di Francesco Petroni

Introduzione e ambientamento

Prima parte

Nello scorso numero di MCmicrocomputer, nel presentare la prova del DB III, abbiamo annunciato l'inizio di un corso pratico di DB II e DB III, che ha il duplice intento di aiutare chi lo usa o ha intenzione di usarlo e di far capire meglio di che cosa si tratta a chi, avendone spesso sentito parlare, vuol rendersi conto delle possibilità offerte dal prodotto e degli ambiti applicativi nei quali può essere produttivamente utilizzato.

Il corso è soprattutto pratico in quanto si baserà principalmente su esempi reali di utilizzazione, per cui a fronte della descrizione di come lavora una certa istruzione ci sarà la corrispondente esemplificazione. Inoltre corso pratico significa che cercheremo non solo di far vedere come funziona ma anche di suggerire dove e come applicare certe istruzioni.

Il DB II e il DB III sono prodotti del tipo Data Management System e quindi la loro principale funzionalità è quella di gestire archivi. Conseguentemente il loro ambito applicativo è quello legato alla manipolazione di grossi volumi di dati, che risiedono istituzionalmente sulle unità di memoria di massa.

Gli archivi gestibili hanno un limite fisico teorico (vedi tabella di fig. 1) elevatissimo che nessuna applicazione su PC raggiungerà mai, e comunque ben più elevato della capacità fisica non solo di un Floppy, ma anche di un Hard Disk.

Ma anche questo è un limite teorico in quanto anche se è fisicamente e «matematicamente» possibile riempire un intero Hard Disk con uno o più archivi, la loro effettiva gestibilità deriva dal numero di movimentazioni e dalla complessità delle elaborazioni che riguardano il contenuto dell'archivio.

In altre, e probabilmente più semplici, parole, nella valutazione se è conveniente realizzare su PC e con il DB II/DB III una procedura che comporta la gestione di archivi voluminosi, il limite superiore di convenienza è dato più dalla capacità delle unità di memoria di massa e dalle prestazioni intrinseche della macchina (ad esempio tempo di accesso) che non dalla capacità dello strumento DB di gestire archivi di grandi dimensioni.

L'altro aspetto che è opportuno anticipare in sede di introduzione è che DB II/DB III non sono prodotti «appariscenti», anzi non utilizzano nessuno di quei ritrovati hard/soft che hanno fatto la fortuna di tanti pacchetti. Non usano grafica, non usano la tecnica a finestre, non usano mouse. A mala pena permettono l'uso del colore per differenziare logicamente i vari elementi delle videate.

Tra DB II e DB III

Dopo aver chiariti, nell'introduzione, gli ambiti applicativi del DB II/DB III, accenniamo alle differenze tra il DB II e il DB III, e spieghiamo i motivi per cui abbiamo preferito unificarli nello stesso corso.

Il DB II nasce per il mondo CP/M e su macchine 8 bit. Il DB III è stato realizzato per il mondo PC IBM e compatibili, e quindi per macchine a 16 bit ma conserva del predecessore soprattutto la filosofia.

Per cui l'ambiente tipico del DB II è un micro 8 bit, con due unità floppy disk, sistema operativo CP/M. Il linguaggio e l'applicativo risiedono (il

TABELLA DI COMPARAZIONE TRA (principali differenze)	DB II	DB III
Numero di Campi per Record	32	128
Numero di Caratteri per Record	1000	4000
Numero di Record per File	65.535	10^9
Numero di File aperti contemp.	2	10
Precisione Calcolo Numerico (cifre)	10	16
Numero di Variabili Attive Contemp.	64	256
Bytes per le Variabili	1536	definib.
Campi DATA	no	si
Campi MEMO	no	si
Funzioni matematiche ** SQRT LOG EXP	no	si
Funzione sulle DATE	no	si
Sort su File di 1000 Record (minuti)	40	1

Figura 1
Tabella delle Differenti Possibilità e Prestazioni tra DB II e DB III. Le differenze riguardano sia i limiti «fisici» dei due prodotti, sia le performance dei comandi più impegnativi.
Descriveremo le differenze pratiche tra i due linguaggi via via che ne nasce la necessità nel corso delle varie puntate.

DB II necessita di circa 50 kbyte) su un dischetto posto nel drive A, gli archivi su uno o più dischi in B.

Il DB III è molto più ingombrante in quanto da solo risiede su due dischetti. Per una applicazione su un PC con due driver è possibile, se l'applicazione è complessa e gli archivi non sono grandi, organizzarsi con tre dischetti. Il primo contiene il DOS e il boot del DBASE e va messo in A in fase di partenza. Il secondo contiene il DBASE. OVL che è, in pratica, l'unico dei file del DBASE che deve essere presente in fase di lavoro.

Poiché il DBASE. OVL occupa circa 180 kbyte occorre ripartire tra lo spazio rimanente su A e lo spazio totale di B sia i programmi applicativi che gli archivi dati e indici.

Per applicazioni voluminose è pressoché obbligatorio il ricorso a sistemi dotati di Hard Disk. E oggi con la diminuzione dei prezzi di tali periferiche questa soluzione è preferibile rispetto a quella di frazionare i programmi e gli archivi su più dischetti, come si usava un tempo.

