Alcune modalità di selezione dei dati sperimentate con MS VB5

In termini musicali si chiamerebbero "variazioni sul tema". Quello che vogliamo fare è risolvere lo stesso problema di accesso ai dati, un problema molto facile, in tanti modi differenti, sfruttando al meglio i vari componenti che Visual Basic 5.0 mette a disposizione dei programmatori, da quelli più semplici, che fanno parte della dotazione di base di Visual Basic, a quelli più evoluti, ad esempio quelli che vengono usati anche nell'interfaccia Win 95.

di Francesco Petroni

Il nostro poblema e le sei soluzioni che sperimenteremo

Abbiamo un Database, in formato MS Access, che si chiama DATI.MDB e che contiene una sola tabella che si chiama persone. I campi della tabella si chiamano cod, cognome, nome, città, qualifica, data, importo.

I due campi città e qualifica sono campi di aggregazione e li utilizzeremo per eseguire le nostre selezioni.

Il problema, come detto, consiste non tanto nell'eseguire una selezione per città e/o per qualifica, quanto nel presentare, in vari modi, i dati selezionati.

Per eseguire la selezione sfrutteremo o il componente DataControl oppure la modalità DAO. In ambedue i casi, che in fondo si assomigliano tantissimo, occorre comporre un'istruzione SQL, da passare come RecordSource al DataControl o per definire il RecordSet di DAO.

Ad esempio se vogliamo l'elenco delle persone di Roma in ordine alfabetico l'istruzione SQL è:

```
SELECT * FROM PERSONE WHERE CITTA="ROMA" ORDER BY COGNOME
```

Invece quando occorre assegnare tale stringa ad una variabile:

```
S = "SELECT * FROM PERSONE WHERE CITTA='ROMA' ORDER BY COGNOME"
```

Nasce la complicazione delle virgolette che servono sia per impostare la stringa che nella sintassi SQL. SQL accetta per la definizione dei criteri di selezione anche la singola virgoletta. Se invece si devono assolutamente usare le virgolette (ad esempio nel caso in cui dovessimo selezionare solo le persone di L'Aquila) possiamo comporre la stringa usando i caratteri ASCII:

```
S = "SELECT * FROM PERSONE WHERE CITTA=" & CHR(34) & "ROMA" & CHR(34)
```

in cui il carattere & (la cosiddetta "e" commerciale) serve per concatenare le stringhe ed il carattere ASCII 34, ottenuto con la funzione CHR, sono proprio le virgolette.

Nel caso infine in cui il nome della città fosse fornito da un controllo, ad esempio da una ComboBox, che si chiama C1, la stringa va così definita:

```
S = "SELECT * FROM PERSONE WHERE CITTA=" & CHR(34) & C1 & CHR(34)
```

Se l'istruzione è troppo lunga la si può costruire "a pezzi", sommando via via i vari pezzetti della stringa.

In questo caso è anche possibile inserire varianti nella costruzione della stringa stessa. Ad esempio:

```
S = "SELECT * FROM PERSONE WHERE CITTA=" & CHR(34) & T1 & CHR(34)
IF T2 <> "TUTTE" THEN
```

```
        S = S & " AND QUALIFICA = "
& CHR(34) & T2 & CHR(34)
  END IF
  IF T3 = "SI" THEN
        S = S & " ORDER BY COGNO-
ME"
  END IF
  MSGBOX S
```

La stringa risultante dipende dal valore delle due variabili T2 e T3.

La prima contiene la qualifica, ma nel caso contenga la parola "TUTTE", significa che si desiderano tutte le qualifiche. In questo caso non si aggiunge la parte "AND QUALIFICA ...".

Se la variabile T3 è uguale a "SI" si vuole l'elenco ordinato per cognome ed in questo caso occorre aggiungere la parte "ORDER BY...".

Un'ultima avvertenza: controllate, anche a vista con una message box che la visualizza, la correttezza dell'istruzione SQL, ovvero della stringa S. Attenzione, ad esempio, quando sommate i pezzetti, ad inserire i caratteri blank, necessari per separare i pezzetti, al punto giusto.

Costruita correttamente la stringa SQL, la si può passare al DataControl, le cui proprietà fondamentali sono:

Figura 1 - Microsoft Visual Basic 5.0 - Enterprise Edition in Italiano. Come al solito, anche la versione 5.0 del Visual Basic della Microsoft è stata tradotta in Italiano. Questo non modifica assolutamente il normale lavoro di programmazione, in quanto la traduzione non altera la parte codice ma solamente gli elementi dell'ambiente operativo, i menu, le finestre di dialogo e la documentazione, sia quella in linea che quella cartacea. La manualistica del Visual Basic è voluminosissima ed importantissima, soprattutto per chi volesse "tirargli" il collo anche nelle funzionalità più evolute, come quelle che riguardano la programmazione Client/Server.

Abbiamo i dati. Ma dove li mettiamo?

Sia che usiamo il DataControl sia che usiamo le tecniche DAO ci ritroviamo con un RecordSet, che corrisponde ad un insieme di dati, in pratica ad una tabella virtuale fatta di record e di campi. Per scorrere i dati si usano due routine standard.

Connect:	Access	è il valore di default
DatabaseName:	DATI.MDB	occorre passare anche la path
RecordSetType:	SnapShot	perché non dobbiamo eseguire aggiornamenti
RecordSource:	S	la stringa appena costruita.

