Il DBGrid in Visual Basic

Indice

Introduzione

	Esercizi
Capitolo 1. L'Oggetto DBGrid	
1.1 Apertura di un Progetto	
1.2 Apertura di un RecordSet tipo Table	Esercizio 01
1.3 Apertura di un RecordSet tipo Dynaset	Esercizio 02
1.4 Modifica degli ordinamenti	Esercizio 03
1.5 Controllo delle Colonne	Esercizio 04
1.6 Estrazione di Dati	Esercizio 05
1.7 Join di Tabelle	Esercizio 06
Capitolo2 . Lettura a tutto schermo	
2.1 Preparazione di una maschera a tutto schermo	Esercizio 07
2.2 La Subroutine DATTEXT e la formattazione Euro	id.
2.3 La Subroutine DATFRESCA	id.
2.4 Simulazione di Rete	id.
Capitolo 3. Aggiunta di un nuovo Record	
3.1 Il bottone CMDNEW	Esercizio 08
3.2 La Subroutine DATCHECK	id.
3.3 La Subroutine DATREPLA	id.
3.4 L'inserimento dati col tasto Return	id.
Capitolo 4. Modifica di un Record	
4.1 Il bottone CMDEDIT	Esercizio 09
4.2 Il bottone CMDCONFEDIT	id.
Capitolo 5. Cancellazione di un Record	
5.1 Il bottone CMDDELE	Esercizio 10
5.2 Il bottone CMDCONFDELE	id.
Capitolo 6. L'Help di dati	
6.1 Il concetto di Help di dati	
6.2 Il sistema AUT	Esercizio 11
6.3 Il sistema DAT	id.
6.4 Il Form AUTHLP1	id.
6.5 Inserimento fuori Help	id.
Capitolo 7. La selezione	
7.1 Il concetto di Selezione	Esercizio 12
7.2 Navigazione nella Selezione	id.
7.3 Estrazione di dati Selezionati	id.
7.4 Estrazione di dati non Selezionati	id.
Capitolo 8. Doppio DBGrid	
8.1 Metodo VscrollBar	Esercizio 13
8.2 Metodo Splits	Esercizio 14
Capitolo 9. Lavorare in Rete	
9.1 Struttura di Rete – App.Path	
9.2 EXE e Dati in Cartelle diverse	Esercizio 15

Commiato

Introduzione

Questo testo si rivolge a coloro che apprezzano il Visual Basic e si divertono a programmare con questo utile ed importante strumento che, ahimé , William Gates vuole mettere in soffitta con grande dolore di noi tutti.

L'argomento affrontato é l'uso del comando DBGrid, la cui potenza e flessibilità non possono essere sufficientemente sottolineate ma per cui esiste pochissima letteratura.

L'impostazione adottata è di tipo didattico ed i vari Esercizi proposti, a carattere esemplificativo, sono di complessità gradualmente crescente. Essi hanno per argomento principale il problema della gestione di dati. Tuttavia si da per scontato che il lettore possegga un minimo di conoscenza di basi di programmazione (cosa sia un Oggetto, una Proprietà, un Metodo, una Funzione, etc.), per apprendere le quali si rimanda ad opere più specifiche per chi inizia.

Viene inoltre affrontato il problema della creazione di programmi capaci di girare in un sistema di Rete dove i dati vengono contemporaneamente condivisi da più utenti.

In ogni momento il lettore può confrontare il suo lavoro con le soluzioni degli Esercizi, presentate nelle Cartelle delle Soluzioni.

In particolare si noti che queste Cartelle sono tutte SottoCartelle della Cartella principale a nome "C:\GRIGLIE" . Inoltre nella SottoCartella a nome "ARCHIVIODATI" sono presenti alcuni Database didattici.

Attenzione ! É necessario che le Cartelle SOLUZIONI e le Cartelle ESERCIZI che il lettore creerà di volta in volta, mantengano questa disposizione perché ad essa si fa continuo riferimento.

Per questo, se già non lo si é fatto, é necessario Unzippare il file GRIGLIE.ZIP in C:\.

Verrà automaticamente creata una Cartella C:\GRIGLIE che contiene una serie di SottoCartelle, quelle delle Soluzioni, nonché la SottoCartella dei Database.

In questa stessa cartella C:\GRIGLIE il lettore creerà le sue SottoCartelle di Esercizio.

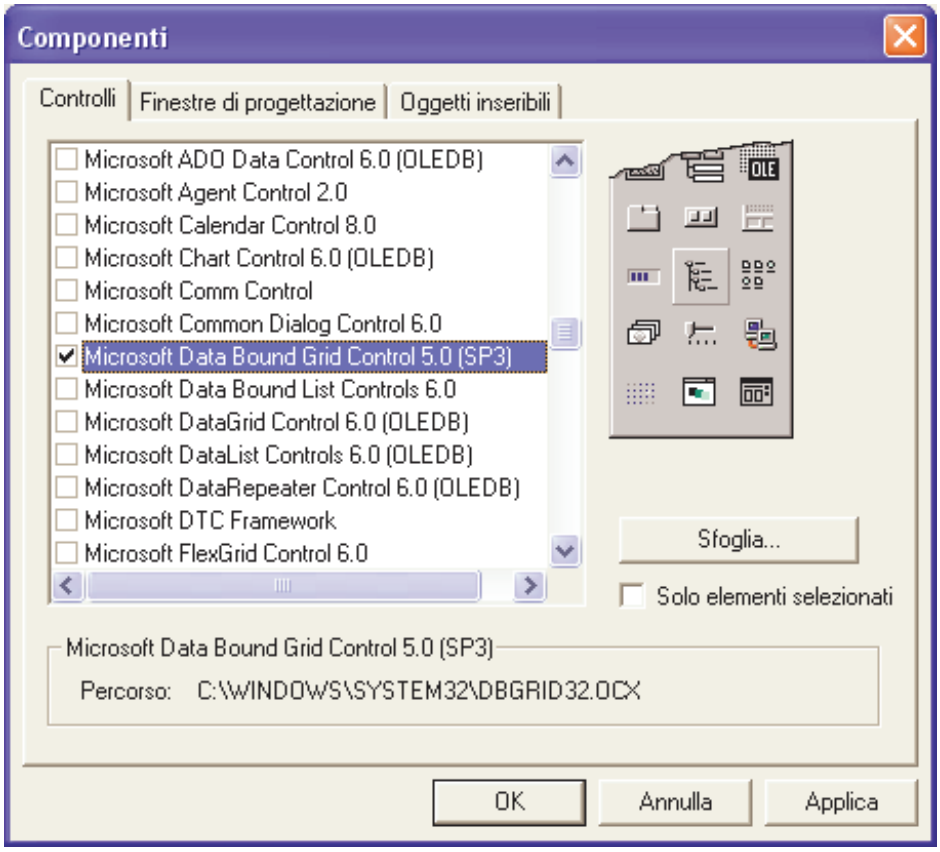
In vista dell'obiettivo di imparare a gestire i dati condivisi in Rete, é importante ricordare che la definizione del Path di indirizzamento é elemento fondamentale e caratteristico di questo modo di lavorare. Ne parleremo più diffusamente nell'ultimo paragrafo.

Capitolo 1. L'oggetto DBGrid

1. 1 Apertura di un Progetto

Si lanci il VB e si aggiunga alla lista dei comandi disponibili (OCX) un DBGrid.

Per fare questo, bisogna partire da menù "Progetto/Componenti" e selezionare dalla lista che appare il comando "Microsoft Data Bound Grid Control":
(C:\Windows\System32\DBGrid32.OCX)
e confermare.



Questo nuovo comando é ora disponibile nel Pannello degli Strumenti e può essere collocato sul Form in uno dei due soliti modi classici:con doppio click sinistro, oppure con click semplice e poi tracciamento sul Form. Si dispone ora di un DBGrid1.

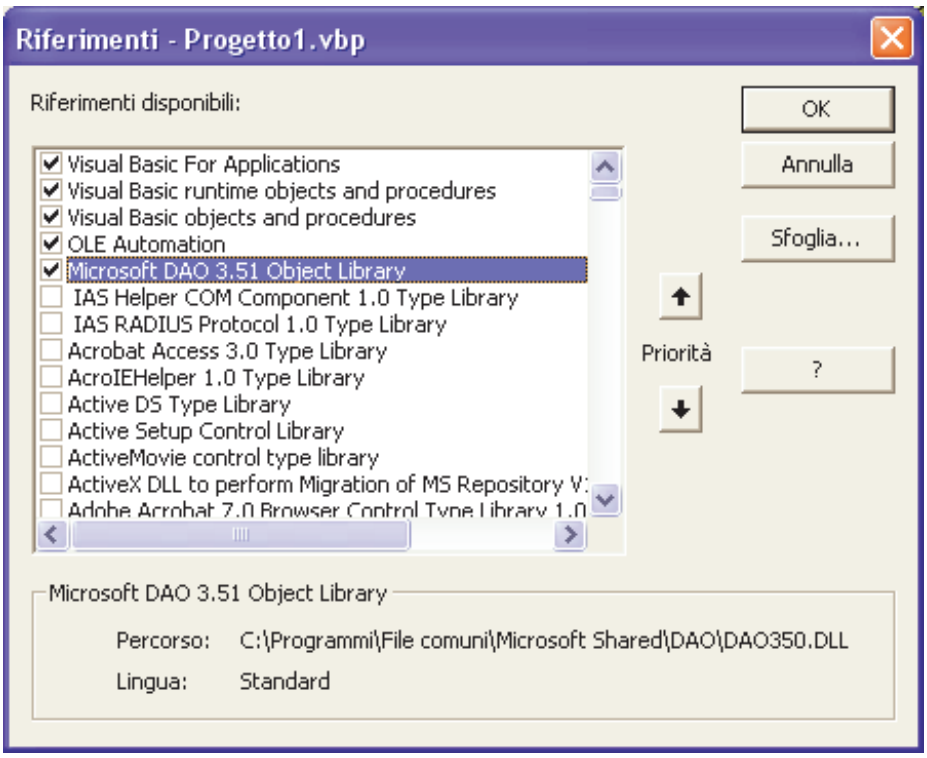
Aggiungiamo ora un oggetto Data, che prenderà il nome di Data1 ed impostiamone la proprietà Visibile a "False".

Apriamo ora il pannello delle proprietà del DBGrid1 e selezioniamo per la DataSource l'oggetto Data1.

Abbiamo così creato la coppia DBGrid/Data su cui vogliamo iniziare a lavorare tenendo presente che tale coppia é sempre inseparabile e che in essa l'oggetto Data agisce da (invisibile) sorgente dati individuando un RecordSet (vedi sotto) da visualizzare, mentre l'oggetto DBGrid agisce da finestra di questo RecordSet.

Prima di cominciare, però , é necessario effettuare un'altra operazione di personalizzazione del sistema: dato che andremo ad utilizzare oggetti Database, RecordSet, Tables, etc, occorre abilitare i Progetti ad operare con questi tipi di oggetti.

Per fare questo occorre aprire da Menù Progetto la Finestra di Dialogo Riferimenti ed aggiungere alle caselle già selezionate anche la Microsoft DAO 3.6 Object Library (DAO360.DLL)



É importante attivare questa casella perché altrimenti gli oggetti di accesso ai dati (DAO) e i tipi Database, RecordSet e TableDef non vengono riconosciuti per cui i relativi riferimenti nel codice generano un errore (Tipo definito dall'utente non definito).

1.2 Apertura di un RecordSet tipo Table

Un RecordSet é un oggetto che individua un gruppo di Records di una Tabella.

Alcuni RecordSet individuano la totalità dei Records, altri una parte secondo un criterio da noi stabilito.

Inoltre i RecordSet possono essere di due tipi diversi, a seconda del metodo usato per la loro creazione, tipo Table e tipo Dynaset. Ci occuperemo ora diffusamente di queste due tipologie sottolineandone le differenze e i rispettivi vantaggi e svantaggi . Note queste caratteristiche, sceglieremo un tipo piuttosto che l'altro a seconda del lavoro che dobbiamo svolgere.

Cominciamo da un primo tipo: il RecordSet Table.

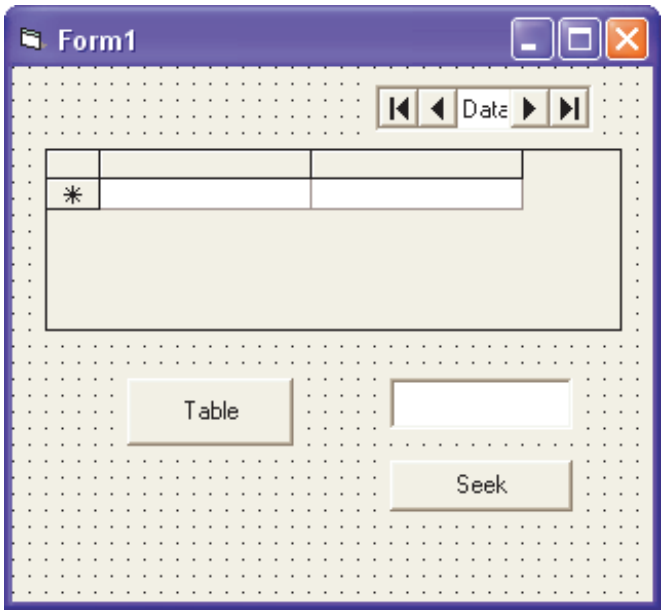
Per creare questo oggetto occorre disporre di un Database (MDB) contenente una o più Tabelle. Non é materia di questo trattato definire come si crea un oggetto Database (cosa che si può fare per esempio utilizzando l'utility Visual Data Manager) per cui supponiamo di possederne già uno, per esempio il BIBLIO.MDB messo a disposizione dalla Microsoft al momento dell'installazione del Visual Basic. Facciamo allora una copia di questo file e prepariamo una nuova Cartella/Directory dal disco C: assegnandole il nome "GRIGLIE". In essa apriamo un'altra SottoCartella e chiamiamola "ARCHIVIODATI". In questa SottoCartella collochiamo la copia del BIBLIO.MDB.

Il nostro BIBLIO di lavoro si troverà pertanto nella sottodirectory "C:\GRIGLIE\ARCHIVIODATI" creata all'uopo.

Il Database BIBLIO dispone di varie Tabelle/Tables e precisamente: Authors, Publishers,Title Author e Titles come ci si può facilmente sincerare partendo dal menu "Aggiunte/Visual Data Manager" e aprendo BIBLIO dalla (sotto)directory sopra indicata.

Poniamo la nostra attenzione sulla Tabella Authors che presenta una serie di Campi/Fields e precisamente AU_ID, Author, Year Born.

Proponiamoci allora di costruire un RecordSet, per ora di tipo Table, contenente tutti i Records di questa Tabella.
Scegliamo allora una collocazione per scrivere la nostre istruzioni VB, per esempio l'evento Click di un CommandButton che avremo collocato sul solito Form e a cui avremo dato nome "cmdRecordSet" (Proprietà Name) e Caption "Table".



Scriviamo allora le seguenti istruzioni per l'evento Click:

```
Private Sub cmdRecordSet_Click()

' path
'--------------------
pathmdb = "C:\GRIGLIE\ARCHIVIODATI\BIBLIO.MDB"

'definizione Data1
'---------------------
Data1.DatabaseName = pathmdb
Data1.Connect = ""
Data1.RecordSetType = 0          'Table

'riempimento Data1
'----------------------
Data1.RecordSource = "AUTHORS"
Data1.Refresh

'definizione del RecordSet
'--------------------
Set aut11 = Data1.RecordSet

End Sub
```

Se ora si lancia il Progetto si avrà la richiesta di salvare il Form1 e il Progetto1. Accettiamo, ma scegliamo come Cartella/Directory la collocazione C:\GRIGLIE\ESERCIZIO_01 che avremo previamente creato all'interno della Cartella GRIGLIE.

Facciamo ora alcuni commenti alle istruzioni proposte.

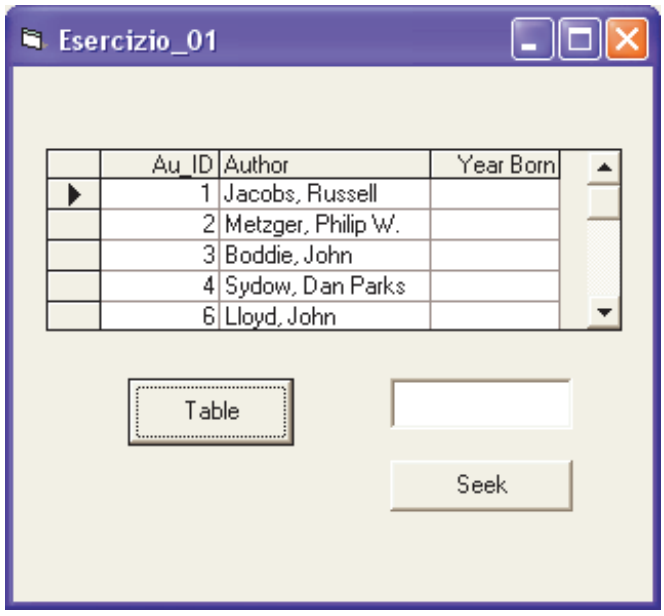
- La variabile a nome (arbitrario) PATHMDB serve per definire la collocazione del Database da cui leggere la Tabella da visualizzare.
- La proprietà DatabaseName dell'oggetto Data1 é impostata su PATHMDB, il che significa che l'oggetto apre il Database BIBLIO.
- La proprietà RecordSource dell'oggetto Data1 é impostata su "AUTHORS", cioè l'oggetto apre la Tabella AUTHORS.
- Il tipo di RecordSet é Table, come definito dal valore Zero assegnato alla proprietá RecordSetType.
- L'istruzione Data1.Refresh é quella che effettua il caricamento del RecordSet definito dai parametri precedenti.
- L'ultima istruzione assegna un nome (arbitrario) al RecordSet cosí creato. Il nome AUT11 é più semplice da scrivere del defatigante "Data1.RecordSet " e risulterà comodo da maneggiare per lavorare in seguito.

In particolare poi vogliamo che il RecordSet AUT11 sia visibile in ogni subroutine del Progetto.

Per ottenere questo é necessari aprire la finestra "Generale" del Form1 ed aggiungere l'istruzione:

Dim aut11 As RecordSet

Se ora si lancia il programma e si clikka sul Bottone "TABLE" si vede che il DBGrid mostra tutti i Records della Tabella AUTHORS del Database BIBLIO



Le caratteristiche di un RecordSet di tipo Table sono le seguenti:

- Interessano sempre la Tabella nella sua totalità: non é cioè possibile estrarre una parte dei Records.
- I Records possono essere ordinati solo se si associa il RecordSet con un indice presente nella Tabella.
- Una volta che i Records sono stati ordinati, si può effettuare una ricerca di uno specifico Record con estrema velocità, usando la potenza dell'indice.
- La ricerca si effettua con il metodo "Seek".

Vediamo in concreto una applicazione di questi concetti.

Torniamo al nostro Form1 e modifichiamo il testo dell'evento Click del bottone cmdRecordSet aggiungendo come ultima riga l'istruzione:

aut11.Index = "AUTHORSAU_ID"

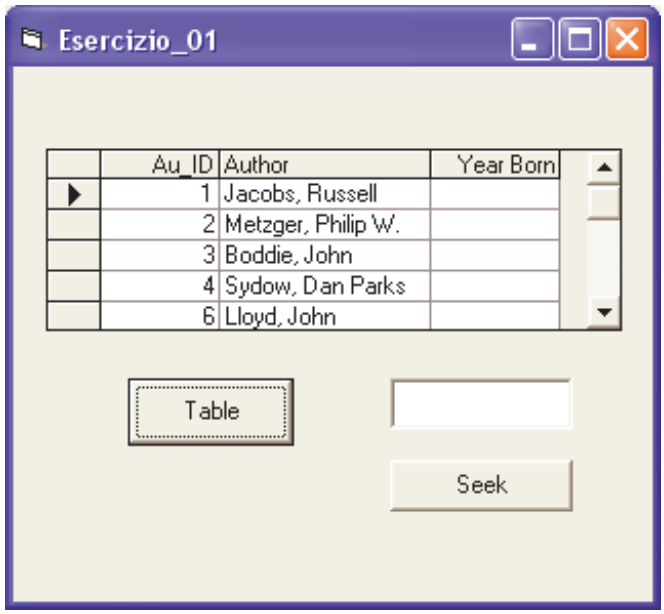
Questa istruzione associa al RecordSet AUT11 l'ordinamento definito dall'indice "AUTHORSAU_ID" ché é un oggetto della Tabella AUTHORS,

Gli indici delle Tabelle possono essere letti, aggiunti o modificati con lo strumento Visual Data Manager, già citato in precedenza.

In particolare, se si apre con il VDM il Database BIBLIO e si batte col mouse destro sulla Tabella Authors, si può scegliere l'opzione Progettazione ed avere una visione dei campi e, quello che qui ci interessa, degli indici associati alla Tabella.

Si potrà in particolare notare la presenza dell'indice AUTHORSAU_ID che definisce un ordinamento dei Records della Tabella secondo il campo +AU_ID, cioè il codice di identificazione di ciascun autore in ordine crescente.

Se ora lanciamo il programma vedremo che l'istruzione che abbiamo aggiunto a proposito di AUT11 ha avuto come risultato che il RecordSet (in questo caso tutti i Records della Tabella AUTHORS) viene visualizzato nel DBGrid1 in ordine di codice autore.



Torniamo ora al nostro Form di esempio ed aggiungiamo:

- Una TextBox a cui imporremo come proprietà Name "txtCerca" ed in cui metteremo un blank nella proprietà Text.
- Un CommandButton a cui imporremo come nome "cmdCerca" e come Caption "Seek".

Per l'evento Click del cmdCerca scriviamo:

```
Private Sub cmdCerca_Click()
zricerca = txtCerca.Text
'----------------------
aut11.Seek "=", zricerca
'----------------------
If aut11.NoMatch Then
MsgBox (zricerca & " non trovato")
aut11.MoveFirst
End If

End Sub
```

Il metodo Seek cercherà nel RecordSet AUT11 il primo Record il cui AU_ID soddisfa esattamente al contenuto della TextBox TXTCERCA.

Proviamo infatti a lanciare il programma e a cercare, ad esempio, il codice di autore 1000.
Il puntatore del DBGrid si posiziona immediatamente sul Record in questione.
La velocità é elevata, indipendentemente dal numero di Records tra cui si cerca.

Se invece il codice cercato non esiste (ad esempio il codice 7) il RecordSet entra in situazione di NoMatch ed il semplice MsgBox provvede a segnalare l'accaduto.

1.3 Apertura di un RecordSet tipo Dynaset

Vediamo ora un esercizio analogo al precedente utilizzando però un RecordSet di tipo Dynaset.

Le caratteristiche principali, di questi RecordSet, da confrontare con le caratteristiche dei RecordSet di tipo Table visti in precedenza, sono qui di seguito riassunte:

- Interessano la Tabella nella sua totalità oppure possono operare una selezione di Records: é cioè possibile estrarre una parte dei Records.

- Per la creazione del RecordSet é necessario usare una istruzione (Query) SQL. Ne vedremo alcuni esempi di seguito.
- I Records possono risultare ordinati se l'istruzione SQL di creazione lo prevede ma in nessun caso possono essere associati ad un indice presente nella Tabella.
- In compenso i Records possono essere ordinati per qualunque Field (o combinazione di Fields) senza che sia stato necessario creare un indice specifico per ciascun ordinamento desiderato.
- Se il RecordSet é ordinato, si può effettuare una ricerca di uno specifico Record in base all'ordinamento imposto. Tuttavia, dato che non si possono usare gli indici, la velocità di ricerca é minore che nel caso di RecordSet di tipo Table. Se il Database é di piccole dimensioni, la differenza é irrilevante mentre diventa importante se ci sono molti Records.
- La ricerca si effettua con il metodo "Find".

