di Francesco Petroni

Uso del DB come linguaggio di programmazione

Quarta parte

Nelle scorse puntate del corso abbiamo visto come si costruisce un archivio, definendone prima la struttura e esaminando poi i vari comandi con i quali si inseriscono, modificano e cancellano i dati, e abbiamo visto le due organizzazioni sotto le quali può essere manipolato un archivio, quella fisica, realizzata inserendo in maniera sequenziale uno dopo l'altro i record, e quella logica, che permette di svincolarsi del tutto dalla rigidità dell'ordine sequenziale.

Nella quarta puntata abbandoniamo questi concetti organizzativi e passiamo ad esaminare gli strumenti che il DB mette a disposizione per manipolare i file, assecondandone la organizzazione sotto la quale sono visti.

Vedremo anche come il DB permette vari livelli di utilizzazione che vanno dall'uso elementare in comandi diretti ad un uso molto sofisticato come generatore di software applicativo.

È evidente che una utilizzazione in comandi diretti può essere limitata ad applicazioni «strettamente personali», e di volumi medio piccoli. Ma l'utente finale, anche quello principiante, ben presto sentirà l'esigenza di memorizzare in una sequenza operativa i comandi che permettono di eseguire determinate operazioni.

Il primo livello di programmazione è proprio quello di trascrivere, in un programma eseguibile, una serie di comandi, evitando in tal modo di doverli digitare direttamente.

*Inoltre il DB tra i comandi diretti (utilizzabili comodamente con la struttura di assistenza ASSIST) e la programmazione tradizionale dispone di strumenti intermedi adatti ad un utilizzo più spinto, ma ancora non sofisticato. Tali strutture sono il Formato, il Report, il Label, e anche il file tipo *.Mem.*

La struttura FORMAT

Il Formato è una struttura che si pone tra l'utente e il file, per cui tutti i comandi di manipolazione del File (come ad es. APPEND, EDIT ecc.) passano attraverso questa struttura. Si scrive come un programma e può contenere solo istruzioni di visualizzazione e di input (ovvero SAY, GET).

La procedura per operare accodamenti in modo diretto è:

```
USE <nome file>
APPEND (oppure EDIT, DISPLAY, ecc.)
```

Se si dispone della struttura formato la sequenza operativa diventa:

```
USE <nome file>
SET FORMAT TO <nome formato>
APPEND (oppure EDIT, ecc)
```

Quando si è terminato il lavoro si disabilita la maschera con:

```
SET FORMAT
```

Il vantaggio rispetto al lavoro in comandi diretti sta nel fatto che è possibile comporre in modalità Full Screen la Maschera, e che si è del tutto liberi di impaginare scritte, spiegazioni, ecc. Mentre l'APPEND in comandi crea una maschera con solo il nome del campo (che se non è mnemonico in genere non permette il facile riconoscimento del significato) e con tutti i campi snocciolati uno dopo l'altro anche in successive videate.

Lo svantaggio rispetto alla forma più evoluta rappresentata dal tradizionale programma di gestione archivio sta nel fatto che il formato non viene eseguito, ma viene usato e quindi non accetta comandi attivi che per esempio permettano di eseguire routine di controllo dei dati.

In figura 1 e 2 vediamo la videata generata dal comando APPEND in assenza e in presenza di un Formato, di cui in figura 3 vediamo il listato.

```
Record n.     257
CODC          3242
RAGR          A
DITT          RASCEL
DTRG          01/08/43
INDI          VIA TORINO, 66
CAP           00099
CITT          ROMA
PROV          NN
SCNT           0
SLDO           3190600
```

Figura 1 - Videata generata dal Comando APPEND. Una volta creato un Archivio il modo più semplice per immettere dati è quello consentito dal comando APPEND, che visualizza una maschera vuota in cui i campi sono messi uno sopra l'altro con a sinistra il nome imposto al campo in fase di Creazione.

In ambedue i casi il passaggio da un record all'altro avviene tramite i tasti PgDn e PgUp, se si raggiunge il BOF () inizio file o EOF () fine file, si esce dalla situazione di maschera. L'ordine con il quale scorrono i record è quello imposto dall'indice in uso, o se non c'è indice, l'ordine è quello di immissione e gestito con il RECNO ().

Per uscire dalla maschera (sia essa quella di default o quella generata di tipo *.FMT) esistono due modalità: cntr END, è l'uscita con memorizzazione, ESC è l'Abort. Queste modalità valgono in altre situazioni operative, come ad esempio in fase di Editor di un programma.

La struttura REPORT

Il REPORT è la struttura con la quale è possibile predisporre tabulati con dati provenienti da uno o più archivi. È un formato di stampa utilizzabile tramite un comando molto sofisticato che ne permette numerose varianti. Ha una logica di riga per cui va stabilita la riga o l'insieme di righe campione. Il comando per realizzare il Report è:

```
USE <nome file>
CREATE REPORT <nome del report>
```

In questa situazione sono accettate solo espressioni che comprendono i nomi dei campi del file. Una utilizzazione più avanzata è:

```
USE <nome file 1>
SELECT 2
USE <nome file 2>
SELECT 1
SET RELATION TO
<nome campo file 1> INTO B
X=1.2
CREATE REPORT <nome del report>
```

In questa situazione sono accettati non solo i campi del primo archivio ma anche i campi del secondo archivio (relazionato al primo) e espressioni che contengono la variabile X. Al contrario una espressione che contenesse una variabile Y, non definita prima, non verrebbe accettata.

La stessa situazione di file aperti e di variabili definite, realizzata al momento della creazione del REPORT deve essere creata di nuovo al momento dell'uso del report, pena un errore del tipo variabile o campo inesistente.