È ovvio che in fase di sviluppo applicazioni è meglio disporre sui dischetti di lavoro non solo del DOS, ma anche dell'HELP e dell'ASSIST, ma in questo caso, essendo necessario lavorare solo su archivi di prova, la configurazione a due dischetti non è penalizzante.

Il differente ambiente hardware in cui nascono DB II e DB III ne differenzia notevolmente le prestazioni sia in termini di numero e dimensione degli archivi (vedi tab. 1), sia in termini di velocità di elaborazione.

Dal punto di vista che interessa al corso, e cioè quello dell'utilizzatore esperto o del programmatore, diremo che il DB III conserva buona parte dei comandi del DB II, altri non li conserva oppure li trasforma, ne ha parecchi di nuovi.

```
TIPOLOGIA FILE DBII DBIII
INDIVIDUATI DAL SUFFISSO

suff.   DBII   DBIII   ASCII

*.DBF     si     si     no
*.NDX     si     si     no
*.DBT     no     si     no
*.FMT     si     si     si
*.FRM     si     si     solo DBII
*.LBL     no     si     no
*.MEM     si     si     no
*.PRG     si     si     si
*.TXT     si     si     si
```

Figura 2 - Tipologia dei file riconosciuti tramite suffisso. Una applicazione, anche poco impegnativa, comporta l'utilizzo di decine di file di vario tipo. È quindi indispensabile saper interpretare immediatamente una Directory.

```
. DO FIGTRE
STRUCTURE FOR FILE:  A:FIGTRE  .DBF
NUMBER OF RECORDS:   00005
DATE OF LAST UPDATE: 01/01/80
PRIMARY USE DATABASE
FLD       NAME       TYPE  WIDTH    DEC
001       NOME       C      012
002       ETA        N      002
003       CITTA      C      012
004       CAP        C      005
005       IMPO       N      007
006       PREF       C      004
007       TELE       C      007
** TOTAL **            00050

00001   GIOVANNA      22 ROMA        00184   1200000    06 4433223
00002   LUIGI         24 VERONA      22345   1000000   043 4321123
00003   MARCO         32 LIVORNO     33445    950000   044 432222
00004   VERONICA      43 ROMA        00199     70000    06 432344
00005   GUIDO         19 ROMA        00182    650000    06 324566

GIOVANNA        22          216000.00
LUIGI           24          180000.00
MARCO           32          171000.00
VERONICA        43           12600.00
GUIDO           19          117000.00

00001   GIOVANNA     4433223
00004   VERONICA     432344
00005   GUIDO        324566



* 04/11/85 FIGTRE
USE FIGTRE
ERAS
STOR .18 TO IVA
DISP STRU
?
LIST
?
LIST OFF NOME,ETA,IMPO*IVA
?
LIST NOME,TELE FOR CITTA="ROMA"
*
```

Figura 3 - Esempio di Sintassi di un Comando (Comando LIST, esempio DB II). Molti dei comandi DB permettono un elevato numero di opzioni, per sfruttare le quali è necessario conoscere la sintassi, oppure basta ricordarsi che esiste un comando e chiederne all'Help interattivo la sintassi (esempio HELP LIST).

Differenze pratiche tra DB II e DB III

Al di là dalle differenze in termini di prestazioni e di possibilità, è bene accennare alle principali differenze tra DB II e DB III dal punto di vista dell'utilizzatore.

Il carattere di ready che indica il fatto che si sta lavorando in DB è, per tutte e due le versioni, un semplice punto (.).

Analoga è pure la funzione del tasto ESC, che proca l'interruzione dell'operazione in corso, ad esempio provoca l'interruzione di un LIST sul video, o dell'EDIT di un Record, ecc.

Invece un'utilizzazione in comandi diretti provoca, in caso di errore, due «comportamenti» differenti. Il DB II permette di correggere il comando sbagliato, con una procedura di sostituzione di caratteri. Questo sistema è tanto più vantaggioso quanto più è lungo il comando. Se il comando è corto si fa prima a riscriverlo. Il DB III invece, di fronte ad un errore chiede «Hai bisogno di Aiuto Sì/No» e se si risponde sì, fornisce l'Help del comando sbagliato, ovvero ne fornisce la sintassi.

Volendo semplificare l'interpretazione del differente comportamento il DB II ipotizza che l'errore sia sempre di battitura, mentre il DB III ipotizza che sia un errore di non conoscenza della sintassi.

Un'altra differenza fondamentale è l'istruzione di assegnazione di valore ad una variabile, che per il DB II è un arcaico STORE <valore> TO <nome variabile>. Il DB III oltre a permettere il comando «arcaico» permette anche il più semplice <nome variabile> = <valore>.

Altre differenze pratiche si incontrano in comandi di uso frequente come quello che «pulisce lo schermo» che in DB II è ERASE e in DB III è CLEAR, oppure nei flag BOF e EOF, che indicano le condizioni logiche di inizio e fine File, che in DB III vogliono una doppia parentesi (BOF (), EOF ()).

In definitiva per chi inizia con il DB II o il DB III ovviamente nessun problema, per chi passa da un ambiente all'altro pochi minuti di adattamento e, solo le prime volte, qualche errore.