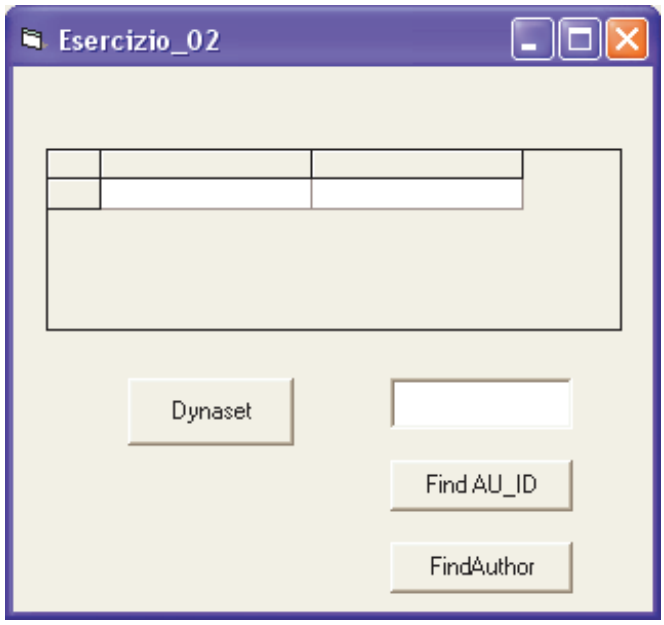
Per vedere una applicazione pratica di questi concetti prepariamo una nuova Cartella/Directory all'interno della Cartella GRIGLIE.

Per esempio possiamo fare un copy della Cartella ESERCIZIO_01 ed incollarla con il nome ESERCIZIO_02.

Carichiamo ora il Progetto1 dalla Cartella ESERCIZIO_02 ed operiamo qualche modifica:

- Modifichiamo la Caption del cmdRecordSet a "Dynaset" e quella del cmdCerca a "Find AU_ID".
- Modifichiamo le istruzioni dell'evento Click del cmdRecordSet come segue:

```
Private Sub cmdRecordSet_Click()

' path
'--------------------
pathmdb = "C:\GRIGLIE\ARCHIVIODATI\BIBLIO.MDB"


'definizione del RecordSet

'definizione Data1
'---------------------
Data1.DatabaseName = pathmdb
Data1.Connect = ""
Data1.RecordSetType = 1         'Dynaset

'riempimento Data1
'----------------------
autq1 = "select * from AUTHORS order by AU_ID"
Data1.RecordSource = autq1
Data1.Refresh
--------------------
Set aut21 = Data1.RecordSet

End Sub
```



Anche qui facciamo alcuni commenti alle istruzioni proposte.

- La variabile a nome (arbitrario) PATHMDB serve per definire la collocazione del database da cui leggere la Tabella da visualizzare.
- La proprietà DatabaseName dell'oggetto Data1 é impostata su PATHMDB, cioè l'oggetto apre il Database BIBLIO.
- La proprietà RecordSource dell'oggetto Data1 é impostata sulla Query AUTQ1
- Questa Query apre la Tabella AUTHORS e ne imposta l'ordine secondo il Field AU_ID.
- In alternativa avremmo potuto omettere del tutto la parte "ORDER BY" dalla Query oppure ancora avremmo potuto scrivere ORDER BY AUTHOR ed in tal caso la Tabella avrebbe preso l'ordinamento crescente del Field AUTHOR.
- Il tipo di RecordSet é Dynaset, come definito dal valore 1 (uno) assegnato alla proprietá RecordSetType.
- L'istruzione Data1.Refresh é quella che effettua il caricamento del RecordSet definito dai parametri precedenti.
- L'ultima istruzione assegna un nome (arbitrario) al RecordSet cosí creato. Il nome AUT21 é semplice e risulterà comodo da maneggiare per lavorare in seguito con tale RecordSet.

In modo analogo a quanto fatto prima, aggiungiamo anche qui l'istruzione:

```
Dim aut21 As RecordSet
```

nella finestra Generale del Form1, per rendere il RecordSet AUT21 visibile in tutte le subroutines del Progetto.

Se ora si lancia il programma, si vede che il DBGrid mostra tutti i Records della Tabella AUTHORS del Database BIBLIO, ordinati per AU_ID, in modo molto simile a quanto visto in precedenza per il RecordSet AUT11 di tipo Table.

La somiglianza é però solo apparente e ce ne accorgeremo sviluppando gli esercizi seguenti.
Vediamo ora come sia possibile operare una ricerca di un Record. Per esempio come si fa a trovare il Record in cui il Field AU_ID sia 1000 ?

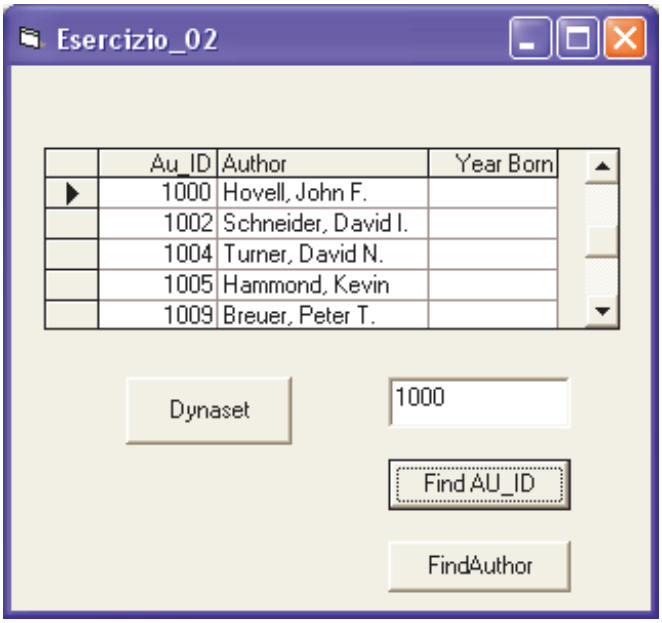
Modifichiamo allora le istruzioni dell'evento Click del cmdCerca (Find AU_ID) come segue:

```
Private Sub cmdCerca_Click()

zricerca = txtCerca.Text
'----------------------
kriterio = "AU_ID = " & zricerca & " "
aut21.FindFirst kriterio
'----------------------
If aut21.NoMatch Then
MsgBox (zricerca & " non trovato")
Aut21.MoveFirst
End If

End Sub
```

Il metodo FindFirst cercherà nel RecordSet AUT21 il primo Record il cui AU_ID soddisfa al contenuto della TextBox TXTCERCA.

Si noti la definizione della variabile a nome (arbitrario) KRITERIO:
La notazione " & iniziale e & " finale indica che la variabile KRITERIO deve essere definita dal "contenuto numerico" della variabile ZRICERCA.
Vedremo più oltre come la definizione di KRITERIO debba essere modificata per indicare "contenuto alfanumerico"

Proviamo allora a lanciare il programma e a cercare, ad esempio, il codice di autore 1000.
Il puntatore del DBGrid si posiziona subito sul Record in questione.



La velocità é ottima perché il Database é relativamente piccolo. In caso di Database grande, cioè con moltissimi Records, la velocità sarebbe stata minore.

Anche qui se il codice cercato non esiste (ad esempio il codice 7) il RecordSet entra in situazione di NoMatch ed il semplice MsgBox provvede a segnalare l'accaduto.

Risolviamo ora un altro problema: come trovare il primo Record in cui il Field AUTHOR cioè il nome dell'Autore cominci per "C" ?

Qui vediamo un'altra differenza tra i due tipi di RecordSet: mentre per i Table la ricerca (Seek) é veloce ma é possibile solo per il campo dell'indice che si é creato in fase di Progettazione

della Tablella (e lo si é aperto), per i Dynaset la ricerca é meno veloce ma può essere effettuata su qualunque Field.

Comunque va chiarito che anche nel caso di Dynaset la ricerca col metodi Find funziona solo se il RecordSet é ordinato secondo il Field su cui si effettua la ricerca. Solamente, non serve indice.

Per verificare quanto sopra, occorre aggiungere un nuovo CommandButton a cui imporremo il Name "cmdCerca2" e la Caption "Find Author".

Nell'evento Click di questo CommandButton provvederemo a:
- Cambiare l'ordinamento del RecordSet da AU_ID ad AUTHOR
- Impostare il nuovo criterio di ricerca con Find.

In pratica la routine é la seguente:

```
Private Sub cmdCerca2_Click()

autq1 = "select * from AUTHORS order by AUTHOR"
Data1.RecordSource = autq1
Data1.Refresh
Set aut21 = Data1.RecordSet

zricerca = txtCerca.Text
'----------------------
kriterio = "AUTHOR > '" & zricerca & "' "
aut21.FindFirst kriterio
'----------------------
If aut21.NoMatch Then
MsgBox (zricerca & " non trovato")
aut21.MoveFirst
End If

cmdCerca.Enabled = False
End Sub
```

La prima parte della Subroutine provvede ad una nuova definizione della Query AUTQ1 che ricostruisce il RecordSet AUT21, questa volta ordinato secondo il Field AUTHOR.
La nuova definizione della variabile KRITERIO tiene conto che la grandezza ricercata (AUTHOR) é ora alfanumerica. Occorre quindi utilizzare la notazione ` " & prima di ZRICERCA e la notazione & " ` dopo di essa.

Inoltre va notato che per l'ordinamento AU_ID la definizione di KRITERIO era finalizzata al rintraccio del Record che avesse esattamente l'AU_ID immesso nella TextBox, da qui il segno uguale all'interno di essa.

Nella situazione attuale, invece, si vuole cercare il primo Record che "cominci" con la lettera immessa, per esempio "C" e non l'autore con nome secco "C". Da qui il segno ">" contenuto nella definizione di KRITERIO ad indicare che si accetta il primo Record successivo al criterio di ricerca: nell'esempio, il metodo FindFirst cerca "C" secco: se c'e,' bene, se no cerca il primo Record dopo la posizione in cui avrebbe dovuto esserci il "C" secco, per esempio un autore che comincia per "CA" eccetera.

Il rischio di finire in Nomatch é allora piccolo ma non nullo. Per convincersene si provi a cercare "ZZ".

L'ultima istruzione della Subroutine, la disabilitazione di cmdCERCA é un artificio didattico per evitare che si facciano ricerche su base AU_ID ora che il RecordSet é ordinato per AUTHOR. In

un programma vero si farebbe dell'altro, per esempio quello che vedremo nell'Esercizio seguente.

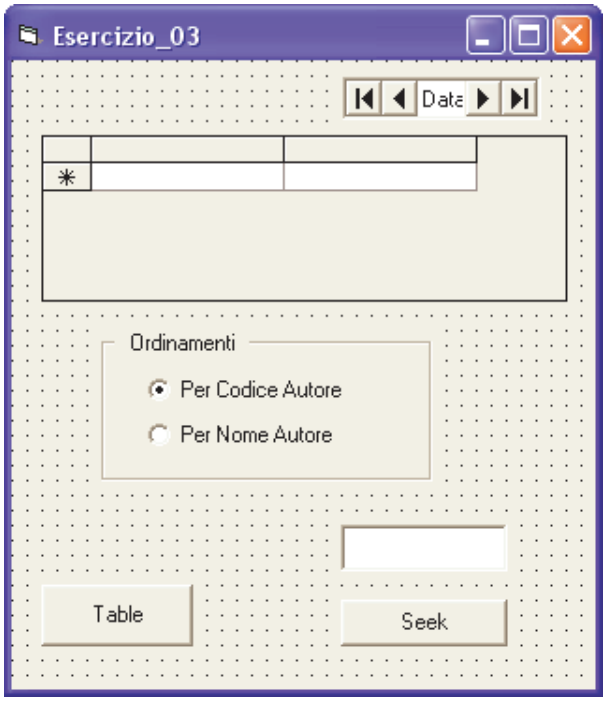
1.4 Modifica degli ordinamenti

Vogliamo ora fare un altro piccolo passo avanti e sviluppare il nostro Esercizio in modo che si possa agevolmente passare da un ordinamento ad un altro per fare ricerche mirate.

Cominciamo allora dai RecordSet tipo Table, riprendiamo il nostro Esercizio 01 e con il solito metodo del copia e incolla creiamo una nuova Cartella ESERCIZIO_03.

Aggiungiamo al Form i seguenti comandi:

- Un Frame con Name "FraOrdinamenti" e con Caption " Ordinamenti " (meglio mettere uno spazio prima e dopo quando si definisce la Caption). Impostare la proprietà Visible di questo Frame a "False".
- Un OptionButton a Name "optCodice" e Caption "Per Codice Autore" collocato sopra il Frame. Impostare la proprietà Value su True
- Un OptionButton a Name "optAutore" e Caption "Per Nome Autore" anch'esso collocato sul Frame. Impostare la proprietà Value su False.



Lasciamo l'evento Click del cmdRecordSet invariata tranne l'aggiunta alla fine dell'istruzione:

```
fraOrdinamenti.Visible = True
```

Nell'evento Click del primo OptionButton scriviamo questa routine:

```
Private Sub optCodice_Click()

Form1.Data1.Refresh
Set aut11 = Form1.Data1.RecordSet
zan11.Index = "AU_ID"

DBGrid1.Caption = "Video List per Codice Autore"
End Sub
```

E nell'evento Click del secondo OptionButton scriviamo:

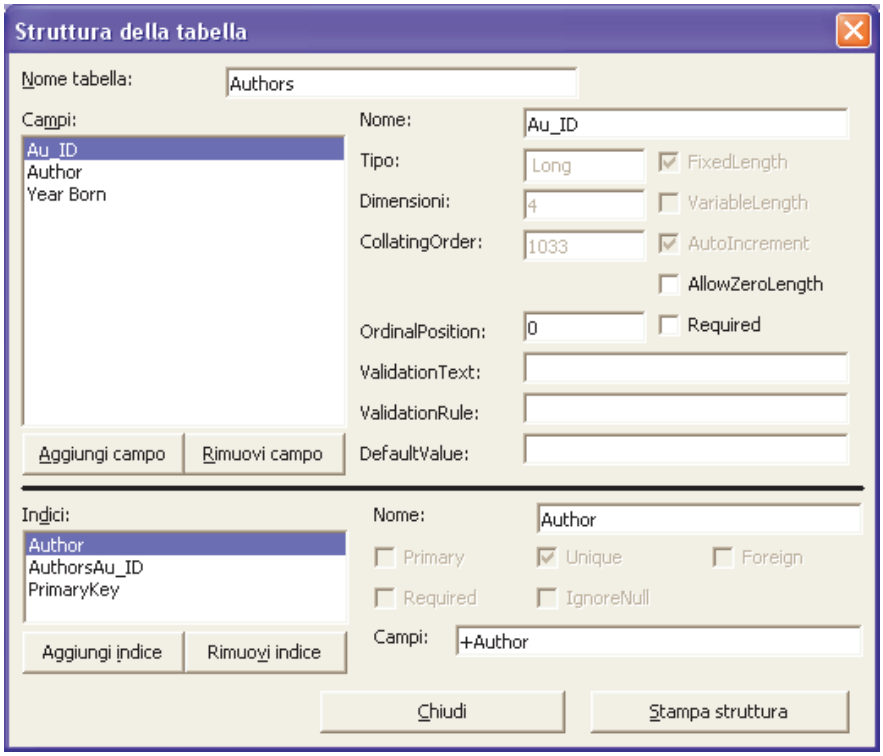
```
Private Sub optNome_Click()

Form1.Data1.Refresh
Set aut11 = Form1.Data1.RecordSet
zan11.Index = "AUTHOR"

DBGrid1.Caption = "Video List per Nome Autore"

End Sub
```

L'idea é che, lanciando il programma, il bottone Table apre un RecordSet di tipo Table, il Frame diviene visibile e si può modificare l'ordine dei Records a piacimento, da Codice Autore a Nome Autore.

C'e solo un inconveniente: l'indice "AUTHOR" non é presente nella Tabella AUTHORS del Database BIBLIO.

Nessun problema. Anche qui il Visual Data Manager ci viene in aiuto e ci permette di costruire questo nuovo indice. Operare come segue:

- Aprire il VBM e caricare il Database BIBLIO (C:\GRIGLIE\ARCHIVIODATI\BIBLIO.MDB)
- Col click destro del mouse selezionare la Tabella Authors/Progettazione.
- Premere il bottone "Aggiungi Indice" e definire un nuovo indice a nome "AUTHOR", deselezionando la casella Primary.
- Scegliere come campo indicizzato il Field "AUTHOR".
- Premere OK ed uscire.



Possiamo ora lanciare il programma ed osservare come la visualizzazione del DBGrid passi da ordine codice ad ordine nome Autore.

Modifichiamo ora l'evento Click del bottone cmdCerca come segue:

```
Private Sub cmdCerca_Click()

zricerca = txtCerca.Text
'---------------------
Select Case aut11.Index
Case "AUTHORSAU_ID"
aut11.Seek "=", zricerca
Case "AUTHOR"
aut11.Seek ">", zricerca
End Select
'---------------------
If aut11.NoMatch Then
MsgBox (zricerca & " non trovato")
aut11.MoveFirst
End If

End Sub
```

Se ora si rilancia il programma, si possono effettuare ricerche per codice Autore oppure per (lettera iniziale) nome Autore.

1.5 Controllo delle Colonne

Negli Esercizi precedenti abbiamo visto un DBGrid che visualizza i Records della Tabella prescelta in modo aderente alla struttura della Tabella stessa: i Fields hanno la sequenza che hanno nella Tabella, le dimensioni delle Colonne del DBGrid sono tutte uguali indipendentemente dalle dimensioni dei Fields e le intestazioni delle Colonne (Headers) sono identiche ai nomi dei Fields.

É venuto il momento di prendere in mano questi parametri.

Creiamo allora una nuova Cartella ESERCIZIO_04 e disponiamo nel Form1 la solita coppia DBGrid1 e Data1 collegati per Data1. Aggiungiamo ancora un CommandButton a nome "cmdCERCA" e a Caption "Cerca", nonché una TextBox a nome "txtCerca".

Per rendere un po' più interessante il Progetto, aggiungiamo anche un Modulo.
I moduli sono una specie di magazzino in cui si possono depositare inizializzazioni, dichiarazioni e subroutines di vario genere, con il vantaggio che tutti questi oggetti sono visibili da qualunque parte del Progetto.

Allora nella finestra "DICHIARAZIONI" del Modulo scriviamo:

```
global generali
'-------------------
Global pathmdb

' RecordSet aut
'-------------------
Global aut11 As RecordSet
Global aut21 As RecordSet
Global autvuoto
```

```
' RecordSet tit
'-------------------
Global tit11 As RecordSet
Global tit21 As RecordSet
Global titvuoto

' RecordSet pub
'-------------------
Global pub11 As RecordSet
Global pub21 As RecordSet
Global pubvuoto
```

E con questo abbiamo inizializzato un bel po' di variabili che utilizzeremo nel corso del nostro Progetto.

Sempre nel Modulo scriviamo inoltre una Subroutine a cui daremo il nome (arbitrario) di "APRIPROGETTO" a cui affidiamo il compito di effettuare i vari interventi e definizioni necessari quando si lancia il Progetto. Per ora il Progetto é semplice e la Subroutine si limita a scrivere:

```
Sub apriprogetto()

' path
'--------------------
pathmdb = "C:\GRIGLIE\ARCHIVIODATI\BIBLIO.MDB"

End Sub
```

Cominciamo ora a lavorare sul Form1 e scriviamo per l'evento "FORM LOAD" le seguenti istruzionI:

```
Private Sub Form_Load()
'apertura Progetto
'---------------------
apriprogetto

'definizione Data1
'---------------------
Data1.DatabaseName = pathmdb
Data1.Connect = ""
Data1.RecordSetType = 0          'Table

'riempimento Data1
'----------------------
Data1.RecordSource = "TITLES"
Data1.Refresh

'definizione del RecordSet
'--------------------
Set tit11 = Data1.RecordSet
tit11.Index = "TITLE"

'file vuoto
'--------------------
If tit11.EOF Then
```

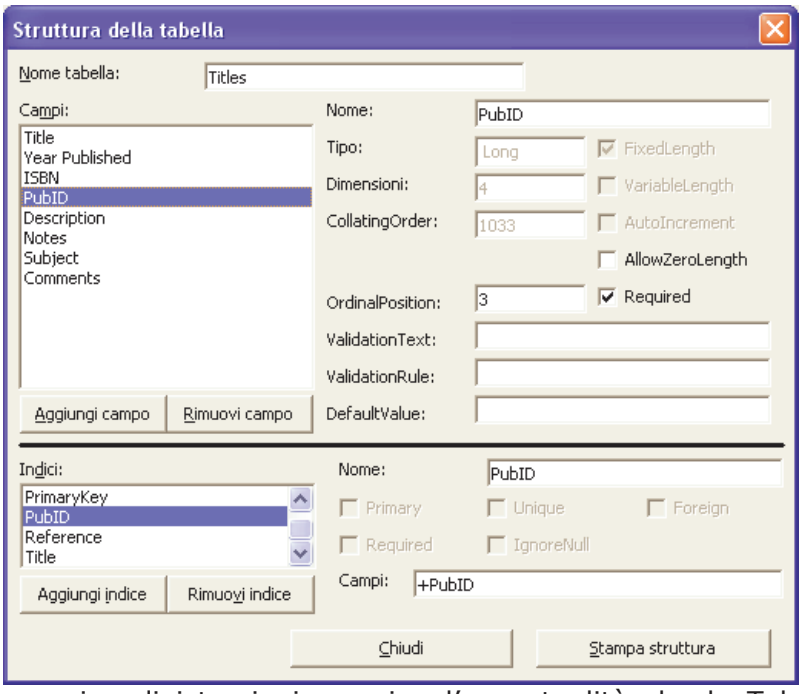
```
titvuoto = True
txtCerca.Enabled = False
else
titvuoto= False
End If

'DBgrid1
'---------------------------------------
DBGrid1.AllowUpdate = False
DBGrid1.Caption = "File Titoli per Titolo"
DBGrid1.Font.Bold = True
DBGrid1.HeadFont.Bold = True
DBGrid1.AllowRowSizing = False

caricacolonne

End Sub
```

Commentiamo ora le istruzioni che abbiamo scritto.

- La prima istruzione lancia la Subroutine APRIPROGETTO che sappiamo essere presente nel modulo e che definisce la variabile PATHMDB, cioè la collocazione di BIBLIO.MDB.

- I successivi tre gruppi di istruzioni aprono con modalità ormai note, un RecordSet di tipo Table che interessa la tabella TITLES e ne fissa l'indice a TITLE.

- Le caratteristiche della Tabella TITLES possono al solito essere studiate con il VDM.



- Il gruppo successivo di istruzioni esamina l' eventualità che la Tabella caricata sia vuota. Il RecordSet si trova allora in situazione di "EOF" ed occorre disabilitare il bottone "Cerca".

- Si passa ora a personalizzare il DBGrid. Per prima cosa si impedisce che l'utente possa fare modifiche ai Records agendo sul DBGrid. Questo é molto pericoloso perché senza ritorno e può causare notevoli mali di testa. Vedremo in seguito come operare in sicurezza.

- Si definisce inoltre la Caption cioè il Titolo del DBGrid e, se si vuole, i caratteri Bold, ben visibili, rispettivamente dei dati (DBGrid1.Font.Bold) e delle intestazioni colonne (DBGrid1.HeadFont.Bold).

- Si impone infine la proprietà RowSizing a False per evitare il fastidioso scarrellamento delle righe quando si toccano i bottoni di selezione Record.

