Lo Zen e l'arte dell'accesso ai dati con VB

Il titolo dell'articolo fa il verso al titolo di un libro di Robert Pirzig, famoso agli inizi degli anni '80, che si intitola appunto Lo Zen e l'Arte della Manutenzione della Motocicletta (Adelphi 1981). L'autore coglieva numerose analogie tra la filosofia orientale e l'attività manuale eseguita su un mezzo meccanico, quale la motocicletta, pieno di simboli di vario genere. Nel nostro articolo vogliamo tornare sull'argomento Visual Basic e l'Accesso ai Dati cercando soprattutto di cogliere gli aspetti più generali, quelli che si possono definire filosofici, di questa attività, che molti appassionati ritengono una vera e propria Arte

di Francesco Petroni

Connessa strettamente con la problematica Gestione dei Dati, c'è, ad esempio, la Teoria Relazionale, teoria che si appoggia esclusivamente su elementi di Logica e che, nella sua concezione estrema, prescinde dall'esistenza del computer.

Riteniamo che un utilizzatore esperto debba raggiungere una propria capacità di vedere i problemi dall'alto, ad esempio intuendo la logica del sistema che sta trattando e la sua architettura, lasciando ad un momento successivo il lavoro materiale e manuale sul database, sulle tabelle, sui campi, ecc.

A supporto della trattazione proporremo, come al solito, una serie di esempi pratici. Questi saranno di tipo «minimalista», per fare il verso in questo caso al nome di una corrente letteraria americana (seconda metà degli anni '80). Diremo cioè il minimo indispensabile, ad esempio citeremo le due o tre operazioni fondamentali necessarie per collegare la nostra applicazione VB con il Database: il Database contiene i dati, l'applicazione VB ne permette la gestione.

Gli esempi che vi proponiamo fanno riferimento ad alcune figure, che in qualche caso conterranno listati di programmi. La lunghezza massima di tali listati, anche per gli esempi più complicati, è comunque tale da essere contenuta in una videata e quindi nella figura stessa.

I tipi di dati da trattare

Il Visual Basic (utilizzeremo la versione 4.0 Professionale, ma il discorso vale quasi sempre anche per la versione Standard e per quelle precedenti), è ben dotato in termini di possibilità di accesso ai Dati. La prima cosa da dire è che non dispone di un suo particolare formato per i Dati, nel senso che non esistono Database in formato Visual Basic. Questa affermazione risulta falsa se si vuole considerare il formato Access come il formato dati di Visual Basic.

La verità sta nel mezzo e la esprimeremo così: il formato Access (il suffisso dei file è MDB) è utilizzato sia dal prodotto Access che dal linguaggio Visual Basic. Però il Visual Basic, ma il discorso vale anche per Access, può utilizzare anche altri formati dati, dal semplice formato testuale sequenziale, all'immortale formato DBF, fino ai più sofisticati formati SQL, raggiunti tramite driver ODBC.

Se, nel realizzare un'applicazione, si fosse liberi di scegliere il formato dei dati la scelta più opportuna sarebbe quella del formato MDB, sia per un problema di prestazioni, che per un problema di controllo dei dati. Di questo parleremo tra un po'.

In caso di applicazioni Aziendali, utilizzate da più utenti che condividono un Database complesso posto su un Server, il formato del Database sarà SQL e l'applicazione VB farà da Front End verso l'utente. Ricordiamo l'esistenza della versione Enterprise del Visual Basic 4.0 che consente la programmazione Client/Server, ovvero di costruire dei programmi eseguiti dal Server ma richiamati dal Client.

Il formato DBF continua ad essere il formato dati più diffuso nelle applicazioni su PC. Visual Basic accede tramite un Driver ISAM (che deve essere installato) a tale tipo di File. VB è, lo diciamo per gli esperti del mondo dBase o Clipper, in grado di gestire anche i file Indice (NDX o NTX) e dispone di una serie di istruzioni che simulano tutti i principali comandi dBase.

Infine i file testuali. VB, in quanto Basic, è in grado di leggerli benissimo. È chiaro che la lettura di un file testuale può essere praticata come funzionalità di Servizio, ad esempio per eseguire un Import «una tantum». Non si può certo appoggiare una complessa procedura gestionale su file testuali.

Figura 1 - Un file dati di tipo testuale: il famigerato «comma delimited».
Per i nostri esercizi useremo gli stessi dati, serviti in file di vari formati: testuale, DBF e MDB. Il meno strutturato di tutti è il formato testuale, costituito solo da dati puri. Nella sua variante «Comma Delimited» ci sono delle virgole che dividono un Campo da un altro e le righe rappresentano i Record. In un file del genere non c'è traccia della struttura, che quindi deve essere conosciuta da chi usa i dati.

Il minimo dello sforzo, il massimo del risultato

Come detto, gli esercizi che vi proponiamo tendono a scarnificare i vari pro-

blemi relativi all'accesso ai dati, nel senso che vogliamo usare solo le istruzioni fondamentali minime, senza le quali l'operazione non potrebbe essere compiuta. Le altre istruzioni, ad esempio tutte quelle per la visualizzazione dei dati, per l'elaborazione di nuovi dati calcolati, quelle per il movimento, avanti ed indietro, sui dati, ecc. sono un di più.

Per realizzare alcuni degli esercizi non è necessario scrivere neanche una riga di programma e quindi, per questi, indicheremo quali Oggetti usare e quali Proprietà impostare per accedere ai dati.

In altri programmi scriveremo del codice e ne evidenzieremo le istruzioni fondamentali. Sia nel primo che nel secondo caso (senza codice, con codice) le proprietà e le istruzioni fondamentali sono pochissime.

Tornando agli esercizi ne abbiamo di quattro tipi:

- esercizi che usano Oggetti e Proprietà e non usano codice,
- esercizi in cui le stesse funzionalità, proprie di Oggetti e Proprietà, vengono ottenute via codice,
- esercizi che usano Oggetti e Proprietà e usano parti di codice per modificarne alcune in fase di esecuzione,
- esercizi con parti di codice che eseguono operazioni non eseguibili utilizzando gli Oggetti e le Proprietà.

Un buon utilizzatore di Visual Basic deve conoscere gli Oggetti, le Proprietà e i Metodi a menadito, non deve avere dubbi su cosa usare e su come usarlo, a seconda delle situazioni operative in cui si trova.