Ogni volta che si ridefinisce il RecordSource occorre anche eseguire il metodo Refresh che serve per "ricalcolare" i dati:

```
Data1.Refresh
```

Se invece si usano le tecniche DAO (Data Access Object) non serve il DataControl:

```
Dim DD As Database
Dim RR As Recordset
Set DD = OpenDatabase("DATI.MDB")
S = "SELECT * FROM PERSONE
WHERE CITTA='ROMA' ORDER BY
COGNOME"
Set RR = DD.OpenRecordSet(S,4)
```

Le istruzioni, come si vede, sono del tutto analoghe.

Nel caso del DataControl, se lo si chiama D1:

```
D1.RecordSet.MoveFirst
Do While Not D1.RecordSet.Eof
        ...
        D1.RecordSet.MoveNext
Loop
```

Invece con DAO, se il RecordSet RR è quello definito prima:

```
RR.MoveFirst
Do While Not RR.Eof
        ...
        RR.RecordSet.MoveNext
        Loop
```

A questo punto dobbiamo scegliere dove e come vedere il risultato dell'interrogazione SQL.

Trattandosi di un RecordSet, ovvero di una tabella virtuale, l'oggetto più adatto è una griglia, in cui ci sono le colonne, che rappresentano i campi, e le righe, che invece sono i record.

Si possono usare sia la DBGrid che la nuova (c'è solo in VB5) FlexGrid. E' sufficiente definirne la proprietà DataSource, si indica il nome del DataControl, per ottenere automaticamente i dati.

Se invece si usano i controlli più tradizionali, come le Combo e le List (che però sono monocolonna), occorre caricarle da programma, usando il metodo AddItem all'interno delle routine viste prima. Ad esempio per caricare una Combo, che si chiama C1, con l'elenco delle città estratte dalla tabella persone:

```
C1.Clear
S = "SELECT DISTINCT CITTA FROM
PERSONE"
Set RR = DD.OpenRecordSet(S,4)
RR.MoveFirst
Do While Not RR.Eof
        C1.AddItem RR!CITTA
        RR.RecordSet.MoveNext
Loop
C1.ListIndex = 0
```

Le proprietà importanti della Combo sono:

C1.ListIndex = X per settare o per leggere il numero dell'elemento

C1.ListCount proprietà (sola lettura) che indica il numero di elementi della Combo.

Oltre alle ListBox ed alle ComboBox, che vanno caricate da programma, ci sono anche le DBList e le DBCombo,

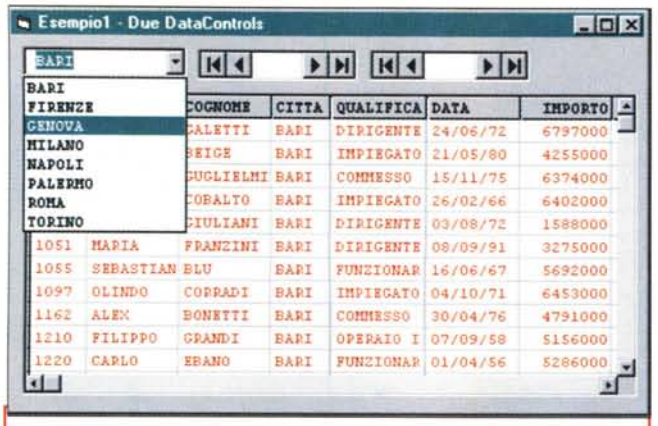

Figura 2 - MS VB5 - Due DataControl – Output.
Abbiamo una tabella contenente dati relativi a persone, potrebbe trattarsi dell'elenco del personale di un'azienda che abbia sedi in varie città. Un altro campo che utilizzeremo per i nostri esercizi è quello con la qualifica. Scopo dell'articolo è quello di realizzare sostanzialmente la stessa cosa, la selezione delle persone per città e per qualifica, in vari modi. Il database, che contiene solo la tabella persone, è in formato MDB, quello di Access, mentre gli esercizi sono scritti in Visual Basic 5.0.

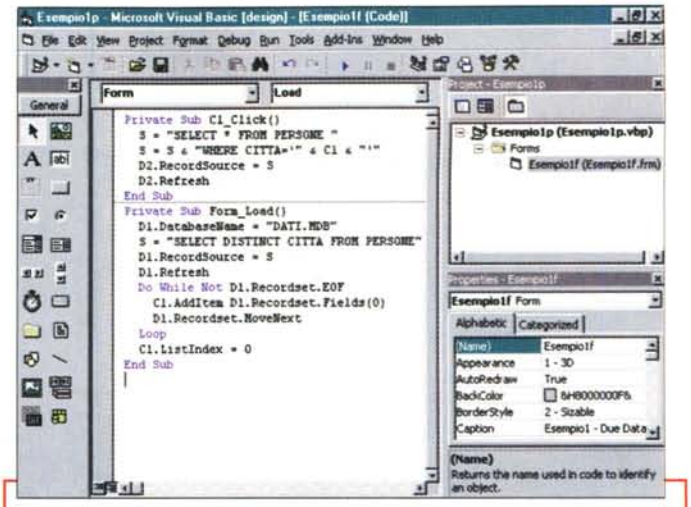

Figura 3 - MS VB5 - Due DataControl – Listato.
Nel primo esercizio, di cui abbiamo visto nella figura precedente l'aspetto esteriore, utilizziamo due DataControl, una ComboBox (non una DBCombo) che si chiama C1 e che alimentiamo manovrando il RecordSet messo a disposizione dal secondo DataControl (D2) ed una DBGrid, il cui nome è G1, e che è collegata al primo DataControl D1. Se si realizza l'esercizio con Visual Basic 5.0 si può usare anche una FlexGrid. All'evento Form Load carichiamo la D2 con l'elenco delle città presenti nella tabella persone. La sincronizzazione tra città scelta nella Combo e dati visualizzati nella Grid si ottiene gestendo l'evento Click sulla Combo. Come noto la DBGrid viene collegata, tramite la sua proprietà DataSource, ai dati del DataControl per cui il programma deve manipolare solo quest'ultimo.

più facili da agganciare ai dati, sempre attraverso un DataControl, ma oggetti aggiuntivi che... occupano memoria. Ricapitoliamo comunque le proprietà più importanti di una DBCombo (o DBList, che funziona allo stesso modo).