Esiste la possibilità di manipolare il formato delle espressioni e la loro larghezza. Ad esempio se si deve visualizzare un campo largo 40, si può imporre una larghezza in uscita di 20 caratteri e il DB si occuperà di scrivere le due parti in due righe successive. Questo vale anche se si vuole, per problemi di spazio, comporre i campi uno sull'altro.

Il REPORT quindi permette anche un uso evoluto, sia in termini di contenuto che di estetica, che risulta spesso utile anche in ambienti programmativi evoluti.

In figura 5 e 6 vediamo due esempi di REPORT ottenuti partendo dallo stesso tracciato record di figura 4. Nel primo vengono semplicemente messi in riga, uno dopo l'altro, i vari campi.

Nel secondo i campi vengono aggregati in modo da occupare più righe, e inoltre viene visualizzato il dato ETÀ ottenuto per mezzo del calcolo della differenza tra l'anno della data di sistema e la data di nascita. I campi quindi si possono comporre o se ne può elaborare il contenuto con le normali funzioni di stringa, matematiche, ecc. come si fa con qualsiasi variabile.

Sintassi del comando REPORT

Il file *.FRM dà il modo con cui visualizzare, ma il cosa visualizzare può essere definito tramite altre componenti della sintassi del comando REPORT. Inoltre la sintassi accetta altre specifiche riguardanti il device di output, che può essere il Video, la Stampante oppure un File.

```
REPORT FROM <nome file>
       HEADING "Provincia di Milano"
       FOR PROV = "MI"
       NOEJECT
       TO PRINT (oppure TO FILE <nome file>)
```

In sostanza si può utilizzare lo stesso formato anche per stampe di contenuto differente, ottenuto variando la condizione FOR, e passando un titolo al volo se ne differenzia anche l'intestazione. Questo rende facilmente «parametrizzabili» i programmi di gestione delle stampe.

Totali e sottototali automatici nel REPORT

Il report rispetta la organizzazione in uso per il file, non ne crea una propria. Per cui l'ordine è esterno al formato. Per utilizzare le opzioni di totali e sottototali (vedi ultimo numero di MC) occorre che l'archivio sia già or-

```
---------- ---------- ----------

Numero Record    257
Codice           3242
Ditta            RASCEL
Data Registr.    01/08/43
Indirizzo        VIA TORINO, 66          Cod.Postale  00099
Citta            ROMA                    Provincia    NN
Importo           3,190,600              Sconto        0

---------- ---------- ----------
Per Muovervi sull' Archivio Usate i tasti        PgUp e PgDn
Per Finire Cntr.END
```

*Figura 2 - Videata generata dal Comando APPEND in presenza di un Formato. Una forma più sofisticata di manipolazione di Archivio si ottiene utilizzando un File tipo *.FMT, che fa da interfaccia tra i record e l'utente. È possibile far scorrere l'archivio con i tasti PgUp e PgDn e modificarne il contenuto utilizzando i tasti freccia per muoversi tra i campi.*

```
* 16/01/86 - MC microcomputer
* *.FMT    - maschera archivio
*
@  1,5  say "---------- ---------- ----------"
@  3,5  say "Numero Record  "+str(recno(),5)
@  5,5  say "Codice          " get CODC pict "####"
@  7,5  say "Ditta           " get DITT pict "!!!!!!!!!!!!!!!"
@  9,5  say "Data Registr.   " get DTRG
@ 11,5  say "Indirizzo       " get INDI pict "!!!!!!!!!!!!!!!!!!!!!!!!!!!"
@ 11,46 say "Cod.Postale " get CAP  pict "#####"
@ 13,5  say "Citta            " get CITT pict "!!!!!!!!!!!!!!!"
@ 13,46 say "Provincia    " get PROV pict "!!"
@ 15,5  say "Importo          " get sldo pict "##,###,###"
@ 15,46 say "Sconto       " get scnt pict "##"
@ 17,5  say "---------- ---------- ----------"
@ 18,5  say "Per Muovervi sull' Archivio Usate i tasti        PgUp e PgDn"
@ 19,5  say "Per Finire Cntr.END"
*
```

*Figura 3 - File di Formato - Estensione *.FMT. Il file *.FMT si realizza con la stessa modalità con la quale si realizza il file *.PRG, ovvero con il comando MODIFY COMMAND <nome file>. Nel file *.FMT possono essere inseriti solo comandi di visualizzazione @ <coord v> <coord. o> SAY <espress.> <picture> di input @ <coord. v> <coord. o> GET <campo> <picture>.*

```
LIST STRU TO PRIN
Struttura del database: A:arcsta.dbf
Totale record inseriti:        11
Ultima revisione      : 01/01/80
Campo  Nome campo  Tipo campo     Dim    Dec
    1  CODICE      Carattere       5
    2  NOME        Carattere      12
    3  COGNOME     Carattere      14
    4  TITOLO      Carattere       5
    5  INDIRIZZO   Carattere      24
    6  CAP         Carattere       5
    7  CITTA       Carattere      15
    8  PROVINCIA   Carattere       2
    9  TELEFONO    Carattere      11
   10  SEX         Carattere       1
   11  DATANASC    Data            8
   12  PROVNASC    Carattere       2
   13  STIPENDIO   Numerico        8
Totale:                          113
```

Figura 4 - Tracciato Record del File Usato per i Report. In fase di creazione del Report va indicato ciascun elemento del prospetto, e ogni elemento può essere un singolo campo o un'espressione (matematica, di stringa, di formato, ecc.) in cui entra il campo. Durante la creazione del Report viene mostrato in alto il tracciato record e le caratteristiche di ciascun campo.

dinato. Ad esempio supponiamo di volere i totali per Regione e, all'interno di ciascuna Regione, per Provincia, di alcuni dati numerici del nostro File.

