110

dBASE II

Sistema di gestione relazionale di Base di Dati

Scritto da

WAYNE RATLIFF

Ratliff Software Production (RSP), Inc.

Manuale d'utente

Caro cliente,

vi siamo veramente grati di aver scelto il dBASE II per il vostro progetto di gestione di dati. Crediamo, d'altra parte, che troverete tutta la potenza necessaria per la creazione delle vostre basi di dati ed il rapido accesso all'esatta informazione che ricercate.

Ashton-Tate è lieta d'informarvi che ora il dBASE II comprende lo ZIP, un programma generatore di formati di schermo, sviluppato da Hal Pawluk. ZIP vi permetterà di determinare facilmente il vostro formato per le entrate e le uscite di schermo, in modo che dBASE II possa operare per voi con una velocità molto superiore.

Per installare lo ZIP sul vostro terminale; eseguite ZIPIN e, subito dopo, ZIP: il vostro schermo è pronto. Se avete osservazioni o suggerimenti circa lo ZIP, scriveteci: Ashton-Tate ve ne sarà grata.

Come parte del servizio di Ashton-Tate ai suoi clienti, periodicamente vi faremo giungere eventuali correzioni, suggerimenti per la programmazione e annunci di nuovi prodotti. A questo scopo, così come per ricevere supporto tecnico, dovrete firmare il dBASE II Software License Agreement (Accordo di Licenza per Software dBASE II). Sempre per vostra comodità, accludiamo un indice comprensivo di ambedue le parti del Manuale.

Riguardo a eventuali richieste ad acquisto avvenuto, il vostro distributore ha le risorse professionali per assistervi mediante i suggerimenti adeguati. Da parte nostra, garantiamo ogni appoggio a quei distributori che ci interpellano a proposito di richieste o problemi dei nostri clienti.

Vi preghiamo di non tralasciare di fare una copia di riserva (backup) tanto del disco del vostro sistema come dello ZIP, prima di intraprendere qualsiasi altra operazione. L'abituale comando CP/M per trasferire gli archivi del disco da A: a B: è

PIP B:=A:<Filename.ext>

Per avere l'elenco dei comandi usate HELP, sia su Demo Disk come su System Disk. Appena vi trovate in dBASE II, battete semplicemente "HELP dBASE": è tutto.

Grazie di nuovo.

Product Marketing Department
ASHTON-Tate

dBASE II™
MANUALE DELL'UTENTE

INFORMAZIONE IMPORTANTE SULL'USO DEL dBASE II E DI QUESTO MANUALE

Il programma per computer dBASE II e questo Manuale dell'utente non possono essere copiati, riprodotti, rivelati, trasferiti o ridotti a qualsiasi mezzo elettronico o forma leggibile senza l'approvazione di Ashton-Tate, espressamente scritta. La copia non autorizzata del programma o del Manuale è una violazione della legge sul copyright e sul segreto commerciale. L'infrazione del copyright o l'appropriazione indebita di segreti commerciali possono condurre al procedimento penale, pagamento di multe e incarcerazione, oltre ai danni civili.

I diritti e gli obblighi dell'utente di questo Manuale e dei programmi software del dBASE II si trovano sottoposti ai termini del Software License Agreement redatto da Ashton-Tate e qui accluso come parte di questo pacchetto.

QUESTO MANUALE DELL'UTENTE SI CONSEGNA COSI' COM'E'. FATTA ECCEZIONE DELLA GARANZIA DESCRITTA NELL'ASHTON-TATE LICENSE AGREEMENT, NON VI SONO ALTRE GARANZIE ESPRESSE O IMPLICITE, COMPRESE, SENZA LIMITAZIONE ALCUNA, LE GARANZIE IMPLICITE DI COMMERCIABILITA' O ADATTAMENTO A PROPOSITI PARTICOLARI, ESSENDO TALI GARANZIE ESPRESSAMENTE E SPECIFICAMENTE RICUSATE.