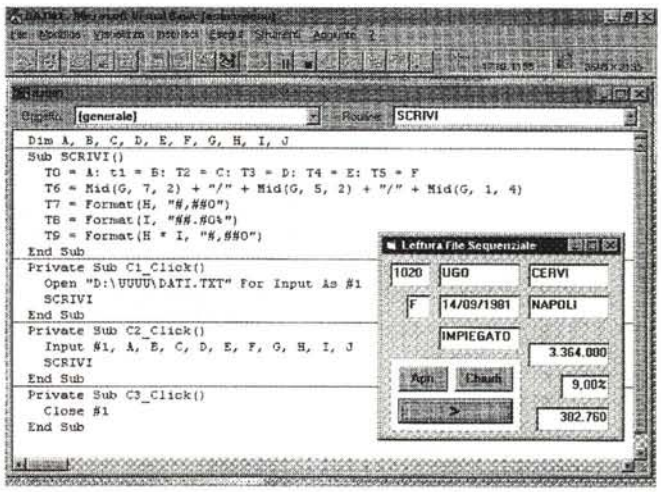


Figura 2 - Visual Basic 4.0 - Un'applicazione che legge e visualizza un file Testuale. Le istruzioni che servono per leggere un file con dati testuali, separati da virgole (Comma Delimited) oppure a lunghezza fissa (Fixed Length), sono tre: OPEN, INPUT, CLOSE. Le altre che vediamo nei listati e che commentiamo nel testo, sono di contorno all'operazione principale. La lettura è sequenziale, si legge un record per volta a partire dal primo fino all'ultimo, su cui si verifica la situazione di fine file, che viene intercettata per far chiudere il file. Non è possibile, con questo sistema, puntare ad un Record predeterminato, occorre in ogni caso leggere tutti i precedenti.

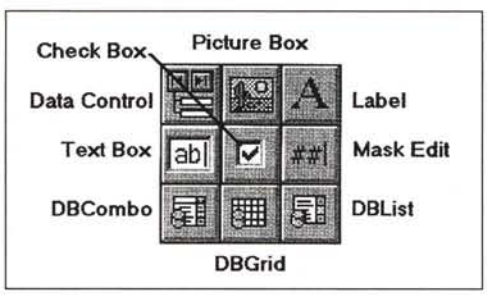


Figura 3 - Visual Basic 4.0 - I Controlli per Gestire i Dati. Visual Basic consente numerose modalità di accesso ai file dati esterni. Abbiamo appena visto la più antica: la lettura, in modo sequenziale, di un file testuale, ottenuta sfruttando alcune classiche istruzioni Basic. L'accesso ai Database veri e propri avviene invece attraverso i servizi svolti dai vari Controlli. Ce ne sono di tre tipi: c'è il DataControl che serve per stabilire un collegamento tra Database e Form. Ce ne sono altri tre, il DBGrid, il DBList e il DBCombo, che vengono riempiti con dati provenienti da un DataControl. Molti degli altri, ad esempio la Casella di Testo (alias TextBox), possono visualizzare i dati messi a disposizione dal DataControl.

I Controlli VB4 che hanno a che fare con i Dati

Anche qui categorizzeremo, facendo riferimento alla figura 3. Abbiamo tre categorie di Controlli (gli oggetti in VB si chiamano Controlli).

- il Controllo, e c'è solo il DataControl, che serve per stabilire un collegamento tra l'applicazione e il Database,
- i tre Controlli, DBGrid, DBCombo e DBList, che mostrano i dati messi a disposizione da un DataControl. Non possono vivere di vita autonoma, debbono essere legati ad un DataControl,
- altri Controlli, una mezza dozzina, legabili, ma non obbligatoriamente, ai singoli dati messi a disposizione da un DataControl.

Per quanto riguarda questi ultimi prendiamo come esempio una semplice Casella di Testo, che serve per mostrare un dato, nel nostro caso un campo di un record. Ebbene una casella di testo può:

- essere legata, attraverso le sue proprietà, ad un Campo proposto da un DataControl,
- essere legata, attraverso codice, ad un Campo proposto da un DataControl,
- essere legata ad un Campo, direttamente via codice (senza utilizzare il DataControl),
- essere legata ad un dato ottenuto via procedura, ad esempio un campo calcolato.

Leggiamo i file sequenziali e non pensiamoci più

Per oltre un decennio gli unici file dati gestibili con il Basic sono stati quelli di tipo testuale, leggibili e scrivibili solo in modo sequenziale, e quelli ad accesso casuale, leggibili e scrivibili puntando direttamente ad un preciso record di cui fosse noto l'indirizzo, ovvero la posizione all'interno del file.

Questi due sistemi, con i quali sono state realizzate fino a pochi anni fa applicazioni anche molto complesse, sono stati ormai abbandonati, in quanto ora ci sono sistemi di accesso a Database strutturati ben più efficaci.

Ciononostante, in molti casi, in un'applicazione Visual Basic può essere utile leggere in maniera sequenziale file di tipo testuale, ad esempio quando occorra eseguire una importazione di dati, che poi magari vengono scaricati nel Database strutturato, oppure quando si utilizzino semplici tabelle di servizio, che però vengono aggiornate al di fuori della procedura.

Insomma pensiamo che non sia raro il caso in cui non solo occorra, ma addirittura convenga, utilizzare file testuali.

In figura 1 vediamo come appare, in un Word Processor, un file testuale di tipo «comma delimited». Si può notare come i campi siano separati da virgole, come i campi stringa appaiano tra virgolette, come quelli numerici invece non abbiano le virgolette, come non esista traccia di intestazioni o di nomi dei campi, e via dicendo. I dati invece ci sembrano, ad occhio e croce, leggibili.

Con il programma Visual Basic vogliamo creare una procedura che riceva i

dati, incasellandoli esattamente uno per uno. La vediamo in figura 2.

Nel listato ci sono una serie di istruzioni, sono sempre le stesse in programmi di questo tipo, che servono per aprire il file testuale (OPEN), per leggere i vari campi del record corrente (INPUT), per chiudere il file (CLOSE).

Nella istruzione OPEN va indicato il numero del file, es.1, numero da utilizzare con le altre istruzioni quando si voglia lavorare su quel file. L'istruzione INPUT va riferita al numero del file, e deve necessariamente esser seguita da tante variabili quanti sono i campi da leggere. In altre parole i campi vanno letti tutti, anche se poi magari se ne utilizzano solo alcuni. Altra particolarità dell'istruzione INPUT è quella di portare il puntatore del record al record successivo, senza che se ne debba occupare il programmatore.

Questo consente di creare facilmente un ciclo DO WHILE... LOOP che legge tutti i record, sequenzialmente, e che termina quando si è raggiunta la situazione di Fine File.

