DAO (+SQL), con VB e VBA

Primi esperimenti

Una delle novità più interessanti, soprattutto dal punto di vista tecnico, presente nell'accoppiata Windows 95 & Office 95, è costituita sicuramente dalla diffusione della tecnologia DAO a tutti i livelli, al punto che in pratica DAO è diventato un "servizio" del sistema operativo Windows 95

di Francesco Petroni

Per DAO si intende una modalità standardizzata di accesso ai dati. In concreto si tratta di una serie di istruzioni di programmazione utilizzabili in qualsiasi programma scritto con qualsiasi tipo di VB, il Visual Basic normale e il Visual Basic for Application, utilizzato nei prodotti Access, Excel e Project della Microsoft. Manca ancora all'appello il solo Word.

L'altra sua caratteristica principale, stiamo parlando di DAO, è che si tratta di una tecnologia Object Based (DAO = Data Access Object). Questo significa che ogni elemento del database, il database stesso, le varie tabelle, i vari campi di ciascuna tabella, ecc. sono "oggetti", sono quindi caratterizzati da una serie di proprietà, che si possono leggere o settare, e da una serie di metodi che agiscono su di essi.

DAO nasce in Visual Basic 3.0 e in Access 1.x. Già all'epoca era possibile scrivere "pezzi di codice" ambivalenti, che potevano "girare" in ambedue gli ambienti.

La tecnologia DAO, ora con VB4 per Win95 e con Access 95, viene confermata e ne viene allargato l'uso anche ad Excel 95. In questo articolo approfondiremo soprattutto questo aspetto, ovvero lo sfruttamento della tecnologia DAO direttamente dalle Macro di Excel.

Soprattutto Excel 95

A mio personalissimo parere la più grossa novità presente in Excel per Windows 95, rispetto alla precedente versione 5.0, è costituita proprio dall'introduzione della tecnologia DAO, con la quale viene, per certi versi, semplificato e reso più diretto l'accesso ai dati esterni.

Excel è diffusissimo in tante Aziende. In tali Aziende l'ostacolo più frequente per la sua utilizzazione ottimale è costituito dalla necessità di sfruttare dati esterni, dati quasi sempre già presenti in Azienda, e disponibili nelle più svariate forme. Una volta portati in Excel (il problema è proprio il come portarli) questi dati possono subire qualsiasi tipo di manipolazione, di tipo statistico, di tipo grafico, ecc. tutte operazioni per le quali Excel è adattissimo.

Prima di parlare di DAO, che in pratica costituisce una nuova modalità di estrazione dei dati, dobbiamo ripercorrere la breve storia dell'evoluzione delle funzionalità di accesso ai dati esterni nei fogli elettronici. In questa storia possiamo individuare tre momenti... "storici".

All'inizio c'erano solo funzionalità, più o meno esterne, di conversione di formato (ad esempio da DBF a WKx) e funzionalità, solo interne e quindi operative solo su dati già importati, di analisi di file testuali di tipo "fixed lenght", la cui finalità è quella di ripartire correttamente i dati, una volta portati sul foglio, nelle righe e colonne di destinazione.

Poi sono nati gli "estrattori" intelligenti, parliamo del DataLens del Lotus 3.x e del Q+E del Microsoft Excel 2.x. Intelligenti nel senso che è possibile impostare delle regole in fase di estrazione in modo che i dati letti vengano filtrati "all'origine", ad esempio selezionando i record desiderati, oppure i campi desiderati.

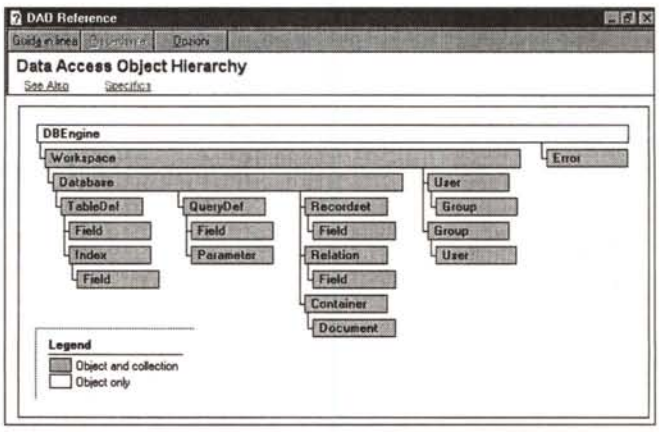


Figura 1 - DAO, primi esperimenti - Ecco il DAO.HLP - La gerarchia degli Oggetti.
La tecnologia DAO, Data Access Object, nata con la versione 3.0 di Visual Basic e con la versione 1.1 di MS Access, consiste in una nuova modalità di accesso a Dati, basata su una filosofia Object Based. Il Database, le sue Tabelle, i Campi delle Tabelle, gli Indici delle Tabelle, ecc. sono definibili e manipolabili come oggetti. Questa tecnologia viene ora, con Windows 95, in un certo senso, promossa in serie A. Diventa parte del sistema operativo ed è sfruttabile anche da altri applicativi. I file relativi a DAO, quindi le DLL, gli Help, ecc. risiedono in una directory comune.

La Microsoft, ai tempi di Excel 5.0, ha proposto un prodotto di estrazione di tipo "stand-alone", che si chiama MS Query, molto potente, ma anche molto impegnativo per la macchina, in quanto viene richiamato come OLE Server. MS Query utilizza un ambiente operativo di tipo QbE e genera un comando SQL. MS Query può essere anche richiamato attraverso una macro di Excel.

Oggi invece, con la tecnologia DAO, è possibile utilizzare, in programmi scritti con i vari prodotti, istruzioni che eseguono un accesso diretto. Rispetto a MS Query viene in pratica saltato un passaggio, per cui il collegamento ai dati esterni è sensibilmente più veloce. Se poi a questa velocità si aggiunge l'aumento generalizzato delle prestazioni dovuto ai motori 32 bit di Windows 95 e Excel 95, si vede come in definitiva si aprano tante nuove possibilità nell'utilizzo di dati prelevati anche da grosse banche dati.

Gli oggetti di DAO

Vediamo, nelle prime tre figure a corredo dell'articolo, alcune videate prelevate dalla documentazione in linea di DAO. Non esiste, nei manuali standard di Office 95, traccia della tecnologia DAO, quindi ci si deve appoggiare sugli Help, installati assieme alle DLL dalla procedura di Setup di Office 95.

Nella prima figura vediamo l'organizzazione gerarchica dell'oggettistica DAO. Ad esempio, partendo per semplicità dall'oggetto Database, si vede come questo contenga una collezione di TableDef (ovvero di definizione di tabelle) e come una singola TableDef contenga una collezione di Field (ovvero di campi della tabella) e una collezione di Index, ognuno dei quali contiene campi di indicizzazione, e così via.