Segue ora una Subroutine che dobbiamo aggiungere nel Form assegnandoli il nome (arbitrario) CARCICOLONNE. Per renderla accessibile anche da altri Form non dichiariamola "PRIVATE" Ecco la Routine:

```
Sub caricacolonne

`caricacolonne
`----------------------
For i = 0 To DBGrid1.Columns.Count - 1
DBGrid1.Columns.Remove 0
Next

For i = 0 To 3
DBGrid1.Columns.Add (i)
DBGrid1.Columns(i).Visible = True
Next

DBGrid1.Columns(0).DataField = "TITLE"
DBGrid1.Columns(0).Caption = "Titolo"
DBGrid1.Columns(0).Width = 2500

DBGrid1.Columns(1).DataField = "ISBN"
DBGrid1.Columns(1).Caption = "Cod.ISBN"
DBGrid1.Columns(1).Width = 1500

DBGrid1.Columns(2).DataField = "YEAR PUBLISHED"
DBGrid1.Columns(2).Caption = "Anno"
DBGrid1.Columns(2).Width = 700

DBGrid1.Columns(3).DataField = "PUBID"
DBGrid1.Columns(3).Caption = "Cd.Ed."
DBGrid1.Columns(3).Width = 700

'allineamento
'--------------------
DBGrid1.RowDividerStyle = 4
For i = 0 To 3
DBGrid1.Columns(i).DividerStyle = 4
DBGrid1.Columns(i).Alignment = 0
Next

'allineamento numeri
'--------------------
DBGrid1.Columns(2).Alignment = 1
DBGrid1.Columns(3).Alignment = 1
End sub
```

Questa Subroutine contiene una serie di istruzioni che servono a personalizzare e a definire il DBGrid. Commentiamole:

- Si comincia con l' eliminare qualunque colonna il DBGrid possegga per creare una situazione pulita e poter poi impostare il numero di colonne da noi desiderato. Per fare questo si fa uso dell' oggetto Columns associato al DBGrid. Una proprietà di questo oggetto é la Columns.Count che conta quante colonne sono presenti e poi con il metodo Columns.Remove si procede a svuotare il DBGrid.

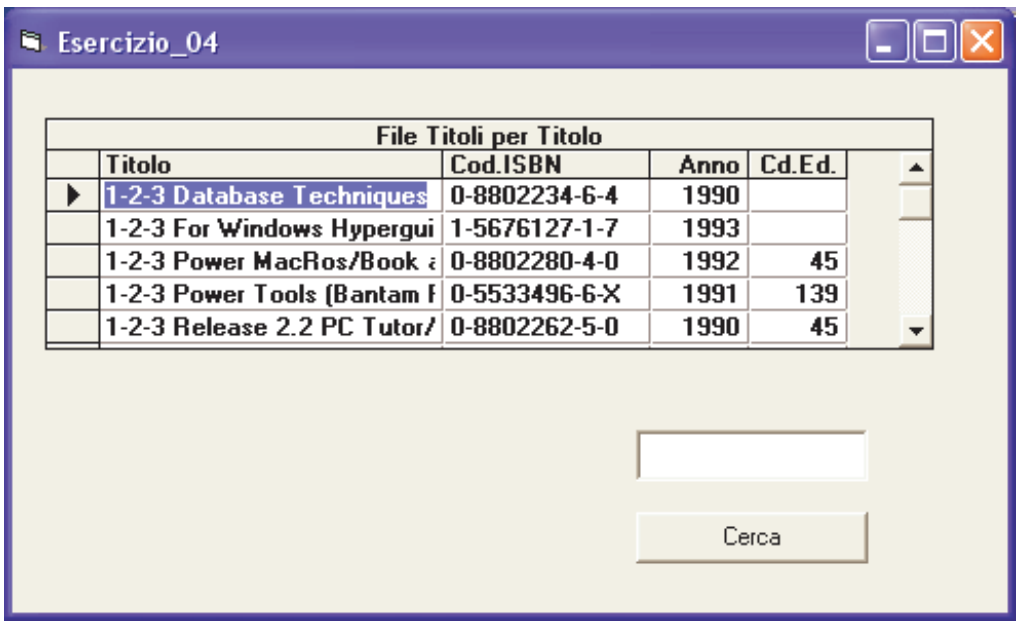
- Si lavora ora con un'altra serie di oggetti associati al DBGrid, i Columns(x). Ogni colonna del DBGrid é infatti caratterizzata da un Columns(x) ove "x" rappresenta l'ordine della colonna a cominciare da zero a salire. Se per esempio decidiamo di impostare un DBGrid con 4 colonne per prima cosa é necessario creare 4 Columns(x) da zero a 3. A questo provvede il ciclo FOR/NEXT di 4 giri (appunto da zero a tre) che ogni volta crea una colonna, rispettivamente Columns(0),(1),(2),(3) usando per questo il metodo Add dell'oggetto Columns associato al DBGrid. Non si faccia confusione tra il Columns e le Columns(x): il primo é un oggetto generale associato al DBGrid con le sue proprietà ed i suoi metodi, mentre il gruppo di seconde consiste in una serie di oggetti associati all'oggetto Columns, per cui anche le Columns(x) risultano a loro volta associate al DBGrid. Lo so, é un po' complicato ma poi ci si abitua… Anche le Columns(x) hanno le loro proprietà ed é su queste che lavoreremo.

- Si procede ora a personalizzare ciascuna Columns(x) così creata definendone le proprietà:il DataField, la Caption e la Width. Questo é il punto nel quale si sceglie quali Fields visualizzare, il loro ordine di incolonnamento, la larghezza della colonna ed il titolo della stessa.

- Si passa quindi a definire l'aspetto delle colonne,cioè il RowDividerStyle del DBGrid (per esempio codice 4, incassato) ed il DividerStyle delle Column(x) (per esempio tutte codice 4, incassato) nonché l'allineamento dei dati (tutti con codice zero, a sinistra).

- Se sono presenti colonne in cui sono presenti dati di tipo numerico, é più estetico mandarli a destra, codice 1. É questo il caso della Column(2) dato che l'Anno é numerico e della Column(3) dato che anche il Codice Editore é numerico

É fatta. Possiamo lanciare il Programma ed osservare il DBGrid personalizzato.



1.6 Estrazione di Dati

Vogliamo ora tornare ad occuparci dei RecordSet di tipo Dynaset perché essi posseggono alcune caratteristiche che apprezzeremo.

Cominciamo con un Esercizio in cui ci proponiamo di estrarre dalla Tabella TITLES un certo numero di Records che soddisfano ad una condizione da noi imposta.
Per esempio vogliamo estrarre tutti e solo quei Records che contengono nel Field TITLE la porzione "DBASE" cioè vogliamo trovare tutti e solo i libri che trattano di questo argomento, come annunciato dal loro titolo.

Procediamo col solito metodo di Copia/Incolla a partire dalla Cartella ESERCIZIO_04 e creiamo una nuova Cartella ESERCIZIO_05. Dopo aver aperto il Progetto1 in questa Cartella operiamo le seguenti modifiche:

- Modifichiamo le prime istruzioni della Subroutine Form_Load in modo che il RecordSet caricato sia di tipo Dynaset e precisamente:

```
'aperura Progetto
'--------------------
apriprogetto

'definizione Data1
'--------------------
Data1.DatabaseName = pathmdb
Data1.Connect = ""
Data1.RecordSetType = 1         'Dynaset

'riempimento Data1
'---------------------
titq1 = "select * from TITLES order by ISBN"
Data1.RecordSource = titq1
Data1.Refresh

'definizione del RecordSet
'--------------------
Set tit21 = Data1.RecordSet

'file vuoto
'--------------------
If tit21.EOF Then
titvuoto = True
txtCerca.Enabled = False
Else
titvuoto = False
End If
```

Per quanto riguarda la personalizzazione di DBGrid1 e la Subroutine "CARICACOLONNE", non occorre fare modifiche.

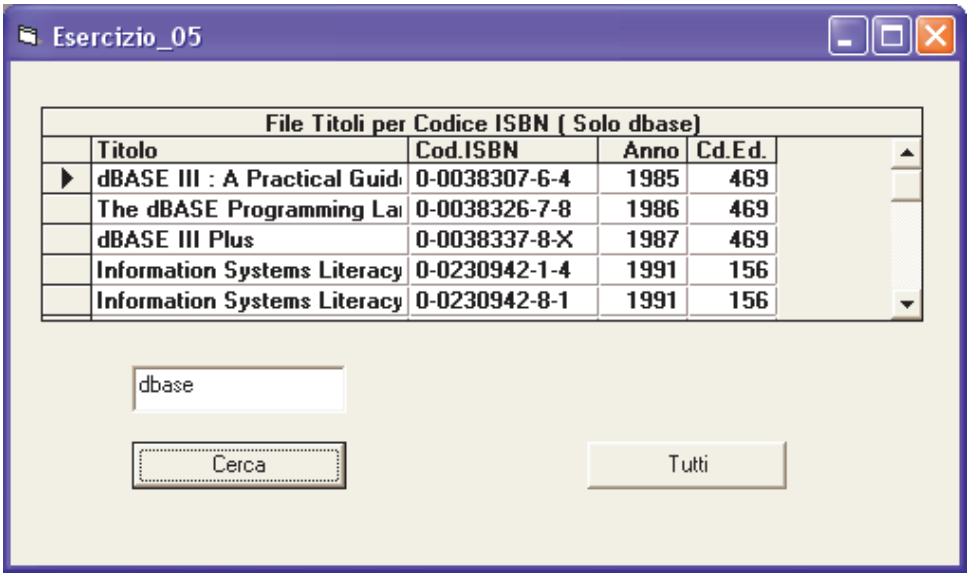
Se si lancia ora il Progetto si otterrà una situazione simile a quella dell'Esercizio precedente con la sola differenza che ora i Records sono ordinati secondo ISBN. Questa scelta é arbitraria e basta modificare la query TITQ1 per avere un altro ordinamento.

Vediamo ora le istruzioni per il CommandButton "CMDCERCA", evento Click:

```
Private Sub cmdCerca_Click()

'riempimento Data1
'---------------------
zricerca = txtCerca.Text
FILTRO1 = "Instr(TITLE, '" & zricerca & "' ) <>0"
titq1 = "select * from TITLES where " & FILTRO1 & " order by ISBN "
Data1.RecordSource = titq1
Data1.Refresh

'definizione del RecordSet
'--------------------
Set tit21 = Data1.RecordSet

caricacolonne
DBGrid1.Caption = DBGrid1.Caption & " ( Solo " & zricerca & ")"

End Sub
```

Come si vede, abbiamo fatto uso di un'altra query con cui si cercano tutti e solo i TITLE all'interno dei quali sia presente il contenuto alfanumerico della variabile ZRICERCA. Per questi Records il valore della funzione INSTR é diversa da zero. Resta inoltre confermato l'ordinamento per ISBN.

Se ora si lancia il programma e si scrive per esempio "DBASE" nella TextBox "TXTCERCA", il bottone cmdCERCA seleziona solo i Records desiderati producendo così un RecordSet parziale dei dati della Tabella "TITLES". Questa operazione non si sarebbe potuta fare se si avesse utilizzato un RecordSet di tipo Table.



Per terminare l'esercizio aggiungiamo un CommandButton a nome "cmdTutti" ed a Caption "Tutti" che serve per ritornare alla situazione iniziale di un RecordSet totale,

Per ottenere questo risultato é sufficiente rilanciare il Form Load con la semplice istruzione:

```
Private Sub cmdTutti_Click()
Form_Load
End Sub
```

1.7 Join di Tabelle

Vogliamo ora occuparci di un altro problema: come realizzare un DBGrid contenente colonne con dati prelevati da varie Tabelle.
Per esempio, vogliamo realizzare un DBGrid che mostri i dati della Tabella "TITLES" ed in cui oltre che al Codice dell'Editore "PUBID" compaia anche il nome e la città di tale editore, da prelevare dall'altra Tabella "PUBLISHERS" cioè i Fields "NAME" e "CITY" (Si controlli con il solito VDM).

Procediamo col solito metodo di Copia/Incolla a partire dalla Cartella ESERCIZIO_05 e creiamo una nuova Cartella ESERCIZIO_06. Dopo aver aperto il Progetto1 in questa Cartella operiamo le necessarie seguenti modifiche.
Per prima cosa modifichiamo la Query di definizione RecordSet come segue:

```
titq1 = "select * from TITLES left outer join PUBLISHERS "
titq1 = titq1 & " on TITLES.PUBID=PUBLISHERS.PUBID order by ISBN "
```

TITQ1 é definita in due step solo per migliore leggibilità e fa uso del metodo JOIN che unisce le due Tabelle TITLES e PUBLISHERS attraverso il Field PUBID. Questa istruzione crea una nuova Tabella virtuale in cui tutti i Records della prima si trovano associati il nome , indirizzo, città etc, cioé tutti i Fields della seconda, secondo una corrispondenza del Field "PUBID".

Si noti comunque che non é necessario che i Field di collegamento abbiano lo stesso nome nelle due Tabelle. Se per esempio il campo Codice Editore della Tabella PUBLISHERS si fosse chiamato con un altro nome, per esempio CODICEDITORE, la seconda parte della Query avrebbe preso questa forma:

```
titq1 = titq1 & " on TITLES.PUBID=PUBLISHERS.CODICEDITORE order by ISBN "
```

Si é scelta l'istruzione LEFT OUTER JOIN perché essa consente di includere tutte le righe della prima tabella specificata, ovvero la prima tabella (a sinistra) della istruzione JOIN, e quindi consente di visualizzare tutti i TITLES, anche quelli per cui non sono disponibili informazioni sull'editore.

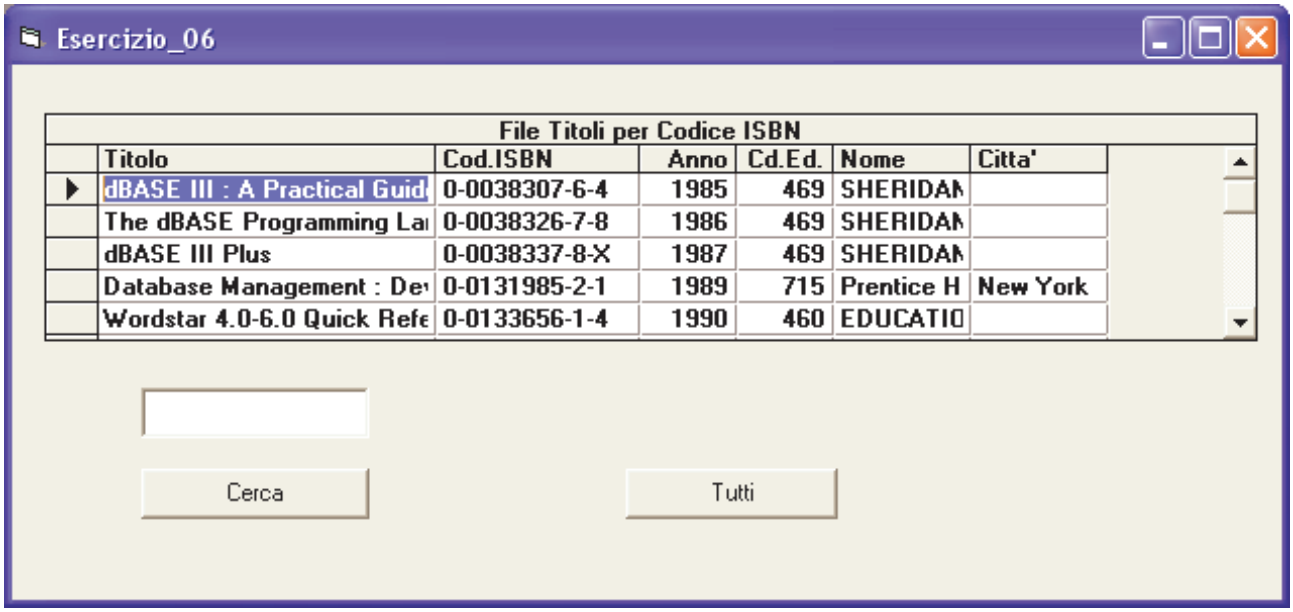
Passiamo ora alla Subroutine CARICACOLONNE. Eccone la nuova scrittura:

```
Sub caricacolonne()

'caricacolonne
'--------------------
For i = 0 To DBGrid1.Columns.Count - 1
DBGrid1.Columns.Remove 0
Next

For i = 0 To 5
DBGrid1.Columns.Add (i)
DBGrid1.Columns(i).Visible = True
Next

DBGrid1.Columns(0).DataField = "TITLE"
DBGrid1.Columns(0).Caption = "Titolo"
DBGrid1.Columns(0).Width = 2500

DBGrid1.Columns(1).DataField = "ISBN"
DBGrid1.Columns(1).Caption = "Cod.ISBN"
DBGrid1.Columns(1).Width = 1500

DBGrid1.Columns(2).DataField = "YEAR PUBLISHED"
```

```
DBGrid1.Columns(2).Caption = "Anno"
DBGrid1.Columns(2).Width = 700

DBGrid1.Columns(3).DataField = "TITLES.PUBID"
DBGrid1.Columns(3).Caption = "Cd.Ed."
DBGrid1.Columns(3).Width = 700

DBGrid1.Columns(4).DataField = "NAME"
DBGrid1.Columns(4).Caption = "Nome"
DBGrid1.Columns(4).Width = 1000

DBGrid1.Columns(5).DataField = "CITY"
DBGrid1.Columns(5).Caption = "Città"
DBGrid1.Columns(5).Width = 1000

'allineamento
'--------------------
DBGrid1.RowDividerStyle = 4
For i = 0 To 5
DBGrid1.Columns(i).DividerStyle = 4
DBGrid1.Columns(i).Alignment = 0
Next

'allineamento numeri
'--------------------
DBGrid1.Columns(2).Alignment = 1
DBGrid1.Columns(3).Alignment = 1
End Sub
```

Come si vede, si é scelto di visualizzare sei colonne (indice da zero a 5) della Tabella "Jointata" ove le ultime due colonne appartengono originariamente alla Tabella "PUBLISHERS".
Per la Columns(3) é stato necessari specificare come DataField il "TABLES.PUBID" perché nel nostro caso il campo di giunzione si chiama "PUBID" in entrambe le Tabelle e quindi bisogna specificare quale PUBID delle due Tabelle si vuole visualizzare. Se i due campi di giunzione avessero avuto nomi diversi, questo non sarebbe stato necessario.

Lanciamo il programma e vedremo come il DBGrid visualizza i dati della Tabella "jointata".



Passiamo ora al bottone “CERCA” e scriviamo per l’evento Click quanto segue:

```
Private Sub cmdCerca_Click()

'riempimento Data1
'---------------------
zricerca = txtCerca.Text
FILTRO1 = "Instr(TITLE, '" & zricerca & "' ) <>0"
titq1 = "select * from TITLES left outer join PUBLISHERS "
titq1 = titq1 & " on TITLES.PUBID=PUBLISHERS.PUBID "
titq1 = titq1 & " where " & FILTRO1 & " order by ISBN"
Data1.RecordSource = titq1
Data1.Refresh

'definizione del RecordSet
'--------------------
Set tit21 = Data1.RecordSet

caricacolonne
DBGrid1.Caption = DBGrid1.Caption & " ( Solo " & zricerca & ")"

End Sub
```

Anche qui la Query “TITQ1” é stata scritta in tre step per ragioni di leggibilità e risulta simile alla Query vista per l’Esercizio precedente, presentando in più l’istruzione di “JOIN”.

Il Bottone “TUTTI” rimanda come al solito al Load del Form.

Come utile esercizio si può modificare il Progetto cambiando queste istruzioni:

- Il terzo step della Query diviene:
titq1 = titq1 & " on TITLES.PUBID=PUBLISHERS.PUBID order by TITLES.PUBID "

- La definizione della Caption del DBGrid diviene:
    'DBGrid1.Caption = "File Titoli per Codice Editore"

- Il terzo step della Query del Bottone “CERCA” diviene
'titq1 = titq1 & " where " & FILTRO1 & " order by TITLES.PUBID"

Con queste modifiche si visualizza il DBGrid in ordine di Codice Editore.

Capitolo2 . Lettura a tutto schermo

2.1 Preparazione di una maschera a tutto schermo

Il DBGrid è certamente uno strumento molto potente ma soffre di una limitazione: ogni Record deve essere visto come una striscia di dati e se il numero di campi della Tabella è elevato occorre fare spesso uso dello scorrimento orizzontale per vedere tutti i campi del Record.

Sarebbe allora interessante poter disporre, oltre che del DBGrid, anche della possibilità di leggere tutti i dati di ciascun Record con un Form/Maschera che occupi tutto lo schermo ed in cui i dati risultino disposti in modo logico e completo.

Si tratta quindi di affrontare il problema di passare dall'uno all'altro dei due Forms:
- quello con il DBGrid
- quello a tutto schermo e ritorno.

Poniamoci inoltre un obiettivo più ambizioso: vogliamo realizzare una procedura che permetta di lavorare in multiutenza, cioè una rete locale a cui possono accedere più utenti. Sarà allora necessario che ogni procedura preveda questa eventualità e blocchi/liberi i Record come necessario per evitare interferenze.

Per questo esercizio abbiamo bisogno di un altro Database, più didattico di BIBLIO. Immaginiamo di avere una libreria piena di Cassette, Dischi e Riviste e di volerli classificare per argomento, autore, attore principale etc . Dovremo costruire e riempire un database come quello allegato nella Cartella C:\ARCHIVIODATI\CDRFILE.MDB. Se lo apriamo con il solito Visual Data Manager si noterà che presenta i seguenti Files/Tabelle:

- ARGFILE, file degli argomenti
- ATTFILE, file dei nomi degli attori principali
- AUTFILE, file dei nomi degli autori/registi
- DATFILE, elenco delle nostre cassette, dischi e riviste.

Quest'ultimo é il file per cui intendiamo presentare i dati del Record a tutto schermo.

Prepariamo allora l'ambiente di lavoro, una Cartella a nome C:\GRIGLIE\ESERCIZIO_07 ed aggiungiamo al Progetto due Form a cui daremo il nome "frmDatsk1", e "frmDatsk2", destinati a contenere, il primo, la Video Lista dei dati (il DBGRid) ed il secondo la visualizzazione dei Records a tutto schermo. Non mancherà un Modulo per immagazzinare le varie Subroutines di uso comune.

Osserviamo dapprima la struttura del DATFILE per studiare quali dati e in quale ordine vogliamo visualizzare nella Videolista/DBGrid.

Campo	Nome campo	Tipo	Dim
1	CODICE	Numerico	Intero
2	TIPO	Testo	1
3	TITOLO	Testo	50
4	AUTCOGN	Testo	15
5	AUTNOME	Testo	15
6	ATTORE	Testo	20
7	ARGOME	Testo	25
8	DATA	Data/Ora	
9	VALORE	Numerico	Precisione Doppia
10	NOTA	Testo	50
11	FLIPRETE	Numerico	Intero

Come si vede sono presenti campi di diversa natura che ci permetteranno di familiarizzarci anche con le Date, l'Euro a due cifre decimali, etc..

Cominciamo da frmDATSK1 ed inseriamo la solita coppia DBGrid/Data. Nel Form Load scriviamo la routine di partenza relativa al RecordSet "DAT11" di tipo Table che qui non riportiamo perché é simile a quelle già viste in precedenza, per esempio per l'Esercizio 04 con la sola differenza che ora apriamo la Tabella "DATFILE" sotto indice "CODICE".

La routine si può esaminare nell'Esercizio svolto in Cartella SOUZIONE_07.

Il "PATHMDB" viene definito nella Subroutine "APRIPROGETTO" inserita nel Modulo 1 risulta la seguente:

```
pathmdb = "C:\GRIGLIE\ARCHIVIODATI\CDRFILE.MDB"
```

Non riportiamo neanche la Subroutine "APRICOLONNE", anch'essa presente nell'Esercizio svolto e ci limitiamo a commentare le definizioni delle Column(x) numero 6 e 7.

La prima:

```
DBGrid1.Columns(6).NumberFormat = "### ###.00;;#"
```

Comporta l'uso della Proprietà NumberFormat ed impone che i valori della Colonna dovranno essere presentati tutti con due cifre decimali (Euro e Centesimi) ed in caso il valore sia zero vogliamo che il campo sia bianco senza un antiestetico "0". Inoltre se sono presenti valori elevati, superiori ai 1000 Euro, le migliaia dovranno essere separate per migliore leggibilità.