Il CREATE REPORT chiede se vogliamo totali e sottototali, ma l'organizzazione la dobbiamo fornire noi, ad esempio:

```
SET INDEX ON <campo regione> +
<campo provi.> TO <nome indice>
REPORT FORMAT <nome report>
```

Solo organizzando così l'ordine del file il calcolo dei totali e dei sottototali funziona.

La struttura LABEL

È struttura analoga al report che permette di stampare dati sotto forma di etichette (Indirizzi, Biblioteche, Magazzino, ecc.) confezionandone il formato partendo direttamente dalla struttura del file.

Valgono le stesse regole sintattiche del comando REPORT, la differenza sta nel fatto che è possibile stampare più etichette sulla stessa riga.

In figura 7 vediamo un esempio di etichette tratte dallo stesso archivio utilizzato per la realizzazione dei Report.

La struttura MEMO

Una funzionalità molto comoda è costituita dal comando Macro, associato al carattere &. Ovvero è possibile assegnare ad una variabile un certo comando e poi eseguirlo senza doverlo digitare ma utilizzando la variabile. Poiché tali variabili possono essere composte è possibile crearsi un proprio insieme di comandi personali, più maneggevoli di quelli «tradizionali», e attribuirli ad un insieme di variabili, che a loro volta possono essere memorizzate in file tipo *.MEM richiamabile con un unico comando quando serve. Esempio per memorizzare i comandi personali:

```
AR1="USE archivio INDEX indice1, indi-
ce2"
SI1="SET INDEX TO indice1, indice2"
SI2="SET INDEX TO indice 2, indice1"
SAVE TO vrarch1.mem
```

Per richiamarli:

```
RESTORE FROM vrarch1.mem
&AR1
LIST
&SI2
LIST
&SI1
LIST
```

Questa opportunità, in ambiti applicativi dove occorra confezionare comandi complessi, prelevando «porzioni» del comando via programma, risulta spesso risolutiva.

Ad esempio supponiamo di voler eseguire una stampa di un certo archivio limitatamente ai dati di una certa provincia, possiamo comporre il comando utilizzando due variabili, una cui corrisponde la parte fissa, e l'altra la parte che vogliamo far variare ad esempio via input.

```
VAR1="REPORT FROM STAMPA PER
PROV="
VAR3="HEADING SIGLA DELLA PROVIN-
CIA"
accept "Immetti Sigla Provincia" to var2
```

Per eseguire il report basterà scrivere:

```
&VAR1 "&var2" &VAR3 "&var2"
```

E in funzione della sigla immessa via input il comando sarà lo stesso, ma cambierà contenuto e titolo del prospetto prodotto.

Programmazione in Data Base editor

Il DB, sia il 2 che il 3, possiedono un proprio editor con il quale è possibile «scrivere» i programmi. Il risultato è un file di tipo TXT (cui il DB fa assumere un'estensione *PRG) e che è quindi leggibile dal DOS e soprattutto editabile con qualsiasi Word Processor. Per cui ad esempio si può scrivere con il WordStar un programma DB, o se è stato scritto in editor DB può essere letto da DOS e da WordStar.

A proposito di WordStar l'editor del DB ne segue parecchie regole sintattiche (quelle di cancellazione, spostamento, ecc.) per cui l'esperto in WS non trova nessuna difficoltà a lavorare sotto editor DB.

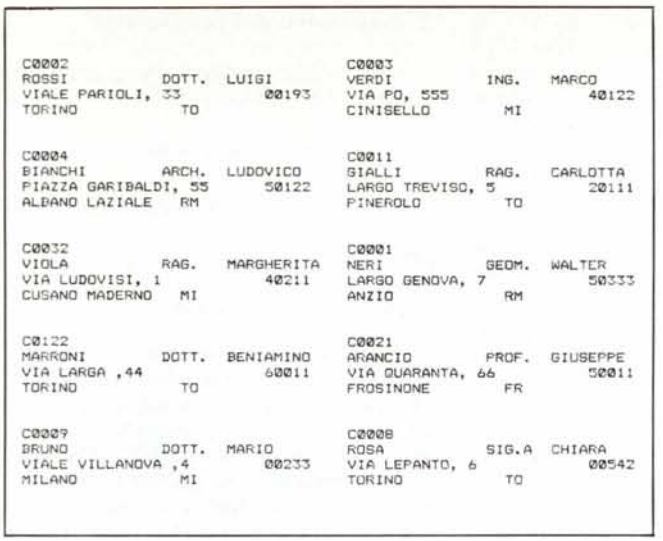
Esistono però alcune limitazioni che rendono il lavoro sotto editor DB più gravoso.