Le cinque istruzioni, che andrebbero scritte ad occhi chiusi, sono:

```
OPEN «NOME FILE» For Input As #1
DO WHILE NOT EOF(1)
INPUT #1, A,B,C...
LOOP
CLOSE #1
```

Nel nostro esercizio invece, vi proponiamo una lettura sequenziale guidata da un tasto identificato da un segno «>». Click sul tasto per leggere, e visualizzare, il record.

Il tasto Apri (C1) si occupa di aprire il file, il tasto «>» (C2) di leggere una riga e di visualizzarla richiamando la routine SCRIVI, il tasto Chiudi (C3) di chiudere il file. Se volete potete provare il programma creandovi, con il NotePad, un filetto testuale di pochi campi e pochi record.

Il DataControl: c'è chi lo ama, c'è chi lo odia. Non lo odiamo, ma preferiamo farne a meno

Supponiamo di avere un file in formato dBase III, insomma un classico file DBF, di cui conosciamo il nome (nel nostro caso si chiama DATI.DBF), la posizione sul disco rigido (nel nostro caso la directory D:\UUUU) e possibilmente la struttura (i suoi campi CODICE, NOME, COGNOME, ecc.).

Per «vederlo» da VB la prima cosa da fare è quella di piazzare un DataControl nel nostro Form e di definirne tre o quattro caratteristiche fondamentali.

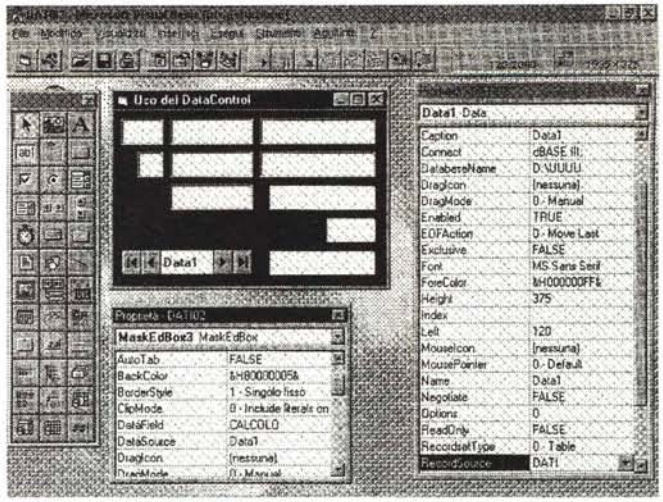


Figura 4 - Visual Basic 4.0 - Le tre proprietà fondamentali del DataControl.
Un DataControl serve a creare un ponte tra i Dati, in qualsiasi formato strutturato essi siano, e i Form di Visual Basic. Sono tre le sue proprietà fondamentali: la prima serve per indicare il tipo dei Dati a cui si accede (Connect), la seconda il nome del Database (DatabaseName) e l'ultima il nome della Tabella (RecordSource). In VB4 c'è una quarta proprietà che serve per definire il tipo di RecordSource. Oltre al tipo Tabella, c'è il tipo Dynaset (vista diretta sui dati, aggiornabili), c'è il tipo Snapshot (vista sui dati, non aggiornabili).

- Connect per indicare il tipo di dati a cui stiamo accedendo. Il tipo è dBase III,
- DatabaseName per indicare il nome della directory in cui c'è il file DATI. DBF,
- RecordSource per indicare il nome della tabella, che è appunto DATI.DBF.

E basta. Siamo già collegati!!!!

Possiamo quindi lanciare il programma e verificare... che non succede nulla. Questo perché siamo collegati al file ma non abbiamo ancora impostato nessun oggetto che visualizzi i suoi Dati. Allora inseriamo una Casella di Testo e impostiamo le due proprietà fondamentali, che sono:

- DataSource va impostato il nome del DataControl da cui è alimentata, ad esempio Data1,
- DataField il campo del DataSource. Ad esempio Cognome.

Insomma abbiamo una Form con due soli Controlli, abbiamo impostato complessivamente solo cinque Proprietà e già vediamo i dati.

Nella figura 4, un collage, vediamo le Box delle proprietà, sia quella del DataControl che quella di una MaskEditBox, una Casella di Testo speciale alla quale si può assegnare direttamente un formato numerico. Anche la MaskEditBox ha le due proprietà DataSource e DataField. La quarta proprietà fondamentale del DataControl è la RecordsetType. Può avere tre valori:

1) se il DataControl viene alimentato da una Tabella,

2) se il DataControl viene alimentato da una Query SQL che permette anche l'aggiornamento dei Dati (Dynaset),

3) se il DataControl viene alimentato da una Query SQL che non permette l'aggiornamento dei Dati (SnapShot).

Ne parliamo dopo.

Nelle successive due figure (la 5 e la 6) vediamo un'applicazione che usa tre DataControl e che lavora sullo stesso file di prima, il file DATI.DBF. I primi due DataControl servono per alimentare due DBCombo in modo che propongano la lista delle CITTA, senza ripetizione, presenti nella tabella DATI.DBF, e la lista delle QUALIFICHE senza ripetizione.

In pratica occorre definire per il primo DataControl:
Connect: dBase III;
DatabaseName: nome della directory che contiene la tabella
Name: Data1, conserviamo il nome di default
RecordType: 2 - Snapshot, set di dati non aggiornabili
RecordSource: SELECT DISTINCT CITTA FROM DATI

Questa è l'istruzione SQL che estrae dalla tabella DATI tutte le città presenti senza ripetizioni. In pratica otteniamo l'elenco delle città per le quali ci sia almeno un dato.

Per la prima DBCombo le uniche proprietà da settare sono:

Name	C1
RecordSource	Data1
Field	Citta

Insomma la DBCombo mostra le CITTA per le quali si possono cercare dati. Lo stesso discorso si può fare per le QUALIFICHE, usando il secondo DataControl e la seconda DBCombo, che chiamiamo C2.

Per complicarci un po' la vita inseriamo anche un paio di Caselle di Testo (D1 e D2) nelle quali scriviamo due date per creare un intervallo temporale.