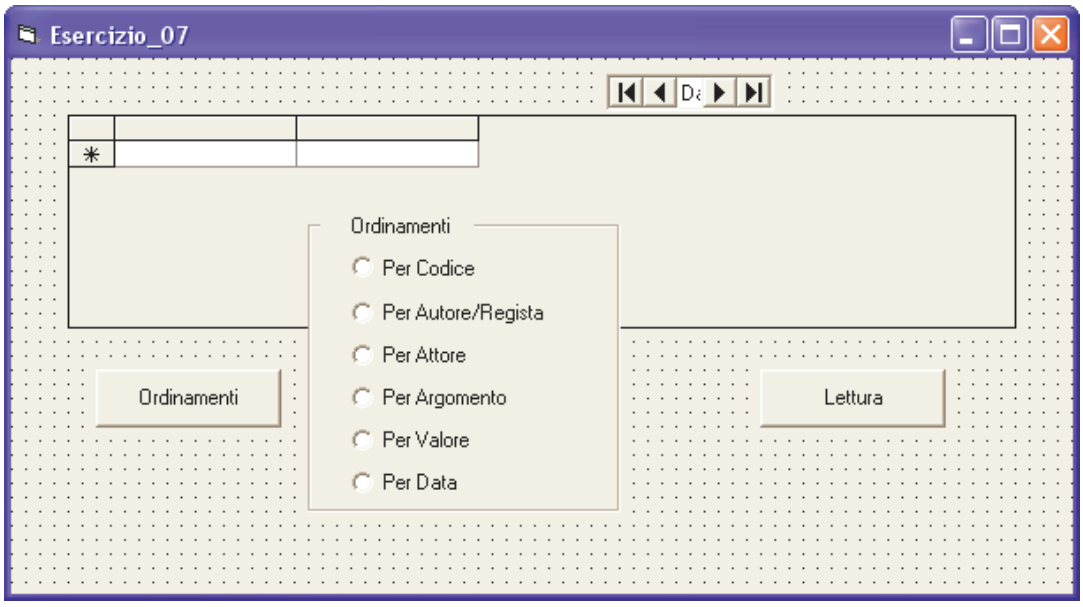
La seconda:

```
DBGrid1.Columns(7).NumberFormat = "dd/mm/yy"
```

indica che I valori della colonna dovranno riportare la data in notazione europea e con l'anno di solo due cifre. Questa notazione tuttavia conviene solo se si é sicuri di gestire dati di un solo secolo. Se si gestiscono dati di entrambi i secoli, 1900 e 2000, l'istruzione va omessa o sostituita con "dd/mm/yyyy".

Per terminare il lavoro sul frmDATSK1 aggiungiamo ancora un Frame (Visibile= False) ed un certo numero di OptonButtons come visto in precedenza per l'Esercizio 03, nonché un CommandButton "CMDORDINAMENTI" la cui funzione é solo quella di far comparire il Frame.

```
Private Sub cmdOrdinamenti_Click()
fraOrdinamenti.Visible = True
End Sub
```



Gli ordinamenti possibili per il file DATFILE sono definiti dagli indici che si sono creati in fase di progettazione Tabella. Essi sono visibili col VDM e sono:

Indice	Fields
ARGOME	ARGOME/CODICE
ATTORE	ATTORE/CODICE
AUTCOGN	AUTCOGN/AUTNOME/CODICE
CODICE	CODICE
DATA	DATA/CODICE
VALORE	VALORE/CODICE

Premendo su un OptionButton si modifica l'indice del RecordSet e si possono visualizzare i dati con diversi ordinamenti.
Le Subroutines degli eventi click sono tutte simili: mostriamone una per tutte:

```
Private Sub optaut_Click()

Data1.Refresh
Set dat11 = Data1.RecordSet
dat11.Index = "AUTCOGN"

caricacolonne
DBGrid1.LeftCol = 3
DBGrid1.Caption = "Video List per Autore/Regista"
datordi = 2
fraOrdinamenti.Visible = False
optaut.Value ( False

End Sub
```

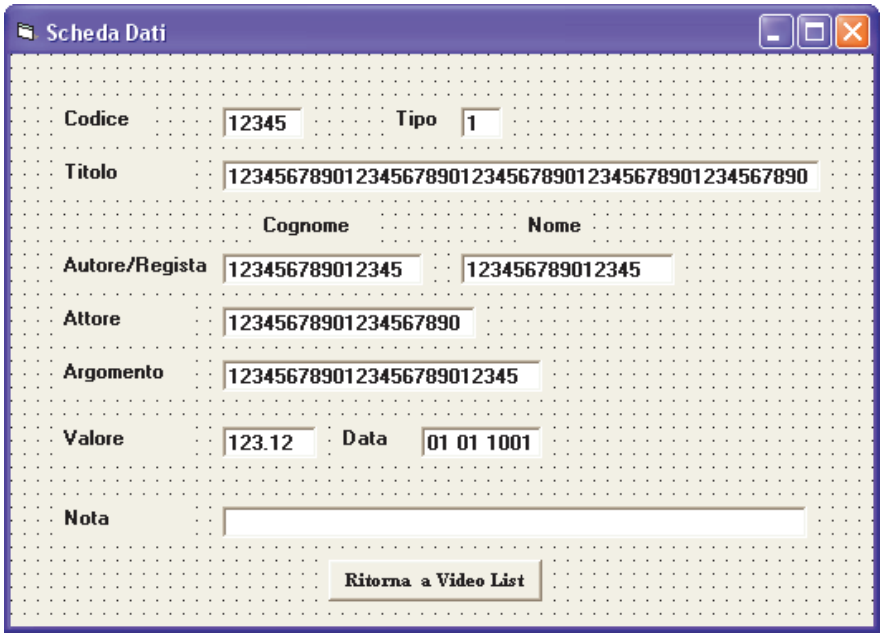
L'istruzione relativa alla "LEFTCOL" del DBGrid (qui presentata al solo fine didattico) fa si che cambiando l'ordinamento, la prima colonna mostrata sul DBGrid sia quella del campo ordinato. Le colonne sono comunque tutte presenti, dato che ogni volta interviene "CARICACOLONNE".

Il Frame viene reso non visibile e l'Option Button ridiviene disponibile.

Passiamo ora alla preparazione del "frmDatsk2". Si tratta di aggiungere sul Form un certo numero di coppie Label/TextBox, tante quante sono i campi della Tabella che intendiamo visualizzare, disponendo questi oggetti in modo da soddisfare la logica e l'estetica.

Questa sarà la scheda che apparirà quando lo richiederemo.
Per richiuderla e ritornare al DBGrid occorre aggiungere un CommandButton a cui assegneremo:

Nome "cmdRitvid"
Caption "Ritorna a Video List":
Proprietà Visibile "False"



Per quanto riguarda i nome delle TextBox conviene utilizzare il nome del canpo corrispondente, preceduto dalle lettere "txt" come di seguito illustrato.

Field	Tipo	Dim	Label	Nome Casella
CODICE	Carattere	5	Codice	TXTCODICE
TIPO	Carattere	1	Tipo	TXTTIPO
TITOLO	Carattere	50	Titolo	TXTTITOLO
AUTCOGN	Carattere	15	Autore/Regista	TXTAUTCOGN
AUTNOME	Carattere	15	Nome	TXTAUTNOME
ATTORE	Carattere	20	Attore	TXTATTORE
ARGOME	Carattere	25	Argomento	TXTARGOME
DATA	Data	10	Data	TXTDATA
VALORE	Numerico	6	Valore	TXTVOTO
NOTA	Carattere	50	Nota	TXTNOTA

La convenienza di questa standardizzazione apparirà chiara con la presentazione delle Subroutines successive. I nomi delle Labels invece possono essere qualunque.

Le proprietà Alignment delle TextBox possono restare col valore de default "Left Justify" per i campi di tipo carattere ma per i campi di tipo numerico è esteticamente meglio imporre un Right Justify. É questo il caso di "txtCodice" e di "txtValore".

Conviene anche delimitare la MaxLength di ciascuna caselle con un numero pari alla lunghezza del campo, in modo che l'utente non possa inavvertitamente introdurre dati più lunghi del consentito. Per i campi numerici imponiamo una lunghezza arbitraria che definisce il numero massimo permesso. Per il CODICE ad esempio 5 caratteri (99999) e per il VALORE 6 (999 Euro, il punto e 99 Cents). Si tratta naturalmente di scelte arbitrarie da adattare al caso.

Si possono mostrare tutti i campi che si desiderano, tranne “FLIPRETE” che, come studieremo più tardi, serve per bloccare/sbloccare i Record quando si lavora in rete.

Per rendere più esteticamente valido il Form "frmDatsk2”, conviene inoltre selezionare il carattere grassetto per le Font delle Label e delle TextBox. Questo si può fare o a livello di ogni casella oppure, meglio, cambiando la Proprietà Font del frmDatsk2 prima di cominciare ad aggiungere le caselle ed imponendo il grassetto. Tutti gli oggetti posti sul Form avranno così la scritta ben marcata e ben visibile.

Per arrangiare e sistemare la varie caselle si useranno i vari strumenti messi a disposizione dal Visual Basic per muovere e sistemare gli oggetti. Particolarmente utile risulta qui la possibilità di selezionare gruppi di oggetti con il Click/Tasto Control.

Un utile espediente per misurare la lunghezza che conviene assegnare alle caselle è quello di imporre artificialmente una proprietà Text con tante lettere o numeri quanto è lungo il campo della Tabella. Ad esempio per un campo lungo 10 si potrà imporre un Text '1234567890'. Questo permette di regolare in prima approssimazione la lunghezza della casella su Form.

A fine lavori conviene controllare la proprietà TabIndex delle varie caselle. Tale proprietà determina l'ordine in cui tali caselle verranno riempite dall'utente quando questi batte il tasto Tab (o come vedremo, Return). Conviene quindi imporre una sequenza (0, 1, 2, 3 etc) corrispondente all'ordine con cui si vuole che poi si passi da una casella all'altra con detto tasto. Premere infine il tasto Tab e verificare di aver impostato l’ordine desiderato.

Per terminare il Form vediamo ancora il bottone che servirà a far scomparire il Form stesso e a ritornare al DBGrid. Si scrivano per l'evento Click() le seguenti istruzioni:

```
Private Sub cmdritvid_Click()
frmDatsk1.Enabled = True
unload frmDatsk2
End Sub
```

Riprendiamo ora il DATSK1, aggiungiamo un nuovo bottone, assegnamogli il nome "CMDREAD", Caption "Lettura" e scriviamo per l’evento CliCk:

```
Private Sub cmdread_Click()

On Error GoTo errori

'prepara maschera
'----------------
dattext
datfresca
dattext

'mostra maschera
'-----------------
frmDatsk2.cmdritvid.Visible = True
frmDatsk2.Show
frmDatsk1.Enabled = False

'routine errori
'------------------
Exit Sub
errori:

Select Case Err
```

```
Case 3167, 3021
MsgBox ("sorry. il Record nel frattempo é stato" & acapo & " cancellato da un altro utente !")
dat11.MoveFirst
Case Else
MsgBox ("errore imprevisto No. " & Err)
End Select

End Sub
```

Le istruzioni del CmdRead_Click() possono essere commentate come segue:

Quando il bottone viene clikkato, vogliamo che il Form "frmDatsk2", descritto in precedenza, venga presentato con le caselle testo riempite con i valori dello specifico Record su quale si trova il DBGrid in quel momento.

La soluzione più semplice sembrerebbe allora quella di preparare il "frmDatsk2" con una serie di istruzioni del tipo:

```
frmDatsk2.txtcodice.Text = dat11!codice
frmDatsk2.txtTIPO.Text = dat11!TIPO
frmDatsk2.txtTITOLO.Text = dat11!TITOLO
frmDatsk2.txtautcogn.Text = dat11!AUTCOGN
frmDatsk2.txtautnome.Text = dat11!AUTNOME
frmDatsk2.txtattore.Text = dat11!ATTORE
eccetera
```

Purtroppo le cose non sono così semplici e questo principalmente per tre motivi:

a) Quando si intende fare una lettura a tutto schermo bisogna per prima cosa rinfrescare il Record che si vuole leggere per essere sicuri che i dati del RecordSet appoggiato alla Tabella sottostante siano i più recenti. Infatti questi dati potrebbero essere stati nel frattempo modificati da qualche altro utente che lavora in rete e che potrebbe essere intervenuto proprio su di essi.

Per tener conto di questa eventualità occorre quindi lanciare sempre una routine di rinfresco che chiameremo DATFRESCA e che studieremo fra poco quando esamineremo il modulo del Progetto, Modulo1. È questo infatti il posto migliore in cui conviene scrivere la routine al fine di renderla accessibile da ogni Form del Progetto.

b) Dato che lavoriamo in rete, non possiamo escludere che qualche altro utente abbia nel frattempo cancellato proprio il Record che intendiamo leggere. È quindi necessario prevedere una routine che catturi un errore di questo tipo.

c) Da ultimo, le semplici istruzioni sopra presentate, del tipo:

```
frmDatsk2.txtAUTCOGN.Text = dat11!AUTCOGN
```

eccetera, potrebbero non funzionare se capita che il Field che viene richiamato (nell'esempio: AUTCOGN, Field di DATFILE) si trova ad essere vuoto, cosa peraltro che può succedere normalmente e spesso, dato che non è obbligatorio riempire tutti i Fields dei Records che si aggiungono ala Tabella.

In questo caso infatti il Visual Basic lancia un messaggio di errore:

```
errore di Run Time 94: Utilizzo non valido di Null.
```

Per essere sicuri non resta altro che aggirare l'ostacolo e trasformare le istruzioni in sequenze di questo tipo:

```
If dat11!AUTCOGN <> Empty Then
frmDatsk2.txtAUTCOGN.Text = dat11!AUTCOGN
Else
frmDatsk2.txtAUTCOGN.Text = Empty
End If
```

Con questo blocchetto, anche se dat11!AUTCOGN é "Empty", non ci sono problemi.
Purtroppo questo blocchetto va riscritto per ogni Field, ma se si fa uso delle capacità dell'Editor, con "Ctrl–H" si può sostituire AUTCOGN con TIPO etc e con un po' di pratica ci vuole un minuto.

Queste istruzioni vanno raccolte in una Subroutine che chiameremo DATTEXT. Anche per questa routine il miglior posto di scrittura è il Modulo1 del Progetto.
Facciamo allora una breve digressione per studiare queste due routines DATTEXT e DATFRESCA, prima di riprendere in esame il funzionamento dell'evento Cmdread_Click().

2.2 La Subroutine DATTEXT e la formattazione Euro

Apriamo allora il Modulo1 e scriviamo:

```
Dichiarazioni

' global generali
'------------------
Global pathmdb
Global acapo

' RecordSet dat
'------------------
Global dat11 As RecordSet
Global dat21 As RecordSet
Global datvuoto
Global datordi
Global fladat
```

```
Sub apriprogetto()

'generali
'--------------------
pathmdb = "C:\GRIGLIE\ARCHIVIODATI\CDRFILE.MDB"
acapo = Chr$(13) + Chr$(10)

'ordinamento
'--------------------
datordi = 1

End Sub
```

```
Sub dattext()

If dat11!codice <> Empty Then
frmDatsk2.txtcodice.Text = dat11!codice
Else
frmDatsk2.txtcodice.Text = Empty
End If

If dat11!TIPO <> Empty Then
frmDatsk2.txttipo.Text = dat11!TIPO
Else
frmDatsk2.txttipo.Text = Empty
End If

If dat11!TITOLO <> Empty Then
frmDatsk2.txttitolo.Text = dat11!TITOLO
Else
frmDatsk2.txttitolo.Text = Empty
End If

If dat11!AUTCOGN <> Empty Then
frmDatsk2.txtautcogn.Text = dat11!AUTCOGN
Else
frmDatsk2.txtautcogn.Text = Empty
End If

If dat11!AUTNOME <> Empty Then
frmDatsk2.txtautnome.Text = dat11!AUTNOME
Else
frmDatsk2.txtautnome.Text = Empty
End If

If dat11!ATTORE <> Empty Then
frmDatsk2.txtattore.Text = dat11!ATTORE
Else
frmDatsk2.txtattore.Text = Empty
End If
If dat11!ARGOME <> Empty Then
frmDatsk2.txtargome.Text = dat11!ARGOME
Else
frmDatsk2.txtargome.Text = Empty
End If

If dat11!Data <> Empty Then
frmDatsk2.txtdata.Text = dat11!Data
Else
frmDatsk2.txtdata.Text = Empty
End If

If dat11!valore <> Empty Then
frmDatsk2.txtvalore.Text = dat11!valore
Else
frmDatsk2.txtvalore.Text = Empty
End If

If dat11!NOTA <> Empty Then
frmDatsk2.txtnota.Text = dat11!NOTA
Else
```

```
frmDatsk2.txtnota.Text = Empty
End If

'formattazioni euro
'------------------
frmDatsk2.txtvalore = virformat(frmDatsk2.txtvalore)

End Sub
```

Come si vede, la Subroutine é un seguito di blocchetti simili a quello descritto sopra.

Per il campo VALORE, Euro/Centesimi, dobbiamo fare alcune considerazioni generali perché vogliamo soddisfare queste esigenze:

- Vogliamo che la TextBox riporti sempre i centesimi anche se nella Tabella c'é un valore intero. Vogliamo cioè vedere "6.00" e non "6".
- Vogliamo poter immettere in questa Casella i valori usando indifferentemente il punto o la virgola per i centesimi.
- Vogliamo poter fare delle operazioni con questa casella, sommarla, moltiplicarla etc ad altre caselle numeriche.

Il Visual Basic non si presta immediatamente a tutte queste esigenze per cui bisogna procurarsi due semplici strumenti che risolvono il problema.

Il primo é una funzione che collocheremo nel solito Modulo1 e che provvede alla formattazione Euro della TextBox.

```
Function virformat(INESAME)

'descrizione
'video formatta con centesimi
'un testo che esprime un numero
'max 999 999 999 -> ### ### ###.00
'--------------------------------
'esempio
'frmDatsk2.txtvalore = virformat(frmDatsk2.txtvalore)
'--------------------------------
VALESAME = virvalore(INESAME)

Select Case VALESAME
Case Is >= 0.1
virformat = Format(VALESAME, "### ### ### ###.00")
Case Is = 0
virformat = ""
Case Is < 0
virformat = Format(VALESAME, "### ### ### ###.00")
Case Else
virformat = Format(VALESAME, "### ### ### ###.##")
End Select

End Function
```

Se per esempio mettiamo l'istruzione

```
txtvalore = virformat(txtvalore)
```

nell'evento "LOSTFOCUS" della casella "TXTVALORE", possiamo stare tranquilli che qualunque importo in Euro con virgola/punto centesimi abbiamo introdotto, questo importo risulterà sempre formattato correttamente. Se poi abbiamo scritto per errore delle lettere, l'importo va a zero e la casella é blank. Questo soddisfa le prime due esigenze.

La funzione visualizza inoltre l'importo con degli spazi di separazione per le migliaia, rendendo il valore ben leggibile. Questo vantaggio va però compensato assegnando un adeguato valore di "MAXLENGHT" alla casella. A seconda dell'importo massimo previsto é meglio non scendere sotto una "MAXLENGHT" di !3/15.

Per la terza necessità, cioè eseguire calcoli con le variabili, occorre che i centesimi siano separate da dei punti e non da virgole.
Per convincersene basta fare un piccolo esperimento con la finestra "IMMEDIATA", raggiungibile p.e. con un Ctrl-G e scrivere:
a=1.2
b=2.2
c=a+b
?c
3,4
Tutto a posto, però se si prova a definire la variabile "A" con una virgola, il VB va in errore.

Occorre allora una funzione opposta a quella vista un precedenza che elimini la formattazione e lasci solo dei punti per separare i centesimi. Ecco la funzione:

```
Function virvalore(INESAME)

'descrizione
'trasforma un campo txt in campo numerico
'se é presente una virgola la trasforma in punto
'per permettere i calcoli
'--------------------------------
'esempio
'A = virvalore(txt) / 2
'esempio
'dat11!valore = virvalore(frmDatsk2.txtvalore.Text)
'--------------------------------
If InStr(1, INESAME, ",") <> 0 Then
zinstr = InStr(1, INESAME, ",")
INESAME = Mid$(INESAME, 1, zinstr - 1) & "." & Mid$(INESAME, zinstr + 1)
End If

virvalore = Val(INESAME)

End Function
```

Con questa funzione posta nel Modulo 1 possiamo operare i nostri calcoli usando non il "TXTVALORE" bensì il "virvalore(txtValore)".

2.3 La Subroutine DATFRESCA

Passiamo ora alla routine di rinfresco, di cui abbiamo parlato sopra.

La routine opera come segue:

- Memorizza i dati del Record corrente.
- Effettua una rilettura della Tabella (Metodo Refresh dell'oggetto Data1)
- Ricostruisce di conseguenza il RecordSet DAT11.
- Associa l'indice opportuno, a seconda del valore della variabile DATORDI. Questa variabile é definita "1" (ordinamento per CODICE) nel Modulo di lancio e assume vari valore, 2,3 etc quando si selezionano i rispettivi OptionButtons di ordinamento.
- Ritorna al Record da cui si era partiti mediante una operazione di Seek utilizzando i dati in precedenza memorizzati. L' operazione di Seek è diversa a seconda della definizione dell'indice sotto il cui ordinamento ci si trova.

La routine è presentata qui di seguito:

```
Sub datfresca()

'memorizza campi Record
'---------------------------------
zcodice = virvalore(frmDatsk2.txtcodice.Text)

If frmDatsk2.txtautcogn.Text <> Empty Then
zautcogn = Left(frmDatsk2.txtautcogn.Text, 15)
Else
zautcogn = Space(15)
End If

If frmDatsk2.txtautnome.Text <> Empty Then
zautnome = Left(frmDatsk2.txtautnome.Text, 15)
Else
zautnome = Space(15)
End If

If frmDatsk2.txtattore.Text <> Empty Then
zattore = Left(frmDatsk2.txtattore.Text, 20)
Else
zattore = Space(20)
End If

If frmDatsk2.txtargome.Text <> Empty Then
zargome = Left(frmDatsk2.txtargome.Text, 25)
Else
zargome = Space(25)
End If

zvalore = virvalore(frmDatsk2.txtvalore.Text)

If frmDatsk2.txtdata.Text <> Empty Then
zdata = CDate(frmDatsk2.txtdata.Text)
Else
zdata = Empty
End If
```

```
Select Case datordi

Case 1
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "CODICE"
dat11.Seek "=", zcodice

Case 2
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "AUTCOGN"
dat11.Seek "=", zautcogn, zautnome, zcodice

Case 3
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "ATTORE"
dat11.Seek "=", zattore, zcodice

Case 4
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "ARGOME"
dat11.Seek "=", zargome, zcodice

Case 5
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "VALORE"
dat11.Seek "=", zvalore, zcodice

Case 6
'--------------------
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
dat11.Index = "DATA"
dat11.Seek "=", zdata, zcodice

End Select

'caricacolonne
'---------------------------------
frmDatsk1.caricacolonne
End Sub
```

Per primo si memorizzano i campi del Record su cui ci si trova per poterci poi ritornare dopo aver ricostruito il RecordSet. Siccome il ritorno si farà con operazione di “SEEK” , bisogna tener conto della eventualità che il campo corrispondente all’ indice sia vuoto. In tal caso la TextBox é Empty , cioè = Space(0).

Per esempio, ammettiamo di essere sotto indice "ARGOME" e di voler fare una Lettura di un Record in cui capita che il campo "ARGOME sia stato lasciato blank da chi ha inserito il Record.

Dato che siamo sotto "DATORDI" = 4 l'operazione di Seek si farà con l'istruzione:

dat11.Seek "=", zargome, zcodice

Affinché la ricerca con l'indice funzioni correttamente, occorre rispettare la struttura dello stesso. L'indice é stato definito come ARGOME+CODICE e quindi si aspetta che noi cerchiamo una sequenza di 25 caratteri più un numero. (25 é la dimensione del campo ARGOME" nella Tabella)
Nel caso che il campo ARGOME sia pieno non ci sono problemi ma se ARGOME é Empty bisogna imporre per la variabile "ZARGOME" una grandezza di 25 spazi.