Nel dischetto che contiene i programmi di utilità del DB III ce ne è uno, DCONVERT, che esegue la conversione dei vari tipi di file DB II nei corrispondenti DB III. Per i programmi questa traduzione non sempre riesce bene, ma il DCONVERT indica le righe non tradotte oppure, a me è successo qualche volta con programmi complessi, si pianta.

In questo caso si può caricare il programma DB II direttamente con l'Editor del DB III ed eseguire sul suo listato le necessarie modifiche che, essendo in genere poche, non comportano molto tempo. Con l'occasione si possono tradurre tutti le istruzioni di assegnazione con un certo risparmio di memoria.

Figura 4
Esempi di utilizzazione di Comandi di Servizio (DIR, COPY FILE, RENAME esempio DB III). DB II e DB III dispongono di numerosi comandi di servizio, alcuni dei quali in pratica eseguono comandi DOS.

```
. DIR
Database          N.record      Revisione          Dim
TELEFO.DBF               4      01/01/80           227
PROVA.DBF                1      01/01/80           243
CODICI.DBF              10      01/01/80           286
AUGURI.DBF               9      01/01/80           428
FIGTRE.DBF        file dBASE II 2.4                1024

   2208 byte,     5 file.
  82944 byte disponibili sul drive.
. COPY FILE CODICI.DBF TO PROVUNO.DBF
    512 byte copiati.
. RENAME PROVUNO.DBF TO PROVA.DBF
C'è già un file con questo nome.
                              ?
RENAME PROVUNO.DBF TO PROVA.DBF
C'è bisogno di aiuto (S/N)? No
. RENAME PROVUNO.DBF TO PROVDUE.DBF
. DELE FILE PROVUNO.DBF
Il file non esiste.
. DELE FILE PROVDUE.DBF
Il file è stato eliminato.
.
```

Programma del corso

Poiché il corso si potrarrà per sei numeri di MC, lo suddivideremo in 6 parti, che possono quindi costituire il Programma.

1) Introduzione ed Ambientamento

L'obiettivo è quello di far capire la filosofia e quindi la modalità di approccio con lo strumento DB, e questi sono i due prerequisiti fondamentali da rispettare per poter lavorare produttivamente con qualsiasi strumento, questo al di là dello studio della sintassi dei singoli comandi. I capitoli sono:

1.1) Da DB II a DB III
1.2) Differenze tra DB II e DB III
1.3) Programma del Corso
1.4) I vari tipi di File e Modalità d'accesso ai File
1.5) La Sintassi dei Comandi
1.6) I Comandi di Servizio
1.7) La Gestione delle Variabili
1.8) I Settaggi e i File CONFIG.SYS e CONFIG.DB

2) Creazione e Gestione dei File Dati

La seconda puntata sarà dedicata alla costruzione e all'uso dei file. Gli argomenti trattati saranno, indicativamente, Concetto di Struttura di un File, Tipo e Logica dei Campi, Come ci si muove all'interno di un file, Comandi per inserire, per modificare Dati nell'Archivio, Comandi che modificano l'Ordine degli Archivi. Parleremo dei rapporti che intercorrono tra variabili attive e file anche nel caso che si lavori con più file.

Introdurremo infine il concetto di indice, che sarà sviluppato nella terza puntata.

3) File Dati e File Indici

La terza puntata sarà dedicata ai file indice che costituiscono la base della «relazionalità» del DB. Vedremo quali applicazioni si possono realizzare con gli indici al di là delle modalità e di costruzione e delle possibilità di uso come «chiave di ricerca».

Vedremo quanti e quali file indici conviene tenere attivi e quali conviene costruire lì per lì e... gettare dopo l'uso.

Faremo esempi di archivi con più file indici e un esempio di costruzione di una tabella tramite un file dati e un file indice. Analizzeremo le possibilità offerte dal DB relazionale.

4) I Programmi

La quarta puntata sarà dedicata alla programmazione e quindi vedremo nel dettaglio l'Editor del DB e quei comandi utilizzabili in programmazione.

Analizzeremo le differenze tra il linguaggio DB e il più diffuso linguaggio per micro che è il Basic.

Vedremo come organizzare un programma di gestione di un archivio e che quindi comprenda funzionalità di immissione, modifica, cancellazione.

5) Elaborazioni e Predisposizioni di Output

Tratteremo tutto quello che non riguarda la gestione degli archivi, ma la loro utilizzazione. Quindi procedure elaborative che manipolano dati provenienti da file e procedure di stampa dei dati risultanti.

Per quanto riguarda le stampe vedremo le tre possibilità offerte dal DB. La stampa diretta tramite specifiche istruzioni da programma, la stampa tramite la facility del DB REPORT, utilizzabile anche in maniera «pesante», e infine la modalità label.

6) Comandi Avanzati di Movimentazione Archivi e Conclusione del Corso

Tratteremo quei comandi DB che permettono movimentazioni anche complesse di archivi e che a loro volta creano nuovi archivi.

Vedremo le possibilità dichiarate e non di «trasportabilità» degli archivi DB in ambienti DOS, BASIC, ecc.

Tratteremo infine le conclusioni del corso.