C1.RowSource nome del DataControl che fornisce i dati
C1.ListField campo da visualizzare
C1.Text (è il valore di default) valore visualizzato
C1.BoundColumn campo che dà il valore a C1 (non necessariamente il C1.ListField)
C1.BoundText valore vero del controllo C1
C1.DataSource nome del secondo DataControl da usare nel caso in cui C1 serva ad aggiornare i dati
C1.DataField campo da aggiornare.

Insomma il buon programmatore deve conoscere bene tutti i controlli e deve scegliere, a seconda delle condizioni al contorno, quale utilizzare.

I primi due esercizi

Nel primo esercizio (figure 2 e 3) usiamo due DataControl.

Il primo (si chiama D1) viene caricato all'evento Form Load e serve ad alimentare una ComboBox (si chiama C1) con un elenco di città. Lo facciamo scorrendo il RecordSet presentato da D1.

Il secondo (D2) invece viene caricato all'evento Click sulla Combo e viene riempito con i soli dati della tabella persone relativi alla città scelta nella Combo stessa. Al secondo DataControl è agganciata una DBGrid (con la proprietà DataSource) che serve solo a visualizzare il RecordSet associato al DataControl.

Si tratta di un automatismo semplice da capire e da realizzare.

In pratica i due DataControl fanno da intermediari tra la nostra applicazione ed i dati, mentre Combo e griglia servono solo a vedere i dati agganciati.

Il secondo esercizio (figure 3 e 4) è simile al primo, con due differenze. La prima è che usiamo oggetti DBCombo e la seconda e che ne usiamo due, per selezionare i nostri dati per città (la selezioniamo dalla prima DBCombo, che si chiama C1) e per qualifica (dalla C2).

Le due DBCombo prendono dati direttamente da due DataControl. Al primo abbiamo assegnato, come RecordSource, direttamente un'espressione SQL (Select Distinct Citta from persone) per avere un elenco di città senza ripetizione, alla seconda la stessa cosa per avere l'elenco delle qualifiche.

Figura 4 - MS VB5 - Due DBCombo e una DBGrid – Output.
Piccola variazione sul tema. Questa volta selezioniamo la città dall'elenco delle città e la qualifica da quello delle qualifiche. I due elenchi vengono mostrati in due DBCombo, che sono direttamente collegabili ai dati tramite un DataControl (mentre se si usano le Combo normali occorre caricarle da programma). Scegliendo una coppia di valori nelle due DBCombo, o anche uno solo dei due, la DBGrid viene immediatamente aggiornata.

Scelta un'accoppiata tra città e qualifica costruiamo (gli eventi interessati sono il Click su C1 e il Click su C2) una nuova istruzione SQL per selezionare i dati per quella città e quella qualifica.

La nuova SQL viene usata come RecordSource del terzo DataControl (D3) e che poi viene aggiornato con il metodo Refresh. D3 è il DataSource della DBGrid,che visualizza in questa maniera i dati di quella città e qualifica al cambiare di una delle due nelle due DBCombo.

Uso brillante dalla TabStrip

Il terzo esercizio si commenta da sé. Basta guardare la figura 6 (il codice re-

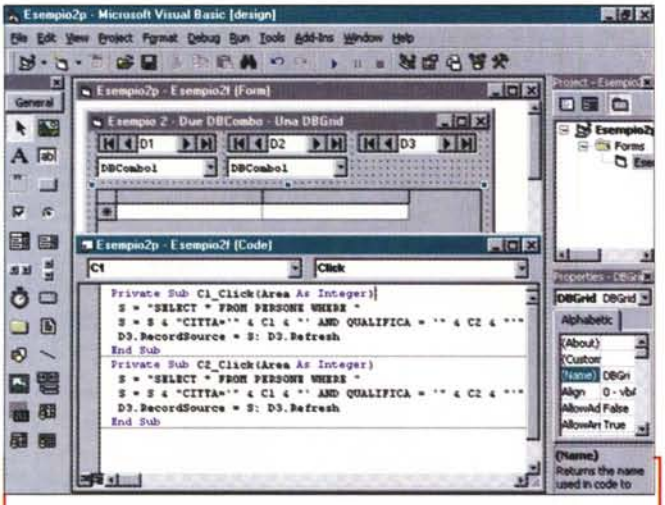

Figura 5 - MS VB5 - Due DBCombo e una DBGrid – Listato.
La scelta tra DBCombo e Combo, in situazioni di questo genere, ossia quando occorra solo selezionare dei dati, dipende dalla situazione. La DBCombo è più facile da alimentare ma, essendo un controllo aggiuntivo, impegna maggiormente la macchina. La Combo è un controllo fondamentale ma va caricato a mano... pardon da programma, scorrendo il RecordSet e trasferendo il campo desiderato del RecordSet sulla Combo stessa, sfruttando il suo metodo AddItem.

Figura 6 - MS VB5 - Uso della TabStrip – Output.
Se le selezioni possibili sono poche e sono sempre le stesse, come nel caso delle nostre otto città, si possono prevedere soluzioni più spettacolari, che sfruttino uno dei numerosi controlli a disposizione in Visual Basic 5 e in Windows 95. Qui usiamo il controllo TabStrip. Lo chiamiamo TB e lo carichiamo da programma: otto linguette per le otto città, più una linguetta usata per poter selezionare tutte le città. Facendo click su una delle linguette viene alimentata la ListBox L1 che mostra quattro campi della tabella persone, riferiti ovviamente alle persone di quella data città.