Un caso speciale é rappresentato dal campo di tipo Data, inserito qui per scopi didattici.

Per i campi Numerici si utilizza la Funzione "VIRVALORE" presentata sopra e si é sicuri che il contenuto della variabile usata per il Seek é privo di qualunque formattazione.

Prima di uscire, la Routine lancia la "CARICACOLONNE". Questo é necessario perché si é ricostruito il RecordSet e quindi bisogna anche rispecificare le colonne del DBGrid che altrimenti prendono di nuovo i nomi e le dimensioni dei Fields della Tabella.

Si notino infine le varie espressioni di ricerca con il "SEEK" espressioni che corrispondono alla natura degli indici di cui é stata dotata la Tabella "DATFILE" al momento della sua creazione. Gli indici sono visibili, come già detto, col VDM.

2.3 Lettura a tutto schermo

Ora che abbiamo studiato le due importanti Subroutines, "DATTEXT" e "DATFRESCA", siamo in grado di capire meglio il funzionamento del Bottone "CMDREAD".

Ripetiamo il frammento di istruzioni per comodità:

```
'prepara maschera
'----------------
dattext
datfresca
dattext

'mostra maschera
'-----------------
frmDatsk2.cmdritvid.Visible = True
frmDatsk2.Show
frmDatsk1.Enabled = False
```

Come si vede. il bottone "CMDREAD" prevede una operazione di rinfresco dei dati del Record su cui é posizionato il DBGrid perché questo potrebbe essere stato modificato da un altro utente in rete. Per fare questo, usa la Subroutine "DATTEXT" per memorizzare nelle varie caselle del "frmDatsk2", che per il momento è ancora invisibile ed in sottofondo, i differenti valori dei campi del Record, valori necessari per l'operazione di rinfresco (vedasi "DATFRESCA").

Dopo il rinfresco, si procede al ricalcolo delle caselle del "FRMDATSK2," ancora con la routine DATTEXT, questa volta però applicata al Record rinfrescato e si mostra il Form/Maschera col metodo Show. Inoltre si disabilita il "frmDatsk1" di partenza per evitare che l'utente possa leggere un altro Record prima di aver premuto il bottone "CMDRITVID" su "frmDatsk2".
Solo premendo "CMDRITVID" il "frmDatsk2" si chiude ed il "frmDatsk1" si riabilita.

Il DBGrid si trova ora sotto lo stesso indice di partenza prima della lettura ed il Record é lo stesso di quello da cui siamo partiti.

Che succede se il Record che intendiamo leggere é stato nel frattempo eliminato da un altro utente ?
La procedura va in errore perché "DATFRESCA non trova più il Record. Questo errore viene però intercettato dall'istruzione "ON ERROR GOTO ERRORI" e l'etichetta "ERRORI" provvede a segnalarci l'accaduto.

2.4 Simulazione di Rete

Per capire meglio i concetti esposti nel paragrafo precedente, vogliamo ora simulare una situazione di Rete e vedere come si comportano le Routines sopra presentate.

Prendiamo in esame lo stesso Progetto1 dell'Esercizio_07 appena visto ed effettuiamo una piccola ma importante modifica: nella Routine del Form_Load del frmDatsk1 disattiviamo a fini didattici l'istruzione che rendeva inattivo il DBGrid1:

```
'DBGrid1.AllowUpdate = False
```

Ora il DBGrid é attivo nel senso che qualunque modifica che effettuiamo sui suoi dati modifica irreversibilmente la Tabella "DATFILE". Si dovrà quindi procedere con cautela. Se si vuole si può anche fare una copia del "CDRFILE.MDB" da ricuperare in seguito. terminato l'esercizio.

Si provveda allora a compilare, cioè creare un file "EXE" del Progetto1 dell'Esercizio_07 così modificato, con nome "PROGETTO1.EXE" in Cartella C:\GRIGLIE\ESERCIZIO_07.

Lanciamolo con doppio Click e poi lanciamo anche lo stesso Progetto1 da Visual Basic.

Ci troveremo davanti a due DBGrid identici che insistono sulla stessa Tabella "DATFILE" del Database "C:\ARCHIVIODATI\CDRFILE.MDB".

In altre parole i dati della stessa Tabella possono essere visti e gestiti da due DBGrid diversi. É la stessa situazione che si verifica in presenza di una Rete.

Vediamo ora il seguente esercizio:

Con Bottone "Ordinamenti" portiamo prima l'uno, poi l'altro DBGrid sotto indice, diciamo, "ARGOME" . Entrambi mostreranno i dati in ordine di Argomento, prima bianco e poi a partire dal Record a Codice 1509 quelli con Argomento "AZIONE" e così via.

Entrambi i DBGrid possono essere posizionati su questo stesso Record ed entrambi permettono di aprire la scheda di lettura a tutto schermo. A chiusura si ritorna al Record Codice 1509.

Portiamo ora il puntatore del mouse del primo DBGrid sulla casella "AZIONE" del Record 1509 e modifichiamola in "BURATTINI" .Toccando qualunque altro punto del DBGrid il Record 1509 scompare e si posiziona un po' più sotto (dopo il Codice 1109) ed il suo Argomento é ormai "BURATTINI" e non più "AZIONE".

Andiamo a vedere il secondo DBGrid. Anche se i dati sono cambiati, il secondo DBGrid non se ne é accorto e non se ne accorgerà fino a che il RecordSet non sarà stato rinfrescato. Quindi noi continuiamo a leggere Argomento "AZIONE".

Se però ora battiamo il bottone "Lettura", si verifica il rinfresco e nel "frmDatsk2" tutto va a posto e l'Argomento é effettivamente "BURATTINI".

Quanto visto dimostra la necessità di rinfrescare il RecordSet prima di effettuare una lettura a tutto schermo perché i dati potrebbero essere stati cambiati nel frattempo da un altro utente e noi vogliamo certamente leggere l'ultima situazione, cioè la vera ed effettiva.

Lo stesso vale nel caso che, colmo di sventura, l'altro utente abbia cancellato proprio il Record su cui siamo e che intendiamo leggere. Il DBGrid non se ne accorge fino a che non si effettua il rinfresco del RecordSet per cui la routine di lettura deve catturare l'errore.

Capitolo 3. Aggiunta di un nuovo Record

3.1 Il bottone CMDNEW

Iniziamo ora con le operazioni di gestione della Tabella. La prima operazione che vogliamo affrontare è l'aggiunta di nuovi Record.

Vogliamo cioè disporre di un bottone, premendo il quale compaia la solita maschera a tutto schermo FRMDATSK2, questa volta però vuota in modo da poter inserire nel nuovo Record i dati che vogliamo.

La procedura deve inoltre prevedere una serie di controlli:

- Deve verificare che il nuovo codice introdotto non esista già nella Tabella perché non vogliamo accettare schede con lo stesso codice.
- Alcuni campi del nuovo Record devono soddisfare a certe condizioni: per esempio vogliamo che il campo "TITOLO" non possa essere lasciato blank, che quanto inserito nel campo DATA sia veramente una data etc..
- La procedura deve verificare i possibili eventi di Rete.

Procediamo allora col solito metodo del copia/incolla a creare una nuova Cartella C:\GRIGLIE\ESERCIZIO_08 copiandola dall'Esercizio_07 precedente.

Assicuriamoci che il pericoloso:

```
DBGrid1.AllowUpdate = False
```

sia attivo ed aggiungiamo sul "frmDatsk1" un nuovo bottone a cui daremo nome "cmdNew" e Caption "Nuovo Record". Nell'evento Click di questo bottone scriviamo:

```
Private Sub cmdNew_Click()

datblank

frmDatsk2.cmdConfNew.Visible = True
frmDatsk2.cmdAbandon.Visible = True
frmDatsk2.Show
frmDatsk1.Enabled = False

End Sub
```

Come si vede, l'evento Click() lancia dapprima la routine DATBLANK che, come vedremo, prepara un Form "frmDatsk2" con tutti i campi vuoti e poi lo mostra pronto per l'inserimento dei nuovi dati . Al solito si disabilita il "frmDatsk1".

La routine "DATBLANK" routine trova posto, come le altre della stessa serie, nel modulo Modulo1.

```
Sub datblank()

frmDatsk2.txtcodice.Text = Space(0)
frmDatsk2.txttipo.Text = Space(0)
frmDatsk2.txttitolo.Text = Space(0)
frmDatsk2.txtautcogn.Text = Space(0)
frmDatsk2.txtautnome.Text = Space(0)
frmDatsk2.txtattore.Text = Space(0)
frmDatsk2.txtargome.Text = Space(0)
frmDatsk2.txtdata.Text = Space(0)
frmDatsk2.txtvalore.Text = Space(0)
frmDatsk2.txtnota.Text = Space(0)

End Sub
```

Passiamo ora al "frmDatsk2" e anche qui aggiungiamo due nuovi bottone.
Per il primo:

Nome	"cmdConfNew"
Caption	"OK Nuovo Record"
Proprietà Visibile	"False"

Per il secondo:

Nome	"cmdAbandon"
Caption	"Rinuncia"
Proprietà Visibile	"False"

Quest'ultimo bottone serve nel caso che, dopo aver chiesto l'aggiunta del nuovo Record, si cambi idea e si voglia annullare l'operazione di aggiunta.

Eccone la semplice routine:

```
Private Sub cmdAbandon_Click()

frmDatsk1.Enabled = True
Unload frmDatsk2

End Sub
```

Vediamo ora la Subroutine di aggiunta Record.

```
Private Sub cmdconfnew_Click()

'campi non validi
'-----------------------
fladat = True
datcheck
If fladat = False Then
Exit Sub
End If
```

```
'controlla codice duplicato
'---------------------
zcodice = Val(txtcodice.Text)
xdatordi = datordi

If datordi <> 1 Then
frmDatsk1.Data1.Refresh
Set dat11 = frmDatsk1.Data1.RecordSet
datordi = 1
dat11.Index = "CODICE"
End If

dat11.Seek "=", zcodice

If Not dat11.NoMatch Then
MsgBox (zcodice & " già esiste!")
frmDatsk1.DBGrid1.Caption = "Video List per Codice"
Exit Sub
End If

'aggiunge il Record
'---------------------------------
dat11.AddNew
datrepla
dat11.Update

'rinfresca il RecordSet
'---------------------------------
datordi = xdatordi
datfresca

'torna a video list
'---------------------------------
datvuoto = False
frmDatsk1.Enabled = True
Unload frmDatsk2

End Sub
```

La routine controlla per prima cosa che non si siano immessi valori non validi per i campi nelle TextBox. Il flag “FLADAT” (Global in Modulo1) é posto a “TRUE” e si lancia la Subroutine “DATCHECK” che vedremo fra poco. Se tutti i controlli di “DATCHECK” sono soddisfatti si prosegue, altrimenti si esce dalla Subroutine e si ritorna al “frmDatsk2”.

Come seconda cosa si controlla che il codice immesso non sia un duplicato di un altro codice immesso in precedenza magari per disattenzione oppure perché, mentre riempivamo il “frmDatsk2”, qualche altro utente ha immesso anche lui un nuovo Record proprio con il codice che volevamo usare noi.

Per effettuare questo controllo di duplicazione é necessario effettuare un rinfresco ed una ricerca del CODICE desiderato. Tale CODICE Non deve esistere, cioè la ricerca si deve concludere con una situazione di “NOMATCH” del “DAT11”.

La ricerca si effettua con una operazione di Seek del codice immesso. Affinché il Seek sia efficace occorre dapprima selezionare l'indice "CODICE", in caso che non fosse questo l’indice con cui siamo partiti da DBGrid1.

Se si trova effettivamente un altro Record (NOMATCH = False) con lo stesso numero di CODICE, l'operazione di aggiunta si interrompe e si torna sul "frmDatsk2" da cui, volendo, si può uscire col bottone "CMDABANDON". Si ritorna allora alla Video Lista posizionati sul Record in questione.

In caso contrario invece, significa che tutto va bene, per cui si può procedere a:

- Aggiungere il nuovo Record nella Tabella.

- Riempire i Fields della Tabella con i valori immessi nelle caselle testo di "FRMDATSK2. Questa operazione è effettuata dalla routine DATREPLA che vedremo in un paragrafo seguente.

- Rinfrescare il RecordSet perché includa il nuovo Record appena aggiunto e quindi reimpostare l'indice che c'era prima di aggiungere il Record, nel caso che esso fosse stato diverso da CODICE.

- Far scomparire DATSK2 e riabilitare DATSK1.

3.2 La Subroutine DATCHECK

Questa Subroutine, collocata in "frmDatsk2", verifica che alcuni campi della TextBox siamo stati riempiti correttamente:

```
Private Sub datcheck()

'codice assente
'-----------------------------------------------
If Len(Trim(txtcodice.Text)) = 0 Then
Messag = "Codice non può essere assente ! "
Tito = "Attenzione prego !"
Rispo = MsgBox(Messag, 0, Tito)
fladat = False
Exit Sub
End If

'titolo assente
'-----------------------------------------------
If Len(Trim(txttitolo.Text)) = 0 Then
Messag = "Titolo non può essere assente ! "
Tito = "Attenzione prego !"
Rispo = MsgBox(Messag, 0, Tito)
fladat = False
Exit Sub
End If

'data non corretta
'----------------------
If Len(Trim(txtdata.Text)) <> 0 And IsDate(Trim(txtdata.Text)) = False Then
Messag = "Formato data errato ! "
Tito = "Attenzione prego !"
Rispo = MsgBox(Messag, 0, Tito)
fladat = False
Exit Sub
End If
End Sub
```

Vediamo i blocchetti di verifica:

- La TextBox "TXTCODICE" non può essere lasciata blank. Se si permettesse ciò si potrebbero avere varie schede vuote tutte con codice blank e non si saprebbe più come distinguerle una dall'altra. Il CODICE deve sempre essere specificato, infatti esso é chiave di ricerca primaria.

- La TextBox "TXTTITOLO" non deve essere lasciata blank. É questa una scelta imposta arbitrariamente da noi ma che é utile per non trovarsi un giorno con un numero di schede che non si sa a cosa si riferiscano.

- La TextBox "TXTDATA" deve contenere un formato Data valido, due cifre per il giorno, due per il mese e due o quattro per l'anno separate da spazi o da barre.

Se il controllo fallisce in una di queste tre TextBox, la variabile Flag Globale "FLADAT" assume valore False e al ritorno nella Subroutine "CMDCONFNEW_CLICK" riporta sul "frmDatsk2" non valicato. Qui potremo effettuare correzioni o rinunciare all'inserimento.

Questa Subroutine é il posto dove si validano i vari camp. Qui si possono aggiungere altri blocchetti, tutti con la stessa struttura. Per esempio, si provi ad imporre che il campo "VALORE" sia sempre e solo compreso tra 3 e 10. od altri esercizi del genere.

3.3 La Subroutine DATREPLA

Questa Subroutine riempie i Fields del nuovo Record della Tabella "DATFILE" . Essa va posizionata nel Modulo1.n "FRMDATSK2". Si noti il diverso trattamento che ricevono i campi tipo Testo, Numero, Data.

```
Sub datrepla()

dat11!CODICE = virvalore(frmDatsk2.txtcodice.Text)
dat11!TIPO = Left(frmDatsk2.txttipo.Text, 1)
dat11!TITOLO = Left(frmDatsk2.txttitolo.Text, 50)
dat11!AUTCOGN = Left(frmDatsk2.txtautcogn.Text, 15)
dat11!AUTNOME = Left(frmDatsk2.txtautnome.Text, 15)
dat11!ATTORE = Left(frmDatsk2.txtattore.Text, 20)
dat11!ARGOME = Left(frmDatsk2.txtargome.Text, 25)
dat11!VALORE = virvalore(frmDatsk2.txtvalore.Text)
dat11!NOTA = Left(frmDatsk2.txtnota.Text, 50)

'controlla validità data
'---------------------------
If IsDate(frmDatsk2.txtdata.Text) Then
dat11!Data = CDate(frmDatsk2.txtdata.Text)
ElseIf Len(Trim(frmDatsk2.txtdata.Text)) = 0 Then
dat11!Data = Empty
Else
dat11!Data = Empty
MsgBox ("pls check data della scheda")
End If
'---------------------------
End Sub
```

3.4 L'inserimento dati col tasto Return

Il VB permette di spostarsi da una casella all'altra del "frmDatsk2" utilizzando il tasto TAB secondo l'ordine stabilito al momento della progettazione del Form e precisamente secondo la numerazione delle proprietà "TABINDEX" delle caselle. Tuttavia l'uso del tasto TAB é di uno scomodo da piangere.

Per impostare una situazione più accettabile, in cui lo spostamento si ottenga con le due frecce in su ed in giù , nonché col tasto Return per scendere da casella a casella, occorre fare come segue:

1. Assegnare alla proprietà "KEYPREVIEW" del Form "frmDatsk2" il valore di "TRUE" (di default é "FALSE").
2. Aggiungere all'evento "KEYDOWN" del Form la seguente Subroutine:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

Select Case KeyCode

Case vbKeyEscape
cmdAbandon_Click

Case vbKeyReturn, vbKeyDown
SendKeys "{TAB}"

Case vbKeyUp
SendKeys "+{TAB}"

End Select

End Sub
```

Capitolo 4. Modifica di un Record

4.1 Il bottone CMDEDIT

Vogliamo ora studiare la possibilità di modificare/editare un Record con la Maschera a tutto schermo che abbiamo preparato.

Si prepari un ambiente di lavoro C:\GRIGLIE\ESERCIZIO_09 copiando per esempio la Cartella ESERCIZIO_08.

Si Aggiunga sul "frmDatsk1" un nuovo bottone con nome "CMDEDIT" e si scriva la seguente Subroutine:

```
Private Sub cmdEdit_Click()

On Error GoTo errori

'rinfresca
'----------------
dattext
datfresca
dattext
```

```
'controlla rete
'---------------
If dat11!fliprete = 1 Then
MsgBox ("il Record è in uso. attendere e ritentare")
Exit Sub
End If

'blocca rete
'---------------
dat11.Edit
dat11!fliprete = 1
dat11.Update

'mostra maschera
'---------------
frmDatsk2.txtcodice.Enabled = False
frmDatsk2.cmdConfEdit.Visible = True
frmDatsk2.cmdAbandon.Visible = True
frmDatsk2.Show
frmDatsk1.Enabled = False

'routine errori
'------------------
Exit Sub
errori:

Select Case Err
Case 3167, 3021
MsgBox ("sorry. il Record nel frattempo è stato" & acapo & " cancellato da un altro utente !")
dat11.MoveFirst
Case Else
MsgBox ("errore imprevisto No. " & Err)
End Select

End Sub
```

La routine effettua il solito rinfresco del RecordSet senza dimenticare la possibilità che nel frattempo qualcuno abbia cancellato proprio quel Record, in modo analogo a quanto visto per la lettura di un Record.

A questo punto la routine verifica se il Record prescelto per la modifica non sia, proprio lui per caso, già in uso da parte di un altro utente della rete.

Questa eventualità può essere constatata perché ogni volta che un utente apre un Record per la modifica, il campo "FLIPRETE," appositamente previsto per questo scopo, viene impostato dal valore zero al valore 1.

Se ora un altro utente cerca di aprire proprio quel Record, la verifica di "FLIPRETE" indica appunto FLIPRETE=1 quindi il Record non può essere aperto e bisogna attendere che il primo utente abbia finito di lavorarci per potervi accedere. Infatti solo in questo momento il campo "FLIPRETE" viene di nuovo rimesso a zero.

Una volta che si è riusciti ad aprire il Record, si imposta il FLIPRETE=1 e si impedisce così ad ogni altro utente di intervenire fino a che il Record non sia stato richiuso.

Un campo come "FLIPRETE" si chiama di tipo "semaforo" poiché questo nome sintetizza bene il suo funzionamento.
Si noti anche la tecnica con cui il Record viene aperto e modificato, grazie all'utilizzo della coppia di metodi Edit/Update.

Le operazioni di modifica e di controllo del Record in editing si fanno sul Form "frmDatsk2".

Per semplificare, abbiamo imposto che il campo "CODICE" che é la chiave primaria, non possa essere modificato (in caso di assoluta necessità si dovrà cancellare il Record con una procedura che vedremo in seguito ed aggiungere un nuovo Record con il "CODICE" desiderato.)

Pertanto si blocca la casella "TXTCODICE" del "frmDatsk2" e si rendono visibili i tasti che ci interessano: "CMDABANDON" già visto in precedenza ed un nuovo bottone "CMDCONFEDIT" che andremo ad aggiungere sul "FRMDATSK2" con Caption "Conferma Modifica" e proprietà Visibile "FALSE".

Per il bottone "CMDABANDON" occorre introdurre una piccola modifica rispetto al bottone visto per l'aggiunta di un nuovo Record. Qui infatti se si cambia idea e si decide di non modificare il Record invocato con il "CMDEDIT" occorre come minimo riportare il Flag "FLIPRETE" al valore zero per permettere ad altri utenti di accedere ad esso.

L'evento Click viene perciò così modificato:

```
Private Sub cmdAbandon_Click()

If cmdConfEdit.Visible = True Then
dat11.Edit
dat11!fliprete = 0
dat11.Update
End If

frmDatsk1.Enabled = True
Unload frmDatsk2

End Sub
```

4.2 Il bottone CMCONFDEDIT

Per il bottone "CMDCONFEDIT scriviamo invece questa Subroutine:

```
Private Sub cmdConfEdit_Click()

'campi non validi
'----------------------
datcheck
fladat = True
If fladat = False Then
Exit Sub
End If

'modifica Record
'-------------------------
dat11.Edit
datrepla
dat11.Update
```

```
datfresca

'sblocco rete
'----------------------
dat11.Edit
dat11!fliprete = 0
dat11.Update

'ritorna a video lista
'--------------------------
frmDatsk1.Enabled = True
Unload frmDatsk2

End Sub
```

La prima parte della Subroutine é simile a quella vista per l'aggiunta di un nuovo Record. Si verifica cioè con la "DATCHECK" che i campi siano stati riempiti correttamente.

Si opera quindi la modifica del Record con la routine "DATREPLA " che già conosciamo, si rinfresca, si sblocca il Record per altri utenti e si ritorna alla Video List dopo aver chiuso il "frmDatsk2".

Capitolo 5. Cancellazione di un Record

5.1 Il bottone CMDDELE

L'ultima operazione importante per la gestione della Tabella è la cancellazione di un Record.

Vogliamo allora creare un bottone, premendo il quale compaia la solita maschera a tutto schermo "frmDatsk2", che mostri tutti i dati del Record su cui eravamo posizionati in modo da permettere un controllo che questo è proprio il Record che vogliamo cancellare.

A questo punto si deve poter procedere alla cancellazione oppure poter abbandonare l'operazione.

Prepariamo nel solito modo la Cartella ESERCIZIO_10 partendo dalla Cartella ESERCIZIO_09.

Aggiungiamo un nuovo bottone, nome "CMDDELE", Caption "Cancella", Proprietà Visibile = False.

Per l'evento Click di questo bottone scriviamo:

```
On Error GoTo errori

'prepara maschera
'----------------
dattext
datfresca
dattext

'controlla rete
'----------------
If dat11!FLIPRETE = 1 Then
MsgBox ("il Record è in uso. attendere e ritentare")
Exit Sub
```

```
End If

'blocca rete
'--------------------
dat11.Edit
dat11!FLIPRETE = 1
dat11.Update

'mostra maschera
'--------------------
frmDatsk2.cmdConfDele.Visible = True
frmDatsk2.cmdAbandon.Visible = True
frmDatsk2.Show
frmDatsk1.Enabled = False

'routine errori
'------------------
Exit Sub
errori:

