qualsiasi prodotto software.

Sui file sequenziali sono possibili solo tre tipi di operazioni:

- Input, lettura a iniziare dal primo record;
- Append, scrittura a iniziare dall'ultimo record (e quindi accodamento);
- Output, scrittura a iniziare dal primo record (e quindi sostituzione dell'eventuale file preesistente).

È chiaro che si tratta di operazioni rudimentali, che permettono solo una gestione «tutto dentro, tutto fuori» dell'archivio.

Tale tecnica può essere comoda per caricare e magari parcheggiare in vettori piccoli archivi, oppure per eseguire elaborazioni Batch, nel senso che il file viene letto riga per riga, la riga subisce un calcolo, viene scaricata e viene subito dopo sostituita dalla riga successiva.

La gestione dei file ad accesso casuale è più sofisticata soprattutto per il fatto che è possibile leggere direttamente un singolo Record, individuato da un numero, senza necessariamente dover leggere tutti i precedenti. Questo comporta la necessità (è intuitivo) di disporre di una lunghezza del Record fissa e dichiarata, in modo che VB possa eseguire il calcolo che dato il numero del record ne individua la posizione nel file.

La struttura del record Random va dichiarata all'inizio e va appoggiata su una variabile (Type Var) che può essere maneggiata con grande facilità.

I dati numerici vanno dichiarati non in termini di numero di caratteri, ma in termini di tipo (Intero, Lungo, Valuta). Questo perché il Basic tramuta il valore da decimale a binario. In questo modo viene risparmiato qualche byte, ma viene persa la possibilità di leggere facilmente il file anche fuori del programma.

Nelle prime due figure dell'articolo vediamo un esercizio di ripasso che contiene le seguenti parti:

- dichiarazione del Record per il file Random,

- dimensionamento dei vettori di appoggio, per i dati del file sequenziale, e definizione della variabile rv, che contiene l'intero record, necessaria per il file Random,
- al caricamento del Form, scrittura del file Sequenziale:

```
Print #1, variabili
```

e scrittura del file Random:

```
Put #2, i, rv
```

- chiusura dei due file,
- apertura e lettura del file sequenziale e trasferimento di tutti i dati nei vettori d'appoggio.

A questo punto il gioco è comandato dalla Scroll Bar cui è delegato il compito di mandare avanti ed indietro il record corrente. Nella Form in alto appaiono i dati del file sequenziale, in basso quelli del file Random, modificati con delle funzioni.

Eccoci ai file in formato dBase

Nella ToolBox esiste in nuovo DataControl, quello a destra nella penultima riga di figura 3. Portato sul Form appare come una ScrollBar con due frecce agli estremi. Per agganciare tale Control ad un file dBase III, occorre definire le seguenti «properties» (a destra nella fig. 3):

- Nome: il nome da usare per riferirsi al Database (default Data1);
- Caption: la scritta che appare sulla parte interna della barra;
- DatabaseName: sarebbe la directory in cui sono i file DBF, nel mio caso C: PARKS;
- Connect: dBASE III, che indica il tipo di file agganciato;
- RecordSource: il nome del file;
- ReadOnly: per definire la scrivibilità o meno del file DBF.

Le altre «properties» sono analoghe a quelle degli altri Control.

Si può lanciare il programma. Se non vengono segnalati errori l'Archivio è collegato... solo che non si vede nulla.

Per vedere i dati occorre definire delle Control che ricevano i dati. Possono ricevere i dati le TextBox, le Label (non editabili), le ListBox, le CheckBox, le ImageBox, ecc.

Ad esempio nel caso si voglia che una TextBox contenga un dato campo occorre definire le sue seguenti «properties»:

- DataSource: ovvero il nome del DataControl cui collegarsi;
- DataField: il nome del campo, presente nella struttura.

L'operazione di aggancio è semplicissima, anche per il fatto che è totalmente guidata, nel senso che via via, nelle varie List Box, Visual Basic propone file DBF e campi disponibili.

Ricordiamo la presenza, nel materiale del Visual Basic 3.0, del programmino accessorio Data Manager, che permette accedere ad un file DBF, sia per leggerne o modificarne la struttura, sia per leggerne o modificarne i dati.

Questo consente di creare un file, magari dotandolo di un indice, e di inserirvi un po' di dati, in modo che si possa testare efficacemente il DataControl da subito (fig. 4).

Il DataControl? Non è indispensabile

In figura 5 vediamo invece come si possa aprire un file DBF senza dover usare lo strumento DataControl.

In questo caso, come in buona parte degli altri esercizi, occorre definire gli

```
Dim n$(3), c$(3), s(3), rv As rt                              ' dimens. vettori

Sub Form_Load ()
    a = Chr$(34): b = Chr$(44): ab = a + b                    ' separatori
    n$(1) = "Pippo": c$(1) = "Rossi": s(1) = 345000           ' def.dati
    n$(2) = "Pluto": c$(2) = "Verdi": s(2) = 456000
    n$(3) = "Paperino": c$(3) = "Blu": s(3) = 567000
    Open "C:\PARKS\DATIS.TXT" For Output As 1                 ' scrive sequenziale
    Open "C:\PARKS\DATIR.DAT" For Random As 2 Len = 24        ' apre random
    For i = 1 To 3
    Print #1, a + n$(i) + ab + a + c$(i) + ab + Str$(s(i))    ' scrive seq.le
    rv.nom = n$(i): rv.cog = c$(i): rv.imp = s(i)
    Put #2, i, rv                                             ' scrive random
    Next i: Close 1: Close 2                                  ' due chiusure
    hscroll1.Min = 1: hscroll1.Max = 3
    l = 1: Open "C:\PARKS\DATIS.TXT" For Input As 1           ' apre sequenziale
    Do While Not EOF(1)
    Input #1, n$(l), c$(l), s(l)                              ' alimenta vettori
    l = l + 1: Loop: Close 1
End Sub

Sub HScroll1_Change ()
    nr = Val(hscroll1.Value)
    text4.Text = nr: text1.Text = n$(nr)
    text2.Text = c$(nr): text3.Text = s(nr)
    Open "C:\PARKS\DATIR.DAT" For Random As 2 Len = 24
    Get #2, nr, rv: Close #2
    text5.Text = UCase(rv.nom): text6.Text = UCase(rv.cog)
    text7.Text = Format(rv.imp * 1.1, "###,###,###")
End Sub
```

Figura 2 - Visual Basic 3.0 - Ripasso dei metodi tradizionali - Listati. La miniapplicazione comporta due eventi. La fase di caricamento, in cui i tre record vengono trasferiti dai vettori N$,C$ e I$ agli archivi DATIS, sequenziale, e DATIR, random. Poi, all'evento «change» della ScrollBar, insomma al variare della ScrollBar, viene associata la visualizzazione dei relativi record nei due file.