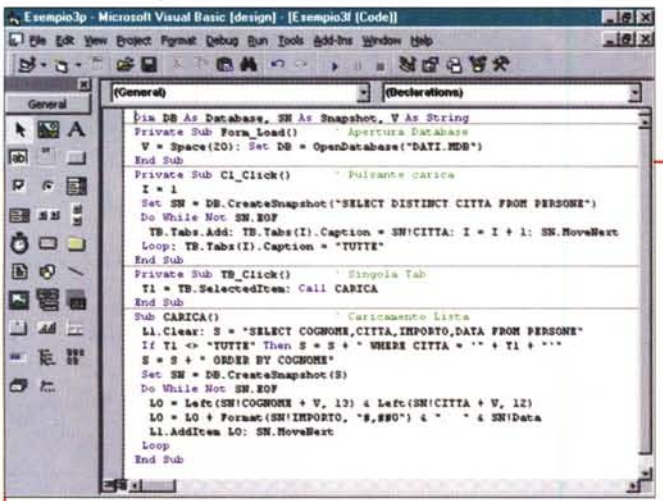

Figura 7 - MS VB5 - Uso della TabStrip – Listato.
Il controllo TabStrip contiene la Collection Tab, che può essere alimentata con il metodo Add. Se ne può definire la proprietà Caption, che corrisponde alla scritta mostrata sulla linguetta. Le linguette possono essere scorse sfruttando le proprietà Item e Count. Per individuare quale sia la linguetta cliccata si può usare la proprietà SelectedItem. E' chiaro che questa soluzione, un po' più complessa delle precedenti, è praticabile solo quando le linguette siano poche. Ricordiamo che questi controlli aggiuntivi dispongono sempre di una finestra delle proprietà custom. Nel nostro caso l'abbiamo usata per impostare "a mano" al valore True la proprietà MultiRow.

trostante è in figura 7). Il problema è sempre lo stesso, selezionare i dati per città, solo che in questo caso, essendo poche città, la selezione la deleghiamo ad otto linguette di una TabStrip.

Vogliamo caricare automaticamente, con la solita Select Distinct, ed al verificarsi dell'evento Form Load, le linguette (il nome della TabStrip è TB) con i nomi delle otto città, mentre, a mano, impostiamo il nome dell'ottava linguetta con la parola TUTTE. Il metodo da usare è:

TB.Tabs.Add

mentre la scritta sulla I-sima linguetta si appone con l'istruzione

TB.Tabs(I).Caption = "TITOLO"

Ora il problema è quello di individuare quale linguetta sia stata cliccata, per poter creare la nuova istruzione SQL con la quale caricare, con il metodo DAO, la nostra lista (il cui nome è L1).

TB.SelectedItem

è la proprietà della TabStrip che ci riporta la Caption della liguetta su cui è stato fatto click.

Il resto del programma è del tutto analogo a quello degli esercizi precedenti in cui la città veniva scelta da una Combo o da una DBCombo.

Da notare la routinetta che serve per caricare, simulando un incolonnamento, la lista L1, che sarebbe, per conto suo, monocolonna. Si usa un carattere non proporzionale (il migliore è il CourierNew) e si inseriscono dei blank come separatori tra i vari campi da incolonnare.

Ulteriore variante, finestre anziché pagine

Una finestra, in un'applicazione VB, è un oggetto e come tale può essere duplicabile a volontà, partendo da una finestra modello. Le finestre possono essere del tipo MDI Child, contenute quindi da una finestra MDI Form che fa da cornice.

Nel nostro prossimo esercizio (figure dalla 8 alla 10) partiamo da una finestra MDI, nella quale abbiamo creato un menu di una sola voce, e da una finestra Child, contenente un DataControl e una DBGrid.

All'evento Form Load dell'applicazione viene generato un RecordSet con il solito elenco delle città. Poi, per ogni città, viene creata una nuova finestra, specifica della città, che contiene nella sua griglia i dati selezionati per quella città.

Il menu della finestra MDI (la madre di tutte le finestre) serve per creare le classiche quattro viste:

finestre sovrapposte
finestre affiancate orizzontalmente
finestre affiancate verticalmente
icone allineate (questo nel caso che qualche finestra sia iconizzata e che le icone siano sparse).

La figura 8 mostra il risultato con un po' di finestre aperte e ben disposte. Quella successiva, la 9, mostra le due finestre di partenza, in alto la MDI ed in basso la MDI Child. Notiamo anche (in

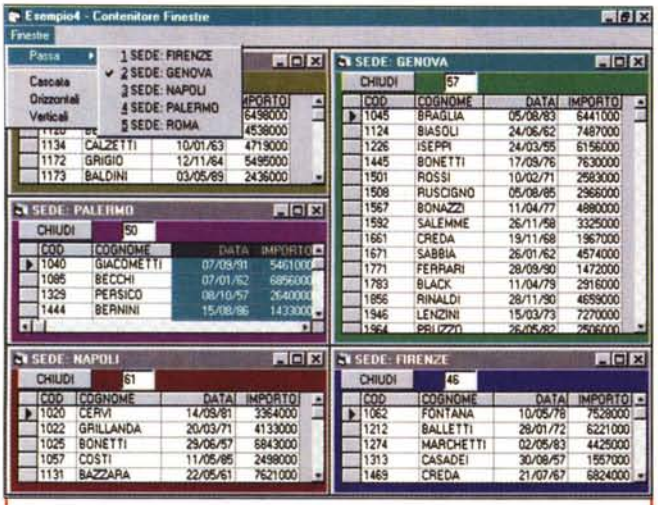

Figura 8 - MS VB5 - Cascata di finestre – Output.
Nell'esercizio precedente abbiamo usato il controllo TabStrip ed abbiamo alimentato... linguette. Ora alimentiamo una serie di finestre di tipo MDI Child, figlie di una finestra MDI, che fa da cornice. Ciascuna delle finestre Child contiene una griglia che viene alimentata con i dati di una singola città. L'aspetto interessante dell'esercizio è la creazione dinamica delle varie finestre partendo da una finestra iniziale che fa da prototipo.