Select Case Err
Case 3167, 3021
MsgBox ("sorry. il Record nel frattempo è stato" & acapo & " cancellato da un altro utente !")
dat11.MoveFirst
Case Else
MsgBox ("errore imprevisto No. " & Err)
End Select
```

Come si vede la routine è abbastanza simile a quella vista per la lettura di un Record, con la differenza che qui si controlla che nessun altro utente in quel momento abbia aperto il Record (FLIPRETE=1), oltre al solito controllo standard che qualche altro utente non abbia nel frattempo eliminato il Record.

Anche qui per il bottone "CMDABANDON" del "frmDatsk2" occorre introdurre la stesa modifica vista per l'Editing di un Record esistente. In entrambi i casi, infatti, se si cambia idea e si decide di non modificare o di non cancellare il Record invocato con il "CMDEDIT" o con il "CMDDELE" , occorre riportare il Semaforo "FLIPRETE" al valore zero per permettere ad altri utenti di accedere ad esso.

L'evento Click viene perciò così modificato:

```
Private Sub cmdAbandon_Click()

If cmdConfEdit.Visible = True Or cmdConfDele.Visible = True Then
dat11.Edit
dat11!FLIPRETE = 0
dat11.Update
End If

frmDatsk1.Enabled = True
Unload frmDatsk2

End Sub
```

5.2 Il bottone CMDCONFDELE

Per il bottone "CMDCONFDELE scriviamo invece questa Subroutine:

```
Private Sub cmdconfdele_Click()

On Error GoTo errori

'cancella
'------------------------
dat11.Delete
dat11.MovePrevious
If dat11.BOF() Then
dat11.MoveFirst
End If

'rinfresca
'------------------------
dattext
datfresca

'ritorna a video list
'------------------------
frmDatsk1.Enabled = True
Unload frmDatsk2

'errore di Tabella vuota
'--------------------------------
Exit Sub
errori:
Select Case Err

Case 3021
datvuoto = True
frmDatsk1.cmdNew.Enabled = True
frmDatsk1.cmdEdit.Enabled = False
frmDatsk1.cmdDele.Enabled = False
frmDatsk1.cmdread.Enabled = False
frmDatsk1.cmdOrdinamenti.Enabled = False

frmDatsk1.Enabled = True
Unload frmDatsk2

Case Else
MsgBox ("errore non previsto " & Err)
End Select
'----------------------

End Sub
```

Come si vede, la routine di cancellazione elimina il Record e sposta il puntatore al Record precedente ("MOVEPREVIOUS").

Nel caso che il Record cancellato sia proprio il primo della Tabella, si verifica l'evento BOF() (beginning of file) questo causa Errore. Per evitarlo si va sul primo Record della Tabella col metodo "MOVEFIRST".

Nel caso poi che il Record cancellato sia l'ultimo rimasto e che quindi la Tabella resti vuota, l'istruzione "MOVEPREVIOUS" comporta l'errore intercettato Numero 3021 . In questo caso si procede alla disabilitazione di tutti i bottoni, tranne quello che permette la introduzione di nuovi Record.

Dopo la cancellazione si opera un rinfresco del RecordSet (basato sul nuovo posizionamento di Record) e si torna quindi sul DBGrid in "frmDatsk1" ove si vede la nuova situazione.

Capitolo 6. L'Help di dati

6.1 Il concetto di Help di dati

Gli help di dati sono degli strumenti che servono a facilitare la compilazione delle maschere di dati facendo riferimento a banche ausiliarie.

Per capire meglio il concetto si pensi ad un esempio pratico: Quando si riempiono i dati di una scheda risulta interessante per alcuni campi, per esempio l'attore, il regista, l'argomento, avere disponibile, in linea, la banca dati degli attori già inseriti in precedenza (o registi o argomenti) per poter prelevare il nome dell'attore nel caso che la scheda in esame si riferisca ad un attore già considerato in precedenza per un'altra scheda.

Questa funzione offre due vantaggi:

- La possibilità di prelievo del nome interessato dalla banca dati precedenti, evitando quindi una fastidiosa ripetizione di battitura dati.

- La certezza che il nome scritto sia esattamente uguale a quello introdotto in precedenza, comprese le abbreviazioni, i caratteri minuscoli e maiuscoli e così via, premessa questa indispensabile se si vuole più tardi procedere ad una estrazione di dati significativa. Per continuare l'esempio cui sopra, se non abbiamo scritto sempre in modo identico il nome dello stesso attore, sarà poi impossibile estrarre tutte le schede di quell'attore.

La funzione di Help può essere risolta con l'uso del DBGrid, che si dimostra particolarmente adatto allo scopo.
Faremo l'Esercizio per un Help di Autore/Regista, Tabella AUTFILE presente in CDRFILE.MDB con questa Struttura:

Campo	Nome campo	Tipo	Dim
1	AUTCOGN	Testo	15
2	AUTNOME	Testo	15
3	AUTNOTE	Testo	50
4	FLIPRETE	Numerico	Intero

Indici: 1. AUTCOGN = AUTCOGN/AUTNOME
2. AUTNOME = AUTNOME/AUTCOGN

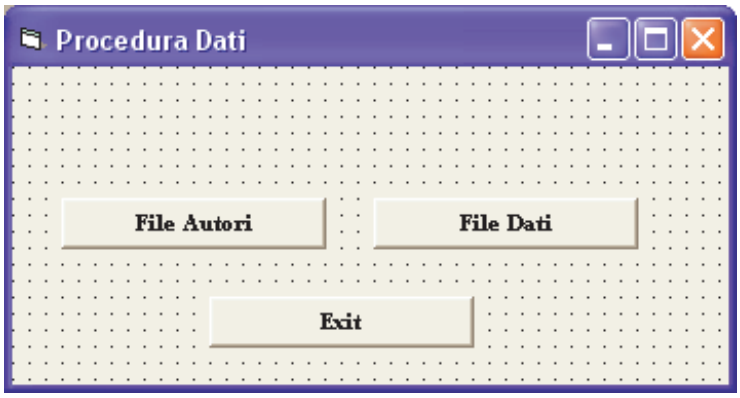
Svilupperemo un Progetto che deve presentare queste caratteristiche:

- Deve poter gestire separatamente le Tabelle "DATFILE" e "AUTFILE", tramite due DBGrid posti rispettivamente sui Form "frmAutsk1" e "frmDatsk1".
- Deve quindi disporre di un Menù di lancio che permetta di scegliere quale Tabella gestire.
- Deve disporre di un Help dei dati Autore/Regista che assista nell'inserimento del relativo campo quando si prepara una scheda dati con "FRNDATSK2".

6.2 Il file FRMMENU

Recuperiamo una buona parte del nostro Progetto dall'ESERCIZIO_10 visto in precedenza e creiamo col solito metodo di Copia/Incolla una Cartella C:\GRIGLIE\ESERCIZIO_11.

Per prima cosa ci serve un menu di lancio per esempio come questo:



Creiamo allora questo nuovo Form con nome "FRMMENU"
Questo Form deve essere quello che parte al lancio della procedura. Ciò si ottiene dal Menù Visual basic "Progetto/Proprietà di Progetto1" impostando l'Oggetto di Avvio su di esso.

Il Form presenta tre bottoni; Ecco i loro eventi Click:

```
Private Sub cmdAutfile_Click()

autordi = 1
frmAutsk1.Show
frmMenu.Enabled = False

End Sub
```

```
Private Sub cmdDatfile_Click()

datordi = 1
frmDatsk1.Show
frmMenu.Enabled = False

End Sub
```

```
Private Sub cmdExit_Click()
End
End Sub
```

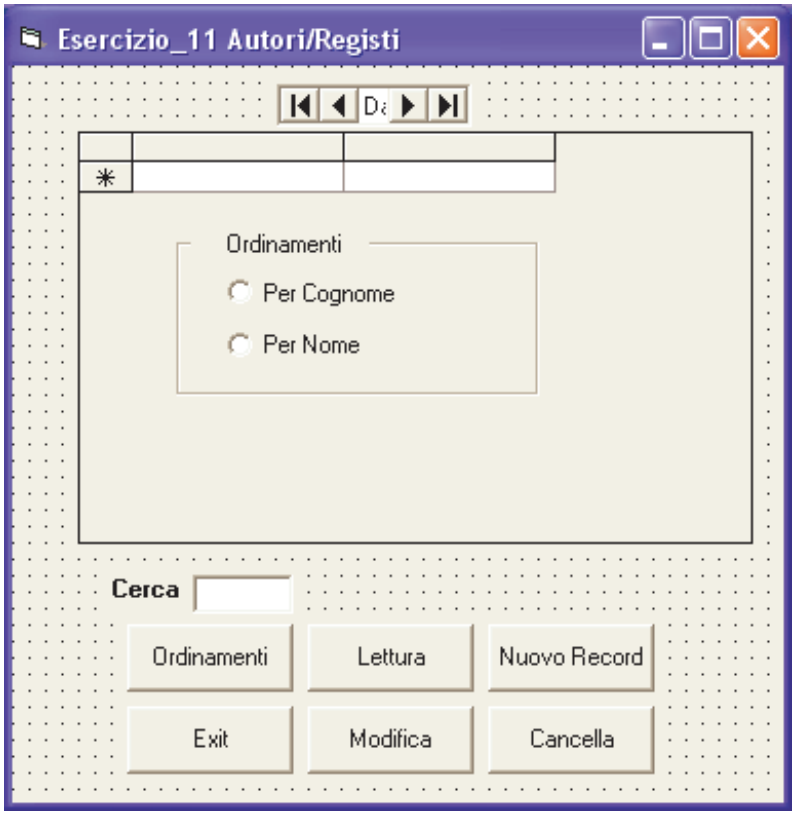
Inoltre, se si vuole, si può mettere la "APRIPROGETTO" nel Form_Load .

6.2 Il sistema AUT

Il sistema "DAT" é praticamente disponibile dall'Esercizio precedente, mentre per il sistema "AUT" dobbiamo procedere con Form analoghi, adattati solo per il diverso nome e numero dei campi della Tabella "AUTFILE".

Se si vuole, si può preparare un "grezzo" di partenza facendo una copia di "frmDatsk1" per creare "frmAutsk1" e poi modificare "DAT" con "AUT" in tutto il Form. Lo stesso per "frmAutsk2". Dopo di che si procede con le personalizzazioni.

Ecco allora il Form "AUTSK1"



Si noti il nuovo bottone "Exit" la cui routine di Click é la seguente:

```
Private Sub cmdExit_Click()

frmMenu.Enabled = True
Unload frmAutsk1

End Sub
```

Naturalmente, analogo bottone va aggiunto anche al "frmDatsk1".

Si noti inoltre la TextBox "CERCA" , a nome "TXTSEEK" che serve a posizionare il puntatore del DBGrid sul primo Record che soddisfa il criterio di ricerca. Per esempio, se scriviamo "R" nella TextBox, il puntatore va sul primo Autore il cui Cognome che comincia con "R" appunto.

Questo comportamento é spiegato dall' evento "CHANGE" della TextBox come segue:

```
Private Sub txtseek_Change()

On Error GoTo fine
marka = aut11.Bookmark

Select Case aut11.Index
Case Is = "AUTCOGN"
aut11.Seek ">=", (Trim(txtseek.Text))
Case Is = "AUTNOME"
aut11.Seek ">=", (Trim(txtseek.Text))
End Select

If aut11.NoMatch Then
MsgBox ("  " + Trim(txtseek.Text) & acapo & " non trovato !")
aut11.Bookmark = marka
End If

fine:
End Sub
```

Non ripetiamo qui le altre Routines del Form perché sono simili o identiche a quelle già viste in precedenza per "frmDatsk1", ove al posto di "DAT" si ha ora "AUT".

In modo analogo si prepara un "frmAutsk2" come questo:

Anche qui le Routines sono simili a quelle viste per "frmDatsk2". Vale solo la pena di commentare la "CMDCONFNEW_CLICK" ;

```
Private Sub cmdconfnew_Click()

'campi non validi
'----------------------
flaaut = True
autcheck
If flaaut = False Then
Exit Sub
End If

'controlla Cognome/Nome duplicato
'----------------------
zautcogn = Left(txtautcogn.Text, 15)
```

```
If txtautnome.Text <> Empty Then
zautnome = Left(txtautnome.Text, 15)
Else
zautnome = Space(15)
End If

xautordi = autordi

If autordi <> 1 Then
frmAutsk1.Data1.Refresh
Set aut11 = frmAutsk1.Data1.RecordSet
autordi = 1
aut11.Index = "AUTCOGN"
End If

aut11.Seek "=", zautcogn, zautnome

If Not aut11.NoMatch Then
MsgBox (zautcogn & "  " & zautnome & acapo & " già esiste!")
frmAutsk1.DBGrid1.Caption = "Video List per Cognome"
Exit Sub
End If

'aggiunge il Record
'---------------------------------
aut11.AddNew
autrepla
aut11.Update

'rinfresca il RecordSet
'---------------------------------
autordi = xautordi
autfresca

'torna a video list
'---------------------------------
autvuoto = False
frmAutsk1.Enabled = True
Unload frmAutsk2

End Sub
```

Come si vede, la ricerca di un eventuale Autore/Regista già esistente, fatta per evitare di inserire un duplicato, si articola in modo un po' diverso da quanto visto per la ricerca di un Codice già esistente.

Qui infatti é necessario tener conto della struttura dell'indice "AUTCOGN" che é composto dai campi "AUTCOGN" ed "AUTNOME". Pertanto il "SEEK" deve tener conto di entrambi i campi e non di uno solo.

```
aut11.Seek "=", zautcogn, zautnome
```

Inoltre anche qui, come già visto per "DATFRESCA", bisogna prevedere la possibilità che il campo "AUTNOM" sia stato lasciato blank. Infatti la Subroutine "AUTCHECK" impedisce che il campo "AUTCOGN", indice primario di questa Tabella "AUTFILE" sia lasciato blank ma lascia passare un "AUTNOME" blank.

Tenendo conto di questa possibilità si impone, se é il caso, per la variabile di ricerca "ZAUTNOME" un valore di "SPACE(15) e allora il Seek può funzionare.

Anche per il sistema "AUT" occorre preparare le tre Routines delModulo1:

- AUTBLANK
- AUTFRESCA
- AUTTEXT

Non riportiamo per brevità queste Routines per cui valgono le considerazioni già fatte per il sistema "DAT".

6.3 Il sistema DAT

Abbiamo già studiato il sistema "DAT" nei paragrafi precedenti: vediamo solo un paio di migliorie possibili. La prima é l'aggiunta di una casella "TXTSEEK" per portare il puntatore del DBGrid sul Record desiderato. Eccone la Routine per l'evento "CHANGE":

```
Private Sub txtseek_Change()

On Error GoTo fine
marka = dat11.Bookmark

Select Case dat11.Index
Case Is = "CODICE", "VALORE"
dat11.Seek ">=", virvalore(Trim(txtseek.Text))
Case Is = "AUTCOGN", "ATTORE", "ARGOME"
dat11.Seek ">=", (Trim(txtseek.Text))
End Select

If dat11.NoMatch Then
MsgBox ("  " + Trim(txtseek.Text) & acapo & " non trovato !")
dat11.Bookmark = marka
End If

fine:
End Sub
```

Come si vede, a seconda dell'indice su cui il RecordSet é impostato, si effettua un Seek diverso per cercare non tanto le lettere esatte scritte nella TextBox (per cui il Seek richiede "=") ma il Record che più si avvicina (Seek con ">=").
La Routine é posta nell'evento "CHANGE" per permettere al puntatore di avvicinarsi gradualmente a mano a mano che si scrive la stringa di ricerca nella TextBox.

IL metodo non funziona per l'indice su base "DATA" per cui bisogna prima finire di scrivere la data che si vuole cercare e poi iniziare il Seek.

La routine di ricerca va allora posta nell'evento "LOSTFOCUS" della TextBox, come segue:

```
Private Sub txtseek_LostFocus()

On Error GoTo fine
marka = dat11.Bookmark

Select Case dat11.Index
Case Is = "DATA"
dat11.Seek ">=", CDate(Trim(txtseek.Text))
End Select

If dat11.NoMatch Then
MsgBox ("  " + Trim(txtseek.Text) & acapo & " non trovato !")
dat11.Bookmark = marka
End If

fine:
End Sub
```

Fino a che si scrive la data non succede niente e la ricerca inizia quando si tocca un altro oggetto diverso dalla TextBox, per esempio il DBGrid.

Per finire, un' ultima modifica per "frmDatsk1" a carattere più che altro estetico:

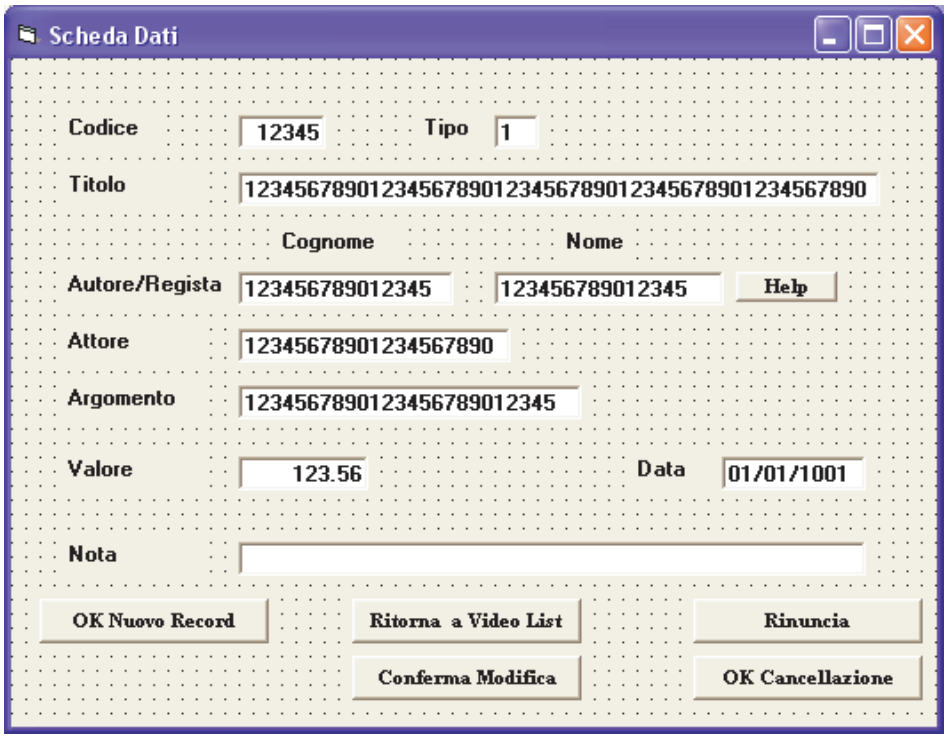
Impostiamo a "FALSE" le seguenti proprietà del Form:

- Control Box
- Max Button
- Min Button

In questo modo l'utente non potrà chiudere o minimizzare il Form con inaspettati e sgraditi effetti. L'unico modo di uscire dal Form resterà il bottone "CMDEXIT".
Analoghe modifiche andranno fatte anche su tutti gli altri Form del Progetto.

I tutti i casi, invece, la proprietà KeyPreview andrà impostata a "TRUE" per rendere il Form sensibile alla tastiera.

Vediamo ora il Form "frmDatsk2". Qui introdurremo un bottone di Help a cui assegneremo nome "CMDHLP1" e Caption "Help".



L'evento Click di questo bottone lancia il Form "FRMAUTHLP1" come segue:

```
Private Sub cmdhlp1_Click()

frhAuthlp1.Show
frmDatsk2.Enabled = False

End Sub
```

6.4 Il Form AUTHLP1

Prepariamo allora un nuovo Form a nome "FRMAUTHLP1" come segue:

Nel Form é presente la solita coppia DBGRid1 e Data1 legate con la proprietà DataSource del DBGrid.

Al lancio del Form corrisponde la solita Routine di carica RecordSet . Ecolab:

```
Private Sub Form_Load()

'RecordSet
'-------------------
Data1.DatabaseName = pathmdb
Data1.Connect = ""
Data1.RecordSource = "AUTFILE"
Data1.Refresh
Set aut21 = Data1.RecordSet
aut21.Index = "AUTCOGN"

'file vuoto
'--------------------
If aut21.EOF Then
autvuoto = True
txtseek.Enabled = False
cmdTake.Enabled = False
End If

'DBgrid1
'----------------------------------------
DBGrid1.AllowUpdate = False
DBGrid1.Caption = "File Autori/Registi per Cognome"
DBGrid1.Font.Bold = True
DBGrid1.HeadFont.Bold = True
DBGrid1.AllowRowSizing = False

caricacolonne

End Sub
```

Dopo il lancio del Form si dispone del RecordSet “AUT21” della Tabella “AUTFILE”.

Vediamo ora l’evento Click del bottone “PRENDI”, a nome “CMDTAKE”

```
Private Sub cmdtake_Click()

frmDatsk2.Enabled = True
frmDatsk2.txtautcogn = aut21!autcogn
frmDatsk2.txtautnome = aut21!autnome

Unload frmAuthlp1
End Sub
```

Appare allora chiaro il funzionamento del Form di Help: quando si lavora su "frmDatsk2" lo si può invocare con il bottone "CMDHLP1", si può cercare il Nome/Cognome che si desidera inserire e, con il bottone "CMDTAKE", imporne i valori alle TextBox del Form. Contemporaneamente il Form di Help scompare.

Per completezza vediamo anche l'evento Click del bottone "RINUNCIA", a nome "CMDABANDON".

```
Private Sub cmdAbandon_click()

frmDatsk2.Enabled = True
Unload frmAuthlp1

End Sub
```

6.5 Inserimento fuori Help

Quanto sopra vale se l'Autore/Regista esiste già nella Tabella "AUTFILE", aperta dal RecordSet "AUT21". In caso contrario dobbiamo aggiungere un nuovo Autore/Regista alla Tabella.

Ciò si può fare naturalmente da "frmAutsk1" ma se stiamo aggiungendo un nuovo Record dati e ci accorgiamo che l'Autore/Regista non é mai stato finora inserito, certamente non abbiamo voglia di uscire dalla operazione di inserimento, cambiare scenario, aggiornare la Tabella da "frmAutsk1" e ritornare ad inserire i dati su cui stavamo lavorando.

Ci occorre uno short cut che andiamo a descrivere. In breve si tratta di aprire un altro RecordSet della Tabella "AUTFILE" ed aggiungere il nuovo il Cognome/Nome dopo aver verificato che già non esiste.

Con l'occasione vedremo anche come é possibile aprire un RecordSet senza appoggiarlo ad un oggetto Data, come abbiamo fatto finora. Il frammento di istruzioni per fare questo é il seguente:

```
'apre RecordSet Table (Seek)
'-----------------------------------
Set AUTDB = OpenDatabase(pathmdb, False, False, "")
Set aut12 = AUTDB.OpenRecordSet("AUTFILE")
aut12.Index = "AUTCOGN"
aut12.Seek "=", zautcogn, zautnome
```

Come si vede, si sono usati i Metodi "OPENDATABASE" ed "OPENRECORDSET" che non richiedono un oggetto Data di appoggio.
É però importante sottolineare che RecordSet aperti in questo modo sono, da un lato, utilissimi per effettuare operazioni di vario tipo all'interno delle Subroutines, ma in nessun caso possono essere visualizzati con un DBGrid. Infatti un DBGrid é solo un occhio aperto su un oggetto Data e vede solo i RecordSet che si appoggiano a tale oggetto Data.

Presentiamo anche, a scopo informativo, il frammento di istruzioni necessarie per aprire un Dynaset:

```
'apre RecordSet Dynaset (FindFirst)
'----------------------------------
AUTQ1 = "select * from AUTFILE"
Set AUTDB = OpenDatabase(pathmdb, False, False, "")
Set aut12 = AUTDB.OpenRecordSet(AUTQ1)
zricerca1 = zautcogn
zricerca2 = zautnome
kriterio = "AUTCOGN = '" & zricerca1 & "'  and AUTNOME = '" & zricerca2 & "' "
aut12.FindFirst kriterio
```

Qui é importante notare che la definizione di "KRITERIO" include Cognome e Nome come definiti dall'utente. C'é però un inconveniente: se l'utente, come é suo buon diritto, ha usato il carattere apostrofo " ' " all'interno di questi campi, la definizione di KRITERIO va in tilt perché questi apostrofi si impicciano con gli altri apostrofi di KRITERIO. Dobbiamo quindi impedire che l'utente usi questo carattere, o meglio, lasciarglieli usare ma poi sostituirglieli con un altro carattere meno "pericoloso".

Questa precauzione non é naturalmente necessaria se si usano RecordSet di tipo Table, perché lì non serve definire "KRITERIO" ma poiché a priori non si sa mai se nel corso d'opera si vorranno usare anche dei Dynaset, é comunque pratica consigliabile operare come sopra indicato.

Al proposito risulta utile la seguente Funzione che potremo installare come al solito, nel Modulo1.