A questo punto abbiamo selezionato una CITTA (C1) e una QUALIFICA (C2) e impostato due date (D1 e D2). Possiamo facilmente confezionare una istruzione SQL che elenchi tutti i dati di quella città, quella qualifica e la cui data sia compresa tra le due indicate. L'istruzione la proponiamo in una Casella di Testo, poi la impostiamo come RecordSource del terzo DataControl, che a sua volta è usato

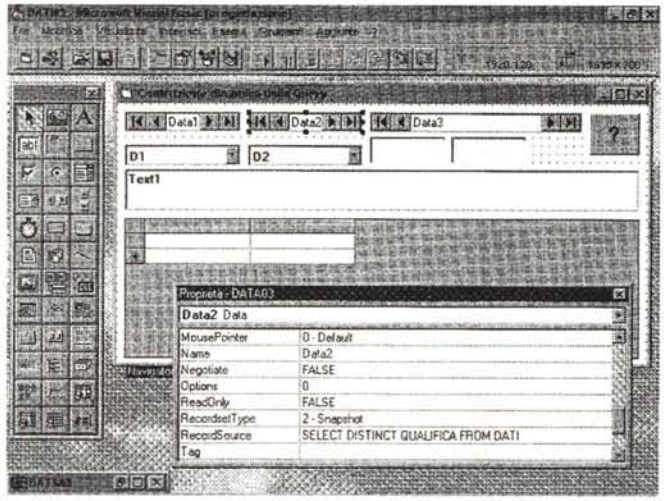


Figura 5 - Visual Basic 4.0 - Sfruttamento del DataControl - Un Form con tre DataControl. Il Form mostrato utilizza tre DataControl ognuno dei quali svolge un preciso servizio. I primi due alimentano due DBCombo, nei quali si scelgono due valori dei campi CITTA e QUALIFICA. Il terzo alimenta la DBGrid, utilizzando una Query costruita ed eseguita al volo in seguito alle scelte operate sulle due Combo. In pratica viene costruita l'espressione SQL che viene assegnata alla proprietà RecordSource del DataControl. Dato il loro lavoro del tutto... sotterraneo, i tre DataControl potrebbero essere resi invisibili, in quanto funzionerebbero anche al buio.

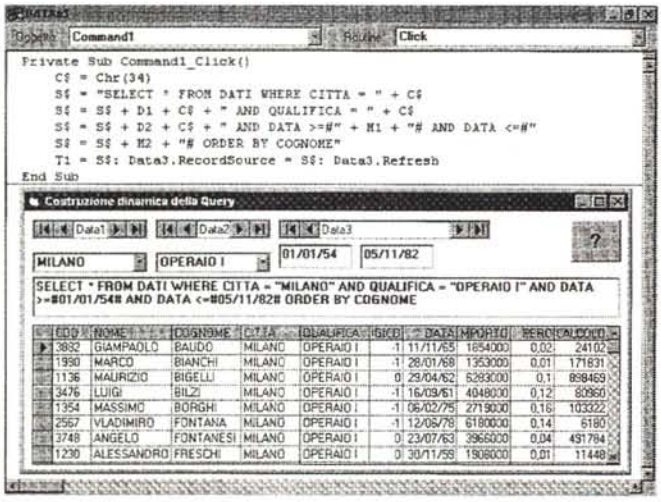


Figura 6 - Visual Basic 4.0 - Sfruttamento del DataControl - Senza una riga di programma. Questo meccanismo, che spero abbiate capito, è molto sofisticato nel senso che esegue un'operazione abbastanza complessa, e lo fa in una maniera efficace ed efficiente. L'aspetto rivoluzionario di questo modo di lavorare con Oggetti, Proprietà e Metodi, è che per fare tutto questo non serve scrivere neanche una riga di programma.

come RecordSource della DBGrid posta in basso.

La singolarità della DBGrid è che può essere alimentata impostando una sola proprietà, quella che indica da quale DataControl «becca» i dati.

Utilizzo nobile del DBCombo

In figura 7, anche questa è un collage, vi proponiamo un esercizio sulla DBCombo. Lavoriamo sempre sulla tabella DATI.DBF ipotizzando di aver sostituito il campo QUALIFICA con il campo CODICEQ, e quindi il valore in chiaro della qualifica, ad esempio DIRIGENTE, con un valore in codice, ad es. A. In altre parole ora il campo che mostra la qualifica farà riferimento ad un codice che verrà scodificato da un'altra tabella, che conterrà almeno due campi, il codice e la sua descrizione. Nel nostro caso:

CODICEQ	QUALIFICA
A	DIRIGENTE
B	FUNZIONARIO
C	IMPIEGATO I
C	OPERAIO I
E	OPERAIO II

In un Form si può utilizzare, per gestire il campo CODICEQ della Tabella DATI, una DBCombo che visualizza però i dati della tabella QUALIFICHE. In questo caso, interessantissimo e un po' difficile da capire, risultano coinvolte le seguenti proprietà della DBCombo:

-RowSource: Data1, il DataControl che propone la tabella Qualifiche

-ListField: Qualifica, campo da vedere, quello con le qualifiche in chiaro

-DataSource: Data2, il DataControl che contiene i dati da aggiornare, ovvero la tabella DATI

-DataField: il campo del Data1 con il quale aggiornare il campo del Data2

-BoundColumn: il campo del Data2, aggiornato con il DataField di Data1, anche se appare il campo indicato come ListField

Anche questo meccanismo, un po' più complesso degli altri, deve essere padroneggiato. Serve spessissimo.

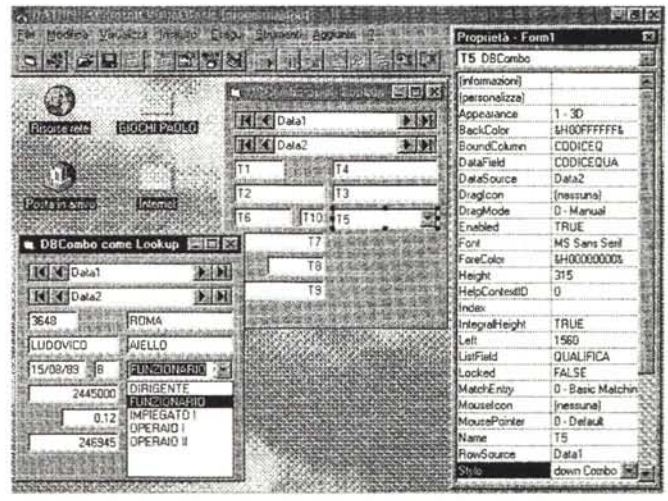
Sperimentando... s'impara

Il bello della programmazione Object Based sta nel fatto che si basa su pochi principi semplicissimi (come qualsiasi teoria filosofica di successo):

- esistono Oggetti, caratterizzati da proprie Proprietà,
- le Proprietà possono essere definite all'inizio, in fase di disegno dell'applicazione e possono essere modificate in fase di esecuzione del programma,
- esistono dei Metodi che servono proprio per modificare le Proprietà degli Oggetti,
- esistono vari tipi di Eventi, subiti da un Oggetto, che scatenano dei Metodi, che cambiano Proprietà di altri Oggetti.