Vari tipi di file Modalità d'accesso

Il rapporto tra l'utente e il disco è regolato tramite due modalità. Una gestita dall'utente che, richiamando un comando, provoca un accesso, in scrittura o in lettura, al disco, ad esempio col comando di COPY FILE di un file in un altro. La seconda gestita dal DB in modo del tutto trasparente per l'utente. Questa modalità interviene durante la manipolazione di archivi, operazione che comporta sempre accessi (al solito per lettura/modifica/cancellazione) al disco. Per esempio se si richiama uno specifico record di un file i comandi sono quelli di richiamo del file (USE <nome file> che apre il file, oppure SELECT <numero del file> che indica il file di lavoro tra quelli aperti in quel momento) e poi un semplice EDIT <numero del record> che presenta i dati del record già sotto forma di maschera pronta per la modifica.

Eseguita la modifica, questa viene immessa direttamente nel file senza dover eseguire altre operazioni se non la chiusura del lavoro; la scrittura fisica del dato sul disco avviene totalmen-

te «a cura» del DB, e anche il momento della scrittura è gestito dal DB, per cui in pratica ad una certa operazione non è immediatamente conseguente l'accensione del drive.

L'altra caratteristica del DB II e ancor più del DB III è che qualsiasi applicazione lavora con molti file suddivisi in numerosi tipi (7 per il DB II e 10 per il DB III), gestiti dallo strumento. Occorre conoscere bene questa tipologia per poter comprendere il modo di lavorare del DB oppure per poter interpretare una directory di un dischetto.

Nella tabella di figura 2 vediamo l'elenco dei vari tipi di file. Va detto che alcuni di questi file sono leggibili direttamente da DOS, tramite un comando TYPE <nome del file>, oppure dal WordStar. Sono leggibili i file tipo *.PRG o *.TXT o, solo per il DB II, i file *.FRM.

Risulta particolarmente interessante la possibilità di utilizzare il WordStar come Editor con il quale scrivere un programma DB II o DB III, soprattutto per il primo che ha un editor molto «spartano».

In pratica ogni archivio comporta

```
. ! type b:figtre.prg
* 04/11/85 FIGTRE
USE FIGTRE
ERAS
STOR .18 TO IVA
DISP STRU
?
LIST
?
LIST OFF NOME,ETA,IMPO*IVA
?
LIST NOME,TELE FOR CITTA="ROMA"
*
```

*Firura 5 - Comando RUN (solo DB III). Uno dei comandi di Servizio più utili è il Run che permette un accesso diretto al DOS, e permette di eseguire file del tipo *.COM o del tipo *.EXE. In pratica è possibile ad esempio eseguire da DB III un programma Basic compilato e poi tornare al DB III. Nell'esempio lo usiamo per eseguire una stampa di un file.*

l'esistenza di un file dati (suffiso *.DBF) e di tanti file indice (suffisso *.NDX) quante sono le modalità di accesso al file.

Per ogni formato di output su video ci sarà un file suffisso *.FMT, per ogni formato di stampa realizzato con il CREATE REPORT ci sarà un file suffisso *.FRM. Inoltre se occorre memorizzare gruppi di variabili in un file richiamabile ad esempio da un programma si utilizzerà un file suffisso *.MEM. L'istruzione per memorizzare un file di variabili è la SAVE TO <nome del file>. Per richiamare il file l'istruzione di lettura è la RESTORE FROM <nome del file>.

In figura 7 presentiamo un programma (valido sia per il DB II che per il DB III) che dapprima crea una variabile linea e la memorizza in un file *.MEM, poi la richiama e la usa.

Esiste poi la possibilità di usare all'interno di un file dei campi del tipo MEMO, ovvero dei campi di lunghezza indefinita e di contenuto libero. Il DB III (questa possibilità è solo del DB III) produce un file *.DBT, collegato al file *.DBF di cui è una specie di appendice richiamabile tramite opportuni comandi.

I programmi veri e propri hanno suffisso *.PRG e sono editabili con un Editor, un po' semplificato nel DB II, un po' più sofisticato nel DB III. Non è possibile eseguire un Debug del programma se non eseguendolo e... vedendo che succede. Gli errori vengono segnalati in questa fase e il messaggio consiste nel tipo di errore e nella ripetizione della riga sbagliata. Se il programma può proseguire malgrado l'errore, il DB chiede il consenso alla prosecuzione... e alla scoperta di altri errori.

Per eseguire il programma l'istruzione è il DO <nome del programma>.

Gli ultimi suffissi sono quelli relativi ai programmi di stampa. Sia il DB II che il DB III hanno una istruzione CREATE REPORT, che permette la costruzione di un tabulato direttamente dalla struttura di un file. Questa istruzione provoca un file tipo *.FRM che è leggibile da DOS e dall'editor del DB II, ma non da quello del DB III, che ha però una specifica istruzione MODIFY REPORT.

In più rispetto al DB II, il DB III ha la possibilità di creare, partendo direttamente dalla struttura del file, un formato etichette, per la stampa delle medesime. Il suffisso del relativo file è *.LBL.

Come detto lavorando in DB si utilizzano sempre decine di file, il che comporta l'inconveniente che fisicamente i file risultano un po «sparsi» sul dischetto. Detto in maniera un po' più tecnica se si esegue un CHKDSK *.* del disco con i file questi risulteranno «non contigui», con ripercussioni in termini di prestazioni.