Nella seconda figura vediamo una tipica videata dell'Help di DAO, dal quale si possono aprire finestre con esempi, immediatamente copiabili, di programmazione DAO.

Office 95 in italiano scarica gli Help di DAO in italiano. Per i nostri esperimenti abbiamo però preferito utilizzare la versione originale, che è risultata più completa.

Nella terza figura invece vediamo DAO dal punto di vista di Excel 95. In pratica quando si scrive un programma, e si è quindi su un foglio di tipo Modulo di Excel, è possibile attivare il Browser degli oggetti (che si chiama Visualizzatore degli Oggetti) che propone una ricchissima serie di oggetti DAO. Anche in questo caso le operazioni sono facilitate.

In definitiva la documentazione è sufficientemente completa, anche se, dato il taglio profondamente tecnico della tecnologia DAO, sarebbe molto comoda anche un'esauriente manualistica cartacea.

Cosa faremo

Nella figura 4 vediamo, in un fotomontaggio basato sulla Finestra delle Relazioni di Access 95, i nostri tre Casi Studio. Abbiamo un primo caso di tipo monotabella, la tabella si chiama Persone, sulla quale eseguiremo gli esperimenti più semplici. Abbiamo poi un caso con tre tabelle, relazionate in cascata tra di loro, relative a Regioni, Province e Comuni, e che utilizzeremo per fare delle ricerche a cascata. L'ultimo caso stu-

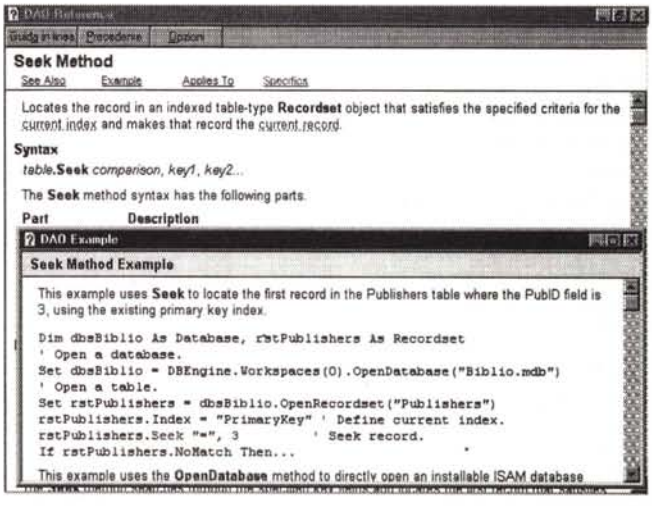


Figura 2 - DAO, primi esperimenti - Ecco il DAO.HLP - Il problema della documentazione. Il materiale per DAO, e quindi i file DLL, i file HLP, ecc., risiedono in una specifica cartella generata dalla procedura di installazione di Office 95. L'Help il linea è molto importante in quanto, attualmente, è l'unica forma di documentazione disponibile. Abbiamo preferito utilizzare quello della versione in inglese, in quanto quello della versione in italiano presenta alcuni disallineamenti. Disseminati nelle pagine dell'Help ci sono esempi importantissimi per capire le varie istruzioni.

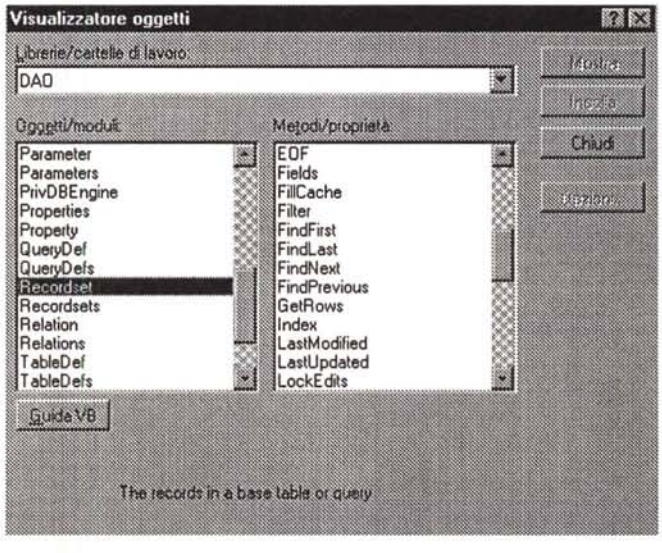


Figura 3 - DAO, primi esperimenti - Excel 95 VBA - Il browser degli oggetti. Il visualizzatore (browser) degli oggetti di Excel serve ad evitare di dover scrivere completamente a mano le varie istruzioni. C'è una specifica sezione DAO, che elenca tutti gli oggetti e per ciascuno di essi tutte le proprietà e i metodi che agiscono su di essi. Il browser è anche una comoda scorciatoia per la consultazione dell'help, che, come detto, è attualmente l'unica fonte di documentazione.

dio coinvolge quattro tabelle: Clienti, Ordini, Righe e Articoli, ed ha quindi una complessità superiore alle altre, ma sempre medio-bassa.

Eseguiremo una serie di esercizi usando, come prodotto di partenza, Excel 95 e facendo delle "puntatine" in Visual Basic 4.0 e in Access 95. Gli esercizi saranno di complessità via via crescente.

C'è da dire che l'ambiente di programmazione di Excel 95 è lo stesso di quello di Excel 5.0 con due "varianti". La prima è, ovviamente, la possibilità di inserire istruzioni DAO, che, se utilizzate, rendono la procedura non compatibile con Excel 5.0. La seconda è costituita dal fatto che le istruzioni VBA sono ora in inglese. In pratica:

- se scriviamo, in una procedura Excel 95, le istruzioni in italiano, queste vengono immediatamente tradotte in inglese,
- se utilizziamo in Excel 95 una procedura scritta, in italiano, con Excel 5 la stessa, miracolosamente, funziona,
- se utilizziamo in Excel 5 una procedura scritta, in inglese, con Excel 95 la stessa, miracolosamente, funziona.

Insomma, in tutti i casi è garantita la compatibilità.