Capita e padroneggiata questa semplice filosofia di base, non è difficile imparare ad usare i vari oggetti, basta scoprirne le proprietà (che sono sempre quelle che ci si aspetta di trovare per quel tipo di oggetto) che servono per le proprie

Figura 7 - Visual Basic 4.0 - Sfruttamento del DBCombo o del DBList. Nell'esercizio appena visto, DBCombo e DBGrid vengono usate solo per vedere i Dati, e quindi vengono utilizzate solo due o tre loro proprietà. Questi controlli possono essere utilizzati anche per modificare o per alimentare il contenuto di una Tabella. In tal caso occorre maneggiare altre proprietà. Qui vediamo come sia possibile usare un controllo DBCombo come Tabella di Lookup, dalla quale pescare un dato necessario per aggiornare una Tabella Principale.



necessità. È anche facile sperimentare le varie proprietà: basta creare un'applicazione che contenga l'oggetto da sperimentare ed un pulsante. All'evento click del pulsante si lega una semplice istruzione che cambia la proprietà in esame dell'oggetto... incriminato.

Due precisazioni: la prima è che spesso è più facile imparare il funzionamento di un Controllo sperimentandolo, che studiarlo dall'Help o dal Manuale. La seconda è che con Visual Basic è possibile creare delle applicazioni del tutto Windows compatibili. Quindi nel suo interno ci sono, basta andarli a cercare, tutti gli oggetti, gli eventi, i metodi che siamo abituati ad usare nelle altre applicazioni Windows, anche in quelle che ci sembrano più familiari.

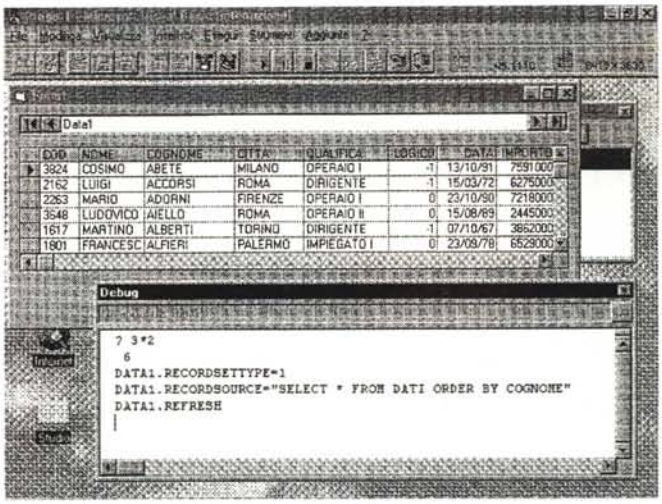


Figura 8 - Visual Basic 4.0 - SQL dalla finestra di Debug.
Uno dei sistemi più immediati per sperimentare l'effetto di certi Comandi, Istruzioni o Funzioni è quello che fa uso della Finestra di Debug, nella quale si possono scrivere espressioni valide che hanno un effetto immediato sugli oggetti della Form. Nel nostro caso abbiamo ridefinito la proprietà Record Source del DBGrid e abbiamo eseguito l'aggiornamento dei dati applicando il metodo Requery al Controllo.

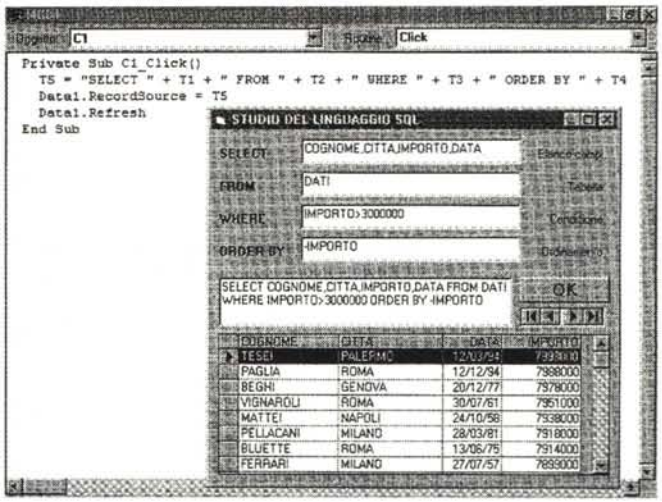


Figura 9 - Visual Basic 4.0 - Palestra di SQL.
Anche in questo esercizio sperimentiamo le possibilità degli Oggetti DB. In pratica costruiamo una espressione SQL impostandone i vari pezzetti in una serie di apposite caselle di testo. C'è un pulsante che manda in esecuzione l'istruzione SQL che viene usata come RecordSource per la DBGrid. Nei nostri esercizi utilizziamo sempre e solo una Tabella. Il discorso si complicherebbe un po' quando si lavora su più Tabelle, in quanto occorre utilizzare anche la clausola Join e le sue varianti, che serve a collegare tra di loro due tabelle.

SQL è ormai obbligatorio

Abbiamo detto che un DataControl può utilizzare tre tipi di RecordSource: la Tabella, il Dynaset e lo Snapshot. Nel caso della Tabella nessuna difficoltà per indicarne il nome. Nel caso di Dynaset oppure di Snapshot occorre impostare un'istruzione SQL.

In figura 8 e 9 vi proponiamo due modi per allenarvi sul linguaggio SQL. Il primo è... senza rete, in quanto si scrivono e si eseguono direttamente le istruzioni SQL.

In una Form inserite un DataControl, per il quale usate il nome di default, Data1, e un DBGrid, associata manco a dirlo al DataControl. Eseguite ed interrompete il programma. A questo punto potete aprire la finestra di Debug nella quale potete scrivere «a manina» le proprietà del DataControl e vedere immediatamente nella Grid sul Form il risultato. La seconda proposta è una palestra di SQL. In una serie di Caselle di Testo inserite porzioni di istruzioni SQL. Il pulsante OK esegue in sequenza tre istruzioni: quella che costruisce la SQL, quella che l'assegna come RecordSource al ControlData e quella che riesegue la Query SQL. Potete complicarlo a volontà.

Ma con un DataControl ci si collega solo ai dati di una Tabella ?

No. Un DataControl permette ben altre cose.