```
Function virapostro(INESAME)

'uso
'elimina gli apostrofi da una stringa
'----------------------------------
'esempio
'Private Sub txtAUTCOGN_LostFocus()
'txtAUTCOGN = virapostro(txtAUTCOGN)
'End Sub
'----------------------------------
virapostro = Trim(INESAME)

If InStr(1, virapostro, "'") <> 0 Then
zinstr = InStr(1, virapostro, "'")
virapostro = Mid$(virapostro, 1, zinstr - 1) & "*" & Mid$(virapostro, zinstr + 1)
End If

End Function
```

Come si vede, la Funzione elimina tutti gli eventuali apostrofi che l'utente ha introdotto e li sostituisce con degli asterischi che risultano visivamente accettabili e non sono "pericolosi" per un eventuale uso di "KRITERIO".

La Funzione va usata per tutte le TextBox potenzialmente interessate, sia per il sistema "AUT" che per il sistema "DAT". E va collocata nell'evento "LOSTFOCUS" della TextBox.

Un esempio per tutti, relativo alla casella “TXTAUTCOGN” del “frmAutsk2”.

```
Private Sub txtautcogn_LostFocus()

txtautcogn = virapostro(txtautcogn)
txtautnome = virapostro(txtautnome)

End Sub
```

Dopo queste necessarie premesse, torniamo al problema da cui siamo partiti, aggiungere un nuovo Autore/Regista nella relativa Tabella.

Ecco allora la Routine “LOSTFOCUS” della TextBox “TXTAUTCOGN” di “frmDatsk2”:

```
Private Sub txtautcogn_LostFocus()

'leva apostrofi
'-------------------
txtautcogn = virapostro(txtautcogn)
txtautnome = virapostro(txtautnome)

If Trim(txtautcogn.Text) <> Empty Then

'cerca in AUTFILE
'--------------------
zautcogn = Left(txtautcogn.Text, 15)

If Trim(txtautnome.Text) <> Empty Then
zautnome = Left(txtautnome.Text, 15)
Else
zautnome = Space(15)
End If

'apre RecordSet Table (Seek)
'-----------------------------------
Set AUTDB = OpenDatabase(pathmdb, False, False, "")
Set aut12 = AUTDB.OpenRecordSet("AUTFILE")
aut12.Index = "AUTCOGN"
aut12.Seek "=", zautcogn, zautnome

'apre RecordSet Dynaset (FindFirst)
'-----------------------------------
‘AUTQ1 = "select * from AUTFILE"
‘Set AUTDB = OpenDatabase(pathmdb, False, False, "")
‘Set aut12 = AUTDB.OpenRecordSet(AUTQ1)
‘zricerca1 = zautcogn
‘zricerca2 = zautnome
‘kriterio = "AUTCOGN = '" & zricerca1 & "'  and AUTNOME = '" & zricerca2 & "' "
‘aut12.FindFirst kriterio

'nuovo Autore
'-----------------
If aut12.NoMatch Then
```

```
'domanda
'-----------------
titolo = "Attenzione !"
Messaggio = "Autore/Regista non esiste. Aggiungo ?"
Stile = vbYesNo + vbQuestion + vbDefaultButton2 ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo, 0, 0)
If Risposta <> vbYes Then
txtautcogn.Text = Space(0)
txtautnome.Text = Space(0)
Exit Sub
End If

'aggiunge Record
'----------------
aut12.AddNew
aut12!autcogn  = Left(txtautcogn, 15)
aut12!autnome = Left(txtautnome, 15)
aut12.Update

End If          'NoMatch
End If          'Empty

End Sub
```

Eccone il funzionamento:

Se (e solo se) la TextBox "AUTCOGN" non é blank, per esempio perché ci abbiamo scritto dentro il cognome dell'Autore/Regista che sappiamo non essere presente in "AUTFILE" (e lo sappiamo perché non l'abbiamo trovato nel Form di Help ove l'abbiamo inutilmente cercato) allora si apre un nuovo RecordSet "AUT12" di tipo Table oppure Dynaset, come vogliamo.

Si effettua poi il "SEEK" oppure il "FINDFIRST" a seconda del tipo di RecordSet ed in caso di NoMatch (Autore/Regista non c'é) si chiede all'utente conferma di poterlo aggiungere alla Tabella. In caso di risposta positiva si opera con "ADDNEW", riempimento campi e "UPDATE". Se la risposta é negativa si sbiancano le caselle "TXTAUTCOGN" e "TXTAUTNOME" e si ritorna sul "frmDatsk2" in attesa di ulteriori ordini da parte dell'utente.

Anche per la TextBox "AUTNOME" é necessario un analogo controllo:

```
Private Sub txtautnome_LostFocus()

'leva apostrofi
'-------------------
txtautcogn = virapostro(txtautcogn)
txtautnome = virapostro(txtautnome)

If Trim(txtautnome.Text) <> Empty Then
'---------------------------------
If Trim(txtautcogn.Text) <> Empty Then
txtautcogn_LostFocus

Else

titolo = "Attenzione !"
Messaggio = "Cogmome Autore/REgista blank. Nome deve essere blank "
Stile = vbOKOnly + vbExclamation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo)
```

```
txtautnome.Text = Space(0)

End If        'autcogn<>Empty
'---------------------------------
End If        'autnome<>Empty

End Sub
```

Come si vede, per questa TextBox, a cui non corrisponde Indice Primario, si verifica che, se essa é stata riempita, deve essere stata riempita anche la TextBox "AUTCOGN" , che invece Indice Primario lo é.

Se questo é il caso, si lancia la stessa Routine di verifica ed eventuale aggiunta alla Tabella "AUTFILE" vista in precedenza per "TXTAUTCOGN_LOSTFOCUS", altrimenti la TextBox viene sbiancata e si torna sul "frmDatsk2".

Capitolo 7. La selezione

7.1 Il concetto di Selezione

Un'altra interessante proprietà del DBGrid é la possibilità di selezionare più Records.
Il metodo manuale consiste nel toccare col mouse il selettore dei vari Records, tenendo premuto il tasto Comtrol ma vedremo presto come operare da Programma.

Ricordiamo il concetto di Bookmark: ogni Record di un RecordSet é caratterizzato da un Bookmark, cioè una posizione. Il Bookmark può essere memorizzato in una variabile, per esempio "MARKA1", quando il puntatore si trova sul primo Record di cui vogliamo memorizzare la posizione.

```
Marka1 = dat11.Bookmark
```

Se ora ci muoviamo nel RecordSet possiamo memorizzare anche la posizione di un secondo Record, per esempio

```
Marka2 = dat11.Bookmark
```

Ora possiamo saltare dal secondo Record al primo imponendo alla proprietà BookMark del RecordSet il valore memorizzato.

```
dat11.Bookmark =  Marka1
```

e cosi via.

Si noti comunque che se il RecordSet viene ricalcolato, per esempio con un rinfresco, Marka1 e Marka2 non hanno più valore.

Il DBGrid possiede una proprietà, chiamata SelBookmarks che permette di memorizzare una serie di Bookmarks di un RecordSet e di mostrare evidenziati i Records inclusi nella collezione. Una sola collezione alla volta é possibile.

Per identificare la collezione si usa una istruzione del tipo:

```
Set SelBk = DBGrid1.SelBookmarks
```

Usiamo il nome (arbitrario) "SelBk" per lavorare con la collezione senza dover scrivere ogni volta il defatigante "DBGRID1.SELBOOKMARKS".

I vari Records evidenziati corrispondono agli elementi della collezione di Bookmarks "SELBK" ed ogni elemento di questa collezione é identificato da un indice a partire da zero e a salire: SelBk(0), SelBk(1), SelBk(2) etc.

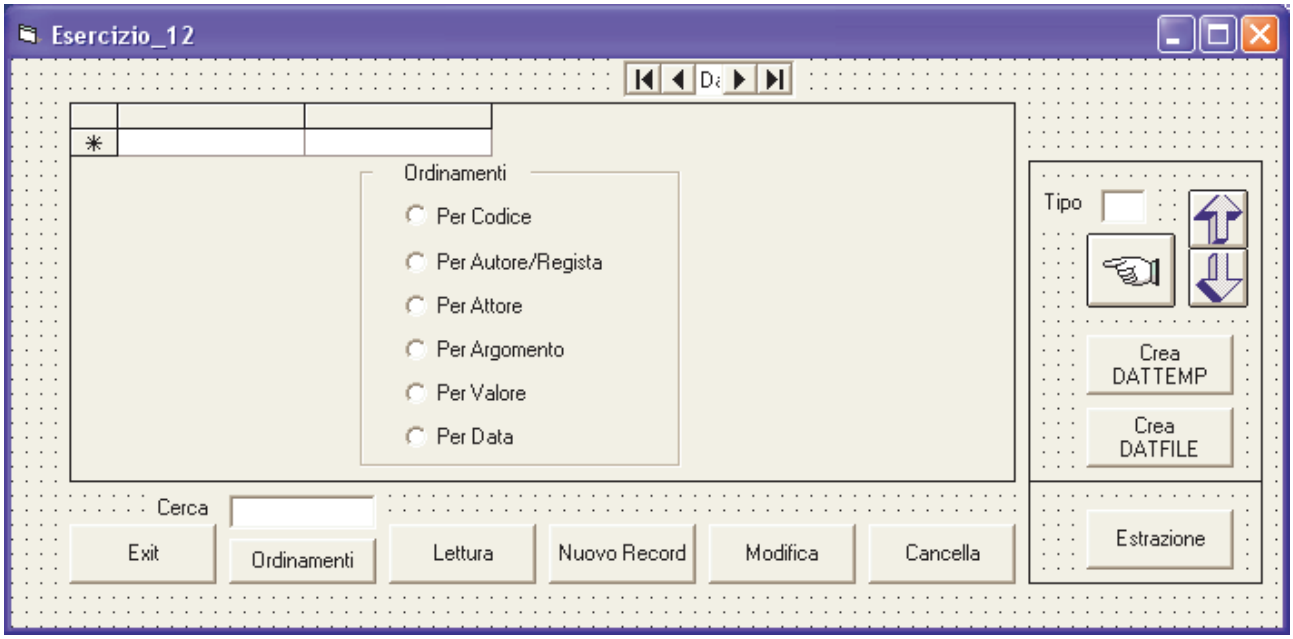
Ad esempio, se su un DBGrid del RecordSet "DAT11" tocchiamo il selettore di tre o quattro Records e poi lanciamo (con un opportuno bottone o altro) l'istruzione:

```
Set SelBk = DBGrid1.SelBookmarks
dat11.Bookmark = SelBk(2)
```

il puntatore si posizionerà sul terzo dei Records Selezionati.
Se però non avremo selezionato un numero sufficiente di Records, la collezione non contiene l'elemento con indice (2) e si avrà un errore (Run time 9- Indice non compreso nell'intervallo).

Vediamo allora di sfruttare questa capacità del DBGrid preparando un nuovo Esercizio, copiando ed incollando l'ESERCIZIO_11 in una nuova Cartella ESERCIZIO_12.

Aggiungiamo un bottone, a nome "cmdEvid" che avrà il compito di mettere in evidenza tutti i Records di un certo "TIPO" del nostro RecordSet "DAT11" e precisamente del Tipo che specificheremo con una lettera in una nuova TextBox a nome "TXTTIPO" con MaxLength = 1, (dato che il Field Tipo ha appunto dimensione 1).



Nella Figura sono presenti anche altri bottoni di cui parleremo tra poco.

Vediamo allora l'evento Click del bottone "cmdEvid":

```
Private Sub cmdEvid_Click()

'elimina evidenze
'----------------------
Set SelBk = DBGrid1.SelBookmarks
For i = 0 To SelBk.Count - 1
SelBk.Remove 0
Next
```

```
'memorizza posizione
'--------------------
marka = dat11.Bookmark
flaevid = False

'cicla
'---------------------
dat11.MoveFirst
Do While Not dat11.EOF()

If UCase(dat11!TIPO) = UCase(txttipo) Then flaevid = True
selbk.Add dat11.Bookmark
End If

dat11.MoveNext
Loop

'imposta puntatore dat11
'----------------------
If flaevid = True Then
dat11.Bookmark = selbk(0)
Puntatore = 0
Else
dat11.Bookmark = marka
titolo = "Attenzione"
Messaggio = "Nessun Record trovato"
a = MsgBox(Messaggio, 0, titolo)
End If

End Sub
```

Il primo blocchetto elimina tutti gli eventuali Bookmarks presenti nella collezione, in modo analogo a quanto visto in "CARICACOLONNE". Anche qui l'oggetto possiede la proprietà Count, numero di Bookmarks della collezione ed il metodo Remove che li elimina.

Si memorizza quindi il Record su cui si é posizionati prima di premere il bottone (per poterci poi ritornare alla fine) e si fa un ciclo del RecordSet "DAT11" dal First Record fino all'ultimo. Per quei Records che soddisfano al criterio specificato in "TXTTIPO", per esempio tutti i Records di TIPO "V", si aggiunge un Bookmark nella collezione, identificando così i Records da evidenziare.

Alla fine del ciclo:

- Se si sono individuati Records da evidenziare (FLAEVID = TRUE) si impone al puntatore di collocarsi sul primo Record evidenziato, dimenticando il Record di partenza ("MARKA")
- Se non si sono individuati Records da evidenziare, si colloca il puntatore su "MARKA" e si lancia un messaggio di avvertimento.

7.2 Navigazione nella Selezione

Proponiamoci ora di percorrere tutti e solo i Records che abbiamo selezionato con l'operazione precedente.

Servono due nuovi bottoni, "CMDSU" e "CMDGIU" e la definizione della variabile (arbitraria) a nome "PUNTATORE" che va definita nella sezione Generale del Form per essere leggibile da tutti gli eventi di esso:

```
Dim Puntatore
```

Vediamo allora l'evento Click del primo bottone

```
Private Sub cmdSu_Click()

Set SelBk = DBGrid1.SelBookmarks
Puntatore = Puntatore - 1
If Puntatore > -1 Then
dat11.Bookmark = SelBk(Puntatore)
Else
Puntatore = Puntatore + 1
End If

End Sub
```

E per il secondo bottone:

```
Private Sub cmdGiu_Click()

Set SelBk = DBGrid1.SelBookmarks
Puntatore = Puntatore + 1
If Puntatore < SelBk.Count Then
dat11.Bookmark = SelBk(Puntatore)
Else
Puntatore = Puntatore - 1
End If

End Sub
```

Come si vede, si definisce il Bookmark del "DAT11" e si muove il puntatore di conseguenza, intercettando i valori di "PUNTATORE" minori di zero e maggiori del "SELBK.COUNT".

7.3 Estrazione di dati Selezionati

Vogliamo ora vedere una applicazione pratica della operazione di selezione. Per esempio vogliamo che tutti e solo i Records selezionati vengano copiati in una nuova Tabella separata, riservandoci poi di operare con questa Tabella per varie operazioni, come stampa, conteggi, etc.

Ci serve quindi una Tabella in cui immagazzinare i Records con queste caratteristiche:

- Deve avere la stessa struttura di "DATFILE, per poter ospitare convenientemente i dati in arrivo. La chiameremo "DATTEMP".

- Deve essere posizionata sul proprio computer e non sul Server di Rete, in luogo quindi separato dal Database principale "CDRFILE" perché deve essere accessibile solo da noi. Infatti ogni utente sarà abilitato a costruirsi la propria Tabella e a riempirsela coi dati che vuole ma va evitata ogni interferenza tra le "DATTEMP" dei vari utenti

Conviene allora per prima cosa creare una nuova Cartella, per esempio "CDR.MIA".

Il miglior posto per questa Cartella é la C:\ ma noi qui, al solo scopo didattico, la collocheremo come SottoCartella della "ARCHIVIODATI" restando inteso che si tratta di una collocazione arbitraria.

All'interno della CRD.MIA creiamo un nuovo Database, per esempio "MIOFILE" facendo uso del citato VDM. Questo Database ci servirà come contenitore di Tabelle specifiche del computer utente e non condivise con altri utenti.

La collocazione di "MIOFILE é quindi la seguente:

```
"C:\GRIGLIE\ARCHIVIODATI\CDR.MIA\MIOFILE.MDB"
```

La prima Tabella che aggiungeremo a "MIOFILE" é proprio la "DATTEMP" che avrà struttura uguale alla "DATFILE di "CDRFILE". Per ottenere praticamente questo risultato é sufficiente usare l'opzione "Importa" del VDM , aggiungere una "DATFILE" al "MIOFILE" e poi cambiarle nome da "DATFILE a "DATTEMP".

Ignoriamo il fatto che "DATTEMP" contenga dei Records (gli stessi di "DATFILE ovviamente), perché a svuotare questa nuova Tabella ci penseranno le istruzioni che andremo a vedere.

Si tratta ora di riempire la "DATTEMP" con tutti e solo i Records che appaiono selezionati sul DBGRid1, indipendentemente che la selezione sia stata fatta manualmente con il tasto Control oppure attraverso il bottone "CMDEVID" visto sopra.

A questo provvede un nuovo bottone "cmdCreatemp" che avremo aggiunto sul Form.

```
Private Sub cmdCreatemp_Click()

'controlla bookmarks
'-------------------------------------
Set SelBk = DBGrid1.SelBookmarks
If SelBk.Count = 0 Then
titolo = "Attenzione !"
Messaggio = "Nessun Record Selezionato"
Stile = vbOKOnly + vbExclamation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo)
Exit Sub
End If

'apre MIOFILE
'-------------------------------------
pathmia = "C:\GRIGLIE\ARCHIVIODATI\CDR.MIA\MIOFILE"
Set DATDB = OpenDatabase(pathmia, False, False, "")

'svuota DATTEMP
'-------------------------------------
DATQ1 = "delete * from DATTEMP"
DATDB.Execute DATQ1

'apre RecorSet
'-------------------------------------
Set dat21 = DATDB.OpenRecordset("DATTEMP")
```

```
'cicla sulla collezione
'----------------------------------
Set SelBk = DBGrid1.SelBookmarks
For i = 0 To SelBk.Count - 1
dat11.Bookmark = SelBk(i)
dattext

'aggiunge in dat21
'----------------------------------
dat21.AddNew
datrepla21
dat21.Update

'ciclo i
'----------------------------------
Next

'chiude database DATDB
'-----------------------------------
'Workspaces(0).Close
DATDB.Close

'ricalcola RecordSet
'-----------------------------------
Data1.DatabaseName = pathmia
Data1.Connect = ""
Data1.RecordsetType = 0         'Table
Data1.RecordSource = "DATTEMP"
Data1.Refresh

'definizione del Recordset
'--------------------
Set dat21 = Data1.Recordset

'Disabilita
'--------------------
cmdNew.Enabled = False
cmdEdit.Enabled = False
cmdDele.Enabled = False
cmdread.Enabled = False
cmdOrdinamenti.Enabled = False

'DBgrid1
'----------------------------------------
DBGrid1.Refresh
DBGrid1.AllowUpdate = False
DBGrid1.Caption = "Video List Estrazioni"
DBGrid1.Font.Bold = True
DBGrid1.HeadFont.Bold = True
DBGrid1.AllowRowSizing = False

caricacolonne
```

```
'fine
'----------------------------------------
titolo = "OK !"
Messaggio = "DATTEMP Creato"
Stile = vbOKOnly + vbInformation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo)

End Sub
```

Si comincia con un gruppo di istruzioni che , grazie al comando SQL “DELETE” svuotano la “DATTEMP”, dopodiché si apre questa “DATTEMP” con un RecordSet a nome (arbitrario) “DAT21” che si affianca quindi al “DAT11” del DBGrid, senza interferire con esso.

Si effettua quindi un ciclo sui Bookmarks componenti la collezione SelBookmarks. Per ogni elemento di tale collezione si va sul Record corrispondente di “DAT11” e se ne copiano i dati su di un Record aggiunto in “DAT21” . Questo RecorSet “DAT21” quindi parte vuoto e poi si riempie gradualmente.

Per effettuare l’operazione di aggiunta nuovo Record a “DAT21” senza scrivere troppe righe di codice, si lancia la “DATTEXT” e si riempie (senza farlo apparire) il Form “FRMDATSK2”, dopodiché si lancia una Routine praticamente identica alla “DATREPLA” vista in precedenza e collocata sul Modulo1. L’unica differenza é che qui si opera su “DAT21” e non su “DAT11”. Con una semplice operazione di sostituzione si può allora scrivere una “DATREPLA21” nello stesso Modulo1.

Terminato il ciclo dei Bookmarks, si chiude il “DAT21” e il suo Database e si passa a ricalcolare il DBGrid utilizzando come RecorSource del “DATA1” la Tabella “DATTEMP”.

A questo punto il DBGrid1 visualizza la Tabella “DATTEMP” che può essere utilizzata con altri eventuali bottoni per fare operazioni specifiche (stampe, calcoli) sul gruppo di Records estratti. I bottoni funzionali Nuovo, Modifica, etc vengono disabilitati.
Per ritornare a vedere la “DATFILE” é necessario un altro bottone , “CREAFILE” che rilancia il Form e riabilita i bottoni funzionali.