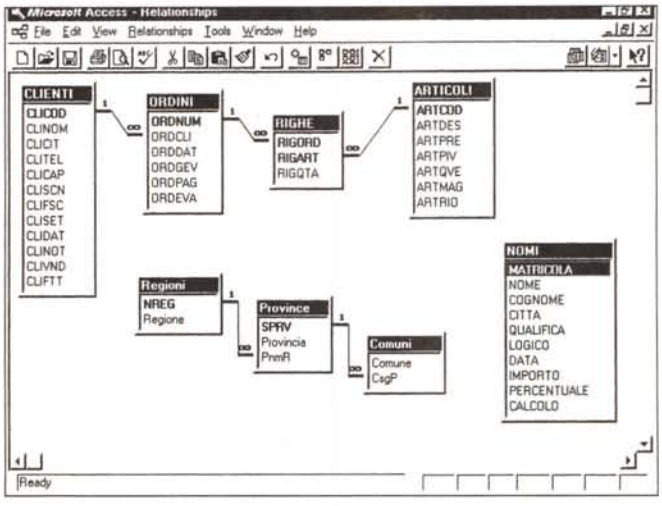
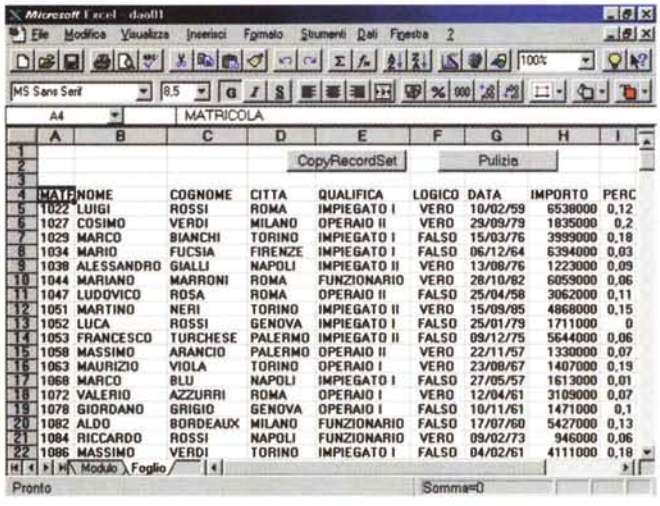


Figura 4 - DAO, primi esperimenti - La struttura delle Tabelle nei tre casi studio.
Per poter capire i vari esperimenti che condurremo occorre innanzitutto conoscere la struttura delle Tabelle utilizzate nei nostri tre Casi Studio. Il primo caso è di tipo "monotabella", e la tabella si chiama Persone. Il secondo riguarda Regioni, Province e Comuni d'Italia e quindi tre tabelle. Nel terzo caso studio le tabelle sono quattro: Clienti, Ordini, Righe ed Articoli.



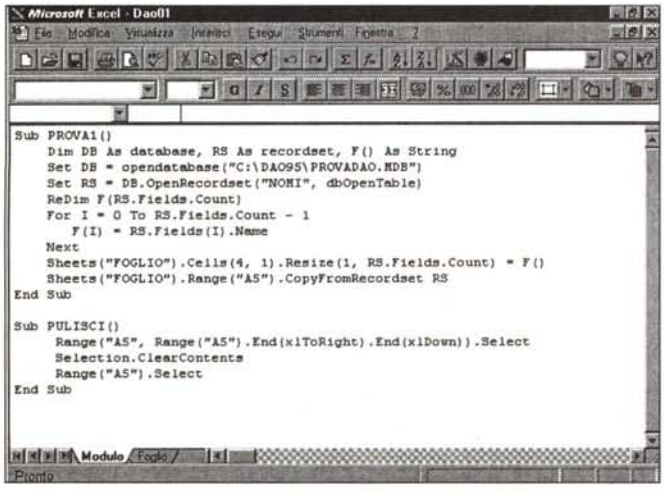


Figure 5, 6 - DAO, primi esperimenti - Excel 95 - Metodo CopyRecordSet - Output e Listato.
Facciamo subito un semplice esperimento che ha come finalità quella di "scaricare" nel foglio Excel una tabella presa da un database esterno. Sul foglio abbiamo piazzato due Pulsanti, uno che esegue l'operazione e uno che pulisce l'area occupata dai dati, e che va utilizzato prima della estrazione successiva. Nella parte Listato dell'esercizio possiamo vedere, tra le altre istruzioni che descriviamo nel testo, l'istruzione CopyRecordSet, che copia, a partire da una data cella, l'insieme dei dati definito come RecordSet. Da notare che ora Excel 95 utilizza comandi di programmazione in lingua originale.

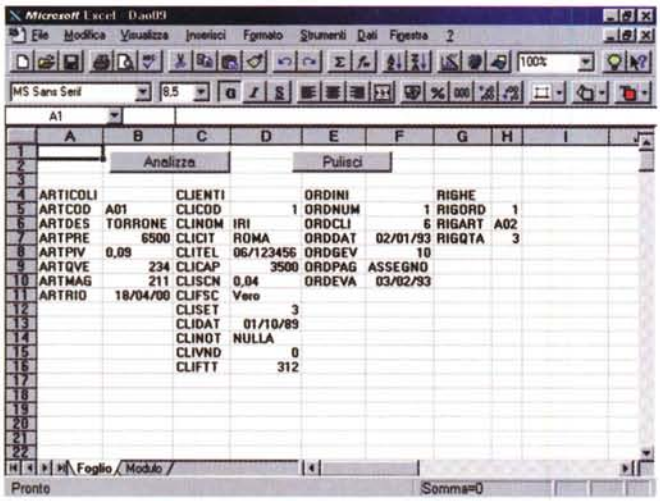
Il metodo CopyRecordSet

Abbiamo preparato un foglio con due Pulsanti (vedi figure 5 e 6). Il primo serve per eseguire un' estrazione di dati da una Tabella esterna per poi riversarli sul foglio a partire da una data cella. Il secondo pulsante serve per ripulire il foglio dai dati così estratti. Esaminiamo subito le istruzioni nel listato della prima procedura:

- dichiarazioni delle variabili,
- definizione dell'oggetto Database, il database MDB (formato Access), che si chiama PROVADAO.MDB,
- definizione dell'oggetto RecordSet, la tabella NOMI presente nel database PROVADAO.MDB,
- scarico, nel vettore F(), di tutti i nomi dei campi, utilizzando la proprietà FieldsCount e la collezione Fields().Name,
- scarico del vettore F() come intestazione delle colonne (a partire da A4),
- scarico dell'intero RecordSet a partire da A5.

Trattandosi del nostro primo esperimento vanno fatte una serie di considerazioni. È necessario dichiarare le variabili di tipo Database e di tipo RecordSet solo nel caso in cui tali variabili fossero utilizzate in più procedure. Nel nostro caso potevamo evitarlo.