Come detto, un DataControl si può collegare a tre tipologie di RecordSet, indicate dalla proprietà RecordsetType:
- Table, una tabella a tutti gli effetti, ad esempio un file DBF, oppure una tabella presente in un'applicazione MDB,
- Dynaset, una tabella «virtuale» ottenuta con un'istruzione SQL, con vista diretta sui dati, che in questa maniera sono anche aggiornabili,
- Snapshot, una tabella «virtuale» ottenuta con un'istruzione SQL, ma come output di dati, che non sono quindi aggiornabili.

È chiaro che va curata l'«accoppiata» tra le due proprietà RecordsetType e RecordSource. Se l'applicazione serve solo per vedere i dati, è preferibile utilizzare dei Snapshot, più veloci da eseguire e facilmente adattabili alle differenti necessità. Se invece occorre gestire i dati si deve lavorare o con le Tabelle o con i Dynaset.

Tra i due ci sono numerose differenze. Un Dynaset va eseguito e quindi richiede, in caso di grossi volumi di dati, tempo, mentre l'accesso alle Tabelle non comporta perditempo. Inoltre, e questo aspetto è ancora più importante, esistono dei metodi che agiscono sulle Table e non sui Dynaset e viceversa. A seconda quindi del volume dei dati da trattare, delle modalità di accesso e di ricerca dei singoli record, si utilizzeranno Table e Dynaset. Qui di seguito vi proponiamo alcune possibili accoppiate, usando la tabella DATI.DBF e i suoi campi CODICE, COGNOME, CITTA, QUALIFICA, DATA, IMPORTO.

```
RecordsetType: Table
   RecordSource: DATI
RecordsetType: Dynaset
   RecordSource: SELECT * FROM DATI
RecordsetType: Dynaset
   RecordSource: SELECT * FROM DATI
ORDER BY COGNOME
RecordsetType: Dynaset
   RecordSource: SELECT * FROM DATI
WHERE CITTA="MILANO" ORDER BY CO-
GNOME
```

Inoltre è possibile, durante lo svolgimento del programma, ad esempio a richiesta, impostare la proprietà RecordSource. Ad esempio passare da:

```
   RecordSource: SELECT * FROM DATI
ORDER BY COGNOME
   a
   RecordSource: SELECT * FROM DATI
ORDER BY CODICE
```

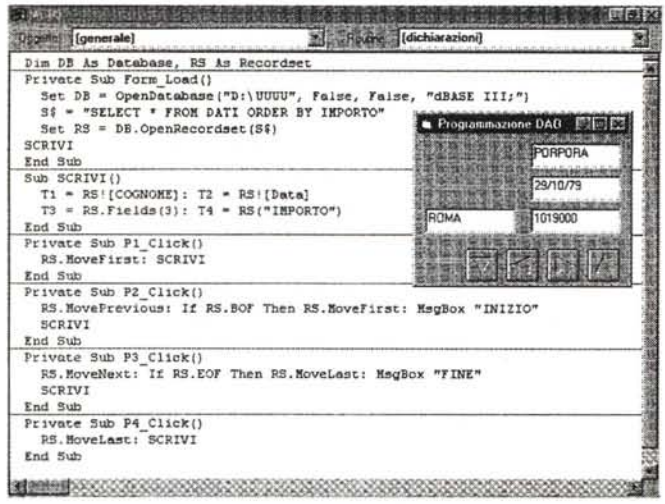


Figura 10 - Visual Basic 4.0 - Simulazione da programma del Control Data.
Come avrete capito il DataControl è utilissimo per accedere ai dati, ma non è indispensabile. Una delle sue caratteristiche più importanti è che può essere nascosto (proprietà Visible=False). Anche nascosto continua a funzionare. In ogni caso facendo un po' di programmazione tradizionale e sfruttando la tecnologia DAO, si possono ottenere le stesse funzionalità del DataControl, anzi molte di più. In pratica si può definire una variabile RecordSet, che è del tutto equivalente al RecordSet fornito dal DataControl.

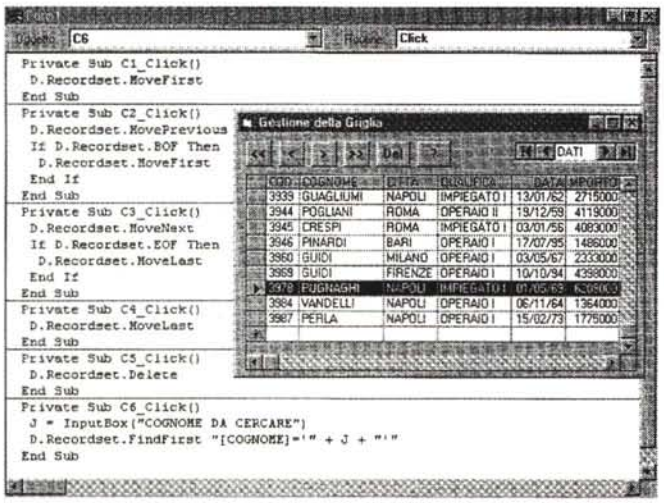


Figura 11 - Visual Basic 4.0 - Sfruttamento del DBGrid.
Il DBGrid è una specie di «fact totum». La sua proprietà principale, DataSource, è quella che serve per collegarla ad un DataControl. Ma siccome il DBGrid può servire per gestire i dati, lavorando direttamente nella vista tabellare e contemporaneamente su campi e record, le sue proprietà e i suoi metodi sono svariate decine. Ne citiamo e verifichiamo qualcuno, tra quelli fondamentali, nell'esercizio proposto: movimento, accodamento, cancellazione.

Spingendosi un po' più in avanti si può ipotizzare una procedura in cui è l'utente che esegue una serie di scelte, in base alle quali un miniprogramma costruisce una variabile A$ che contiene la nuova istruzione SQL, che poi viene passata al DataControl. In pratica le tre istruzioni sono:

A$="SELECT * FROM DATI WHERE CITTA='ROMA'" l'istruzione SQL, posta in una variabile
Data1.RecordsetType=Dynaset il RecordsetType, può essere un Dynaset
Data1.RecordSource=A$ definizione del RecordSet
Data1.Refresh ricalcolo del RecordSet

Vanno tenute presenti due cose. La prima è che una condizione WHERE fa uso di virgolette quando lavora con dei campi di tipo testuale, così come l'intera istruzione SQL va chiusa tra virgolette. Per non pasticciare si possono usare, anche se non è molto elegante, gli apostrofi per isolare la stringa, le virgolette per chiudere la frase SQL.

L'altra è la necessità, quando si cambia al volo la proprietà RecordSource, di sfruttare un metodo Refresh, che in pratica manda in esecuzione la query SQL. Se i dati sono tanti, il tempo necessario può essere sensibile. Cambiare l'origine dei record di un DataControl non è come fare 2 + 2.