IN NESSUN CASO ASHTON-TATE SI RITIENE RESPONSABILE DI QUALSIASI DANNO, INDIRETTO O CASUALE, NE' DI PERDITA DI PROFITTI, COMPRESO IL CASO IN CUI ASHTON-TATE FOSSE STATA AVVERTITA DELLA POSSIBILITA' DI TALE DANNO. (Alcuni stati non permettono la limitazione o l'esclusione della responsabilità per danni accidentali o casuali, nel qual caso le limitazioni o esclusioni sopra dette resterebbero senza applicazione).

DEFINIZIONE DEL PRODOTTO

All'acquisto del sistema completo dBASE II dovete ricevere:

Dischetto Sistema serializzato,
Dischetto dimostrativo limitato a 15 record per file,
Ashton-Tate Software License Agreement serializzato,
Manuale dell'utente di dBASE II,
Programmi di esempio,
Assistenza ad acquisto avvenuto, attraverso il vostro distributore di software.

Altri prodotti e servizi disponibili presso Ashton-Tate:

Aggiornamenti del sistema dBASE II,
Bollettino per l'utente finale,
Annunci di nuovi prodotti,
Catalogo dBASE delle applicazioni esistenti.

Se credete di non aver ricevuto il prodotto completo o se volete chiedere chiarimenti sui vostri diritti e obblighi, oppure se desiderate informazioni su questi nostri servizi, indirizzatevi a:

Ashton-Tate
10150 West Jefferson Boulevard
Culver City, California 90230
(213) 204-5570

PARTE A

INDICE

S E Z I O N E I

S E Z I O N E II

S E Z I O N E I°I I

S E Z I O N E I V

S E Z I O N E V

INTRODUZIONE

dBASE II è uno strumento di gestione di base-dati che permette la facile manipolazione di base-dati piccole e medie usando comandi verbali. Con dBASE II potete:

- Creare sistemi completi di base-dati.
- Aggiungere, eliminare, correggere, visualizzare e stampare con facilità i dati della vostra base-dati, con un minimo di duplicazione di dati nel file.
- Ottenere un'ampia indipendenza fra programma e dati, in modo che al momento di cambiare i vostri dati non avete bisogno di cambiare i programmi e viceversa.
- Generare rapporti da una o più base-dati, ed eseguire automaticamente moltiplicazioni, divisioni, totali, sottototali ed altri tipi di operazioni, anche non aritmetiche, ogni volta che usate la base-dati.
- Usare la capacità di configurazione dello schermo (full-screen editing) per stabilire un formato, in modo da vedere esattamente quello che state ottenendo e introdurre i dati semplicemente "riempiendo gli spazi".

dBASE II è un sistema estremamente potente. Per trarne il massimo rendimento, prendetevi il tempo necessario per leggere le istruzioni prima di cominciare ad usarlo. Non ve ne pentirete.

CONVENZIONI TIPOGRAFICHE USATE IN QUESTO MANUALE

LE MINUSCOLE (fra parentesi quadre) che appaiono sullo schermo indicano i comandi che voi avete introdotto.

LA MAIUSCOLE che appaiono sullo schermo indicano le chiamate e le risposte del dBASE II. Nel testo, le maiuscole si usano per le istruzioni o comandi.

"..." si useranno nel testo di questo Manuale per indicarvi i comandi e i comandi dBASE II che dovete battere. Occasionalmente, essi possono apparire nelle rappresentazioni su schermo per maggiore chiarezza. VOI NON DOVETE BATTERE QUESTI SIMBOLI.

[...] si useranno per indicare le parti opzionali di un'istruzione.

<...> indicheranno le porzioni di un'istruzione contenenti un'informazione reale. Per esempio, <file> vorrà dire che si deve inserire il nome di un file. Nel testo, si usano per racchiudere i nomi dei file e dei campi.

<enter> significa premere il ritorno-carrello o il tasto "enter" sulla tastiera. NON BATTETE QUESTA PAROLA, NE' QUESTI SIMBOLI.