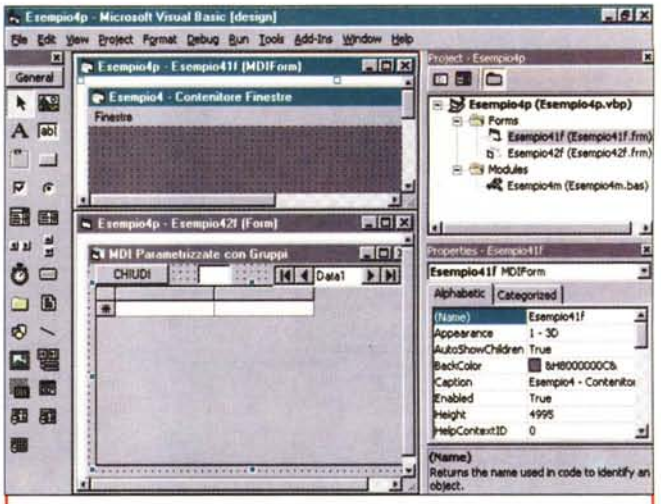

Figura 9 - MS VB5 - Cascata di finestre - Organizzazione del progetto.
Qui vediamo le due finestre di partenza. Quella MDI, che conterrà le varie finestre figlie, e quella, di tipo MDI Child, con la griglia per i dati, che fa da prototipo per le successive copie. In alto a destra vediamo la finestra Project, che contiene anche un modulo che serve a contenere le variabili necessarie per passare i dati tra le varie finestre. Abbiamo anche creato, nella finestra MDI, il menu necessario per gestire la visualizzazione delle otto finestre. I comandi sono quelli classici del menu finestre e quindi sovrapposte, finestre orizzontali e verticali, allinea icone.

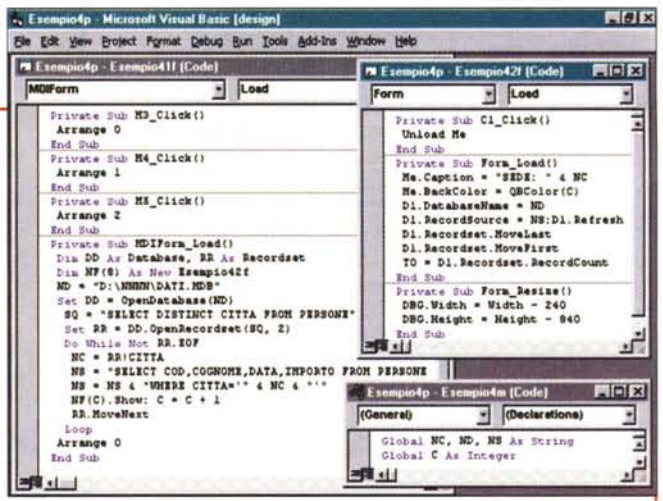

Figura 10 - MS VB5 - Cascata di finestre – Listati.
Vediamo i vari listati necessari al nostro programma suddivisi in tre finestre. In basso a destra le dichiarazioni inserite nel modulo. In alto a destra il codice relativo alla singola form MDI Child nella quale, all'evento Load, viene caricata di dati la griglia. In alto a sinistra vediamo il programma che prepara le variabili e crea le otto finestre. Per creare una nuova finestra in pratica basta dichiararla con l'istruzione Dim .. As New .. dopodiché la nuova form è già utilizzabile, pronta a ricevere i dati.

alto a destra nella figura) la presenza del modulo BAS, che serve per la dichiarazione delle variabili che vengono passate dalla finestra madre alle varie figlie, via via che vengono create.

La figura 10 mostra i listati dell'applicazione. Ce ne sono tre gruppi, quello nel modulo BAS, di cui abbiamo appena parlato, quello della finestra MDI che crea le otto finestre, e quello della singola MDI Child che serve per caricare il DataControl (D1) e quindi la DBGrid di ciascuna finestra.

Dal listato e dalle figure dovreste essere in grado di ricostruire gli esercizi.

Intervallo: uso in VB di una query di Access

Si tratta di un esercizio fuori tema, lo facciamo per distrarci un po'...

Stiamo usando un database in formato MDB e quindi in formato MS Access. Possiamo usare il prodotto Access come strumento di supporto per la nostra applicazione VB.

Ad esempio possiamo creare una query con Access, sfruttando il formidabile aiuto prestato dal suo generatore di query.

La query che desideriamo deve eseguire un raggruppamento per città ed un conteggio per gruppo. Inoltre (lo vediamo bene nella figura 11) inseriamo un criterio di selezione sul campo qualifica.