La semplice e ovvia soluzione a questo inconveniente è quella di eseguire copie di Backup con il comando Dos COPY *.* e non con il DISKCOPY che riproduce anche la disorganizzazione del dischetto origine.

La sintassi dei comandi

La prima cosa da fare è imparare la sintassi dei comandi, infatti molti comandi del DB permettono svariate opzioni e quindi è bene saper interpretare immediatamente la loro sintassi.

Inoltre una volta ambientatisi e capita la filosofia con la quale lavora il DB non serve consultare il voluminoso manuale, è sufficiente richiamare l'HELP interattivo, se installato, o ricorrere alle guide di consultazione, che sono dei libretti pieghevoli in dotazione sia al DB II che al DB III.

In ogni comando vi è una parte DB, ovvero riconosciuta dalla sintassi del DB, e una parte utente, riconosciuta in quanto in quel momento sono attive quelle variabili o quei campi. Inoltre in ogni comando c'è una parte obbligatoria, ovvero il minimo indispensabile, e parti opzionali, che possono all'occorrenza essere aggiunte.

Le convenzioni tipografiche per i comandi sono semplicissime, in pratica sono quattro.

In carattere MAIUSCOLO sono riportati i comandi sintassi DB.

In carattere minuscolo sono riportate le porzioni di comandi fornite dall'utente.

Le parti opzionali del comando sono racchiuse tra parentesi quadre, [].

Le parti indicata dall'utente sono racchiuse tra segni maggiore, minore, < >.

In figura 3 facciamo un esercizio con il comando LIST, che utilizziamo in quanto ha un significato semplicissimo. Con il comando LIST si ottiene la visualizzazione del contenuto dell'archivio. La sua sintassi è:

```
LIST [OFF]
[<intervallo>]
[<elenco di campi, espressioni>]
[FOR/WHILE <condizione>]
[TO PRINT]
```

Interpretiamo la sintassi.

Il comando LIST «liscio» produce la visualizzazione di tutti i campi, nell'ordine in cui sono nella struttura del file, di tutti i record, nell'ordine in cui è organizzato l'archivio.

Con l'opzione OFF si elimina dalla visualizzazione il numero record, che è l'elemento identificativo che il DB attribuisce a ciascun record.

Con l'opzione [<intervallo>] si limita l'out ai record il cui numero sia compreso nell'intervallo stesso.

Con l'opzione [<elenco dei campi, espressioni>] si indica il contenuto da visualizzare per singolo record. Ovvero quali campi e in quale ordine, oppure quale espressione di stringa o matematica. Per cui se ad esempio serve eseguire un calcolo su dei campi questo può essere eseguito direttamente sotto il comando LIST.

L'istruzione [FOR/WHILE <condizione>] permette di condizionare l'out, ovvero di listare solo i record che soddisfano una certa condizione, anche complessa.

Infine l'istruzione TO PRINT permette di incanalare l'out verso la stampante.

Tutte le parti opzionali dei comandi

possono combinarsi tra di loro e quindi possono dare luogo ad un numero pressoché infinito di combinazioni.

In realtà per la predisposizione di menti calcolati, oppure funzioni di stringa all'interno delle espressioni, consente di utilizzare il comando LIST anche per funzioni, un po' primordiali, di stampa.

In realtà per la predisposizione di stampe sotto forma di tabulati esiste il comando REPORT, che vedremo nella quarta puntata. Il comando Report ha in più la possibilità di inserire campi di totali e sottototali e inoltre permette di gestire l'impaginazione, tramite controllo delle intestazioni e dei salti pagina, cosa che ovviamente il LIST non fa.

In figura 3 eseguiamo una serie di esercizi sul LIST prendendo come base un piccolo archivio DB II di cui diamo all'inizio la struttura e il list completo. La sequenza dei comandi è stata inserita in un programma (file tipo *.PRG) di cui diamo alla fine dell'esempio il listato.

Va detto che sia DB II che DB III accettano una abbreviazione dei comandi ai primi 4 caratteri, il che permette di velocizzare la digitazione e di ridurre la dimensione dei listati dei programmi, esempio:

```
DISP STRU = display structure
SET DEFA TO B = set default to B
```

I comandi di servizio

Esistono alcuni comandi di Servizio e di Utilità che permettono una serie di operazioni non direttamente utilizzabili nella gestione dei file, ma che facilitano il lavoro. Alcuni di questi sono attribuiti ai tasti funzione, ma possono essere richiamati anche direttamente o da programma, così come si possono riattribuire i tasti funzioni.

Sono fondamentali i comandi:

— LIST FILE (DB II) e DIR (DB III) che elencano tutti i file del tipo *.DBF presenti sul dischetto e danno alcune informazioni supplementari (vedi fig. 4).

Per avere l'elenco di altri tipi di file valgono le regole dei caratteri jolly dei comandi DOS, per cui ad esempio il comando DB III DIR *.NDX elenca tutti i file del tipo indice.

— DISPLAY MEMORY elenca tutte le variabili attive in quel momento specificandone il tipo.

— DISPLAY STATUS presenta due pagine. La prima elenca la situazione dei file aperti in quel momento. Indica i file tipo *.DBF, indica i file di tipo *.NDX e la composizione della relativa chiave, indica le Relazioni attive. La seconda pagina indica lo stato delle istruzioni SET.