Potevamo evitare anche lo scarico delle Intestazioni delle colonne, nel caso in cui queste fossero state fisse e quindi avremmo potuto scriverle direttamente sul foglio. In pratica per scaricare i dati, senza le intestazioni, bastavano solo tre istruzioni. Al momento della definizione dell'oggetto Record-



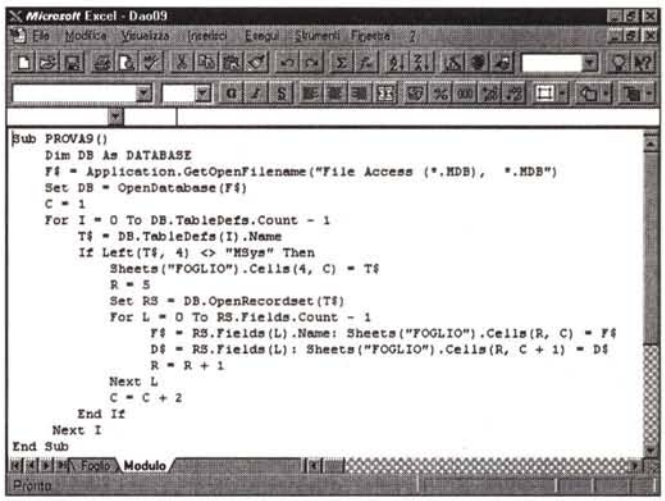
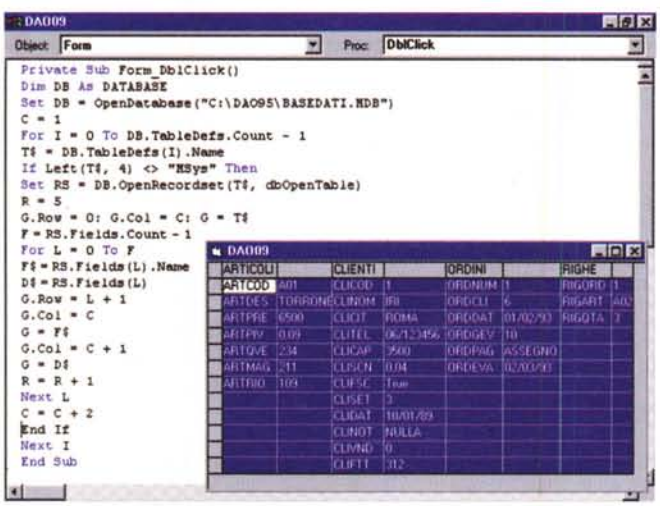


Figure 7, 8 - DAO, primi esperimenti - Lettura della Struttura del Database da Excel 95.
Questo programma, che può utilizzare qualsiasi database in formato MDB, ne individua le varie Tabelle e di ciascuna tabella legge Struttura e Contenuto del primo Record. I risultati di queste analisi vengono incolonnati nelle celle del foglio Excel. Nei vari esperimenti proposti utilizziamo anche le varie istruzioni che servono per scaricare dati nelle celle del foglio.



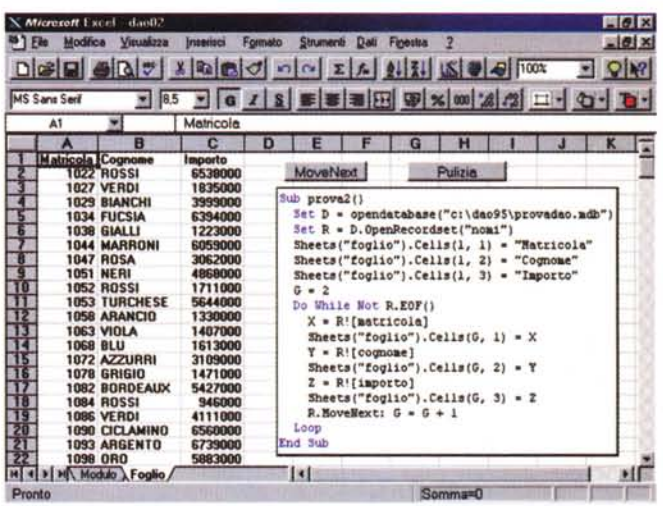


Figura 9 - DAO, primi esperimenti - Lettura della Struttura del Database da Visual Basic 4.0.
In questo fotomontaggio vediamo come sia possibile realizzare con il Visual Basic 4.0 lo stesso programma DAO visto in Excel 95. Come si vede il programma è assolutamente identico per quanto riguarda le istruzioni di accesso al database e di lettura delle sua caratteristiche. Variano ovviamente le istruzioni di visualizzazione. In particolare in VB utilizziamo, per vedere i dati della Struttura del Database, una griglia che si chiama G. Per poter "risparmiare una figura" abbiamo compresso il listato, togliendo le indentature.

Figura 10 - DAO, primi esperimenti - Excel 95 - Scorrimento semplice con il metodo MoveNext.
Il metodo CopyRecordSet è utilissimo e velocissimo. Non va bene però quando alla lettura dei dati esterni si voglia associare anche una loro elaborazione, da eseguire in concomitanza con lo scarico dei dati sulle celle del foglio. In questo caso occorre sfruttare un meccanismo di scorrimento, record per record, basato sulla classica struttura di programmazione Do While Not Eof .. MoveNext .. Loop, da eseguire sul RecordSet.

Set si può specificarne il tipo tra:

Table, una tabella normale,

Dynaset, una query con dati aggiornabili,

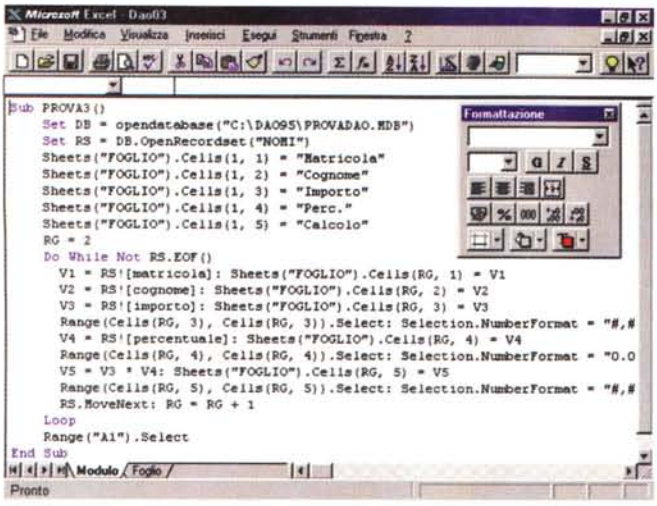
SnapShot, una query con dati non aggiornabili. È importante definire, o perlomeno conoscere, il tipo di RecordSet, in quanto alcuni metodi possono agire solo su alcuni tipi di RecordSet e non su altri.

Sulla procedura che fa "pulizia" non c'è nulla da dire, se non il fatto che viene selezionata tutta la zona piena a partire da A5. Le varie istruzioni sono quelle che "vanno fino in fondo", individuando automaticamente la dimensione della zona effettivamente piena di dati.