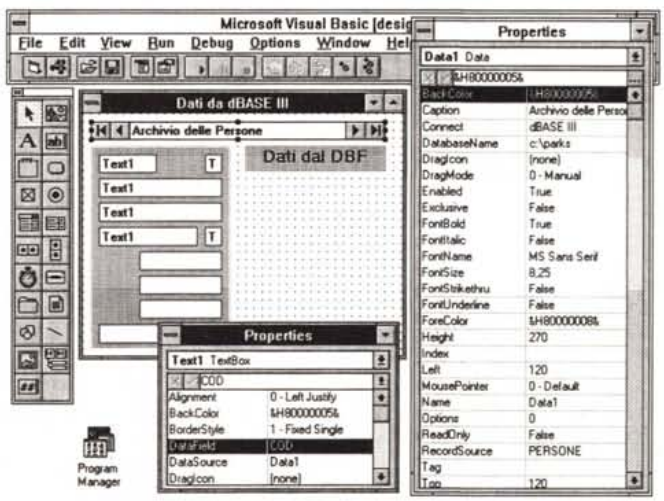


Figura 3 - Visual Basic 3.0 - Data Control - Connessione dBase. Il Data Control si concretizza in una specie di ScrollBar, all'interno della quale si può inserire una Caption, una scritta, che ai margini presenta due coppie di frecce, che servono per spostarsi sul primo record, sul precedente, sul successivo, sull'ultimo. Le properties del Control con le quali si attiva il collegamento sono «dBase Name», che identifica la SubDirectory, Connect, che identifica il tipo di collegamento e il RecordSource, che identifica il File vero e proprio.

oggetti, DataBase e Table, che verranno utilizzati, cosa che si può fare nella sezione Declaration dell'Object General. La riga da scrivere è:

```
Dim miodb as DataBase, miatb as Table
```

Al verificarsi dell'evento Load della Form viene aperto l'archivio PERSONE.DBF, che si posiziona sul primo record.

Vengono attivate una serie di «properties» che descrivono struttura e contenuto dell'archivio e vengono attivate una serie di istruzioni che agiscono sull'archivio. Ad esempio:

```
miatb(1) dà il valore del secondo campo del record corrente
miatb.Eof dà il valore logico True se ci si trova a fine file
miatb.MoveNext posiziona sul record successivo
```

Con queste poche istruzioni siamo in grado di leggere tutto l'archivio. Nel caso presentato poi il contenuto dell'archivio viene riversato in una Griglia.

La griglia è un altro Control, presente già dalla versione 2.0, che dispone di istruzioni di vario tipo che permettono di spostarsi sulle celle della griglia, che permettono di leggere il contenuto delle celle o, al contrario, di definirlo. È utilissima quando si vogliano organizzare dati per riga e per colonna.

Nella figura successiva (fig. 6) vediamo invece una serie di istruzioni che consentono di analizzare la struttura dell'archivio DBF.

In questo esercizio i dimensionamenti debbono essere:

```
Dim SDB as Database, STB as Table, SDF as TableDefs, SFL as Fields
```

Le funzionalità che agiscono sui Database DBF sono numerosissime, in quanto permettono in pratica di simulare tutti i comandi dBASE, sia per la creazione delle strutture che per la manipolazione dei dati.

Il ruolo dell'SQL

Fino ad ora abbiamo parlato di operazioni su un solo archivio.

Per trattare più archivi Visual Basic si appoggia all'SQL, confermando ancora una volta il fatto che questo linguaggio di interrogazione e di manipolazione degli archivi è ormai standard.

Nella figura 7 vediamo sulla destra una piccola Form, che mostra in alto la istruzione SQL con la quale si selezionano quattro campi dell'archivio, sempre DBF, Persone e si filtrano con la condizione Città="ROMA".

In basso nella figura vediamo la definizione degli oggetti. In pratica per utilizzare il risultato di un'interrogazione SQL bisogna dapprima dimensionare una variabile:

```
Dim DB as Database, QD as Dynaset
```

poi per eseguire materialmente la istruzione SQL occorre l'istruzione VB

```
Set DB = OpenDatabase... (come prima)
```

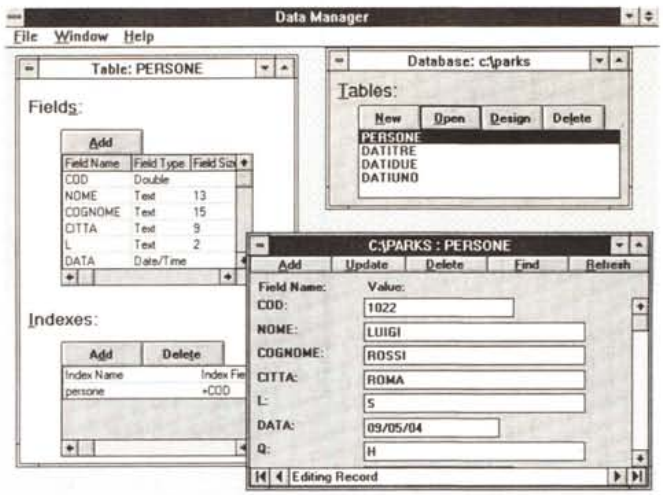
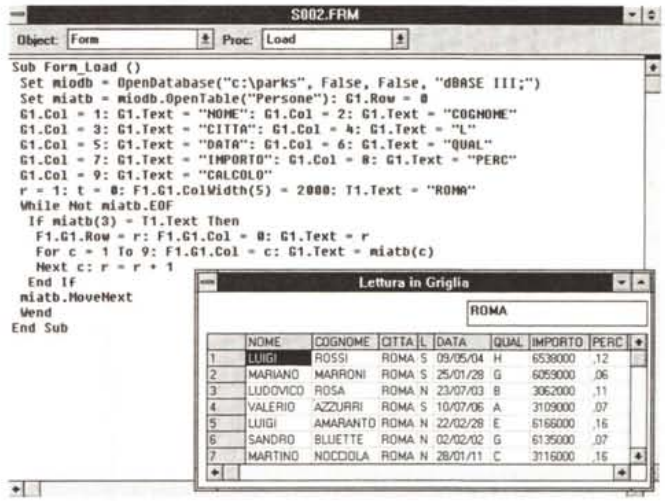