Quando si lavora su procedure complesse, che manipolano molti archivi e molti indici, è indispensabile sapere in ogni momento quali file siano aperti e quali siano attivi. La maniera più semplice per saperlo è il DISP STAT, anzi è opportuno metterne anche all'interno dei programmi di cui si sta eseguendo il DEBUG, in quanto non ne provoca l'interruzione.

— DISPLAY STRUCTURE dà tutte le informazioni relative all'archivio attivo in quel momento, ecc.

Altri comandi di Servizio sono quelli che permettono la copia, la cancellazione, la rinominazione dei file. Sono simili ai corrispondenti comandi DOS, ma in più rispetto a questo, chiedono conferma per le operazioni che provocano la cancellazione di altri file.

Un altro comando di servizio è il RUN (solo DB III) che permette l'esecuzione direttamente da DB III di un comando DOS. Per poterlo utilizzare sono necessarie due condizioni, che esistano sia il COMMAND.COM che i *.COM richiamati sul dischetto che si sta usando, e che in sede di settaggio iniziale si sia lasciata libera una porzione di memoria. Per esempio:

```
! type B: program.prg > prn
```

Con questo comando si esegue una stampa, via comando DOS TYPE, del listato del programma PROGRAM. PRG che sta sul disco B.

La gestione delle variabili

Poiché vi è un limite nel numero e

```
DO FIGSEI
00001  GIOVANNA      22 ROMA        00184   1200000   06 4433223
00004  VERONICA      43 ROMA        00199     70000   06 432344
00005  GUIDO         19 ROMA        00182    650000   06 324566



* 04/11/85 FIGSEI
*
SET TALK OFF
STOR "FIGTRE" TO VAR:1
STOR "ROMA" TO VAR:2
STOR "10000" TO VAR:3
ERAS
*
USE  &VAR:1
*
LIST FOR CITTA="&VAR:2" .OR. IMPO<VAL("&VAR:3")
*
```

Figura 6 - Esempio di Gestione Variabili. Uso dell'«&» (esempio DB II). DB II e DB III differiscono non solo nel numero delle variabili utilizzabili contemporaneamente, ma anche nel modo di gestirle all'interno di una procedura. È identico, per ambedue i linguaggi, l'uso della «&», cui si riferisce l'esempio pubblicato.

```
. do figset
linea.mem già esiste, va riscritto (S/N)? Si
La variabile non è stata trovata.
      ?
? linea
Chiamato da B:figset.prg
Chiusura file comandi (S/N)? No
_____________________________________________________

_____________________________________________________



Elaborazione testi dBASE III                      INS
* 04/11/85 figset
clear all
set talk off
stor 1 to cn
linea=chr(196)
do while cn<79
 linea=linea+chr(196)
 cn=cn+1
enddo
save to linea.mem
clear all
? linea
restore from linea
? linea
?
? linea
*
```

*Figura 7 - Uso di un file del tipo *.MEM per memorizzare variabili. In pratica si possono costruire veri e propri archivi di variabili da caricare e scaricare al momento opportuno. L'esempio pubblicato (vale per ambedue) mostra come costruire e memorizzare una variabile, che per essere usata deve essere prima richiamata.*

```
Direttrici di ricerca:
Drive per difetto: B:
ALTERNATE  - OFF  DEBUG      - OFF  ESCAPE     - ON   MENU       - ON
BELL       - ON   DELETED    - OFF  EXACT      - OFF  PRINT      - OFF
CARRY      - OFF  DELIMITERS - OFF  HEADING    - ON   SAFETY     - ON
CONFIRM    - OFF  DEVICE     - SCRN HELP       - ON   STEP       - OFF
CONSOLE    - ON   ECHO       - OFF  INTENSITY  - ON   TALK       - ON
UNIQUE     - OFF


Margine sinistro:     0

Funzione 1  F1   - help;
Funzione 2  F2   - assist;
Funzione 3  F3   - list;
Funzione 4  F4   - dir;
Funzione 5  F5   - display structure;
Funzione 6  F6   - display status;
Funzione 7  F7   - display memory;
Funzione 8  F8   - display;
Funzione 9  F9   - append;
Funzione 10 F10  - edit;
```

Figura 8 - Controllo Elenco dei Settaggi Tramite Display Status. I settaggi possibili sono 30 in DB II e 45 in DB III e consentono una serie pressoché infinita di situazioni di lavoro.

dimensioni delle variabili utilizzabili è bene imparare subito a gestirle con parsimonia. In realtà il problema esiste solo per il DB II, che permette 64 variabili fino a un totale di 1500 byte, mentre il DB III ne accetta 256, permette il settaggio della memoria riservata e soprattutto le rilascia automaticamente con una logica che vedremo poi.

Invece con il DB II, in una applicazione complessa, è probabile che si raggiunga il limite di 64 variabili, per cui occorre rilasciarle dopo averle utilizzate. Poiché esiste anche per il comando RELEASE la possibilità di usare i caratteri jolly la procedura da usare è quella di battezzare con un prefisso uguale tutte le variabili usate in un dato programma così la procedura di rilascio si riduce ad un'unica istruzione.