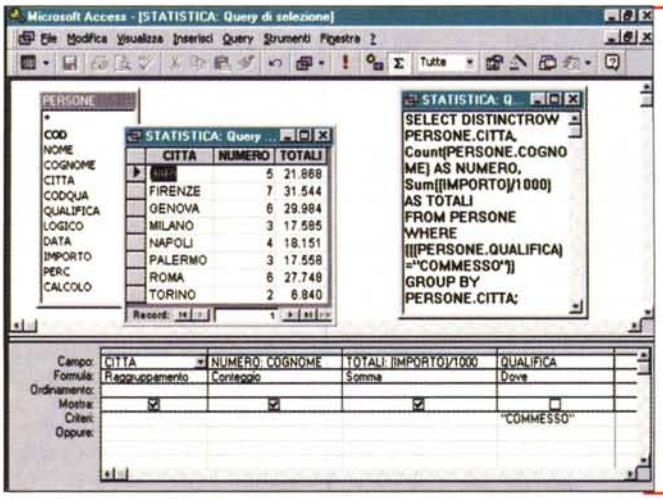

Figura 11 - MS Access 97 - Creazione di una query campi incrociati.
Se si vogliono maneggiare al meglio i dati messi a disposizione dal database occorre conoscere bene il linguaggio SQL con il quale si costruiscono i RecordSet. Se non si conosce bene SQL e si ha a disposizione MS Access è possibile creare l'SQL sfruttando il suo ambiente interattivo facilitato QbE. Poi basta copiare il comando SQL corrispondente ed incollarlo (ed adattarlo) come codice SQL dell'applicazione VB che si sta creando

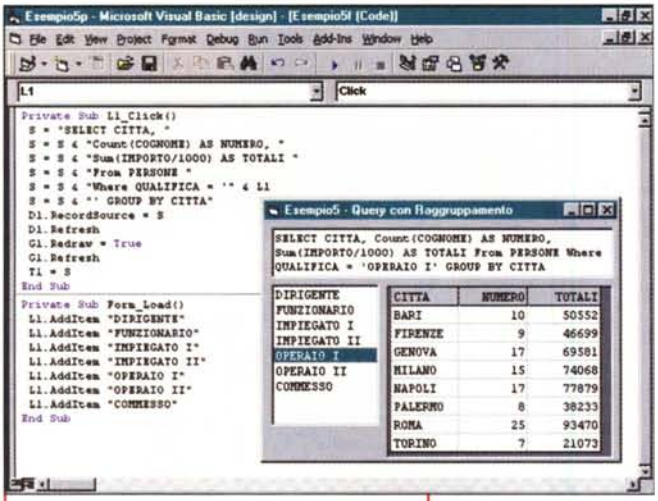

Figura 12 - MS VB5 - Uso della query campi incrociati - Output e listato.
Con MS Access (lo abbiamo visto nella figura precedente) abbiamo creato una query contenente un criterio di selezione fisso (qualifica='COMMESSO'). Una volta incollata l'istruzione SQL, generata automaticamente, nel nostro listato VB, la modifichiamo per renderla variabile in funzione della qualifica scelta nella ListBox L1, che abbiamo caricato a mano con l'elenco di tutte le qualifiche.

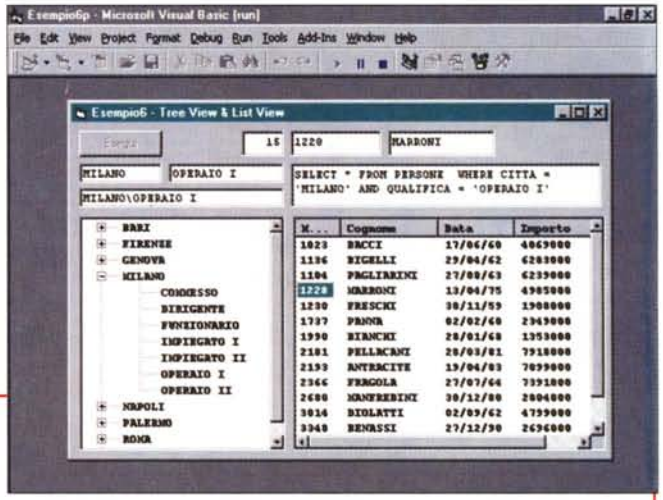

Figura 13 - MS VB5 - TreeView & ListView – Output.
Questo è un esercizio un po' più complesso, soprattutto per il fatto che usa due controlli che vanno sincronizzati tra di loro. A sinistra una TreeView che mostra città e qualifiche. Scelta una qualifica di una città, oppure semplicemente una città, viene alimentata la ListView sulla destra con l'elenco delle persone di quella città e qualifica. Le istruzioni con le quali caricare TreeView e ListView non sono molto intuitive per cui vanno imparate a memoria. Le intestazioni delle colonne della ListView le abbiamo inserite "a mano" nel foglio delle proprietà della ListView stessa.

Copiamo il codice SQL creato in Access e lo incolliamo nella nostra applicazione VB (lo vediamo in figura 12, figura che mostra anche il listato). Nella riga SQL sostituiamo il criterio secco, creato in Access, con il valore della qualifica preso da una lista.

Ci sono poi un DataControl, D1, che non è necessario che sia visibile, ed una DBGrid che ospita i dati.

Il risultato è interattivo, si sceglie la qualifica e la tabella risultato viene ricalcolata.

Creiamo un'applicazione Explorer like

L'Explorer di Windows 95, in italiano si chiama gestore delle risorse, costituisce una modalità di visualizzazione di dati che si presta a molti utilizzi, oltre a quelli istituzionali e che mostrano cartelle, sottocartelle e file.