La prima è quella che non esiste la possibilità di copiare e spostare blocchi all'interno del file.

```
Pagina n.    1
01/01/80

CODICE COGNOME     TIT.  NOME       INDIRIZZO          CAP   CITTA          PR. TELEFONO    S  NATO IL  PR. STIPENDIO

C0002  ROSSI       DOTT. LUIGI      VIALE PARIOLI, 33  00193 TORINO         TO  56444       M  09/06/58 RM   1200000
C0003  VERDI       ING.  MARCO      VIA PO, 555        40122 CINISELLO      MI  556666      M  20/03/64 PA   1250000
C0004  BIANCHI     ARCH. LUDOVICO   PIAZZA GARIBALDI, 55 50122 ALBANO LAZIALE RM  23424       M  29/05/51 LI   2190000
C0011  GIALLI      RAG.  CARLOTTA   LARGO TREVISO, 5   20111 PINEROLO       TO  465665      F  21/05/45 LI   1875000
C0032  VIOLA       RAG.  MARGHERITA VIA LUDOVISI, 1    40211 CUSANO MADERNO MI  3345343     F  05/10/54 RM   1090000
C0081  NERI        GEOM. WALTER     LARGO GENOVA, 7    50333 ANZIO          RM  4434344     M  11/01/55 RM   1100000
C0122  MARRONI     DOTT. BENIAMINO  VIA LARGA ,44      60011 TORINO         TO  3434545     M  29/06/57 RM   1295000
C0021  ARANCIO     PROF. GIUSEPPE   VIA QUARANTA, 66   50011 FROSINONE      FR  6546456     M  24/09/58 LI   1450000
C0009  BRUNO       DOTT. MARIO      VIALE VILLANOVA ,4 00233 MILANO         MI  4343434     M  28/12/62 TO   1345000
C0008  ROSA        SIG.A CHIARA     VIA LEPANTO, 6     00542 TORINO         TO  454222      F  25/03/60 MI   1456000
C0033  AZZURRI     SIG.  ALDO       VIA GUERRIERI, 6   33321 ROMA           RM  3213443     M  15/02/56 MI   2009000
```

```
Pagina n.    1
01/01/80

NOMINATIVO          INDIRIZZO                    TELEFONO    ETA'  STIPENDIO

DOTT. LUIGI         VIALE PARIOLI, 33
ROSSI               00193 TORINO           TO    56444       22     1200000
ING.  MARCO         VIA PO, 555
VERDI               40122 CINISELLO        MI    556666      16     1250000
ARCH. LUDOVICO      PIAZZA GARIBALDI, 55
BIANCHI             50122 ALBANO LAZIALE   RM    23424       29     2190000
RAG.  CARLOTTA      LARGO TREVISO, 5
GIALLI              20111 PINEROLO         TO    465665      35     1875000
RAG.  MARGHERITA    VIA LUDOVISI, 1
VIOLA               40211 CUSANO MADERNO   MI    3345343     26     1090000
GEOM. WALTER        LARGO GENOVA, 7
NERI                50333 ANZIO            RM    4434344     25     1100000
DOTT. BENIAMINO     VIA LARGA ,44
MARRONI             60011 TORINO           TO    3434545     23     1295000
PROF. GIUSEPPE      VIA QUARANTA, 66
ARANCIO             50011 FROSINONE        FR    6546456     22     1450000
DOTT. MARIO         VIALE VILLANOVA ,4
BRUNO               00233 MILANO           MI    4343434     18     1345000
SIG.A CHIARA        VIA LEPANTO, 6
ROSA                00542 TORINO           TO    454222      20     1456000
SIG.  ALDO          VIA GUERRIERI, 6
AZZURRI             33321 ROMA             RM    3213443     24     2009000
```

Figure 5 e 6 - Due Report ottenuti dallo stesso tracciato Record. Come si vede dalle due figure a parità di dati base, la forma esteriore del prospetto può essere la più varia in quanto è possibile in fase di creazione comporre i vari campi anche in espressioni complesse.



```
C0002                                   C0003
ROSSI           DOTT.  LUIGI            VERDI           ING.   MARCO
VIALE PARIOLI, 33          00193        VIA PO, 555                 40122
TORINO           TO                     CINISELLO        MI

C0004                                   C0011
BIANCHI         ARCH.  LUDOVICO         GIALLI          RAG.   CARLOTTA
PIAZZA GARIBALDI, 55       50122        LARGO TREVISO, 5            20111
ALBANO LAZIALE   RM                     PINEROLO         TO

C0032                                   C0001
VIOLA           RAG.   MARGHERITA       NERI            GEOM.  WALTER
VIA LUDOVISI, 1            40211        LARGO GENOVA, 7             50333
CUSANO MADERNO   MI                     ANZIO            RM

C0122                                   C0021
MARRONI         DOTT.  BENIAMINO        ARANCIO         PROF.  GIUSEPPE
VIA LARGA ,44              60011        VIA QUARANTA, 66            50011
TORINO           TO                     FROSINONE        FR

C0009                                   C0008
BRUNO           DOTT.  MARIO            ROSA            SIG.A  CHIARA
VIALE VILLANOVA ,4         00233        VIA LEPANTO, 6              00542
MILANO           MI                     TORINO           TO
```

Figura 7 - Esempio di Output Permesso dalla Struttura Label. Da un archivio è possibile realizzare un output di tipo «etichette». La carta in modulo continuo con etichette autoadesive ormai è molto diffusa e le applicazioni possibili (indirizzi, biblioteche, discoteche, magazzino, ecc.) anche in campo personale, sono moltissime.

Figura 8

```
Database selezionato:
Area di lavoro:  1, database attivo: C:MGDAT.dbf, alias: MGDAT
    File indice: C:MGND1.ndx, chiave: mcli+mtab+mcst+mprg
    File indice: C:MGND3.ndx, chiave: mtab+mcst
    File indice: C:MGND2.ndx, chiave: mtab+mcli+mcst+mprg
    File indice: C:MGND4.ndx, chiave: mcst+mtab
    File indice: C:MGND5.ndx, chiave: mcli+mtab+mprg+dtoc(mdeq)
    In relazione con: TTDAT
    Relazione: MTAB

Area di lavoro:  2, database attivo: C:CLDAT.dbf, alias: CLDAT
    File indice: C:CLND2.ndx, chiave: cnom
    File indice: C:CLND1.ndx, chiave: cdos

Area di lavoro:  3, database attivo: C:TTDAT.dbf, alias: TTDAT
    File indice: C:TTND1.ndx, chiave: tabi
    File indice: C:TTND2.ndx, chiave: tnom
```

Figura 8 - Esempio del Comando Display Status. È un comando di utilità che permette di sapere quali file dati e indice siano aperti (file Primario, Secondario, ecc.), quali di questi siano in uso al momento, e quale Relazione sia attiva. È importante sapere in ogni momento quali file siano attivi.

```
LINEA       pubb  C  "----------------------------------------
                     ----------"
TOTALE      pubb  N   234567890  ( 234567890.00000000)
ARCHIVIO    pubb  C  "clienti"
DATA        pubb  D  20/01/86
CONTATORE   pubb  N          22  (         22.00000000)
COGN        pubb  C  "fanfaroni"
FLAG        pubb  L  .T.
COGNM       pubb  C  "FANFARONI"
MESI        pubb  C  "GENFEBMARAPLMAGGIULUGAGOSETOTTNOVDIC"
LUNGHEZZA   pubb  N          36  (         36.00000000)
   10 variabili definite,      157 byte utilizzati.
```

Figura 9 - Esempio del Comando Display Memory. Serve per conoscere lo stato delle variabili, ovvero quante siano quelle definite al momento (il massimo è di 64 in DB II e 256 in DB III) e quanta memoria occupano (il massimo è di 1.500 byte in DB II ed è settabile in DB III).