I REQUISITI DEL SISTEMA

dBASE II richiede l'ambiente hardware e software che specifichiamo:

- Un microprocessore di base 8080, 8085 o Z-80 (come TRS-80/II, Northstar, Apple II con Z-80, ecc.) con sistemi operativi CP/M Z.X, CDOS o CROMIX.
- Un microprocessore 8086 o 8088 (come IBM-PC, NEC APC ecc.) con sistemi operativi CP/M-86 o MSDOS.
- Memoria di 48K byte come minimo (dBASE II utilizza locazioni da 5CH a A400H) per la maggior parte dei microprocessori, 56k per Apple (Standard), Heath, Northstar e 128k per IBM-PC.
- Una o più memorie di massa (normalmente dispositivi a dischetti).
- Un video a cursore indirizzabile, se si prevedono operazioni a schermo intero.
- Stampante opzionale (per alcune istruzioni).

CARATTERISTICHE DI dBASE II

Record per file	65535 massimo
Caratteri per record	1000 massimo
Campi per record	32 massimo
Numero maggiore	± 1.8x10**63
Numero minore	± 1x10**-63
Precisione numerica	10 cifre
Lunghezza massima della stringa	254 caratteri
Lunghezza massima della riga d'istruzioni	254 caratteri
Lunghezza massima del titolo delle relazioni	254 caratteri
Lunghezza massima della chiave degli indici	100 caratteri
Espressioni nell'istruzione SUM	5 massimo

PRIMA DI FARE QUALSIASI ALTRA COSA, FATE UNA COPIA DEL DISCO dBASE II. CONSERVATE L'ORIGINALE IN UN POSTO SICURO E USATE LA COPIA.

Installate un disco nel drive A e il disco di dBASE II nel drive B. Ora inserite:

^PIP A:=B:*.*[OV^] o un comando equivalente.

La lettera "O" è necessaria per assicurarvi che il vostro sistema operativo copierà tutti i dati del disco originale.

Se lavorate con un solo drive, usate le istruzioni COPY o BACKUP e seguite le istruzioni dello schermo.

I backup sono essenziali e devono ripetersi con frequenza. Se le vostre sedute al computer sono brevi, basterà uno per seduta, in caso contrario dovrete farlo molto più spesso. Naturalmente voi siete in grado di valutare il bilancio tra il costo della copia e quello dei vostri dati meglio di noi, ma noi possiamo dirvi che, dato che potete riscrivere i dischi, il costo dei backup rimane basso. E quanto vale il vostro file?

Crediamo, a questo punto, di aver insistito a sufficienza.

INSTALLAZIONE DEL dBASE II NEL VOSTRO SISTEMA OPERATIVO

Caricate la copia del dBASE II (l'avete fatta, non è vero?) nel drive predisposto ed eseguite le inizializzazioni necessarie (controllo-C, reset, ecc.). Ora digitate ^INSTALL^ per personalizzare dBASE II al vostro sistema (RICORDATEVI DI NON BATTERE IL SIMBOLO "^").

Se il vostro terminale non ha il posizionamento X-Y del cursore (guardate nel vostro manuale), battete ^N^ in risposta alla richiesta dello schermo. Altrimenti battete ^Y^. Con ciò avrete ottenuto la possibilità di correggere a "schermo intero", che è la miglior maniera di introdurre dati e di lavorare con la vostra base-dati. In questo modo invece di trovarvi a battere sull'ultima riga dello schermo, con tutto il resto che scompare verso l'alto, potete posizionare il cursore dovunque vogliate, usando i comandi e le istruzioni di dBASE II.

A questo punto, dBASE II elenca per voi i tipi di terminali compatibili. Se il vostro è nella lista, battete la lettera indicata. In caso contrario, battete ^Z^.

```
A Install

dBASE INSTALLATION PROGRAM VER 2.8

ARE FULL SCREEN OPERATIONS WANTED (Y/N)? y
SELECT TERMINAL TYPE
A - HAZELTINE 1500              B - SOROC 120, 140. TELEVIDEO
C - HEATH 89                    D - PERKIN ELMER FOX 1100
E - ADM-3A                      F - ADM-31
G - VDP-80                      H - INTECOLOR
I - GNAT-SYSTEM 10              J - TRS-80 PICKLES TROUT
K - APPLE                       L - VECTOR GRAPHICS
M - SUPERBRAIN                  N - VISUAL 100
Y - CHANGE/MODIFY PREVIOUSLY INSTALLED TERMINAL
Z - USER SUPPLIED TERMINAL CHARACTERISTICS

K
```

Se scegliete uno dei terminali in lista, dBASE II vi domanderà quale carattere volete usare per la sostituzione macro (descritta nella Sezione IV e definita nella Parte II di questo Manuale). Se l'"ampersand" (il segno "&") non entra in conflitto con il vostro programma di video scrittura, rispondete <enter>. Altrimenti, premete il simbolo che volete usare.