Figura 4 - Visual Basic 3.0 - Data Manager - Preview sul File. Attraverso la voce di menu Data Manager si accede ad un'applicazione (è un eseguibile a tutti gli effetti) che consente di vedere gli oggetti inerenti al Database. Si può notare come nel caso del dBase III il Database sia in realtà una subdirectory che può contenere più archivi DBF. Sfruttando la tecnica MDI si può accedere sia alle strutture che ai dati dei vari archivi. Le une e gli altri sono, in caso di necessità, modificabili.

Figura 5 - Visual Basic 3.0 - Accesso diretto al file DBF. L'accesso al file DBF può essere ottenuto indipendentemente dalla presenza sul Form di un oggetto Data Control. In altre parole VB 3.0 dispone di una cospicua dotazione di nuove istruzioni dedicate alla gestione dei file dati in formato interno (Access), esterno (DBF, Paradox, BTRIEVE), che si aggiungono a quelle tradizionali, presenti anche nei vecchi Basic, di accesso ai file testuali, sequenziali o Random, e binari.



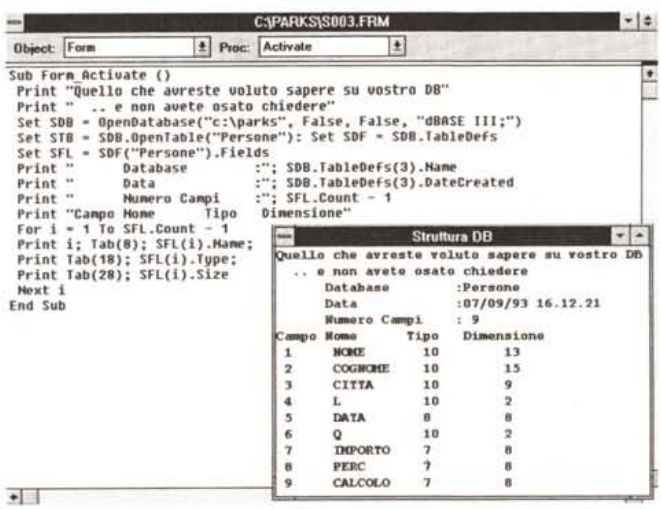


Figura 6 - Visual Basic 3.0 - Accesso alla Struttura. Una volta stabilito il collegamento sono utilizzabili decine di istruzioni che servono non solo per accedere ai dati ma anche per accedere alla struttura dei record. Un uso più evoluto di tali istruzioni è quello che prevede la creazione o la manipolazione della struttura.

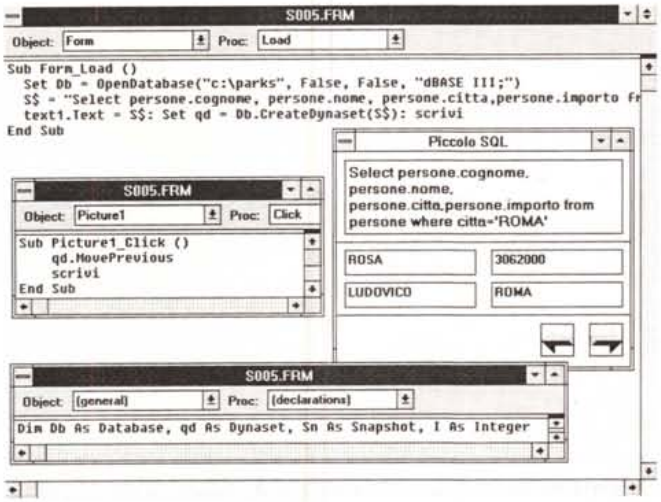


Figura 7 - Visual Basic 3.0 - Accesso via SQL. Il linguaggio SQL si sta affermando come linguaggio comune ai vari prodotti. Anche Visual Basic 3.0 lo riconosce al punto che si possono facilmente creare dei Dynaset o degli Snapshot (tabelle virtuali di dati, aggiornabili o meno) con delle istruzioni eseguibili su vari tipi di file. In questa figura vediamo un'istruzione SQL leggera, ma vediamo soprattutto la semplicità delle istruzioni iniziali che servono per definire i vari oggetti utilizzati (il Database) o creati (il Dynaset).

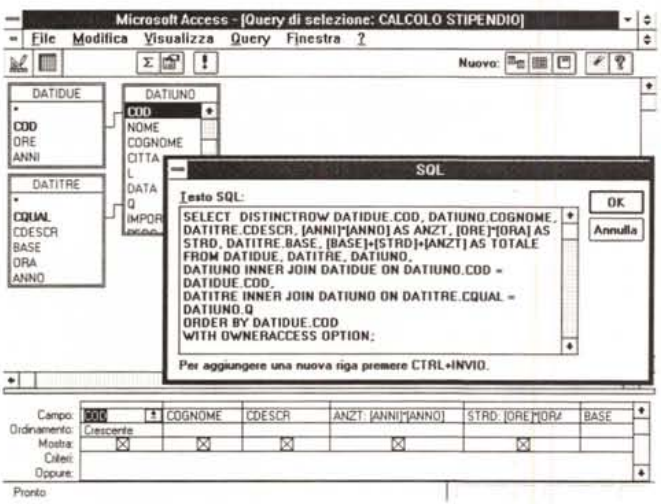


Figura 8 - Visual Basic 3.0 - SQL di Access. Il formato DB «nativo» del Visual Basic 3.0 è l'Access, inteso come formato dati e non come prodotto. Ci sembra quindi logico sperimentare anche l'allineamento delle funzionalità che agiscono sul database tra Access, prodotto, e VB. Ad esempio andiamo a ripescare questa interrogazione SQL, descritta negli articoli relativi ad Access (da MC 128 a MC 130) e vediamo quanto si può riutilizzare in Visual Basic 3.0.