```
Private Sub cmdCreafile_Click()
Form_Load
'Abilita
'--------------------
cmdNew.Enabled = True
cmdEdit.Enabled = True
cmdDele.Enabled = True
cmdread.Enabled = True
cmdOrdinamenti.Enabled = True
'fine
'----------------------------------------
titolo = "OK !"
Messaggio = "DATFILE Creato"
Stile = vbOKOnly + vbInformation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo)
End Sub
```

7.4 Estrazione di dati non Selezionati

L'utilizzo della Tabella "DATTEMP" può essere illustrato ancora da un altro esempio:

Supponiamo di voler isolare i Records che si riferiscono ad un "TIPO" come specificato nella casella "TXTTIPO" e che, inoltre, abbiano un "VALORE" maggiore o uguale a, diciamo, 8 Euro.

Possiamo allora aggiungere un nuovo bottone "CMDESTRAI" con questo evento Click;

```
Private Sub cmdEstrai_Click()

'apre MIOFILE
'-------------------------------------
pathmia = "C:\GRIGLIE\ARCHIVIODATI\CDR.MIA\MIOFILE"
Set DATDB = OpenDatabase(pathmia, False, False, "")

'svuota DATTEMP
'-------------------------------------
DATQ1 = "delete * from DATTEMP"
DATDB.Execute DATQ1
DATDB.Close

'riempie DATTEMP
'-------------------------------------
Minvalore = 8
percorso = pathmdb
FiltroStri = "TIPO =  '" & txtTipo.Text & "' "
FiltroNume = "VALORE >= " & Minvalore & ""

DATQ2 = "Insert into DATTEMP select * from DATFILE in '" & percorso & "'  where " & FiltroStri
& " and " & FiltroNume & " "
Set DATDB = OpenDatabase(pathmia, False, False, "")
DATDB.Execute DATQ2
DATDB.Close

'ricalcola RecordSet
'-------------------------------------
… Vedi cmdCreatemp
```

La Routine definisce le variabili "FILTROSTRI" e "FILTRONUME" e poi applica l'istruzione SQL "INSERT" per riempire la "DATTEMP" da "DATFILE" filtrando i Records che interessano.

Anche qui, una volta creata la "DATTEMP" la si visualizza ricalcolando opportunamente il DBGRid1, in modo analogo a quanto visto sopra per "CMDCREATEMP".

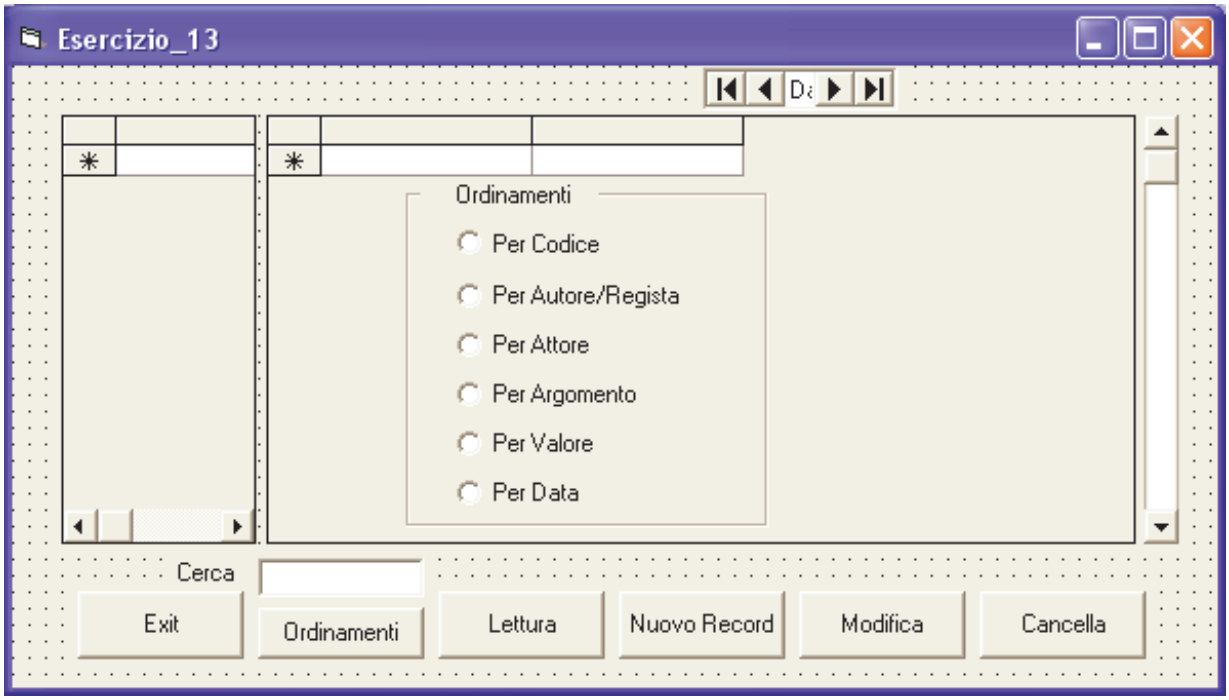
Capitolo 8. Doppio DBGrid

8.1 Metodo VScrollBar

Quando la Tabella visualizzata con il DBGRID possiede molti campi e quindi il DBGRid molte colonne, lo scorrimento orizzontale può dare qualche problema perché si perde la)(le) prima(e) colonna (e).
Spesso però, come nel nostro caso, la prima colonna contiene un dato importante, quale é il Codice del Record, per cui sarebbe più interessare avere sempre in vista come prima colonna il campo Codice scheda e scorrere orizzontalmente gli altri campi del DBGrid, vedendo però sempre a quale Codice essi sono riferiti.

Esistono due metodi per ottenere questo risultato. Cominciamo col primo e creiamo una nuova Cartella ESERCIZIO_13 copiando per esempio i dati dall'ESERCIZIO_11.

Aggiungiamo un secondo DBGRid, anch'esso agganciato a Data1, ed una Barra di Scorrimento Verticale.



Modifichiamo l'evento Form_Load del "FRMDATSK1" aggiungendo prima dell' "End Sub" il seguente ulteriore frammento:

```
'DBgrid2
'----------------------------------------
DBGrid2.AllowUpdate = False
DBGrid2.Caption = Space(1)
DBGrid2.Font.Bold = True
DBGrid2.HeadFont.Bold = True
DBGrid2.AllowRowSizing = False

caricacolonne2
```

```
'arrangiamenti
'----------------------------------------
VScroll1.Max = dat11.RecordCount

DBGrid1.ScrollBars = vbHorizontal
DBGrid1.RecordSelectors = False
DBGrid1.RowHeight = 220

DBGrid2.ScrollBars = vbHorizontal
DBGrid2.RowHeight = 240

End Sub
```

Abbiamo cioè:

- Impostato le proprietà del "DBGRID2" in modo analogo al "DBGRID1".
- Eliminato i selettori dal "DBGRID1" , lasciandoli solo al "DBGRID2".
- Eliminato le barre di scorrimento verticale di entrambi i DBGRid perché intendiamo valerci della sola "VSCROLL1" aggiunta.
- Definito l'altezza delle righe nei due DBGrid in modo che non ci siano scarrellamenti tra i due. I valori da usare si trovano con un paio di tentativi lanciando il Progetto.
- Impostato il valore Max di "VSCROLL1" pari al numero di Records del RecordSet.
- Aggiunto una "CARCACOLONNE2" che definisce la sola colonna "CODICE".

Vediamo ora l'evento Change del "VSCROLL1":

```
Private Sub VScroll1_Change()

'primo Record
'-------------------------------------
If VScroll1.Value = 0 And datvuoto = False Then
dat11.MoveFirst
datposiz = 0
Exit Sub
End If

'ultimo Record
'-------------------------------------
If VScroll1.Value = VScroll1.Max And datvuoto = False Then
dat11.MoveLast
datposiz = VScroll1.Max
Exit Sub
End If

'scroll in giù
'------------------------------
If VScroll1.Value > datposiz Then
Do While Not datposiz = VScroll1.Value
dat11.MoveNext
datposiz = datposiz + 1
Loop
End If
```

```
'scroll in su
'-------------------------------------
If VScroll1.Value < datposiz Then
Do While Not datposiz = VScroll1.Value
dat11.MovePrevious
datposiz = datposiz - 1
Loop
End If
'------------------------

End Sub
```

La variabile “DATPOSIZ” va inoltre definita nella sezione “GENERALE” del Form:

```
Dim datposiz
```

A questo punto se si lancia il Progetto si vedranno i due DBGRid allineati e mostranti lo stesso RecordSet, dato che entrambi sono allacciati a “DATA1” con la loro “DATASOURCE”.

Lo scorrimento della Barra Verticale provoca il movimento del puntatore Record in modo identico sui due DBGrid. Il “DBGRID1” può essere fatto scorrere il modo orizzontale ma il “DBGRID2” assicura che il Codice sia sempre visibile.

Per completare l’Esercizio occorre infine aggiungere le istruzioni di aggiornamento del “MAX” della “VSCROLL1” in caso di aggiunta o cancellazione Record. Ecco i relativi frammenti in “FRMDATSK2” per “cmdConfnew”

```
'aggiunge il record
'---------------------------------
dat11.AddNew
datrepla
dat11.Update
VScroll1.Max = VScroll1.Max + 1
```

E per “cmdConfdele”

```
'cancella
'-------------------------
dat11.Delete
VScroll1.Max = VScroll1.Max - 1
dat11.MovePrevious
If dat11.BOF() Then
dat11.MoveFirst
End If
```

8.2 Metodo Splits

Vediamo ora il secondo metodo: Esso si basa sull' uso di un altro oggetto associato al DBGrid, la collezione Splits. É questa la terza collezione che troviamo. Facciamone un riassunto:

Collezione	Elementi della Collezione
Columns	Columns(0), (1), (2), etc
SelBookmarks	SelBookmarks(0), (1), (2), etc
Splits	Splits(0), (1), (2), etc.

Non si confondano la Collezione e i suoi Elementi.

La Collezione Splits permettere di aggiungere al DBGrid delle divisioni/partizioni che possono essere gestite separatamente.

Vediamo come al solito un esempio pratico: creiamo una nuova Cartella ESERCIZIO_14, copiando ed incollando la Cartella ESERCIZIO_11.

Andiamo nel Form_load di "FRMDATSK2" ed invece di "CARICACOLONNE" scriviamo il seguente frammento:

```
'splits
'----------------------------------------
DBGrid1.Splits.Add 0
DBGrid1.Splits(0).Size = 1000
DBGrid1.Splits(1).Size = 4500
DBGrid1.Splits(1).RecordSelectors = False
CaricaSplits
```

Come si vede, abbiamo aggiunto uno "SPLITS" alla Collezione ed ora disponiamo di due partizioni del DBGRid che dobbiamo gestire separatamente.
Abbiamo gia' definito le dimensioni dei due "SPLITS" ed eliminati i Selettori del secondo.

Vediamo ora la nuova Subroutine "CARICASPLITS":

```
Sub caricaSplits()

'elimina colonne
'--------------------------------
For i = 0 To DBGrid1.Columns.Count - 1
DBGrid1.Columns.Remove 0
Next

'Splits(0)
'--------------------------------
DBGrid1.Splits(0).Columns.Add (0)
DBGrid1.Splits(0).Columns(0).Visible = True
DBGrid1.Splits(0).Columns(0).DataField = "CODICE"
DBGrid1.Splits(0).Columns(0).Caption = "Codice"
DBGrid1.Splits(0).Columns(0).Width = 1000
DBGrid1.Splits(0).Columns(0).DividerStyle = 4
DBGrid1.Splits(0).Columns(0).Alignment = 1
```

```
'Splits(1)
'--------------------------------
For i = 0 To 6
DBGrid1.Splits(1).Columns.Add (i)
DBGrid1.Splits(1).Columns(i).Visible = True
Next

DBGrid1.Splits(1).Columns(0).DataField = "TIPO"
DBGrid1.Splits(1).Columns(0).Caption = "Tipo"
DBGrid1.Splits(1).Columns(0).Width = 500

DBGrid1.Splits(1).Columns(1).DataField = "TITOLO"
DBGrid1.Splits(1).Columns(1).Caption = "Titolo"
DBGrid1.Splits(1).Columns(1).Width = 3000

DBGrid1.Splits(1).Columns(2).DataField = "AUTCOGN"
DBGrid1.Splits(1).Columns(2).Caption = "Autore"
DBGrid1.Splits(1).Columns(2).Width = 2000

DBGrid1.Splits(1).Columns(3).DataField = "ATTORE"
DBGrid1.Splits(1).Columns(3).Caption = "Attore"
DBGrid1.Splits(1).Columns(3).Width = 2500

DBGrid1.Splits(1).Columns(4).DataField = "ARGOME"
DBGrid1.Splits(1).Columns(4).Caption = "Argomento"
DBGrid1.Splits(1).Columns(4).Width = 2000

DBGrid1.Splits(1).Columns(5).DataField = "VALORE"
DBGrid1.Splits(1).Columns(5).Caption = "Valore"
DBGrid1.Splits(1).Columns(5).Width = 700
DBGrid1.Splits(1).Columns(5).NumberFormat = "### ###.00;;#"

DBGrid1.Splits(1).Columns(6).DataField = "DATA"
DBGrid1.Splits(1).Columns(6).Caption = "Data"
DBGrid1.Splits(1).Columns(6).Width = 1200

'allineamento
'--------------------
For i = 0 To 6
DBGrid1.Splits(1).Columns(i).DividerStyle = 4
DBGrid1.Splits(1).Columns(i).Alignment = 0
Next

'allineamento numeri
'--------------------
DBGrid1.Splits(1).Columns(5).Alignment = 1

End Sub
```

Si comincia con l'eliminare tutte le Colonne del DBGRid per partire da una situazione pulita, poi si comincia a lavorare sulla prima Partizione, la "SPLITS(0)". Per questo Splits si inserisce una sola Colonna, relativa al campo "CODICE".

Per la seconda Partizione invece, la "SPLITS(1), aggiungiamo sette colonne (indici da zero a sei) e per ciascuna Colonna definiamo il Field del RecordSet che intendiamo visualizzare, l'ordine, la dimensione etc. Si noti in particolare che qui si é omesso il Field "CODICE" ma volendo lo si sarebbe potuto aggiungere, per esempio in ultima posizione, perché no ?

La Subroutine "CARICASPLITS" sostituisce quindi la Subroutine "CARCACOLONNE" che avevamo studiato in tutti gli Esercizi precedenti. Per completare l'Esercizio occorre quindi sostituire nel Progetto tutte le occorrenze di una Subroutine con l'altra utilizzando il "CTRL-H".

Capitolo 9. Lavorare in Rete

91 Struttura di Rete

Vogliamo fare qui alcune considerazioni di carattere generale che possono risultare utili a chi intende preparare un sistema di Rete con vari utenti che lavorano da macchine diverse ma che accedono tutti ad un'unica banca dati.

La Rete si realizza con il necessario Hardware, del quale non vogliamo qui occuparci ed il risultato concreto dell'esistenza di una rete é che ogni computer collegato vede, oltre che ai propri dischi specifici (certamente il C: ed eventualmente D. , E: etc.) anche un alto disco (Server di Rete) , non presente fisicamente sulla propria macchina e contraddistinto con una lettera prefissata durante la configurazione del sistema di Rete.

Per alcune configurazioni il Server di Rete é visto da tutte le macchine con la stessa lettera, ad esempio , che so, "W". In tal caso i programmi presentati finora sono validi senza modifica con la sola avvertenza che la variabile "PATHMDB" venga definita come :

```
pathmdb = W:\CARTELLA_DEI_DATI\CDRFILE
```

In altri casi, una macchina vede per esempio il disco remoto con la lettera "H:", un'altra con la lettera "J:" etc. pur restando comunque il fatto che i dati sono collocati una volta sola sul disco remoto, indipendentemente dalla lettera sotto la quale li si vede.

Esiste allora il problema di indirizzare la procedura al momento del lancio nella giusta Cartella: Disco, Percorso e Database. Il problema si risolve modificando la routine "APRIPROGETTO" come segue:

```
Sub apriprogetto()

'generali
'--------------------
acapo = Chr$(13) + Chr$(10)
'pathmdb = "C:\GRIGLIE\ARCHIVIODATI\CDRFILE.MDB"
pathmdb = App.Path & "\CDRFILE"
End Sub
```

Il miglior posto per lanciare la "APRIPROGETTO" é l'evento "FORM_LOAD" del Form Menù.

La variabile App.Path identifica sempre la Cartella di partenza della Procedura, quindi il trucco funziona sempre e soltanto se avremo messo i Dati e l'EXE nella stessa Cartella.

Per verificare questi concetti si proceda allora alla compilazione del Progetto1 di uno qualunque degli Esercizi precedenti dopo aver apportato la modifica sopra indicata e scegliendo imperativamente come Cartella di creazione del file "EXE" la:

"C:\GRIGLIE\ARCHIVIODATI"

Lanciando ora l'EXE da questa Cartella, l'istruzione App.Path permette di cercare i dati in questa stesa Cartella e tutto funzionerà

Il metodo funziona per qualunque nome di Cartella si sia scelto sul Server, a patto che in tale Cartella siano presenti sia i Dati che l'EXE di lancio della Procedura.

92 EXE e Dati in Cartelle diverse

Quanto sopra va bene, come detto, se l'EXE ed i Dati sono nella stessa cartella.

Ci sono però situazioni di Rete in cui questo, per vari motivi, non si può fare.

In questo caso, se ammettiamo per esempio che il File dati "CDRFILE" si trovi nella Cartella "ARCHIVIODATI" o qualunque altro percorso ("PIPPO\ARCHIVIO\ETC") del disco remoto, succede che, per il primo computer, la Cartella dati é la "H:\ARCHIVIODATI\CDRFILE" mentre per il secondo computer é la "J:\ARCHIVIODATI\CDRFILE". In altre parole la variabile "PATHMDB" , con cui abbiamo lavorato finora, va definita macchina per macchina.

Ciò significa che ciascun computer deve conservare memorizzati sul proprio disco fisico i dati che compongono il percorso e li deve richiamare al momento del lancio.

Il modo migliore di memorizzare dei dati é quello di usare un Database specifico da collocare su ogni macchina. L'abbiamo già incontrato: é il "MIOFILE" , di cui abbiamo parlato sopra e che, per il momento, possiede solo la Tabella "DATTEMP". É il posto giusto per aggiungere una seconda Tabella, sempre facendo uso del VDM, con nome ad esempio "RETFILE" , con queste caratteristiche:

Campo	Nome campo	Tipo	Dim
1	DISCO	Testo	1
2	PERCORSO	Testo	50
3	DATABASE	Testo	50

Dotiamo questa Tabella di un solo Record, nel quale scriveremo gli elementi per indirizzare il Path di Rete.

DISCO= "C" perché per noi il disco "Server" é proprio il "C"
PERCORSO= "GRIGLIE\ARCHIVIODATI" perché é in questa Cartella che c'é il Database.
DATABASE= "CDRFILE" nome del Database su cui lavoriamo.

Prepariamo allora una nuova Cartella la ESERCIZIO_15 copiando ed incollando per esempio la Cartella ESERCIZIO_11 e modifichiamo la Subroutine "APRIPROGETTO" presente sul Modulo1 come segue:

```
Sub apriprogetto()

On Error GoTo errori:

'acapo
'-------------------
acapo = Chr$(13) + Chr$(10)

'rete
'-------------------------------------
percorsoMiofile = "C:\GRIGLIE\ARCHIVIODATI\CDR.MIA\MIOFILE"
'-------------------------------------
Set RETDB = OpenDatabase(percorsoMiofile, False, False, "")
Set ret11 = RETDB.OpenRecordset("RETFILE")
'-------------------------------------
pathmdb = ret11!disco & ":\"
'-------------------------------------
If Trim(ret11!percorso) <> Empty Then
pathmdb = pathmdb & Trim(ret11!percorso) & "\"
End If
'-------------------------------------
If Trim(ret11!Database) <> Empty Then
pathmdb = pathmdb & Trim(ret11!Database)
End If
'-------------------------------------
Set CDRDB = OpenDatabase(pathmdb, False, False, "")

'errori
'-------------------
Exit Sub

errori:
MsgBox ("Errore di setting dei dati !")
If ret11!disco <> Empty Then
frmRete.txtDisco.Text = ret11!disco
Else
frmRete.txtDisco.Text = Empty
End If

If ret11!percorso <> Empty Then
frmRete.txtPercorso.Text = ret11!percorso
Else
frmRete.txtPercorso.Text = Empty
End If

If ret11!Database <> Empty Then
frmRete.txtDatabase.Text = ret11!Database
Else
frmRete.txtDatabase.Text = Empty
End If

frmRete.Show
Unload frmMenu

End Sub
```

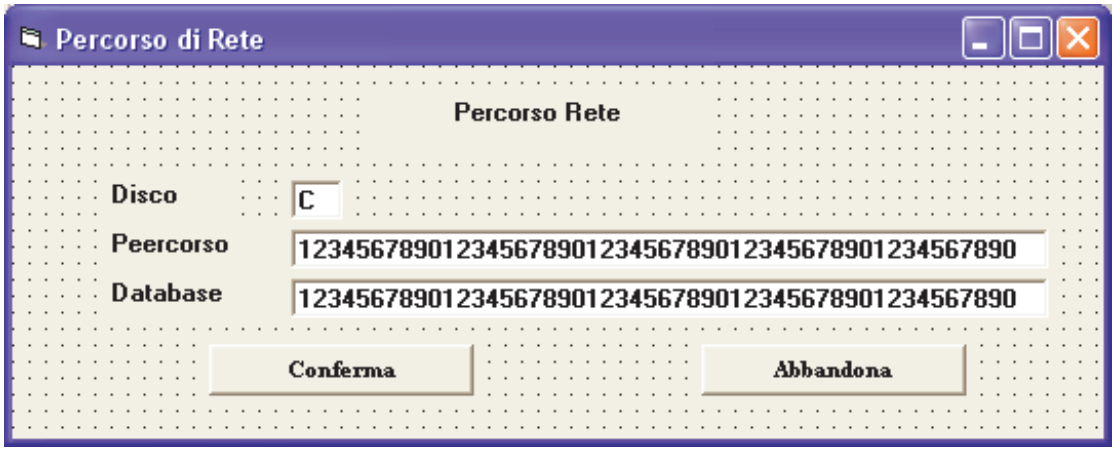
La Routine apre un RecordSet "RET11" dalla Tabella "RETFILE" del Database "MIOFILE" posizionato nella Cartella di lancio standardizzata "C:\GRIGLIE\ARCHIVIODATI\CDR.MIA" e costruisce la variabile "PATHMDB" con i dati lì immagazzinati.

In caso di errato "PATHMDB", l'istruzione:

Set CDRDB = OpenDatabase(pathmdb, False, False, "")

manda il Progetto in Errore.

L'Errore si può verificare perché il sistema Rete non é attivo oppure perché il Database ha collocazione diversa da quella che abbiamo lasciato l'ultima volta etc, . Questo Errore viene intercettato aprendo un apposito Form "FRMRETE" che avremo aggiunto al nostro Progetto e che si presenta come segue:



Nel Form sono presenti tre TextBox a nome "TXTDISCO", "TXTPERCORSO" e "TXTDATABASE" che vengono riempite nella Routine di errore o lasciate blank a seconda del contenuto del campo dell'unico Record della Tabella "RETFILE"

Per esempio ammettiamo che il campo "DISCO" sia stato lasciato vuoto; apparirà la seguente situazione:

Le TextBox di questo Form vanno riempite con dati tali che il Disco di Rete ed il Percorso di Rete siano reali. Nel nostro caso didattico il Disco é "C" ma per un sistema vero dovremo impostare "H" oppure "J" a seconda di come la nostra macchina vede il Server di Rete.

L'evento Click del bottone "CONFERMA" verifica questi dati e se essi sono buoni, cioè se il "PATHMDB" che ne risulta corrisponde ad una vera situazione di Rete, li memorizza nel "RETFILE" e la procedura non li chiederà più ai prossimi rilanci, pur controllandoli sempre.

```
Private Sub cmdConferma_Click()

'controlla directory
'-------------------------
On Error Resume Next
zdire = txtDisco & ":\" & Trim(txtPercorso)
If Dir(zdire, 16) = "" Then
titolo = "Attenzione"
Messaggio = "      Directory " & acapo & zdire & acapo & "      Non Esiste "
Stile = vbOKOnly + vbExclamation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo, 0, 0)
Exit Sub
End If

'ricalcola pathmdb
'-------------------------
On Error GoTo errori
pathmdb = txtDisco & ":\"
If Trim(txtPercorso) <> Empty Then
pathmdb = pathmdb & Trim(txtPercorso) & "\"
End If
If Trim(txtDatabase) <> Empty Then
pathmdb = pathmdb & Trim(txtDatabase)
End If

'controlla database
'-----------------------------------
Set CDRDB = OpenDatabase(pathmdb, False, False, "")

'registra i dati di rete
'-------------------------
ret11.Edit
ret11!disco = Left(frmRete.txtDisco.Text, 1)
ret11!percorso = Left(frmRete.txtPercorso.Text, 50)
ret11!Database = Left(frmRete.txtDatabase.Text, 50)
ret11.Update

'lancia menu
'-----------------------------------
frmMenu.Show
Unload frmRete

'errori
'-------------------
Exit Sub
errori:
titolo = "Attenzione !"
Messaggio = "Errore nel Nome del Database !"
Stile = vbOKOnly + vbExclamation ' Definisce pulsanti.
Risposta = MsgBox(Messaggio, Stile, titolo, 0, 0)
frmRete.Show

End Sub
```

Commiato

Termina qui il percorso che l'Autore ha avuto il piacere di fare in compagnia dei volonterosi che l' hanno voluto accompagnare.

L'Autore ha cercato di travasare nel migliore dei modi la sua modesta esperienza e si augura che il lavoro svolto possa essere di ausilio e di stimolo per i programmatori in questo validissimo ambiente che è il Visual Basic ed in particolare il DBGrid.

rg+