Al principio vorrete usare il dialogo per la correzione di errori. Per far ciò premete <enter>. In questo modo potrete correggere un errore senza dover introdurre di nuovo l'istruzione completa (pag. 10). Quando vorrete attivare, più avanti, questa procedura, userete l'opzione "Y-CHANGE/MODIFY".

```
ENTER A CHARACTER TO BE USED FOR INDICATING MACROS OR A
RETURN FOR DEFAULT CHARACTER OF AMPERSAND (&) <return>

TYPE A RETURN IF THE ERROR CORRECTION DIALOGUE IS TO BE
USED OR ANY OTHER KEY IF NO DIALOGUE IS WANTED: <return>

TYPE "Y" TO SAVE, ANY OTHER CHAR TO ABORT INSTALL
Y

SAVING INSTALLATION PARAMETERS
```

Terminata la procedura d'installazione, potete terminare premendo "Y", oppure annullare l'operazione in corso. Se il vostro terminale non è in lista e battete "Z", dBASE II vi presenta l'elenco delle istruzioni richieste per completare la procedura d'installazione del vostro terminale. Se volete, potete usare questa procedura anche per cambiare i valori normali che avete introdotto nel vostro terminale (per esempio il video inverso con certe istruzioni).

```
USER SUPPLIED SPECS ROUTINE

FOR THIS METHOD,  YOU WILL NEED THE HEX OR DECIMAL CODES
THAT CAN BE SENT FROM YOUR COMPUTER TO THE VIDEO TERMINAL
TO CONTROL IT

THE CODES OR SEQUENCE THAT YOU WILL NEED ARE:

     DELETE A CHAR SEQUENCE
     DIRECT CURSOR POSITIONING SEQUENCE
     CLEAR SCREEN COMMAND
     HOME CURSOR COMMAND
        (CLEAR AND HOME CAN BE COMBINED)
     OPTIONAL:BRIGHT/DIM COMMANDS OR

TYPE "Y" IF YOU WISH TO CONTINUE
Y
```

Se conoscete i codici del vostro terminale, premete "Y" per continuare. dBASE II vi richiederà l'immissione di questi codici. Il prossimo esempio si riferisce a un terminale IBM 3101/12, che non permette l'aumento di intensità di luminosità né il video inverso, per cui alla richiesta relativa si è premuto il tasto <enter>.