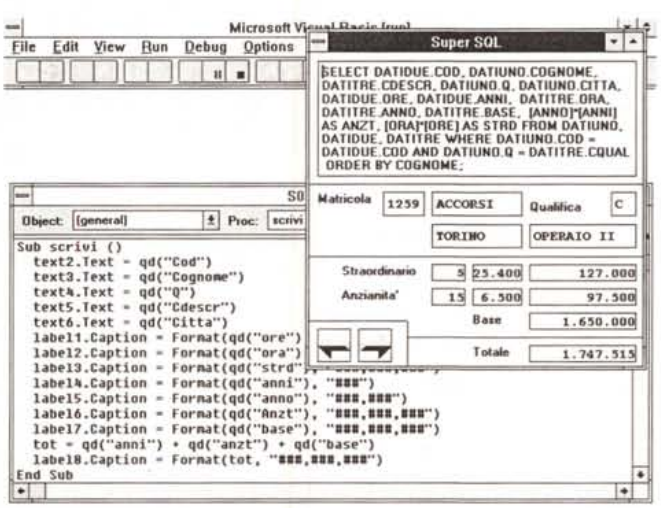


Figura 9 - Visual Basic 3.0 - Istruzione SQL «pesante». L'esperimento è riuscito. In pratica l'istruzione, che vedete nel riquadro in alto, apre tre archivi, li collega con relazioni, ne estrae dei dati, altri ne calcola. Poi il VB tradizionale si preoccupa di piazzare questi dati nelle varie TextBox e Label.

Set QD = DB.CreateDynaset(espressione SQL)

In pratica bastano due comandi.

Ricordiamo che l'istruzione Set è quella che assegna le caratteristiche di un oggetto ad una variabile.

A questo punto esiste l'oggetto QD, che è il Dynaset, una tabella «virtuale» creata dalla istruzione SQL. Su tale tabella possono agire i comandi di visualizzazione, che noi, sempre riferendoci alla figura, abbiamo concentrato in una Sub «scrivi», e i comandi di movimento tra i record della tabella.

Verificata la possibilità di eseguire un'istruzione SQL «facile», ci proponiamo subito l'obiettivo di eseguire una istruzione SQL più complessa.

Usiamo questa volta tre archivi DBF e l'istruzione la riprendiamo da un esercizio proposto in uno degli articoli pubblicati da MC su MS Access (figg. 8 e 9). Il caso studio è quello presentato nel nostro minicorso di Access al quale vi rimandiamo per una comprensione del significato dei tre archivi.

In effetti, con alcune piccole varianti, l'SQL riconosciuto da Visual Basic è altrettanto efficiente.

Nella figura 8 vediamo l'ambiente Access con, sullo sfondo, la struttura dei tre archivi usati, e i segmenti che rappresentano le relazioni, in basso la struttura della tabella virtuale «Dynaset» in uscita, e in primo piano l'istruzione.

Nella figura 9 vediamo la stessa istruzione eseguita su tre archivi DBF (analoghi a quelli usati per Access, solo in formato DBF) posta in una Text.Box, assieme ad una serie di campi in uscita, riempiti dalla routine Scrivi, mostrata in secondo piano. La routine Scrivi viene «azionata» dai due Bottoni con le frecce che fanno scorrere nelle due direzioni il record corrente.

La sezione dichiarazioni è identica a quella dell'esercizio precedente, in quanto la dichiarazione del Dynaset è indipendente dalla complessità dell'istruzione e dal numero dei campi. Per chi non conoscesse l'SQL precisiamo che l'istruzione comprende più «clausole»:

SELECT, che serve per indicare i campi desiderati,

C1*C2 AS NOME, esempio di campo calcolato, che non c'è negli archivi ma viene prodotto solo in uscita, e al quale in uscita viene attribuito un Nome,

WHERE A1.C1=A2.C2, che mette in relazione i due archivi A1 e A2, tramite ii loro rispettivi campi C1 e C2,

ORDER BY C1, per mettere in ordine i dati del Dynaset rispetto ad un campo di uno degli archivi.

Su SQL dobbiamo dire due o tre cose. La prima è che a pagina 125 del manuale Professional Features Book 2 c'è una sezione intitolata Microsoft Access SQL, che illustra tutte le caratteristiche dell'SQL di Access e le confronta anche con ANSI SQL, quello ufficiale.

La seconda è che SQL in realtà consente anche delle operazioni di gestione del Database, con creazione di Tabelle e loro alimentazione, mentre nei

nostri due esercizi ci siamo limitati ad interrogazioni.

La terza e ultima considerazione è che siamo un po' agli inizi, non per nulla l'articolo è intitolato Primi Approcci. Per usare SQL pesantemente occorre una perfetta padronanza di tutte le sue numerose sfaccettature, padronanza che si ottiene con il tempo e con l'ausilio di testi che ancora non esistono. Anche il manuale VB dedica a tale complesso argomento poche pagine, destinate peraltro a chi già conosce SQL, e del tutto prive di esempi pratici.

Il ruolo degli Array di Control

Facciamo un piccolo intervallo nel quale vi proponiamo un programma di calcolo molto dinamico (fig.10).

L'interesse del programma sta nella brillante gestione degli Array.Controls. L'Array.Control è un insieme di Control omogenei ai quali si può associare un unico evento che riguarda tutti i componenti, che vengono identificati da un indice.

Le «properties» dei vari componenti dell'Array possono a loro volta essere considerate parti in causa dell'Array stessa.

Ad esempio si supponga di dover utilizzare otto TextBox, dalle caratteristiche simili. Tracciata la prima la si può copiare. VB chiede se si vuole creare una Array.

Se i vari Control attivano la stessa Subroutine è conveniente creare l'Array. In pratica invece di n Control:

```
T1.Text
T2.Text
T3.Text
```

se ne avrebbe uno solo

```
T1(index).Text
```

identificato da un index che viene definito dal VB stesso.