Una vista di tipo Explorer consta di due oggetti sincronizzati tra di loro: a sinistra una TreeView ed a destra una ListView. Nel nostro caso sulla sinistra vediamo città e qualifiche e poi, scelta

Figura 14 - MS VB5 - TreeView & ListView - Listato stampato e commentato.
Questo è il listato del nostro penultimo esercizio. La TreeView l'abbiamo chiamata TV, colleziona Node e supporta gli eventi NodeClick e Click. Il nome della ListView è LV e colleziona Items e SubItems. I suoi eventi sono il click sulla colonna e quello sulla riga.

```
Dim DD As Database
Dim RR As Recordset

Private Sub C1_Click()
Set DD = OpenDatabase("DATI.MDB")
S = "SELECT DISTINCT CITTA,QUALIFICA FROM PERSONE"
Set RR = DD.OpenRecordset(S, 4)
Do While Not RR.EOF
  K1 = RR.CITTA
  Set Node = TV.Nodes.Add(, , K1, K1)
  Do While K1 = RR.CITTA
    K2 = RR.QUALIFICA
    Set Node = TV.Nodes.Add(K1, tvwChild, , K2)
    RR.MoveNext: If RR.EOF Then Exit Do
  Loop
Loop
C1.Enabled = False
End Sub

Private Sub TV_NodeClick(ByVal ND As Node)
  T3 = ND.FullPath
  T2 = TV.SelectedItem: T1 = TV.SelectedItem
  If InStr(T3, "\") > 0 Then
    T1 = Left(ND.FullPath, InStr(ND.FullPath, "\") - 1)
  End If
End Sub

Private Sub TV_Click()
  S = "SELECT * FROM PERSONE "
  S = S + " WHERE CITTA = '" & T1 & "'"
  If T1 <> T2 Then S = S + " AND QUALIFICA = '" & T2 & "'"
  T4 = S: Set RR = DD.OpenRecordset(S, 4)
  LL.ListItems.Clear
  Do While Not RR.EOF
    Set IT = LL.ListItems.Add()
    IT.Text = RR!cod
    IT.SubItems(1) = RR!cognome
      IT.SubItems(2) = RR!Data
      IT.SubItems(3) = RR!IMPORTO
      RR.MoveNext
    Loop
    T5 = RR.RecordCount
End Sub

Private Sub LL_ColumnClick(ByVal CH As ComctlLib.ColumnHeader)
  LL.SortKey = CH.Index - 1
  LL.Sorted = True
End Sub

Private Sub LL_ItemClick(ByVal IT As ComctlLib.ListItem)
  T6 = LL.SelectedItem.Text
  T7 = LL.SelectedItem.SubItems(1)
End Sub
```

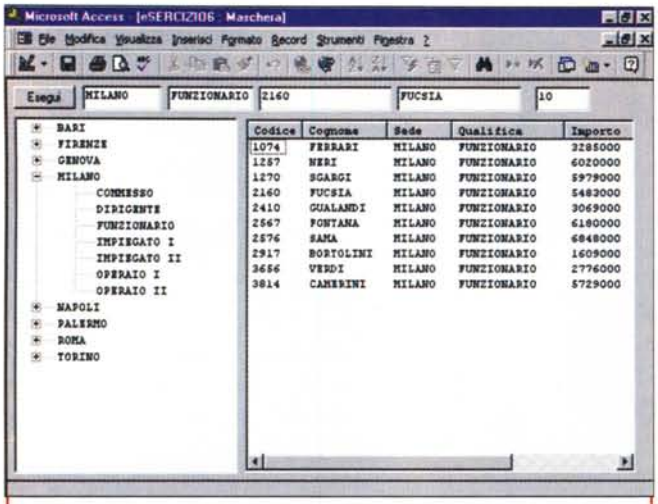

Figura 15 - MS Access 97 - TreeView & ListView - Output con Access 97. Abbiamo realizzato lo stesso esercizio di prima con MS Access 97, che, essendo programmabile in Visual Basic for Application, permette di caricare controlli nelle form e di utilizzare la programmazione DAO. Sia gli OCX, ovvero i controlli aggiuntivi, che le librerie DLL, sono richiamabili facilmente da tutto l'ambiente VBA.

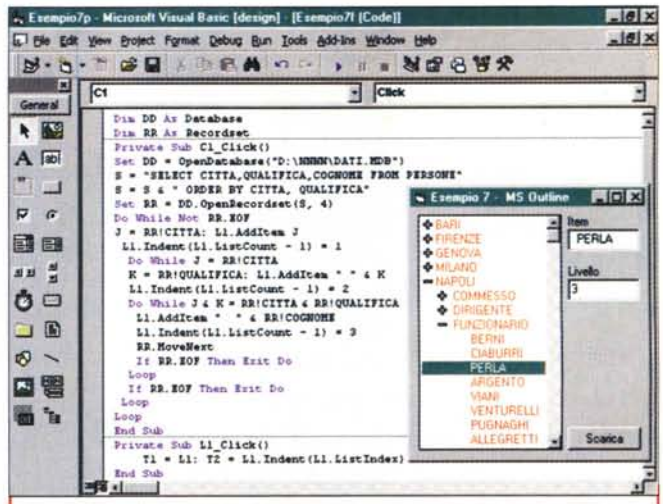

Figura 16 - MS Access 97 - Attenzione c'è anche il controllo MS Outline. Nelle versioni 3.0 e 4.0 di Visual Basic c'era un vecchio controllo, che si chiamava MS Outline, a cui ero molto affezionato. Nella versione 5.0 del VB è sparito, penso per il fatto che viene ben sostituito dal più moderno TreeView. Fortunatamente, soprattutto per garantire la compatibilità con i programmi scritti con le versioni precedenti di VB, ne viene fornita (è un po' nascosta nel CD del VB5) una versione compatibile con VB5. L'esercizio, che vediamo in un'unica figura, ne mostra l'efficacia in termini di visualizzazione dei dati.

un'accoppiata città + qualifica, riempiamo la lista sulla destra con l'elenco delle persone di quella città e quella qualifica. Se si seleziona solo la città devono apparire tutte le persone della città indipendentemente dalla qualifica.