Non solo i dati

Vi proponiamo una coppia di esercizi identici, il primo realizzato con Excel 95 e il secondo con Visual Basic 4.0 (vedi rispettivamente figure 7, 8 e 9). Per quanto riguarda quest'ultimo prodotto lo abbiamo usato come se fosse una versione 3.0. Non ne abbiamo sfruttato, per ora, i controlli Windows 95 e le istruzioni DAO sono le stesse che avremmo potuto già usare nella versione 3.0.

Per vedere i dati abbiamo usato il fo-



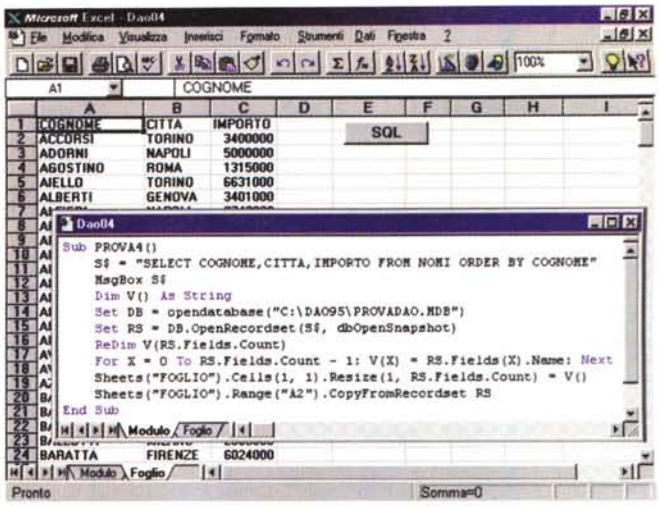


Figura 11 - DAO, primi esperimenti - Excel 95 - MoveNext e maggiore interazione con il foglio.
Qui vediamo, ma per motivi di spazio ne mostriamo solo il listato, un programma che, oltre a scorrere, con il sistema visto prima, via via i vari record del RecordSet, esegue dei calcoli (inserisce una colonna con un dato calcolato) e sistema la formattazione dei nostri dati. Anche in questo caso i nomi dei campi li abbiamo inseriti da programma (in quanto ipotizziamo di conoscerli a priori) e non li leggiamo dal RecordSet.

Figura 12 - DAO, primi esperimenti - Excel 95 - In fondo basta una istruzione SQL.
Il RecordSet può essere una Tabella, un Dynaset (ovvero una Query "aggiornabile") oppure uno Snapshot, ovvero una Query non aggiornabile, più veloce e sicura quando i dati non debbano essere aggiornati. Per definire un RecordSet è sufficiente definire una istruzione SQL. In questo fotomontaggio vediamo sia il programma DAO sia, sullo sfondo, il risultato della sua esecuzione.

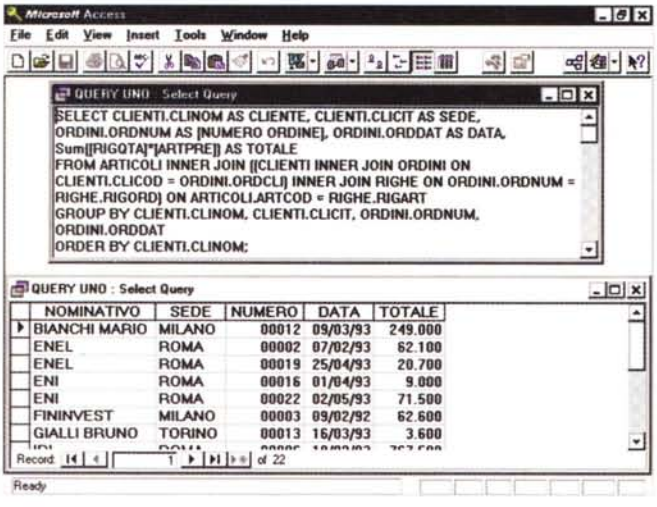


Figure 13,14,15 - DAO, primi esperimenti - La stessa SQL da MS Access 95, da VB 4.0, da Excel 95. Abbiamo appena detto e dimostrato che un RecordSet può essere ottenuto semplicemente utilizzando un'istruzione SQL. In queste tre figure, meglio commentate nel testo, vediamo la stessa istruzione SQL, utilizzata in MS Access 95 per generare una Query, utilizzata in VB 4.0 per generare la stessa query che riversa il suo risultato in una DBGrid e infine utilizzata in Excel 95. Possiamo affermare che per sfruttare bene DAO è necessario saper maneggiare bene anche SQL.

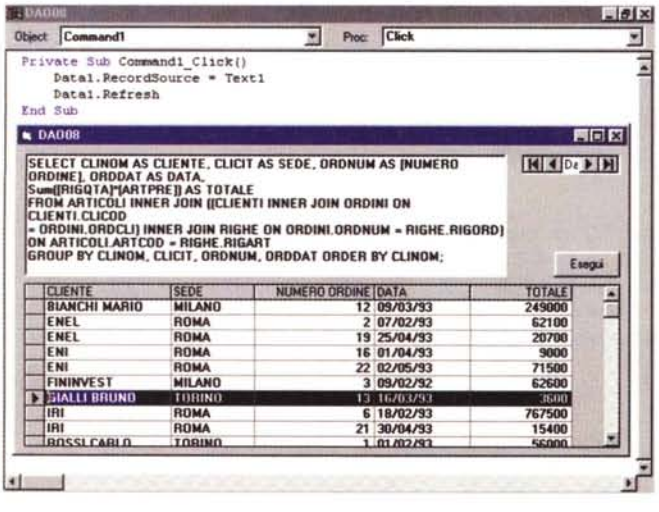
glio di Excel e una Griglia, quella normale, di Visual Basic.

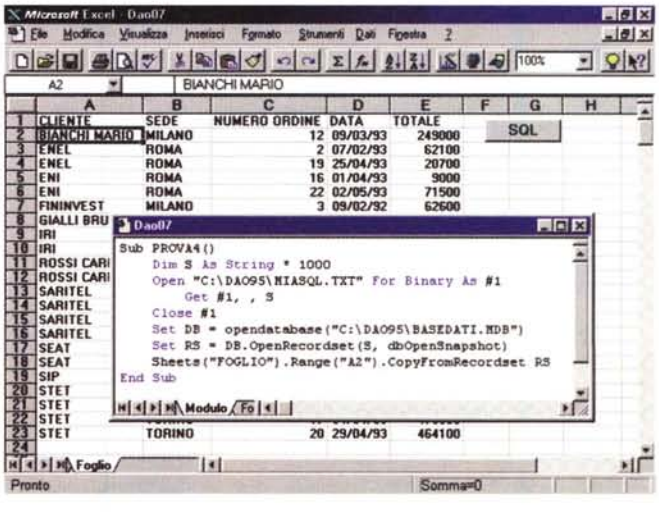
Il programma esegue due cicli FOR .. NEXT. Il primo scorre tutte le tabelle del database, "zompando" quelle di sistema che cominciano con MSys, il secondo scorre, nella singola tabella, tutti i campi. Le tabelle appartengono alla collezione TableDefs(I), i campi alla collezione Fields(L).Name. Ogni RecordSet ha, tra le altre, la proprietà FieldsCount che ne indica il numero totale dei campi.