Ne vediamo un esempio chiarificatore in figura 10. Nel nostro programma vogliamo eseguire uno scorrimento del solito archivio Persone, del quale ci interessano i campi Città e Importo. Durante lo scorrimento vogliamo riempire una serie di Text.Box con il conteggio delle frequenze della città, indicate in una Array di Label, e del totale dell'importo per città.

Il valore della città, ad esempio Roma, è una delle «properties» della Label. Insomma si può scrivere un'istruzione del tipo:

```
If label1(index).caption = "ROMA" then...
```

senza dover, ad esempio, creare delle Array tradizionali con il valore della singola Città.

È chiaro che il programma diventa più snello, nel nostro caso invece di otto subroutine ne abbiamo una sola.

Il programma esegue come detto lo scorrimento dell'archivio, record per record. Poi per ogni record viene eseguito un ciclo per individuare quel record a quale città corrisponde. Trovata la città vengono incrementati sia il contatore

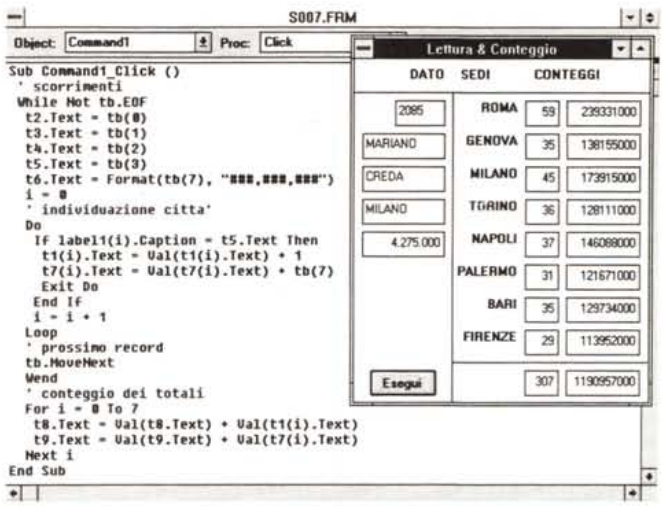


Figura 10 - Visual Basic 3.0 - Un programma di Calcolo. In questo esercizio vi proponiamo un programma di calcolo molto dinamico. L'interesse del programma, anche questo descritto con maggior dettaglio nel testo, sta nella brillante gestione degli Array.Control. L'Array.Control è un insieme di Control omogenei ai quali si può associare un unico evento che riguarda tutti i componenti che vengono identificati da un indice. Le properties dei vari componenti dell'Array possono a loro volta essere considerati dati riferibili all'Array stessa.

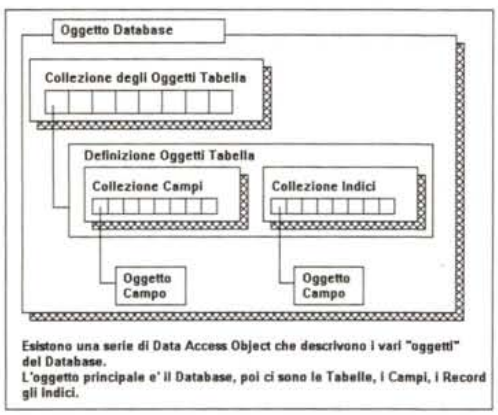
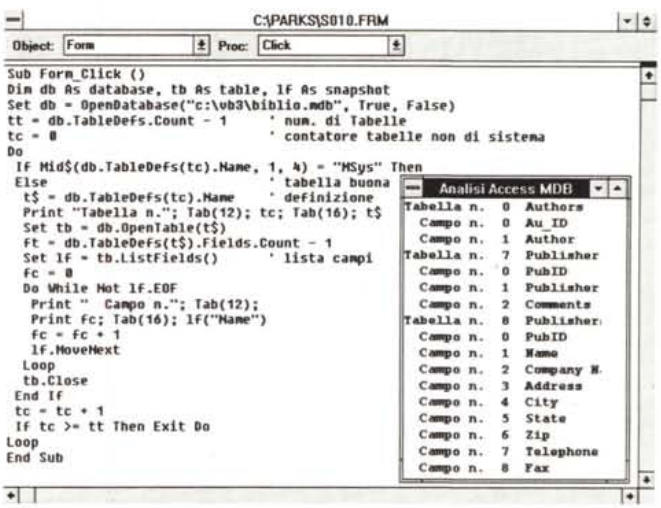


Figura 11 - Visual Basic 3.0 - Schema della gerarchia degli oggetti. Nel trattare i Database (non parliamo più di singolo archivio) occorre capire come sono organizzati gli oggetti. Esiste l'oggetto Database, che contiene una «collezione» di oggetti Tabella. Ogni oggetto Tabella contiene una collezione di oggetti Campo e una collezione di oggetti Indice. Ognuno di questi oggetti dispone di «properties» che ne individuano sia le caratteristiche (es. nome, tipo, lunghezza di un campo) sia il contenuto (es. valore del campo).

Figura 12 - Visual Basic 3.0 - Vivisezione di un file Access. Questo programma individua l'organizzazione di un qualsiasi DB Access (basta cambiare il nome del file nell'istruzione OpenDatabase). Ne individua e visualizza sia le Tabelle e per ciascuna Tabella i campi.



delle frequenze, sia il totalizzatore degli importi. Dopodiché si passa al prossimo record.

Alla fine vengono calcolati anche i due totali con un piccolissimo ciclo.

Organizzazione per Oggetti

Con Visual Basic per maneggiare i Database (non parliamo più di singolo archivio) occorre assolutamente capire come sono organizzati i vari oggetti coinvolti (fig.11).

Esiste l'oggetto Database, che contiene una «collezione» di oggetti Tabella. Se si usa il formato dBase il Database è la directory con i file, e le Tabelle i file DBF. Invece Access non usa singoli archivi ma definisce, nel file MDB, l'insieme degli archivi e delle loro caratteristiche.

Ogni oggetto Tabella contiene una collezione di oggetti Campo e una collezione di oggetti Indice, identificabili, definibili, manipolabili ciascuno con specifiche istruzioni.

Ognuno di questi oggetti dispone infatti di «properties» che ne individuano sia le caratteristiche (es. nome, tipo, lunghezza di un Campo) sia il contenuto (es. valore del campo).