La seconda è quella che pur essendo possibile l'operazione di copia passando per un file esterno (questo vale solo per DB III) il limite massimo del file rimane di soli 4.000 byte e questo rende le operazioni di taglio e cucito molto pericolose se sono al limite di tali dimensioni.

Altra funzionalità mancante, ma necessaria in un editor serio è la ricerca/sostituzione che rende gravoso il lavoro di manutenzione dei programmi.

Tali difetti diventano evidenti solo quando si fa un uso «pesante» del DB, e tale strumento diventa produttivo solo quando è usato in maniera pesante.

Per entrare in Editor il comando è:

```
MODIFY COMMAND <nome del file>
```

Se già esiste un programma con quel nome ne appare il listato, altrimenti appare la pagina vuota per il nuovo listato. Per quanto riguarda l'estensione quella di default è la *.PRG, per cui se si vuol scrivere un file *.FMT bisogna specificarne l'estensione.

I REPORT e le LABEL hanno una loro sintassi, strettamente legata al file in uso, e producono file *.FRM e *.LBL non modificabili se non attraverso la loro sintassi specifica.

Discorso a parte per il DB II che permette solo il Report e che produce un file «leggibile» e quindi modificabile con altri linguaggi.

```
CREATE REPORT   <nome report>
CREATE LABEL    <nome file>
```

oppure

```
MODIFY REPORT   <nome file>
MODIFY LABEL    <nome file>
```

tali operazioni sono possibili solo se è in uso il file dal quale Report e Label prelevano i dati.

Per cancellare un file, di qualsiasi tipo la sintassi è:

```
DELETE FILE <nome file>
```

È obbligatoria l'estensione, e se si tratta di file Dati o Indici, non debbono essere in uso.

Per copiare e per rinominare un file di qualsiasi tipo:

```
COPY FILE <nome file> TO <nome file>
RENAME <nome 1> TO <nome 2>
```

Tutti questi comandi permettono ovviamente anche l'indicazione del drive di partenza e/o di arrivo.

I comandi ora citati duplicano i corrispondenti comandi DOS, ma sono più rigidi, in quanto chiedono sempre conferma per operazioni di tipo «pericoloso».

Tralasciamo di trascrivere i comandi di editor in quanto interessano solo chi lavora col DB e quindi ha comunque la possibilità di impararli e sperimentarli direttamente.

La programmazione in DB

Un programma DB accetta tutti comandi usabili in modo diretto, e in più permette l'uso di altri comandi che hanno senso solo se inseriti in un programma.

Non sono però accettati i comandi di DATA DEFINITION ovvero quelli per creare Strutture oppure Report oppure Formati che vanno creati preventivamente con le procedure specifiche. È viceversa possibile da programma creare e distruggere archivi indice, copiare strutture (partendo da una già esistente), ecc.

Molti dei comandi diretti, pur avendo senso solo se usati in tale modalità, possono essere utili in fase di debug dei programmi. È il caso in particolare dei comandi di utilità che permettono di avere il «polso» della situazione. Ad esempio:

```
DISPLAY (oppure LIST) MEMORY
DISPLAY (oppure LIST) STATUS
```

Per sapere in che situazione di file dati e indici e di relazioni attive si sta lavorando, oppure per sapere il tipo e il valore delle variabili in uso in un certo momento (vedi esempi in figg. 8 e 9).

Nel caso di lavoro con più archivi solo uno tra quelli aperti risulta in uso, per cui per passare ad un altro occorre eseguire il comando SELECT <numero file>. Se ad un file Dati sono associati più file indice, prevale il primo richiamato (come ordine imposto al file) per cui se ad esempio se ne vogliono usare due alternativamente, ma si vuole che siano sempre aggiornati tutti e due, occorre invertirne l'ordine.

Comandi specifici di programmazione

I comandi utilizzabili solo in programmazione sono:

— NOTE oppure * che permette di inserire un commento nel listato;

— TEXT e ENDTEXT, che permettono di «scrivere» in maniera diretta il contenuto di videate.

```
DO WHILE <condizione>
...
ENDDO
```

Con il quale si costruisce un loop dal quale si esce solo quando si verifica la condizione, oppure con una uscita Forzata (ottenuta tramite il comando EXIT).

```
IF <condizione>
   ...
   ELSE
   ...
   ENDI
```

Scelta tra una condizione e il suo opposto

```
   DO CASE
   CASE <condizione 1>
   ...
   CASE <condizione 2>
   ...
   CASE <condizione n>
   ...
   (opzionale OTHERWISE)
ENDCASE
```

Scelta tra svariate condizioni e (opzionale) condizione residua, mutuamente escludentesi.

La possibilità di «complicare» le condizioni con strutture di AND e OR e la possibilità di innestare una dentro l'altra più serie di comandi consentono di risolvere qualsiasi situazione logica all'interno del programma.

Nel numero scorso ne abbiamo visto un esempio applicativo in un programma di gestione archivio.