Oggetti, proprietà, metodi, come si può vedere dai listati, sono identici in Excel e in VB.

Il metodo MoveNext

Il metodo CopyRecordSet è formidabile ma non permette di eseguire, sui





dati letti, nessuna forma di intervento a livello di singolo Record (vedi figure 10 e 11).

Allora quando occorre lavorare il singolo record si deve ricorrere ad un altro sistema. Ad esempio si può utilizzare il classico programma di scorrimento record che si basa sulla struttura:

```
Do While Not R.Eof()
...
R.MoveNext
Loop
```

In cui, ovviamente, R è il nostro beneamato RecordSet.

All'interno del ciclo, e quindi record per record, è possibile leggere campi, magari solo quelli desiderati, è possibile eseguire dei calcoli, è possibile eseguire istruzioni di riversamento nelle celle, è possibile formattare le celle di destinazione dei dati.

Tutto ciò che accade all'interno del ciclo è indipendente dal funzionamento del ciclo.

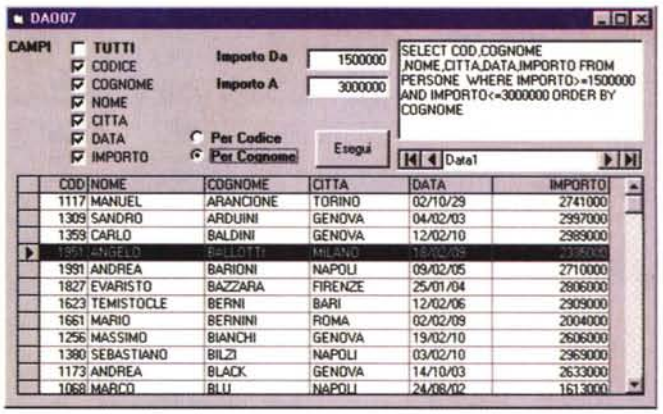
In figura 10 vediamo il ciclo più semplice che c'è. Il RecordSet viene scorso e vengono scaricati nelle celle solo i campi di interesse. È, ovviamente, necessario un contatore che permetta di definire e quindi di impostare la riga corrente.

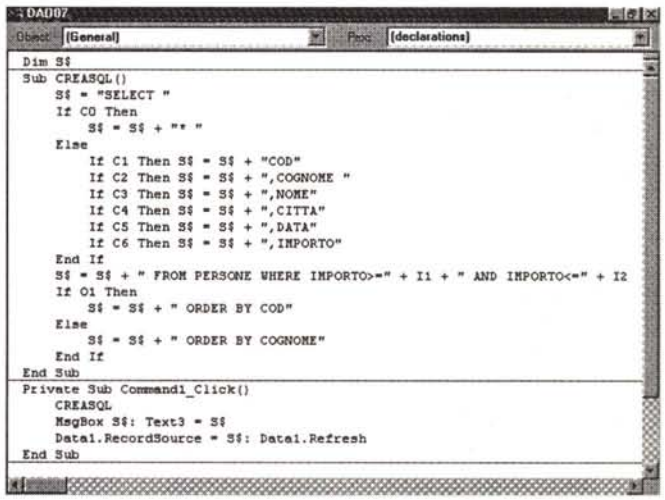
In figura 11 un esempio più complesso, ma solo per il fatto che viene eseguito un calcolo (in pratica viene creato un campo calcolato) e vengono formattate le celle in cui vengono riversati i dati numerici.

Va chiarito il fatto che mentre è possibile impostare il contenuto di una cella senza doverci "andare sopra", non è possibile impostarne il formato numerico. Quindi in questo caso il ciclo produce anche lo scorrimento della Cella Attiva, con il metodo Select.

Notare anche come in questi due esempi non abbiamo dovuto dichiarare le variabili in quanto tutte le operazioni vengono svolte da una sola procedura.

Figure 16, 17 - DAO, primi esperimenti - Costruzione, in modalità interattiva, della regola SQL.
In questo esercizio, realizzato per comodità con Visual Basic 4.0, vediamo come sia facilmente costruibile una SQL personalizzata, scegliendo i campi desiderati (o tutti o alcuni), un paio di condizioni (importo compreso tra due valori) e il tipo di ordinamento (tra due possibili). Vediamo anche come il listato sia relativamente succinto. Può diventare un po' più lungo se si inseriscono ulteriori "varianti" nelle modalità di estrazione. Una volta disponibile l'istruzione SQL, l'estrazione attraverso il DataControl e il riversamento nella DBGrid sono immediati.





DAO e SQL: inseparabili

Chi vuole utilizzare a fondo la tecnologia DAO deve conoscere a fondo anche SQL per il semplice fatto che la definizione di un RecordSet va fatta usando appunto la sintassi SQL (vedi figure dalla 12 alla 15).

Vediamo, nell'esercizio mostrato in figura 12, che è in pratica una semplice variante del primo esercizio, come sia facilmente definibile un RecordSet basato su un'istruzione SQL. In questo caso non è possibile specificare, come modalità di apertura, la modalità Table, ma solo quella Dynaset, con la quale i dati sono modificabili, o quella SnapShot, con la quale i dati non sono modificabili. La sintassi SQL è molto semplice, si basa su una decina di espressioni, qualora la si utilizzi solo per estrarre dati da un database, diventa più complessa in caso di aggiornamento di dati in un database già esistente, e ancora più complessa in caso di creazione di un nuovo database o di nuovi elementi in un database già esistente.

Se si usano tutti i prodotti DAO, e quindi Excel,Visual Basic ed Access si può sfruttare l'ambiente QbE di quest'ultimo per costruire in maniera guidata ed interattiva l'istruzione SQL, per poi ricopiarla, con un semplice Copia ed Incolla, là dove debba essere utilizzata (in una TextBox di una Form Visual Basic, in una cella di un Foglio di Excel).