Il tutto rientra nella filosofia della programmazione Object Oriented, in cui esistono oggetti, esistono proprietà degli oggetti, esistono gerarchie tra gli oggetti. Ad esempio un Campo è un componente di una Tabella, a sua volta componente del Database, così come una TextBox può essere un componente di una Form.

Ancora una volta occorre riferirsi a concetti generali, riutilizzabili, per cui una volta «capiti la prima volta» rimangono validi per sempre.

Il formato dati di Visual Basic Access

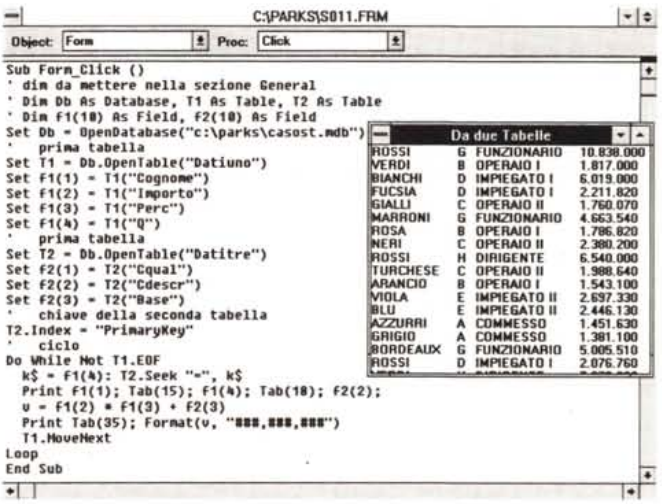
Come specificato nella prova, della quale questo articolo rappresenta il seguito, il formato «nativo» del Database di Visual Basic è Access, inteso come formato file e non come prodotto Microsoft.

Lo studio delle funzionalità di Database va quindi fatto riferendosi a tale formato che deve essere capito soprattutto nella sua organizzazione.

Con Access si gestiscono Database, composti di varie tipologie di oggetti, le Tabelle, le Query, le Schede, i Report, le Macro e le Procedure. Il tutto viene memorizzato in un unico grosso file, che ha desinenza MDB.

Anche Visual Basic utilizza file MDB. Può generare e gestire file MDB, può gestire file MDB realizzati con Access.

Figura 13 - Visual Basic 3.0 - Seek.
Qui vediamo come si possano aprire due Tabelle e come, partendo da un campo di collegamento, si possa ricercare il corrispondente valore sull'altro Archivio. Insomma il problema del collegamento tra due file «relazionati» si può risolvere o ricorrendo all'SQL, oppure ricorrendo a questa istruzione Seek, il cui uso è del tutto analogo alla Seek del dBase III.



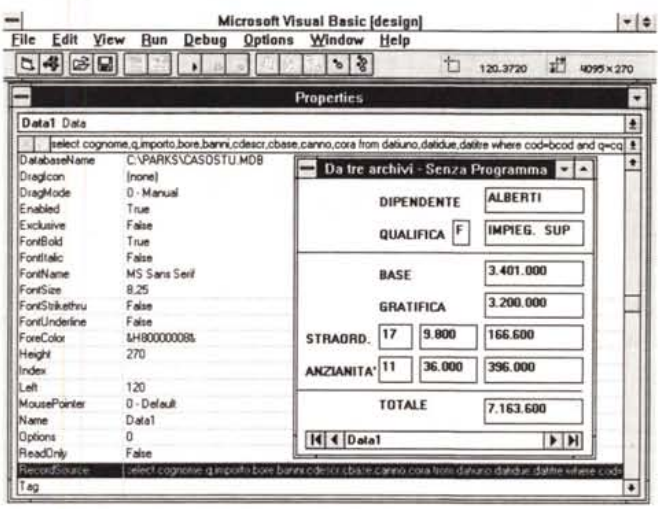


Figura 14 - Visual Basic 3.0 - Da tre Archivi Senza programmare Vista dal Data Control.
Come d'accordo non si programma. L'unica cosa da fare è costruire nella property RecordSource dell'oggetto Data Control l'istruzione SQl che genera il Dynaset. Poi al momento di definire il RecordSource di ciascuna Text Box si potrà attingere alla List Box che mostra tutti i campi resi disponibili dalla riga SQL. L'unico appunto sta nella ristrettezza della riga dell'editor delle properties.

In realtà da Visual Basic non si può accedere a tutti gli oggetti di un DB Access, ad esempio non si può utilizzare una Macro, non si può utilizzare direttamente una Form o un Report, e non si possono eseguire alcuni dei comandi eseguibili con Access.

D'altra parte se si deve da Visual Basic usare Form e Report realizzati con Access diventa inutile il Visual Basic stesso, tanto è fare tutto con Access.

Lo sfruttamento totale delle possibilità comporta un discorso molto complesso, in quanto occorre valutare cosa si riesce a fare con Visual Basic, e cosa non si riesce a fare con Visual Basic di quanto invece si riesce a fare con Access.

Per ora, in attesa dei dovuti approfondimenti che stiamo facendo e sui quali vi informeremo, limitiamoci ad alcune cose essenziali.

La prima cosa che facciamo (fig.12) è un programmino generalizzato che analizza la struttura di un qualsiasi Database Access, individuandone le Tabelle e per ciascuna Tabella i Campi.

In questo caso le definizioni, che abbiamo, scorrettamente, legate come tutto il programma all'evento Click sul Form, riguardano il Database, la Tabella generica e lo SnapShot che è un Tabella virtuale, risultato di una Query, che differisce dal Dynaset, per il fatto che non è aggiornabile.

Aperto il Database c'è un Ciclo che va avanti fin quando ci sono Tabelle (il numero delle tabelle non è noto a priori). Vengono, con un'istruzione IF, saltate le Tabelle di sistema (interne al file MDB il cui nome comincia con «MSys»).