Non essendoci assolutamente istruzioni di salto, né semplice né condizionato, occorre «per forza» risolvere il programma a livello di flow, e questo per i programmatori «disordinati» abituati a risolvere i problemi di programmazione con istruzioni del tipo IF <condizione> THEN GOTO <indirizzo> risulterà abbastanza ostico.

L'ambiente è al contrario ideale per i puristi della programmazione, che magari lavorano preparando i flow dei loro programmi e che preferiscono la programmazione strutturata.

Un programma può essere lungo al massimo 4.000 byte per cui ogni procedura, anche di bassa complessità ne richiederà un certo numero. È quindi necessario organizzare i vari programmi tra di loro in maniera «strutturata». In parole semplici ci saranno programmi a alto livello (chiamanti) che chiameranno programmi a basso livello (chiamati) e tale catena in generale richiede dai tre ai cinque livelli.

È indispensabile capire il meccanismo per ottenere, in fase di esecuzione della procedura, il passaggio tra i vari programmi che viceversa in fase di scrittura della procedura risultano indipendenti l'uno dagli altri.

L'esecuzione dei programmi

Una volta scritto il Programma si esegue con il comando

```
DO <nome del programma>
```

e la fine del programma è il comando

RETURN che torna al programma chiamante oppure
QUIT che torna al DOS

Ciascun programma può a sua volta richiamare uno o più altri programmi, che vanno visti come strutture di livello inferiore e che permettono la riemersione con il citato RETURN.

È interessante la gestione delle variabili che se definite valgono per tutti i livelli inferiori, ma che vengono automaticamente rilasciate se dal programma nel quale vengono definite si torna a un livello superiore. Questa logica permette una effettiva economia di variabili e segue una logica quasi sempre verificata.

Se occorre «forzare» questa logica, ad esempio se serve il risultato di un calcolo eseguito a livello inferiore, anche nel programma di livello superiore, si può utilizzare il comando di definizione delle variabili Pubblica e/o Privata, oppure più semplicemente si definisce la variabile direttamente nel programma a livello superiore per cui al rientro non viene rilasciata.

Questo discorso delle variabili vale solo per il DB III, e chi ha lavorato o lavora in DB II ricorda il pesante problema della loro definizione e del loro

```
  246 variabili disponibili, 16227 byte disponibili.
* 04/01/86 - MC microcomputer
*          - gestione Menu
DO WHILE .T.
clear
@  1,10 say "GESTIONE TABELLA MENU'"
@  3,10 say "-------------------------"
@  5,10 say "0   Ritorno in Ambiente DB"
@  6,10 say "1   Gestione Archivio     "
@  7,10 say "2   Stampe Varie          "
@  9,10 say "-------------------------"
var1=" "
do while .not. var1$"012"
 @ 11,10 say "Scegli   : :"
 @ 11,20 get var1 picture "#"
 read
enddo
do case
CASE var1 ="0"
 clear
 clear all
 release all
 return
CASE var1 = "1"
 DO prgest
CASE var1 = "2"
 DO prstam
ENDCASE
ENDDO
```

Figura 10 - Esempio di Programma Chiamante (di Menu). Il tipico programma chiamante altri programmi è quello di Menu, che offre una serie di opzioni per le quali è idonea l'istruzione DO CASE .. CASE.. ENDCASE. IL DO WHILE .T. .. ENDDO esterno crea un ciclo «eterno» dal quale non si esce mai se non tramite un RETURN (al livello superiore), un DO (al livello inferiore), QUIT (al DOS) oppure un EXIT (forzatura).

```
* 04/01/86 - MC microcomputer
*          - gestione archivio
DO WHILE .T.
clear
@  1,10 say "GESTIONE ARCHIVIO"
@  3,10 say "-------------------------------"
@  5,10 say "0   Ritorno al Menu' Precedente"
@  6,10 say "1   Immissione Record          "
@  7,10 say "2   Modifica / Visualizzazione "
@  8,10 say "3   Cancellazione              "
@ 10,10 say "-------------------------------"
var1=" "
do while .not. var1$"0123"
 @ 12,10 say "Scegli   : :"
 @ 12,20 get var1 picture "#"
 read
enddo
do case
CASE var1="0"
 clear
 return
CASE var1="1"
 DO gest1
CASE var1="2"
 DO gest2
CASE var1="3"
 DO gest3
ENDCASE
ENDDO
```

Figura 11 - Esempio di Programma Chiamato. Si ritorna al programma di livello superiore con il comando RETURN, si va a eventuali livelli inferiori con DO <nome programma>.

rilascio nelle applicazioni appena un po' impegnative.

In figura 10 e 11 presentiamo due programmi. Il primo si chiama Menu, ed è un programma solo chiamante in quanto gestisce, per mezzo del comando di scelta DO CASE... CASE la scelta tra tre funzioni a livello più basso. Non avendo un chiamante il RETURN (attivato dalla scelta 0) provoca il rientro in ambiente DB comandi.

Il secondo programma è intermedio in quanto risulta chiamato dal primo e a sua volta chiama livelli sottostanti. Il RETURN in questo caso provoca un rientro al programma superiore di Menu.

In ambedue i programmi viene utilizzato il loop «eterno». DO WHILE .T. .. ENDDO. Questo permette di «canalizzare» bene l'esecuzione in quanto non si deve mai uscire da un programma se non per tornare al livello superiore oppure per richiamare un livello inferiore.