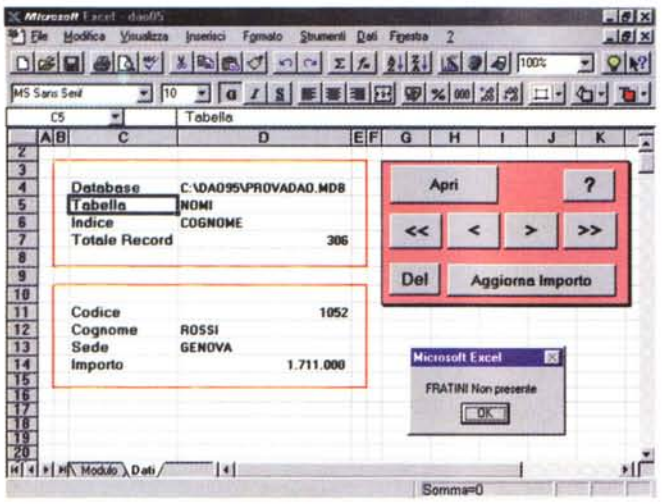
È chiaro che questa soluzione risulta comoda quando o non si conosca affatto SQL o quando l'istruzione sia particolarmente complessa e quindi si preferisca costruirla sfruttando una modalità guidata. Nelle tre figure che vanno dalla 13 alla 15 vediamo una stessa istruzione SQL, generata nell'ambiente Query di Access, riutilizzata in una DBGrid di Visual Basic 4 e infine usata come definizione del RecordSet in Excel.

Vanno fatte tre precisazioni.

La prima è che il generatore di SQL di Access è un po' prolisso. Ad esempio definisce ogni campo indicandone anche la tabella di appartenenza. Questa precisione non è necessaria quando si abbia l'accortezza di evitare di dare nomi di campi uguali in tabelle dello stesso database.

Quindi, nel portare l' istruzione SQL in VB e in Excel, l'abbiamo debitamente alleggerita.

La seconda è su come può essere utilizzato un controllo DBGrid di Visual Basic. Se esaminiamo la figura 14 vediamo che sulla Form ci sono quattro Oggetti: una TextBox, nella quale abbiamo ricopiato l'istruzione SQL, un Da-




```
Global D As database, R As recordset

Sub APRI()
  Set D = opendatabase("c:\dao95\provadao.mdb")
  Set R = D.OpenRecordset("NOMI"): R.Index = "Cognome"
  Range("D4").Formula = UCase(D.Name)
  Range("D5").Formula = UCase(R.Name)
  Range("D6").Formula = UCase(R.Index)
  Range("D7").Formula = R.RecordCount
  SCRIVI
End Sub

Sub SCRIVI()
   X = R![matricola]: Range("D11").Formula = X
   Y = R![cognome]: Range("D12").Formula = Y
   W = R![citta]: Range("D13").Formula = W
   Z = R![importo]: Range("D14").Formula = Z
End Sub

Sub primo()
    R.MoveFirst: SCRIVI
End Sub

Sub indietro()
    R.MovePrevious
    If R.BOF Then: MsgBox "Primo Record": R.MoveFirst
    SCRIVI
End Sub

Sub avanti()
    R.MoveNext
    If R.EOF Then: MsgBox "Ultimo Record": R.MoveLast
    SCRIVI
End Sub

Sub ultimo()
    R.MoveLast: SCRIVI
End Sub

Sub cerca()
    B = R.Bookmark
    K = InputBox("Cognome desiderato", "Metodo Seek", "ROSSI")
    R.Seek "=", K
    If R.NoMatch Then
        MsgBox K + " Non presente"
        R.Bookmark = B
    Else
        SCRIVI
    End If
End Sub

Sub aggiorna()
    R.Edit
    R![importo] = Range("D14").Value
    R.Update
End Sub

Sub cancella()
    R.Delete
End Sub
```

taControl, la cui proprietà RecordSource è proprio il contenuto della TextBox, una DBGrid, la cui proprietà RecordSource è il DataControl, e infine un Pulsante che manda in esecuzione la Requery del DataControl e conseguentemente l'aggiornamento del contenuto della DBGrid.

L'ultima precisazione. L'istruzione SQL è una stringa che può essere anche molto lunga. Se supera i 256 caratteri non può, ed esempio, essere riportata in una cella di Excel. Se per memorizzarla si utilizza una variabile di tipo stringa, occorre dichiararne la lunghezza (come lo si vede chiaramente nella figura 15).

Figure 18, 19 - DAO, primi esperimenti - Metodi Seek, Edit, Update - Il foglio e il modulo.
Qui vediamo un altro tipo di problematica, ben differente da quella risolta con il RecordSet: quella che riguarda l'accesso al singolo Record e la "lavorazione" del singolo Record. Abbiamo realizzato una minipulsantiera che serve per spostarsi tra i record, per aggiornare il record corrente, per cercare il record del tizio desiderato indicandone il cognome. Non abbiamo inserito, solo per motivi di spazio, il pulsante Inserimento nuovo Record. Nel testo spieghiamo come fare.

La costruzione della Istruzione SQL

Se le varianti della nostra istruzione SQL non sono tantissime risulta abbastanza facile costruirsi un proprio, personalissimo, generatore di istruzioni SQL (vedi figure 16 e 17). Ve ne proponiamo uno realizzato in Visual Basic. In figura 16 vediamo il suo aspetto esteriore e nella successiva figura 17 vediamo il breve listato che serve per confezionare l'istruzione SQL sulla base delle scelte, effettuate dall'utente, sulle varie CheckBox e nei vari controlli piazzati sulla Form.

La grossa semplificazione sta nel fatto che si lavora su una sola tabella. Di questa in pratica si possono scegliere i Campi, l'ordinamento e si può impostare un filtro, basato su un criterio di tipo "Maggiore di .. Minore di".

Nella casella di testo più grande viene mostrata l'istruzione SQL generata.

La stessa Form si può costruire con Excel 95 e con Access 95. Lasciamo a voi l'onore.

Altri metodi che agiscono sul RecordSet

Nel successivo esercizio vediamo come si possa simulare, anche (oserei dire addirittura) su un foglio Excel, una Form di scorrimento, visualizzazione e gestione dati (vedi figure 18 e 19). In pratica vogliamo che sul foglio appaia un record per volta e che i campi, quelli del Record attivo al momento, appaiano sempre nelle stesse celle.

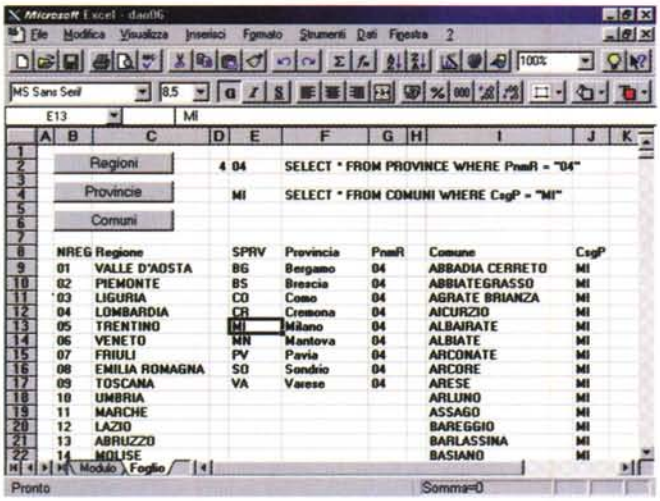
Osserviamo la videata in figura 18. Sul foglio riconosciamo tre zone: quella con i dati generali riferiti al RecordSet in uso, posti in alto a sinistra, quella con i dati riferiti al Record Corrente, in basso a sinistra, e la pulsantiera con tutti i bottoni necessari per eseguire le varie operazioni.