Per ogni tabella individuata ne viene letta la lunghezza in termini di numero di campi (db.TableDefs(nomtab)Fileds.-Count) necessaria per poter scorrere ed

```
Dim v1, v2, v3, f$

Sub Form_Load ()
    v1 = 0: v2 = 0: v3 = 0: f$ = "###,###,###"
End Sub
Sub T1_GotFocus ()
    t1.Text = Str$(Val(v1))
    t1.SelStart = 0: t1.SelLength = Len(t1.Text)
    l4.Caption = "Digitare il numero di pezzi venduti"
End Sub
Sub T1_LostFocus ()
    v1 = Val(t1.Text): t1.Text = Format(v1, f$)
    v3 = v1 * v2:t3.Text = Format(v3, f$):l4.Caption = ""
End Sub
Sub T2_GotFocus ()
    t2.Text = Str$(Val(v2))
    t2.SelStart = 0:    t2.SelLength = Len(t2.Text)
    l4.Caption = "Digitare la quantita' venduta"
End Sub
Sub t2_LostFocus ()
     v2 = Val(t2.Text): t2.Text = Format(v2, f$)
     v3 = v1 * v2: t3.Text = Format(v3, f$): l4.Caption = ""
     If v3 < 1000000 Then
       MsgBox ("Valore Minimo 1.000.000")
       t2.SetFocus
     End If
End Sub
```

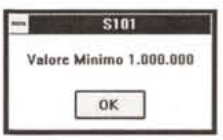


Figura 15 - Visual Basic 3.0. Qui vediamo come si possa addomesticare una Text.Box per gestire un'immissione numerica controllata. In caso di campi calcolati o di campi che debbano subire dei controlli risultano molto utili alcuni... eventi. Il GetFocus, quando il cursore «entra» nella Text.Box e quindi occorre attribuirgli il valore iniziale, e se del caso, selezionarlo, il LostFocus, quando si «esce» e si possono eseguire le istruzioni di controllo. L'istruzione T2.SetFocus, riporta il cursore nella TextBox T2.

moltiplicazione dei due valori (fig.15).

Semplicissimo, però vogliamo risolvere questi problemi:

- i numeri debbono apparire nel formato ###,###,### e debbono essere allineati a destra. La soluzione è usare l'istruzione Format, che converte un numero in una stringa del formato desiderato. L'allineamento a destra è una property della TextBox, che si attiva solo se la property MultiLine è True,
- il prodotto è accettato solo se dà un risultato superiore a 1.000.000. In caso negativo viene mandato una MsgBox di avvertimento dal quale si esce con il bottone OK,
- nel caso in cui il risultato non sia accettato il cursore deve tornare sulla seconda Text.Box. È l'istruzione SetFocus che porta il cursore nel punto desiderato,
- vogliamo gestire una Label in cui dinamicamente inseriamo un messaggio relativo alla situazione in cui ci si trova,
- in ogni caso entrando in una TextBox vogliamo che il contenuto precedente sia già selezionato. Questo che è il comportamento standard nelle Dialog Box di Windows si ottiene, un po' macchinosamente, usando le specifiche SetStart e SelLength, che però attiviamo sul valore delle Text.Box depurato dei caratteri di separazione.

Gli eventi gestiti sono, in questo caso, il GotFocus, che indica l'entrata nel Control, e il LostFocus, che indica l'uscita.

In entrata si seleziona il contenuto, in uscita si eseguono i calcoli, si sistemano i formati, si eseguono i test sui valori.

È chiaro che se le TextBox sono tante e se subiscono operazioni analoghe può essere opportuno usare delle Array di Control.

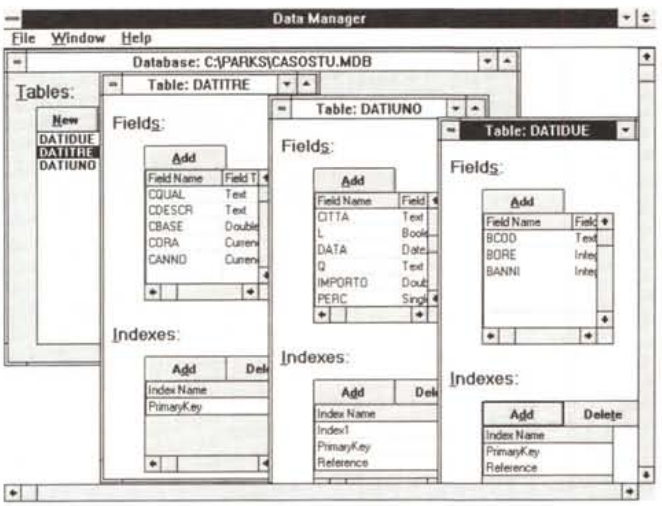


Figura 16 - Visual Basic 3.0 - Da tre Archivi Senza programmare Vista del Database. Volete risolvere tutti i vostri problemi dal Data Control e senza dover scrivere una riga di programma (o quasi). La prima cosa da fare e conoscere bene la struttura del Database che si intende manipolare. Per far questo usiamo ancora una volta il DataManager, che, rispettando la tecnologia MDI, permette di aprire più oggetti in contemporanea.

elencare tutti i campi.

Nella figura vediamo sia il listato tutto concentrato sotto l'evento Click sul Form che il risultato sul Form stesso.

Nel successivo esercizio lavoriamo con due archivi. Ci ripromettiano di analizzare il funzionamento dell'istruzione Seek (fig.13).

In pratica scorriamo un archivio (il solito del caso studio) che si chiama Datiuno. Questo archivio contiene un campo Q, che è in corrispondenza con il campo Cqual, dell'archivio Datitre. Il campo Cdescr di tale archivio costituisce in pratica la descrizione del codice Q presente nel primo archivio.

L'istruzione Seek, del tutto analoga a quella presente nel linguaggio dBase, permette di localizzare direttamente il record nel file che deve essere necessariamente dotato di un Indice.

Insomma si può fare anche questo, collegare due archivi, un po' rudimentalmente (senza scomodare in SQL), ma altrettanto efficacemente.

Senza programma

Nella figura 14 vediamo invece come sia possibile inserire un'istruzione SQL come property RecordSource di un Data Control. Questo consente, senza dover scrivere una sola riga di programma, si accedere ad un Database con più tabelle. Poi, continuando a non scrivere pezzi di programma, è possibile agganciare gli oggetti del Form ai singoli campi, siano essi campi esistenti o campi calcolati, per vederne il contenuto.

Piccoli trucchi del mestiere

Prima di chiudere vediamo un piccolo programma nel quale vengono affrontati quei piccoli problemi che si incontrano quando, in fase di input di dati, occorre fare un po' di estetica e un po' di controllo sui campi.