Dato che il dBASE II mostra i valori dei byte di controllo introdotti previamente, vi indichiamo quelli introdotti da voi chiudendoli fra due "^". NON BATTETE QUESTI SIMBOLI.

```
WILL YOU BE ENTERING COMMANDS AS HEX OR DECIMAL? TYPE "D"
FOR DECIMAL OR "H" FOR HEXADECIMAL
h

COMMANDS ARE ENTERED AS A SEQUENCE OF NUMBERS TYPE A
CARRIAGE  RETURN TO END A SEQUENCE

NOW  ENTER THE CODES FOR CHARACTER  DELETION THIS IS THE
SEQUENCE "BACKSPACE,  SPACE,  BACKSPACE"  ON MOST TERMINALS
IF THIS IS TRUE FOR YOUR TERMINAL THE TYPE "Y"
Y
```

```
                - DIRECT CURSOR POSITIONING -

    THE CURSOR CONTROL SEQUENCE IS USUALLY A 3 TO 4 BYTE SE-
    QUENCE. THE FIRST ONE OR TWO BYTES ARE USUALLY FIXED AND
    THE REMAINING BYTES CONTAIN THE LINE AND COLUMN NUMBERS

    FIRST,  ENTER THE POSITION IN THE SEQUENCE THAT   HOLDS
    THE COLUMN NUMBER
    "4"
    NEXT,  ENTER THE POSIZION IN THE SEQUENCE THAT    HOLDS
    THE LINE NUMBER
    "3"

    MANY TERMINALS ADD A CONSTANT TO THE LINE AND COLUMN NUM
    BERS.  ENTER THE CONSTANT BIAS FOR YOUR TERMINAL
    "20"

    NOW ENTER THE SKELETON FOR THE DIRECT CURSOR COMMAND, EN
    TER ZERO IN THE PLACES WHERE COLUMN AND LINE NUMBERS  GO
    (11 BYTE MAX)

    ENTER CONTROL CODE BYTE 1:03 "13"
    ENTER CONTROL CODE BYTE 2:00 "59"
    ENTER CONTROL CODE BYTE 3:00 "0"
    ENTER CONTROL CODE BYTE 4:00 "0"
    ENTER CONTROL CODE BYTE 5:00 <return>

    IS THIS CORRECT (Y/N)? y
```

```
            - DIM/BRIGHT VIDEO/REVERSE VIDEO -

    ENTER THE COMMAND THAT WILL SWITCH TO HIGH INTENSITY  OR
    NORMAL VIDEO
    (5 BYTE MAX)

    ENTER CONTROL CODE BYTE 1:1D  <return>

    IS THIS CORRECT (Y/N)? y
```

```
- CLEAR AND HOME COMMAND(S) -

ENTER THE COMMAND(S) THAT WILL CLEAR THE SCREEN AND PLACE
THE CURSOR IN THE UPPER LEFT CORNER OF THE TERMINAL
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1:00 ^1B^
ENTER CONTROL CODE BYTE 2:00 ^4C^
ENTER CONTROL CODE BYTE 3:00 <return>

IS THIS CORRECT (Y/N)? y
```

```
ENTER THE COMMANDS TO BE ISSUED WHEN ENTERING THE    FULL
SCREEN EDITING MODE (IF ANY)
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1:00 <return>

IS THIS CORRECT (Y/N)? y
```

```
ENTER THE COMMAND THAT WILL  SWITCH TO STANDARD  INTENSI-
TY OR NORMAL VIDEO TO RESET THE SCREEN AFTER FULL  SCREEN
OPERATIONS
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1:1D  <return>

IS THIS CORRECT (Y/N)? y
```

```
ENTER THE COMMANDS TO BE ISSUED WHEN LEAVING  THE  FULL
SCREEN EDITING MODE

SUGGESTION:  USE DIRECT CURSOR POSITIONING TO PUT CURSOR
ON THE BOTTOM LINE OF THE SCREEN
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1:1B ^1B^
ENTER CONTROL CODE BYTE 2:17 ^59^
ENTER CONTROL CODE BYTE 3:03 ^31^
ENTER CONTROL CODE BYTE 5:3E <return>

IS THIS CORRECT (Y/N)? y
```

```
ENTER A CHARACTER TO BE USED FOR INDICATING MACROS OR A
RETURN FOR DEFAULT CHARACTER OF AMPERSAND () <return>

TYPE A RETURN IF ERROR CORRECTION DIALOGUE IS TO BE USED
OR ANY OTHER KEY IF NO DIALOGUE IS WANTED <return>

TYPE "Y" SO SAVE,  ANY OTHER CHAR TO ABORT INSTALL

Y
SAVING INSTALLATION PARAMETERS
```

```
MODIFY EXISTING SPECS ROUTINE

FOR THE METHOD, YOU WILL NEED THE HEX OR DECIMAL CODES
THAT CAN BE SENT FROM YOUR COMPUTER TO THE VIDEO TERMI
NAL TO CONTROL IT

TYPE "Y" IF YOU WISH TO CONTINUE
Y

WILL YOU  BE ENTERING COMMANDS AS HEX OR DECIMAL?
TYPE "D" FOR DECIMAL OR "H" FOR HEXADECIMAL
h

COMMANDS ARE ENTERED AS SEQUENCE OF NUMBERS TYPE A CAR
RIAGE RETURN TO END A SEQUENCE

1. DELETE A CHAR SEQUENCE
2. DIRECT CURSOR POSITIONING SEQUENCE
3. CLEAR AND HOME SCREEN COMMAND
4. BRIGHT/STD VIDEO COMMANDS
5. DIM/REVERSE VIDEO COMMANDS
6. INITIALIZATION SEQUENCE
7. EXIT SEQUENCE
8. RESET TO STANDARD VIDEO MODE

SELECT ITEM TO CHANGE
ANY CHAR THAN 1-8 TERMINATE SESSION
```