Simulazione, con Istruzioni, del lavoro di un DataControl

Il DataControl fornisce un RecordSet, un insieme di Record maneggiabile in svariate maniere.

A questo punto occorre dire tre cose importantissime. La prima è che se si imposta a False la proprietà Visible di un DataControl, questo continua a funzionare... nell'ombra. La seconda è che esistono una serie di metodi che agiscono direttamente sul RecordSet. In altre parole è possibile creare un Form in cui c'è un DataControl invisibile ed in cui c'è un tasto Next, che applica il Metodo MoveNext al RecordSet definito dal DataControl. La terza cosa, ancora più importante delle prime due, è che un RecordSet può essere definito da programma senza dover usare per forza un DataControl.

Vi proponiamo un esercizio, in figura 10, nel quale viene simulato un DataControl in tutto e per tutto. Vi trovate tutte le istruzioni che servono per definire, da programma, un RecordSet e per simulare i classici quattro tasti di movimento.

Approfondiamo il DBGrid

Il DBGrid, di cui abbiamo parlato prima, deve essere collegato ad un DataControl tramite la sua proprietà RecordSource. Il DBGrid è il controllo più utile, in quanto permette di vedere e di gestire contemporaneamente i Record e i Campi del nostro RecordSet.

In figura 11 vediamo un campionario minimo di azioni sul RecordSet visto nel DBGrid. Abbiamo piazzato, in cima al DBGrid, sei pulsanti, quattro per eseguire i quattro movimenti classici sul RecordSet (Primo, Precedente, Successivo, Ultimo), il quinto per eseguire il Metodo Delete, e quindi per cancellare il Record corrente, e l'ultimo per eseguire una ricerca sul Cognome. Trattandosi di un RecordSet si può usare solo il metodo FindFirst e non il metodo Seek.

Insomma il nostro programmino fa un bel po' di cose, tutte però documentate dai listati.

Il formato MDB

All'inizio dell'articolo abbiamo detto che il formato Dati proprio del Visual Basic 4.0 è quello Access (i file MDB). In particolare VB4 ha due motori, uno per Access a 16 bit, utilizzabile da Access 2.0, uno per Access a 32 bit, utilizzabile da Access 95. I motori si chiamano Jet 2.5 e Jet 3.0, rispettivamente.

Un Database Access si concretizza in un unico file, suffisso MDB, onnicomprensivo, con Tabelle, regole interne alle tabelle, Relazioni, regole tra le tabelle. Comprende anche gli altri oggetti che servono per gestire e/o manipolare i dati, che risiedono esclusivamente nelle tabelle.

Un file MDB può contenere vari tipi di oggetti, ma quelli che interessano a Visual Basic sono solamente due:

le Tabelle, fatte di struttura e di dati,

le Relazioni, che legano le tabelle tra di loro.

Potrebbe interessare anche l'oggetto Query. Ma poiché una Query di Access è memorizzata, nel file MDB, come istruzione SQL, basta memorizzare, al limite in una variabile, la stessa istruzione e usarla, in uno dei modi visti prima, per definire un RecordSet.

Figura 12 - Visual Basic 4.0 - Una collezione completa di... messaggi di errore.
Nel capitoletto Sbagliando s'impara dell'articolo elenchiamo i principali messaggi di errore segnalati dal Visual Basic quando si compie qualche operazione sbagliata sui dati, nel caso si utilizzi un formato MDB. Alcuni messaggi vengono generati da errori sul tipo dei dati, altri da errori sulle regole di validità impostate per ciascun campo, altri ancora da errori commessi sulle chiavi, altri, infine, da errori sulle regole di integrità referenziale, che legano coppie di tabelle.

Figura 13 - Visual Basic 4.0 - Da Sequenziale a MDB, via programma.
Torniamo un attimo al file testuale visto prima. Proponiamo un programma che legge un File Testuale, lo stesso di prima, e lo scarica in una Tabella Access. Il passaggio non è diretto in quanto i vari campi in entrata sono tutti in formato testo, mentre in uscita, verso il formato MDB, vanno adattati alla tipologia del campo di destinazione.

Figura 14 - Visual Basic 4.0 - Con la programmazione DAO si legge la struttura.
Oltre a gestire i dati, con la programmazione DAO si può fare una vera e propria «radiografia» del Database, leggendo tutte le caratteristiche delle tabelle e dei campi delle tabelle. Nel nostro caso, come sappiamo, abbiamo una sola tabella e quindi leggiamo una sola tabella. Ricordiamo che la caratteristica principale della tecnologia DAO è quella di essere in comune con gli altri prodotti della Microsoft. Il codice che vedete può essere usato anche con Excel 95 e con Access 95.

```
Private Sub Form_Load()
  Set DB = OpenDatabase("D:\UUUU\DATI.MDB")
  Set RS = DB.OpenRecordset("NDAT")
End Sub
Private Sub P_Click()
  Open "D:\UUUU\DATI.TXT" For Input As #1
    Do While Not EOF(1)
      Input #1, A, B, C, D, E, F, G, H, I, J
      If F = "T" Then FF = 1 Else TT = 0
      GG = DateSerial(Mid(G, 7, 2), Mid(G, 5, 2), Mid(G, 1, 4))
      RS.AddNew
      RS("Cod") = A: T1 = A
      RS("Nome") = B: T2 = B
      RS("Cognome") = C: T3 = C
      RS("Citta") = D: T4 = D
      RS("Qualifica") = E: T5 = E
      RS("Logico") = FF:  T6 = FF
      RS("Data") = GG: T7 = GG
      RS("Importo") = H: T8 = H
      RS("Perc") = I: T9 = I
      RS("Calcolo") = J: T10 = J
      RS.Update
    Loop
  Close #1
End Sub
```

Da File Sequenziale a File MDB

3987	OPERAIO I
ULISSE	1
PERLA	26/06/20
NAPOLI	1775000
	0,19
Esegui	179275

```
Dim DB As Database
Dim RS As Recordset
Private Sub Command1_Click()
  G.Row = 0: G.Col = 0: G = "Database"
  G.Col = 1: G = DB.Name
  G.Row = 1: G.Col = 0: G = "Tabella"
  G.Col = 1: G = RS.Name
  G.Row = 2: G.Col = 0: G = "Num.Rec."
  G.Col = 1: G = RS.RecordCount
  k = 1
  G.Col = 0: G.Row = 3: G = "Campo"
  G.Col = 1: G.Row = 3: G = "Tipo"
  G.Col = 2: G.Row = 3: G = "Lung."
  For k = 0 To RS.Fields.Count - 1
    G.Row = k + 4
    G.Col = 0: G = RS.Fields(k).Name
    G.Col = 1: G = RS.Fields(k).Type
    G.Col = 2: G = RS.Fields(k).Size
  Next k
End Sub
Private Sub Form_Load()
  Set DB = OpenDatabase("D:\UUUU", False, False, "dBASE III;")
  S$ = "DATI"
  Set RS = DB.OpenRecordset(S$)
End Sub
```