Supponiamo di avere tre TextBox, nelle prime due digitiamo dei numeri, nella terza mettiamo il risultato della

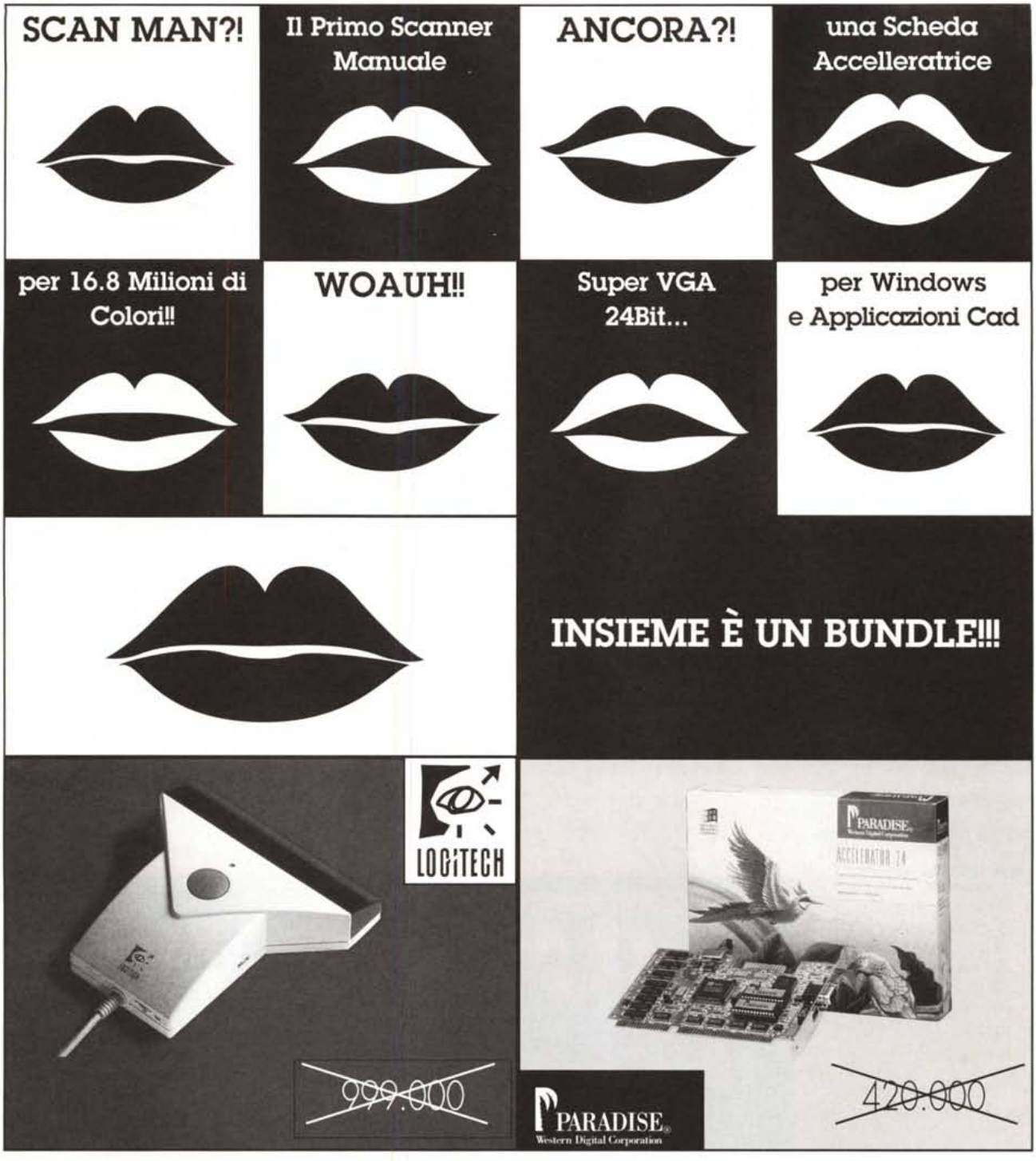
Conclusioni

Al solito non abbiamo parlato di tante cose. Non abbiamo parlato molto di gestione (modifica, inserimento, cancellazione), non abbiamo parlato per niente dell'uso degli Indici.

Abbiamo comunque visto un bel po' di cose per cui possiamo affermare che i nostri primi approcci sono stati positivi, vedremo in seguito di renderli ancora più produttivi.

Con Visual Basic è possibile gestire, in vario modo, i Database.

Se i Database sono in formato esterno, DBF, Paradox, BTRIEVE, oppure Access, l'utente deve padroneggiare innanzitutto tale formato, deve sapere come è fatto, cosa contiene, quali tipi di dati, ecc. Può studiare questa «materia», se non dispone di altri prodotti più vicini ai formati di suo interesse, direttamente dal Data Manager (fig.16). MG



Lire 980.000*

*PREZZO UTENTE IVA ESCLUSA

MIXEL

36040 - Torri di Quartesolo (VI) - Via Roma, 171
Tel. 0444/583998 - 380799 r.a.

PER MAGGIORI INFORMAZIONI SPEDIRE IL COUPON A:
MIXEL S.r.l. Via Roma, 171 - 36040 - TORRI di QUARTESOLO (VICENZA)
Fax 0444/583994

NOME, COGNOME N.
VIA CITTÀ C.A.P.

Per entrare nel mondo delle tecnologie e dei prodotti dedicati all'ascolto in automobile c'è una strada sicura: le pagine di Audiocarstereo. Recensioni dagli alti contenuti tecnici, prove di installazione, un vasto panorama di aggiornamenti mensili - anche sui prezzi - sono una lettura obbligata per i professionisti del settore come per i semplici appassionati, e costituiscono il migliore osservatorio per ascoltare al meglio. Infine le sezioni dedicate alla telefonia cellulare, ai test sugli antifurto, alle recensioni musicali completano Audiocarstereo, accompagnando chiunque voglia percorrere in auto la strada dell'alta fedeltà.

La strada migliore per l'alta fedeltà in auto.

AUDIOCARSTEREO. Per superare i limiti di alta fedeltà.

technimedia
Pagina dopo pagina, le nostre passioni.