In pratica pigiando il pulsante Apri viene aperto il database PROVADAO.MDB, la tabella NOMI e l'indice sul COGNOME. Vengono anche riempite le celle con le informazioni sul Database e sul RecordSet in uso.

La routine SCRIVI è quella che si occupa di riversare sulle singole celle il contenuto dei campi del record corrente. Per motivi di spazio vediamo solo quattro campi.

I vari pulsanti attivano una serie di metodi che agiscono sul RecordSet (nel nostro esempio si chiama R):

R.MoveFirst	si sposta sul primo record
R.MovePrevious	si sposta sul precedente record
R.MoveNext	si sposta sul successivo record
R.MoveLast	si sposta sull'ultimo record
R.Bookmark	definisce un segnalibro
R.Index	definisce l'indice attivo




```
Global DB As database
Global S$

Sub primo()
    start
    pulisciR
    regioni
End Sub

Sub secondo()
    Range("D2").Value = ActiveCell.Value
    S$ = "SELECT * FROM PROVINCE WHERE PnmR = "
    S$ = S$ + Chr(34) + Range("E2").Value + Chr(34)
    Range("F2").Value = S$
    pulisciP
    province
End Sub

Sub terzo()
    Range("E4").Value = ActiveCell.Value
    S$ = "SELECT * FROM COMUNI WHERE CsgP = "
    S$ = S$ + Chr(34) + Range("E4").Value + Chr(34)
    Range("F4").Value = S$
    pulisciC
    comuni
End Sub

Sub start()
    Set DB = opendatabase("c:\dao95\italia.mdb")
End Sub

Sub regioni()
    Set R = DB.OpenRecordset("regioni")
    Sheets("foglio").Range("B9").CopyFromRecordset R
    Range("a1").Select
End Sub

Sub province()
    Set R = DB.OpenRecordset(S$, dbOpenSnapshot)
    Sheets("foglio").Range("E9").CopyFromRecordset R
    Range("A1").Select
End Sub

Sub comuni()
    Set R = DB.OpenRecordset(S$, dbOpenSnapshot)
    Sheets("foglio").Range("I9").CopyFromRecordset R
    Range("A1").Select
End Sub

Sub pulisciR()
     Range("B9", Range("B9").End(xlToRight).End(xlDown)).Select
     Selection.ClearContents: Range("B9").Select
End Sub

Sub pulisciP()
     Range("E9", Range("E9").End(xlToRight).End(xlDown)).Select
     Selection.ClearContents: Range("E9").Select
End Sub

Sub pulisciC()
     Range("I9", Range("I9").End(xlToRight).End(xlDown)).Select
     Selection.ClearContents: Range("I9").Select
End Sub
```

Figure 20, 21 - DAO, primi esperimenti - SQL direttamente dal foglio.
Vediamo tre tabelline, poste sullo stesso foglio di Excel, messe in sequenza. La prima, che rimane fissa, con le Regioni d'Italia, la seconda con le Province della Regione scelta nella prima tabella, e la terza con i Comuni della Provincia scelta nella seconda tabella. Abbiamo poi creato un meccanismo "a valanga", che, sempre sul stesso foglio, mostra le Province di una data Regione e i Comuni di una data Provincia, scelta tra quelle della data Regione.

R.Seek	cerca sul campo dell'indice attivo
R.NoMatch	indica la situazione di record non trovato
R.Edit	attiva la situazione di aggiornamento
R.Update	aggiorna i campi
R.Delete	cancella il record attivo

Di tutti questi metodi trovate traccia nei vari listati che costituiscono l'applicazione. Manca all'appello, per poter avere un quadro completo delle possibilità, il metodo:

R.AddNew

ed parecchi altri meno importanti.

A questo punto facciamo qualche considerazione.

La prima è che questo codice è assolutamente identico in VB ed in Access, varia solo la routine SCRIVI che non dovrà scrivere in Celle del Foglio, ma in Caselle di Testo.

La seconda è la necessità di capire bene la differenza tra Table e Dynaset.

Con la prima struttura si può usare il metodo Seek (che necessita di un indice che deve esistere nella struttura della tabella e che deve essere indicato prima di poter usare il Seek), con la seconda si può invece usare il metodo FindFirst e il FindNext. Non è possibile il viceversa nel senso che su una Table non si può usare il FindFirst, e su un Dynaset, e meno che mai su uno SnapShot, si può usare il metodo Seek.

Altra considerazione è quella che tali metodi, che per semplicità abbiamo utilizzato su Database in formato Access, sono utilizzabili anche su altri tipi di dati, dBase, SQL, ecc.

L'ultimissima considerazione la riserviamo a coloro che conoscono bene la sintassi xBase. Costoro avranno notato la corrispondenza di buona parte dei comandi, tra DAO e dBase. Ad esempio:

R.Index	Set Index
R.FindFirst CR	Locate for
R.FindNext	Continue
R.Seek KEY	Seek KEY
R.MoveNext	Skip

ecc.

Conclusioni

Se siete riusciti ad arrivare fino a qui significa che siete in grado di capire, senza il nostro aiuto, il funzionamento dell'ultimo esercizio (proposto nelle figure 20 e 21) nel quale realizziamo un processo a cascata. Si sceglie una Regione (selezionando la cella con il suo codice) e se ne estraggono le Province. Si sceglie la Provincia e si estraggono i suoi Comuni. Tutto accade in foglio Excel. Anche in questo caso il listato è talmente ridotto da essere contenuto in una pagina formato A4. Circa un terzo delle istruzioni sono quelle che servono per ripulire le celle prima della successiva estrazione.

I Comuni d'Italia sono 8.000 e le prestazioni di questo motore di estrazione sono assolutamente interessanti, ben superiori a quelle, modeste per la verità, del vecchio MS Query.

In conclusione pensiamo di avervi dato un'idea abbastanza chiara di cosa vuol dire DAO.

Dal nostro punto di vista ci ripromettiamo di riprendere il discorso, in un successivo articolo, per proporvi altri approfondimenti. Uno sicuramente riguarderà l'utilizzo di altri formati di dati, dBase e SQL in testa. Ci interessa anche sperimentare DAO con Database più complessi, con più tabelle e più relazioni e più voluminosi, con decine di migliaia di dati. A presto.

MC