Internal DAO — Esegui

Database:	D:\UUUU	
Tabella	DATI	
Num.Rec.	500	
Campo	Tipo	Lung.
COD	10	4
NOME	10	15
COGNOME	10	15
CITTA	10	10
QUALIFICA	10	12
LOGICO	1	1
DATA	8	8
IMPORTO	7	8
PERC	7	8
CALCOLO	7	8

La struttura di una Tabella MDB è ricca, contiene una decina di specifiche che servono anche, in fase di gestione, al controllo della validità del dato.

Ad una Relazione tra due tabelle si possono assegnare alcune specifiche, la più importante delle quali è quella che riguarda il controllo della Integrità Referenziale.

In altre parole un Database in formato MDB contiene una serie di sistemi di controllo, definiti a livello di struttura, che valgono sempre, anche quando vi si accede da un Form Visual Basic.

Un Database in formato dBase III, invece, è fatto di tanti file separati e dispone di un minor numero di controlli. Se servono, ed in genere servono, vanno realizzati da programma.

In figura 12 mostriamo un piccolo campionario di messaggi di Errore, che appaiono quando, da Visual Basic, cerchiamo di fare qualche cosa che non va in un Database MDB. In particolare quando scriviamo un dato di tipo sbagliato, un dato troppo lungo rispetto alla dimensione del campo, un dato che non rispetta le regole di integrità, quando inseriamo un nuovo record e un suo campo chiave che già esiste. Ribadiamo il fatto che di tutti questi controlli si occupa la struttura MDB e non se ne deve occupare il programmatore.

Conclusioni

Prima delle conclusioni due esercizietti utili.

Nel primo, in figura 13, vi proponiamo un programma che legge, in maniera sequenziale, un file testuale, e, record per record, lo scarica in una struttura Access preconfezionata (il programma scarica i dati non la struttura). Via via che scorre i record li visualizza anche nel Form. Il secondo, in figura 14, è un programma che apre, come RecordSet, la nostra Tabella Dati e ne legge alcune caratteristiche, come il nome dei campi, il tipo (VB usa una codifica numerica), la lunghezza, ecc. I risultati di questa lettura vengono scaricati in un controllo Griglia.

Il concetto chiave attorno a cui ruota il sistema di accesso ai dati di VB4 è il RecordSet. Per RecordSet si intende l'insieme dei dati, siano essi una Tabella vera e propria, oppure un Dynaset creato dinamicamente per l'occasione.

Un RecordSet può coincidere con un DataControl. È possibile bypassare il DataControl definendo il RecordSet mediante istruzioni di programmazione. Le istruzioni sono sempre le stesse e quindi è consigliabile impararle a memoria.

Un RecordSet può essere visualizzato e/o gestito attraverso altri Controlli, semplici, come le Text Box, o complessi, come i DBGrid.

Un RecordSet ha una serie di proprietà secche, ad esempio il Numero dei Record che contiene, ha una serie di «collezioni» di proprietà, ad esempio i nomi dei suoi Campi.

Le tecniche di utilizzo, con istruzioni di programmazione, dei RecordSet, ma anche dell'intero Database o dei singoli elementi del RecordSet, rientrano nella tecnologia DAO, Data Access Object, che sta accomunando i vari prodotti della Microsoft.

Abbiamo, un paio di numeri fa, visto alcuni di questi stessi esercizi realizzati con Excel e con il suo linguaggio VBA. Le istruzioni di accesso ai dati sono le stesse.

MC

NEW

ITALSEL presenta:

CD-ROM

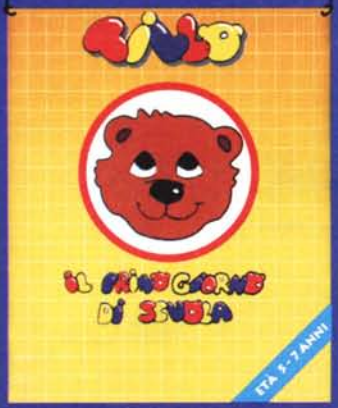
in italiano

MUSEI E MONUMENTI IN ITALIA

Oltre 2600 musei e 2000 monumenti di tutta Italia: Castelli, Ville, Aree Archeologiche, Murales, Catacombe e Grotte naturali.
Una capillare raccolta di informazioni per orientarsi tra le ricchezze artistiche, archeologiche, storiche, scientifiche e naturali.
Oltre 1500 splendide fotografie a colori.
Varie possibilità di ricerca: per località, per regione, per genere e per tipologia.
Ricerche incrociate, 32 categorie, 2500 parole chiave.
In italiano e in inglese. Versione Windows.

SCIARE IN ITALIA

Un Cd-Rom per tutti gli amanti dello sci!
Tutte le località, gli impianti e le scuole.
Lunghezza e difficoltà delle piste, indirizzi utili, immagini.
Ipertesti con collegamenti tra le località.
Un archivio enorme con informazioni utili e curiose per chi ama lo sci.
In italiano. Versione Windows.



TILLO IL PRIMO GIORNO DI SCUOLA

Il primo di una serie di racconti interattivi e multimediali che hanno per protagonista l'orsetto Tillo.
Tillo è un orsetto buffo e simpatico che vive in una graziosa casetta con i suoi genitori.
Il bambino seguirà l'orsetto Tillo durante tutte le fasi della sua giornata, sempre in compagnia di oggetti animati e ambienti multimediali.
Inoltre, in ogni scena, sono presenti alcuni oggetti che, cliccati, danno la possibilità di aprire delle videate su semplici esercizi didattici.
La realizzazione di Tillo è stata seguita da un'èquipe di educatori e pedagogisti.
Per bambini delle scuole elementari. Scatola gioco.
In italiano, inglese, francesce, tedesco e spagnolo. Versione Mac e Windows.

A PARTIRE DA
L. 34.000

.. è un' idea
HEAD ON

Prodotto e distribuito da: ITALSEL S.r.l. tel. 051-320409 fax 051-320449
e-mail italsel@italsel.nettuno.it http://www.nettuno.it/italsel