Per modificare un sistema dBASE II già installato, introducete "INSTALL" seguito da "Y" o "N" in risposta alla richiesta di schermo intero e poi optate per "Y" per quel che riguarda la lista di terminali. dBASE II risponde con una sequenza di comandi. Nell'esempio che segue abbiamo voluto cambiare la sequenza "EXIT" per posizionare il cursore sulla ventitreesima riga invece che sulla diciassettesima al momento di lasciare la modalità di schermo intero. (Su questo punto troverete le spiegazioni più avanti sul manuale).

Da notare che i numeri sono introdotti in esadecimali, le righe sono numerate da 0 a 23 e le colonne da 0 a 79.

```
ENTER COMMANDS TO BE ISSUED WHEN LEAVING THE FULL SCREEN
EDITING MODE

SUGGESTION:  USE DIRECT CURSOR POSITIONING TO PUT CURSOR
ON THE BOTTOM LINE OF THE SCREEN
(11 BYTE MAX)
CURRENT SEQUENCE:
1B
59
31
20

IS THIS CORRECT (Y/N)? n
ENTER CONTROL CODE BYTE 1:1B ^1B^
ENTER CONTROL CODE BYTE 2:59 ^59^
ENTER CONTROL CODE BYTE 3:31 ^31^
ENTER CONTROL CODE BYTE 4:20 ^20^
ENTER CONTROL CODE BYTE 5:00 <return>

IS THIS CORRECT (Y/N)? y
1. DELETE A CHAR SEQUENCE
2. DIRECT CURSOR POSITIONING SEQUENCE
3. CLEAR AND HOME SCREEN COMMAND
4. BRIGHT/STD VIDEO COMMANDS
5. DIM/REVERSE VIDEO COMMANDS
6. INITIALIZATION SEQUENCE
7. EXIT SEQUENCE
8. RESET TO STANDARD VIDEO MODE

SELECT ITEM TO CHANGE
ANY CHAR OTHER THAN 1-8 TERMINATES SESSION

<return>

ENTER A CHARACTER TO BE USED FOR INDICATING MACROS  OR A
RETURN FOR DEFAULT CHARACTER OF AMPERSAND (&) <return>

TYPE A RETURN IF THE ERROR CORRECTION DIALOGUE IS TO  BE
USED OR ANY KEY IF NO DIALOGUE IS WANTED:  <return>

TYPE "Y" TO SAVE, ANY OTHER CHAR TO ABORT INSTALL
Y

SAVING INSTALLATION PARAMETERS
```

Adesso il sistema dBASE II è installato e potete cominciare ad usarlo immediatamente.

Entrate nel sistema battendo "dBASE".

Una linea di richiesta vi chiede la data. Se rispondete, la data sarà registrata nei vostri record come ultimo accesso ogni volta che aggiungerete o toglierete qualcosa dai vostri record e può essere utile per un controllo degli aggiornamenti. Se preferite ignorarla, premete semplicemente <enter> .

Una volta che il dBASE II è caricato in memoria, trasmette un messaggio di inizio e mostra l'indicatore di richiesta o prompt (.) indicando che è pronto ad accettare istruzioni.

Per mostrarvi la potenza e facilità di uso di dBASE II, la prima cosa che faremo sarà quella di creare una base-dati ed introdurvi immediatamente le informazioni. Ci metteremo solo pochi minuti.