Esempio:

```
var:01 = «GENNAIO»
var:02 = «01»
var:03 = «1985»
RELE ALL LIKE var:*
```

In una procedura DB III invece le variabili valgono solo per i programmi di livello sottostante a quello nel quale sono state create. Per conservarne il valore oltre tale limitazione occorre dichiararle PUBLIC, oppure basta definirle al livello superiore. Ad esempio se si deve eseguire un programma che produce un calcolo del quale serve solo il risultato nel programma chiamante, basta definire, all'interno di questo, la variabile «risultato».

Un insieme di variabili può essere memorizzato in un file tipo *.MEM, le istruzioni sono:

SAVE TO <nome del file> [<elenco di variabili>] per scrivere RESTORE FROM <nome del file> per leggere.

Con tali istruzioni, combinate alla RELEASE, si possono caricare e scaricare insiemi di variabili predefinite una volta per tutte, e quindi non è necessario definirle all'interno dei programmi.

La funzione «&» (e commerciale, ampersend, ecc.) è molto piccola, ma molto importante e ha il significato di MACRO sostituzione. Ovvero l'apposizione di & davanti al nome di una variabile fa assumere all'insieme il significato del contenuto della variabile, e questo anche all'interno dei caratteri apice o doppio apice, che delimitano le stringhe.

In pratica si può attribuire ad una variabile il contenuto di un comando, esempio var = «LIST», e poi per eseguire il comando si può passare direttamente la variabile con il prefisso Macro sostituzione, esempio «&var».

Questa opportunità permette la completa parametrizzazione anche dei comandi complessi per cui risulta particolarmente utile in programmazione.

In figura 6 vediamo un esercizio sulle variabili e sull'uso della Macro sostituzione e in figura 7 vediamo un programma che costruisce un file *.MEM in cui immagazzina una variabile LINEA, poi richiama il file e usa la variabile richiamata.

La variabile LINEA è costituita da una sequenza di 78 caratteri 196, trattino orizzontale.

I settaggi e i file CONFIG. SYS e CONFIG. DB

L'ultima categoria di comandi che esaminiamo nella prima puntata è quella dei settaggi. Esistono 28 comandi in DB II e 38 in DB III la cui sintassi è:

SET <nome del settaggio> ON/OFF
oppure
SET <nome del settaggio> TO

che permettono di definire i correnti parametri di lavoro. Queste definizioni valgono fino a quando non viene dato il comando di SET differente.

Il primo gruppo è costituito da switch, che vengono accesi o spenti tramite ON/OFF. Ad esempio SET PRINT ON/OFF accende o spegne la stampante, oppure SET DELETED ON/OFF permette o meno la visualizzazione, all'interno di un LIST, dei record cancellati logicamente (ma non ancora fisicamente dal file).

Il secondo gruppo setta parametri che non sono dei semplici switch. Ad esempio SET DECIMALS TO 16 indica che le cifre significative dei calcoli numerici debbono essere 16, oppure SET INDEX TO <elenco indici *.NDX>, attiva una serie di archivi indici relativi al file attivo in quel momento.

Non elenchiamo pedissequamente i SET possibili, diremo solo che permettono di predisporre in ogni momento le migliori condizioni di lavoro. Ad èsempio i settaggi da definire in sede di sviluppo di un programma saranno del tutto differenti da quelli da definire in sede di applicativo.

In particolare è possibile definire al momento del caricamento del DB III (solo DB III) tutti i settaggi iniziali, più una serie di configurazioni iniziali (come memoria da destinare alle variabili, massimo della memoria da riservare al DB III, ecc.), utilizzando il file CONFIG. DB. Questo, viene consultato dal DB III al momento del caricamento, e permette anche la definizione di un file programma (tipo *. PRG) da eseguire al caricamento.