Un'applicazione di tipo Explorer viene prodotta automaticamente anche dal Wizard di VB, solo che i due oggetti non vengono popolati.

Nel nostro esercizio, lo vediamo in figura 13, creiamo una Form con una TreeView (nome TV) ed una ListView (LL), poi alcune TextBox (da T1 a T7) che servono a mostrare alcuni valori.

Per caricare i due oggetti occorre scrivere del codice, un po' lunghetto per la verità, che vi mostriamo integralmente in figura 14, e che non commentiamo.

Nella figura 15 vediamo la stessa identica applicazione realizzata con Access, a dimostrazione del fatto che la tecnologia Component, e quindi i vari componenti, possono essere usati anche da altre applicazioni. Il codice, che non vi mostriamo, è assolutamente lo stesso.

Per chiudere uno sguardo al passato ed uno al futuro

In figura 16 vediamo un'applicazione che fa uso del vecchio Outline Control, che è stato lasciato nel VB5 solo per garantirne la compatibilità delle applicazioni sviluppate con le versioni precedenti. E' ovvio che la vista Outline viene sostituita dalla TreeView, che è più versatile, soprattutto in fase di caricamento, che si basa sulla codifica degli Item.

I vari componenti... abbandonati si trovano comunque nel CD del Visual Basic 5.0, nella cartella Tools Controls.

Per concludere vorrei semplicemente far notare come i vari esercizi proposti, anche se esteriormente producono effetti molto differenti tra di loro, sono invece molto simili in termine di codice. Anche gli oggetti usati sono molto simili tra di loro, si tratta, è la mia opinione personale, di varianti del concetto generico di lista, che è l'oggetto che ha il compito di mostrare insiemi di dati.

In questo articolo abbiamo visto accessi diretti, tramite DataControl, oppure accessi tramite codice DAO. Nel prossimo articolo parleremo più specificamente di modalità di accesso ai dati, ancora DAO, ma anche RDO, ADO, UDA con tutte le varianti. Voi intanto cominciate a scaricare da Internet (www.microsoft.com/data) i nuovi componenti ed a sperimentarli.

MC

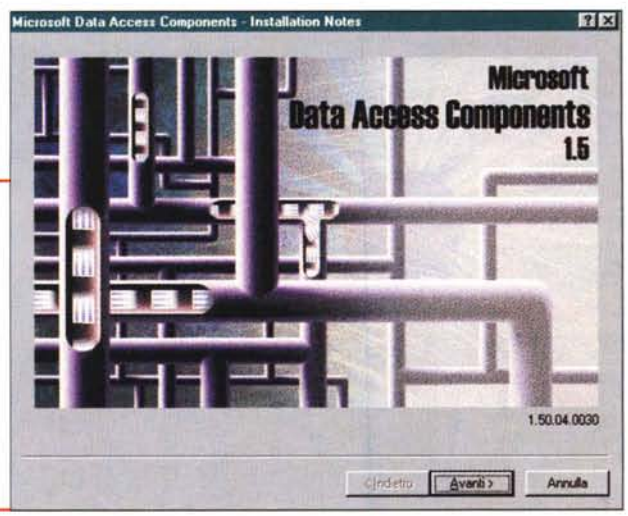

Figura 17 - Microsoft Data Access Components.
Molte tecnologie bollono nelle pentole della Microsoft. Dopo DAO, RDO, ADO si profila all'orizzonte UDA, Universal Data Access, nuova strategia, che racchiude vecchi e nuovi componenti (come questo in figura), che ha come ambizioso obiettivo quello di unificare le modalità di accesso ai dati, rendendole indipendenti sia dall'applicativo Client che dal tipo di fonte dei dati. Altro obiettivo è l'allargamento delle tipologie di dati gestibili, non solo dati di tipo strutturato, ma anche dati di tutti gli altri tipi, testuali non strutturati, documenti di vario genere, materiale multimediale, ecc.

Per un '98 multimediale, una tecnologia

eccezionale!

HIGHSCREEN XA

SkyMIDI 97-line

- **Processore:** **Intel Pentium® II Processor 233 MHz**
- **Mainboard:** Intel ATX
- **Cache:** 512 K
- **RAM: 32 MB EDO**
- **Hdd:** HDD da 2 GB Fast/Ultra DMA & 3,5" FDD
- **Scheda grafica:** ATI 3D Charger Rage II + DVD 4 MB EDO DRAM
- **Tastiera:** Indus-Keyboard Win'95
- **Lettore CD-ROM:** 24X
- **Scheda audio:** Highscreen Sound Boostar® 3D 16 PNP
- **Software:** 97-line
- Abbonamento incluso fino fino ad Aprile '98 a Italia OnLine: Internet + e-mail 24 ore al giorno!

Software

- *Windows 95* 97-line
- *MSInternet Explorer 4.0*
- *MSWord 97*
- *MSWorks 4.0*
- *Autoroute Express*
- *Corel Draw 6*

PRONTA CONSEGNA

COMPRESO INTERNET + E-MAIL

2.599.000

PREZZO SEMPRE IVA INCLUSA!

MMX intel inside pentium II

Vieni in uno degli oltre 200 punti vendita Vobis d'Italia. Gli indirizzi sono sulle Pagine Gialle e sulle Pagine Utili Mondadori alla voce "Personal Computer". Per saperne subito di più, chiama la Hot-Line Vobis: 02-6125898.

La prima catena europea dell'informatica

VOBIS MICROCOMPUTER