S E Z I O N E I

In questa Sezione, creeremo una base-dati introducendo i dati. Inoltre, vi presenteremo alcuni comandi di dBASE II che svilupperemo e amplieremo attraverso il resto del manuale.

Per una definizione completa di un comando, consultate la Parte II.

COME CREARE UNA BASE-DATI

Cominceremo creando una base-dati di nomi per un indirizzario. Ogni record della base-dati conterrà:

```
NOME:      fino a 25 caratteri di lunghezza
INDIRIZZO: fino a 30 caratteri di lunghezza
CAP:       fino a  5 caratteri di lunghezza
CITTA':    fino a 35 caratteri di lunghezza
```

Per prima cosa, scrivete ^CREATE^.

dBASE II risponde ENTER FILENAME:.

Introducete un nome di file iniziando con una lettera e proseguendo fino ad un massimo di 8 caratteri (limite CP/M) senza due punti né spazi. Dato che si tratta di un file di nomi, chiamiamolo appunto: ^Nomi^.

Quando premete il CR, dBASE II crea un file chiamato <NOMI.DBF>. La parte di questo titolo che si trova dopo il punto è l'estensione CP/M dei file e consiste nell'abbreviazione di "file di base-dati" (data base file) (Vedi Sezione V, Tipi di file).

In un sistema gestionale di base-dati, ogni elemento che vogliamo introdurre nei singoli gruppi di relazione si chiama campo (field) ed il loro insieme si chiama record. (Vedi Sezione V, Fondamenti della base-dati). Nel nostro esempio, ogni record ha quattro campi. dBASE ha bisogno di sapere il nome di ciascuno dei campi, che tipo di dato conterrà, di che lunghezza e con quanti posti decimali, se il dato è numerico.

```
.create
ENTER FILENAME:nomi
ENTER RECORD STRUCTURE
AS FOLLOWS:
   FIELD     NAME,TYPE,WIDTH,DECIMAL
   001       PLACES
```

I campi possono essere lunghi fino a 10 caratteri e possono venire introdotti sia in lettere maiuscole sia in lettere minuscole. Il nome deve cominciare con una lettera e non può contenere spazi né apici, ma può contenere cifre e i due punti. Abbreviate solo ciò che è necessario perché, anche se il computer vi capirebbe, le persone forse no.

Il tipo di dati si specifica con una lettera: C per Carattere, N per numerico, L per Logico. Nel nostro caso, tutti i campi contengono dati alfanumerici, ossia caratteri.

L'ampiezza del campo può arrivare ai 254 caratteri. Quando il campo è numerico e il posto dei decimali viene specificato, ricordate che il punto decimale occupa anch'esso uno spazio.

Ora che sappiamo il nome dei nostri campi, il tipo di dato che conterranno e la loro lunghezza, introducete subito le informazioni. Quando avrete finito, lo schermo vi apparirà così:

```
.create
ENTER FILENAME:nomi
ENTER RECORD STRUCTURE AS FOLLOWS:
          NAME,TYPE,WIDTH,DECIMAL
FIELD     PLACES
001       nome,c,25
002       indirizzo,c,30
003       cap,c,5
004       citta prov,c,35
BAD NAME FIELD
004       citta:prov,c,35
005       <return>
```

Notate cosa è successo nel campo 4: il nostro errore è stato quello di lasciare uno spazio all'interno del nome del campo. dBASE II ci segnala l'errore e ci dà la possibilità di correggerlo.

Notate altresì che il tipo di dato per il codice postale è anche esso specificato come "carattere", benché si pensi alle cifre in generale come a numeri.

Questa scelta si è fatta perché il dBASE II ha comandi, come "TOTAL", che possono sommare tutti i campi numerici in un record, senza che voi ne facciate una lista. Ma fare questo con un campo di codici postali sarebbe una semplice perdita di tempo. D'altra parte, possiamo usare gli operatori relazionali ("maggiore di", "minore di", "uguale e non uguale a") sui caratteri, così che non esiste interferenza alcuna con qualsiasi classificazione del codice postale che eventualmente vorremo fare più tardi.