È infine buona norma quando si esce definitivamente dalla procedura, eseguire i comandi di chiusura generale delle variabili e dei file (comandi CLEAR ALL, RELEASE ALL). In quanto successive esecuzioni potrebbero trovare file o variabili già impegnate e questo potrebbe generare una condizione di errore.

```
* Programma PARAM: Con Passaggio Parametri
* X alto a sinistra L larghezza
* Y alto a sinistra H altezza
* eseguibile con DO PARAM WITH 3,4,30,12
parameters X,Y,L,H
set talk off
Xi=X+L
Yi=Y+H
clea
C=Y
do whil C<=Xi
 @ Y,C  say "*"
 @ Yi,C say "*"
 C=C+1
endd
C=Y
do whil C<Yi
 @ C,X  say "*"
 @ C,Xi say "*"
 C=C+1
endd
retu
```

Figura 12 - Esempio di Programma Parametrizzato. Il comando DO permette opzionalmente il passaggio di parametri, per cui il programma chiamato diventa a sua volta parametrizzato e utilizzabile anche in situazioni differenti. Nel caso illustrato il programma realizza un quadrato sul video dimensionato in funzione dei parametri X,L,Y,H che gli vengono passati.

Programmazione avanzata

Sempre alla ricerca della programmazione strutturata esiste la possibilità di trasferire parametri tra le procedure per cui il comando di richiamo di un programma diventa:

DO <nome file> WITH <elenco di parametri>

Interpretando in senso estensivo tale sintassi si possono costruire sottoprogrammi parametrizzati che vengono totalmente definiti dai parametri che fornisce il programma chiamante. In figura 12 vediamo un esempio di tale possibilità.

Abbiamo realizzato un programma che disegna sul video un quadrato fatto di asterischi.
Gli vanno passati i quattro parametri che ne identificano la posizione sul video. È quindi una logica di subroutine che può essere richiamata da qualsiasi posizione e/o livello all'interno della procedura.

Altra funzionalità che risulta utile in programmazione avanzata è costituita dalla possibilità di raccogliere più programmi in sequenza all'interno di una «procedura» oltre il limite (molto penalizzante) di 4 kbyte.

Il vantaggio sta nel fatto che non si spezzetta la procedura in tanti piccoli programmi e che i tempi di esecuzione migliorano.

Lo svantaggio sta nel fatto che la «costruzione» del file procedura è conveniente solo quando il programma è consolidato e cioè quando il lavoro di programmazione è già finito.

MK

LE PIÙ GRANDI NOVITÀ DEL 1986 AI PREZZI PIÙ BASSI D'EUROPA

PERIFERICHE PER QL

QINTERAM - Espansione interna di memoria da 256 o da 512 Kb
Incrementa la memoria delQL a 384 o a 640 Kb, si inserisce all'interno del computer **senza effettuare nessuna saldatura, dissaldatura o manomissione**, non invalida pertanto la garanzia ed il montaggio è velocissimo ed accessibile a tutti. Il connettore di espansione resta libero e si possono inserire contemporaneamente altre periferiche. Costruita con tecnologia TTL C-MOS non richiede alcuna alimentazione esterna. Raddoppia la velocità di esecuzione dei programmi.
QINTERAM 256 Kb L. 199.000
QINTERAM 512 Kb L. 259.000

Q+DISK - SISTEMA PER FLOPPY DRIVES
Il sistema è composto da un'interfaccia per floppy drives e da drives da 3,5" e 1 Mb. L'interfaccia Q+DISK è stata progettata anch'essa dall'autore del QDOS Tony Tebby ed oltre a possedere la piena compatibilità con tutto il software e hardware SINCLAIR offre i seguenti ulteriori vantaggi: **espansione di memoria** fino a 256 Kb residente sulla stessa scheda, gestione diretta della RAM come **RAM-Disk** (carica ad esempio uno screen in meno di un secondo), nuovi **comandi** del **Toolkit N. 2** residenti sulla Eprom della scheda la quale possiede ancora 3,5 Kb di memoria libera per inserire nuove estensioni al Superbasic.
Q+DISK senza memoria (espandibile) + 1 Drive 3,5" 1MB L. 525.000
Q+DISC con 256Kb RAM + 1 Drive 3,5" 1MB L. 725.000
Q+DISC con 256Kb RAM + 2 Drives 3,5" 2Mb L. 985.000

QL CARPROM
Cartuccia porta EPROM completa di EPROM da 128 Kb. Si inserisce nella porta ROM esterna **L. 15.000**

QL PROM
Elaboratissimo e professionale programmatore di EPROM per il QL che si inserisce nel connettore di espansione. Sistema operativo residente su EPROM per una rapidissima programmazione **L. 300.000**

QL UVPROM
Compatto cancellatore di EPROM a UV. Cancella fino a tre EPROM contemporaneamente. Timer automatico da 15' **L. 110.000**

SOFTWARE PER QL

QL MOON
Superbo e professionale programma sull'astronomia lunare, informazioni e statistiche su oltre 300 punti, banca dati di oltre 50 Kb. Gestito da menu con grafica eccezionale. **su cartuccia L. 34.000**

3D SLIME
Gioco con grafica tridimensionale, scritto dagli stessi autori di M-PAINT. **su cartuccia L. 34.000**

QL PLAGIO
Professionale copiatore universale per QL. Copia settore per settore qualsiasi tipo di cartuccia. Scritto completamente in L/M. Funziona con qualsiasi tipo di programma e versione ROM. Il programma viene venduto esclusivamente per proprio uso personale. **su EPROM L. 90.000**

CARTUCCE PER MICRODRIVE 1 L. 5.900
50 (cad.) L. 5.400
100 (cad.) L. 4.900

FLOPPY DISK 3,5" DD - SD - 135 TPI BASF L. 7.000

ATTENZIONE PER TUTTI I POSSESSORI DELLO SPECTRUM

INTERFACCIA DUPLEX

Permette di duplicare e di trasferire su:
- NASTRO - MICRODRIVE - FLOPPY DISK qualsiasi tipo di programma commerciale oggi esistente sul mercato:
- TURBO - TURBO-PULSANTI - MAXI - CON L/M NEL LOADER, ecc.