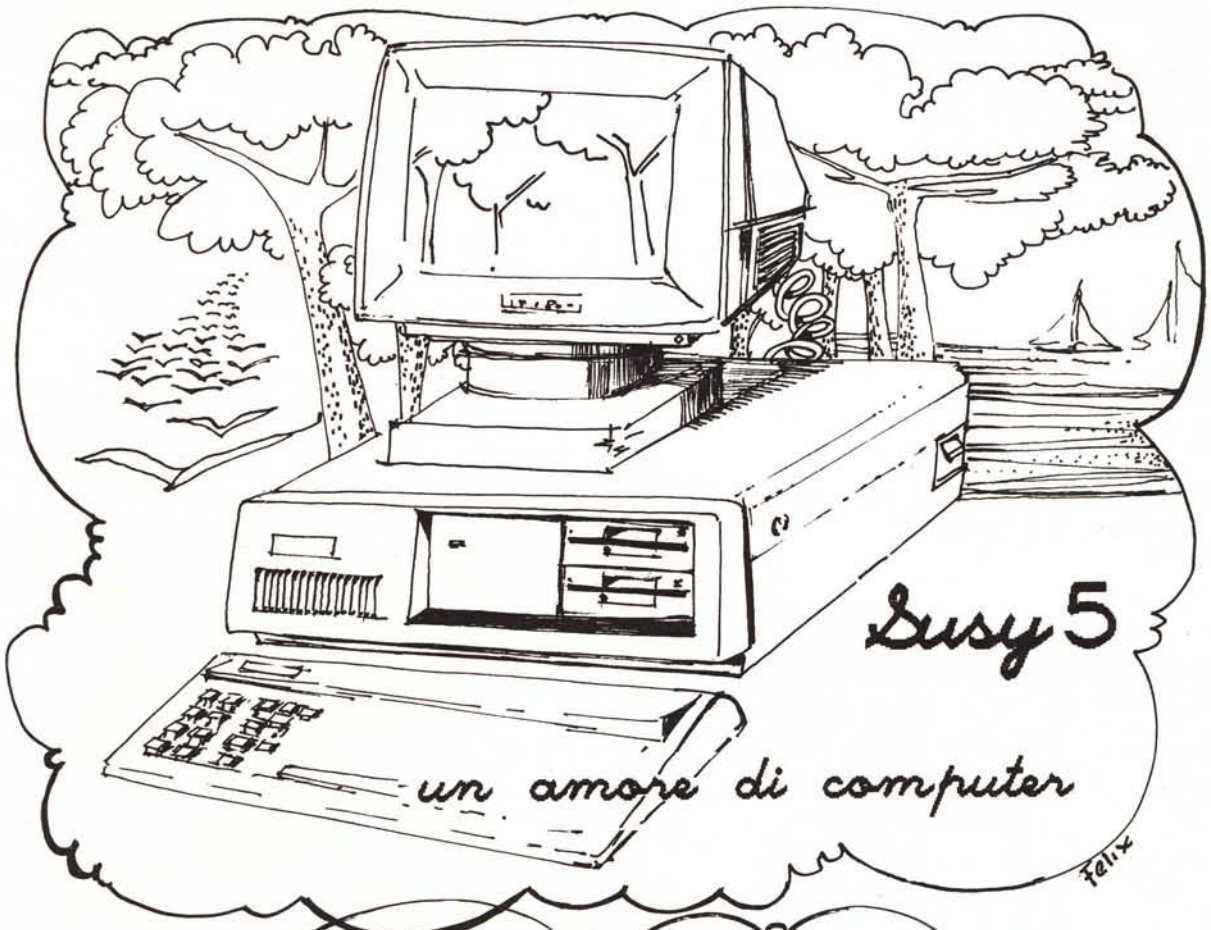
In pratica se si carica il DB III con l'Autoexec del DOS, e l'applicativo sotto DB III con il CONFIG. DB, si può entrare nell'ambiente applicativo direttamente all'accensione della macchina. La stessa cosa si può fare con il DB II, creando un file *.PRG (es. START.PRG), che esegue tutti i settaggi voluti e richiama l'applicativo con DO <nome del file>. E da Autoexec si richiama l'applicativo con il comando composto DBASE START.

Esiste, se si lavora in DB III, che come detto può lavorare con 15 file aperti contemporaneamente, la necessità di comunicare al DOS i settaggi dei buffer e dei file. Di questo si occupa il comando DOS CONFIG. SYS che deve essere presente al momento del caricamento del DOS. Nei dischetti del DB III c'è un CONFIG. SYS già confezionato allo scopo, o altrimenti sul manuale c'è la spiegazione di come costruirlo.

già IL BITTEGONE di felice pagnani

computerline srl

via ubaldo comandini 49 00173 roma - t. 6133025 7970559 tx.621166 fepag i



Ama il duro lavoro da
sola e collegata in rete
con altre compagne,
colloquia con l'host come
con un vecchio amico.
Sopratutto e' fedele:
Non ti pianta mai in asso!

CORSO PRATICO DI UTILIZZO DEL

SOFTWARE

APPLICAZIONI
LINGUAGGI
SISTEMI OPERATIVI
E PROGRAMMAZIONE
DEI PERSONAL COMPUTER

WORD PROCESSOR • DATA BASE • FOGLI ELETTRONICI • COMPUTERGRAFICA • BASIC • LOGO • PASCAL • FORTRAN • COBOL • "C"... MS DOS • C/PM • XENIX • UNIX • UCSD

Software si compone di 52 fascicoli settimanali, da rilegare in 5 splendidi volumi:
BASIC I E II • SISTEMI OPERATIVI • LINGUAGGI • APPLICAZIONI •

È IN EDICOLA
1° E 2° FASCICOLO A SOLE Lire 2.200

Software, ultimissima novità del Gruppo Editoriale Jackson, è la prima opera completa sulla programmazione del personal computer in 5 volumi.
Un'opera diversa e assai più approfondita rispetto a un semplice corso di Basic.
Se è vero, infatti, che il Basic fornisce un'utile chiave d'accesso al mondo della programmazione, è altrettanto vero che quest'ultima abbraccia un campo assai più vasto e complesso rispetto al popolare linguaggio.
Sistemi Operativi, **Linguaggi di Programmazione**, **Software Applicativo**: questi i tre cardini su cui si fonda **Software**, che fornisce tutti gli strumenti teorici, ma soprattutto pratici, per acquisire la padronanza completa del personal computer. Per risolvere, finalmente, i problemi legati all'uso pratico della macchina; per comprenderne le soluzioni applicative più idonee.
Ottimo per il principiante, che intende accedere al mondo dellInformatica dalla porta principale, ideale per chi desidera approfondirne la conoscenza e acquisire in tal modo una professionalità sempre maggiore.

GRUPPO EDITORIALE JACKSON
DIVISIONE GRANDI OPERE