Semplicissima da usare, si collega l'interfaccia al connettore di espansione, al termine premendo un tasto di break si ottiene una copia a velocità normale che si carica in maniera autonoma senza interfaccia collegata.

I possessori dell'interfaccia 1 potranno scegliere l'opzione microdrive al momento del trasferimento ed ottenere su cartridge una copia del programma preferito.

Il prezzo dell'INTERFACCIA DUPLEX, con il manuale e le spese di spedizione contrassegno è di

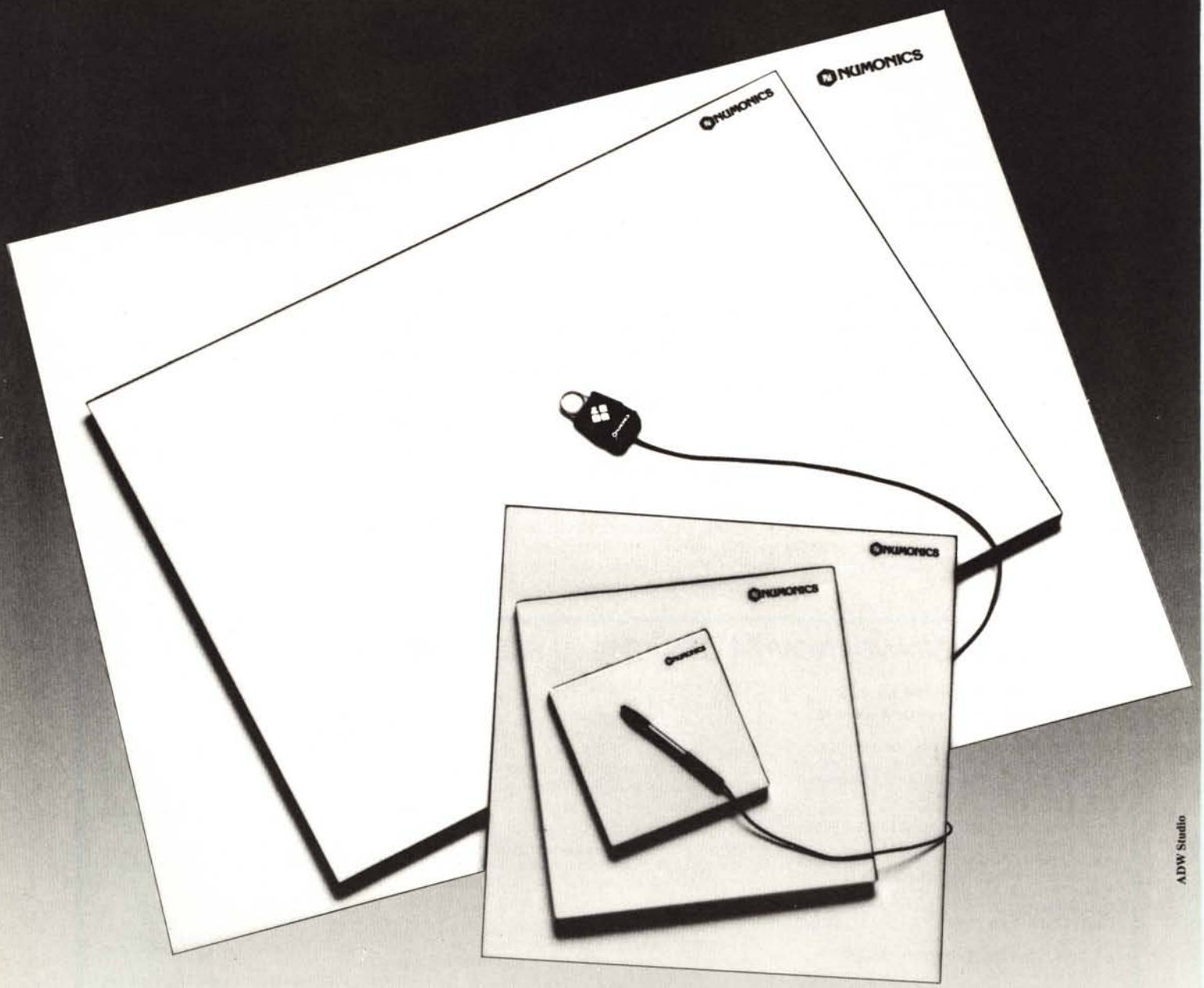
L. 95.000

COMPUTER CENTER

ATTENZIONE I PREZZI SOVRAESPOSTI SONO IVA COMPRESA

PER INFORMAZIONI E/O ORDINAZIONI
VIA FORZE ARMATE, 260 - 2052 MILANO - TEL. 02-4890213

TAVOLETTE GRAFICHE NUMONICS 2200/2210



ADW Studio

Numonics serie 2200 e 2210. Le tavolette grafiche sulle quali potete fare affidamento e sulle quali potete anche dimenticare il vostro dischetto programma.
La tecnologia utilizzata evita i campi elettromagnetici e ne elimina le conseguenze negative. Le tavolette NUMONICS sono disponibili nei seguenti formati: 15x15 cm; 30x30 cm; 30x43 cm; 50x50 cm; 60x90 cm; 90x120 cm.
La precisione è di $\frac{1}{1000}$'' (0,00251 mm), l'interfaccia standard è seriale RS 232C,

inoltre le tavolette NUMONICS possono essere fornite in emulazione di altri modelli. Ampia gamma di accessori per tutte le necessità.

TELAY INTERNATIONAL S.r.l.

COMPUTER PERIPHERALS DIVISION

MILANO: Via L. da Vinci, 43 - 20090 Trezzano S/N
Tel. 02/4455741/2/3/4/5 - Tlx: TELINT I 312827

ROMA: Via Salaria, 1319 - 00138 Roma
Tel. 06/6917058-6919312 - Tlx: TINTRO I 614381