Quando il dBASE II ci ha richiesto le specifiche per un quinto campo, abbiamo premuto il tasto <enter> per terminare la definizione dei dati. Il dBASE II ha salvato la struttura dei dati e ci ha domandato se vogliamo introdurvi dati.

La base-dati <Nomi.DBF> è quindi pronta ad accettare l'informazione: battete "Y". Vi diremo subito come introdurre i vostri dati.

L'INTRODUZIONE DEI DATI NELLA VOSTRA BASE-DATI

Se non avete l'opzione "editing" sul vostro terminale, il numero del record e il nome dei campi vi appariranno uno alla volta in basso anche se voi avete scritto sulla parte alta fino adesso. La lunghezza di ogni campo viene segnalata dai due punti e il cursore si trova nel punto in cui dovete cominciare a scrivere. Appena avete riempito il campo o premuto <enter>, vi apparirà il campo seguente. Dopo aver riempito (o ignorato) l'ultimo campo, cominciate un nuovo record.

Per interrompere l'introduzione di dati, battete <enter> quando il cursore si trova nella posizione del primo carattere del primo campo di un nuovo record.

Se invece avete potuto installare dBASE II con l'opzione di "editing" a schermo intero, il video sarà cancellato e il numero del record e tutti i suoi campi verranno visualizzati a cominciare dall'angolo superiore sinistro dello schermo, con il cursore posizionato al primo carattere del primo campo.

(Se avete scelto uno dei terminali standard del listato d'installazione, i nomi dei campi possono apparire con sfondo inverso o a

media intensità. Se in seguito vorrete cambiare la situazione, potete farlo usando l'opzione esistente nella procedura d'installazione: "Y-CHANGE/MODIFY")

```
RECORD 00001

NOME       :                              :
INDIRIZZO  :                                        :
CAP        . :     :
CITTA:PROV:                                                 :
```

NOTA: Se il nostro esempio non coincide con il vostro schermo, deve esserci qualche problema con la procedura d'installazione. Ripetete tutta la procedura.

La lunghezza dei campi è indicata dai due punti. Sia che il campo sia stato riempito oppure premendo <enter>, il cursore salta al campo seguente. Il cursore può essere riportato a un campo precedente se, mantenendo premuto il tasto di controllo, battete una volta la lettera E: ^control-E^, abbreviato ^ctl-E^. Quando avrete finito con l'ultimo campo, dBASE II vi presenterà un altro record da riempire.

Introducete i nomi e gli indirizzi seguenti. Li useremo presto, per mostrarvi alcune caratteristiche potenti del dBASE II.

```
AROSIO,ANGELA            V.Induno 12           20092 Cinisello
BANDINO,PAOLO            V.Savelli 30          40138 Bologna
BATTAGLIA,CESARE         V.Pitré 22            00162 Roma
DE PAOLIS,LIANA          C.so Garibaldi 8      40126 Bologna
FRASSINETTI,MARIA PIA    V.Nannetti 15         40069 Zola Pedrosa,
GIANNESINI,ANDREA        V.Cesarea 89/19       16121 Genova
GIORELLO,FRANCESCO       V.Fabio Filzi 37      35100 Padova
MALDACEA,LUIGI           V.Notarbartolo 7      90141 Palermo
```

Nel caso in cui facciate qualche errore che non possa essere corretto tornando indietro e riscrivendoci sopra, leggete le prossime due pagine su "editing" (composizione) *prima* di muovervi dal record da correggere per passare a quello nuovo. Se poi, per errore, tornate al punto di richiamo (dot prompt), battete:

^USE Nomi^
^APPEND^

e continuate il vostro lavoro di introduzione di dati. (Più avanti, il manuale vi spiegherà perché).

Per interrompere l'introduzione di dati, dopo aver riempito l'ultimo campo di un record e trovandovi al primo carattere del primo campo del record seguente, battete <enter>. Se avete già introdotto una parte dell'informazione o se avete mosso il cursore, mantenete abbassato il tasto di controllo e battete la lettera "Q" (^control-Q^).

In questo modo, dBASE II interrompe il comando d'introduzione dati e presenta il punto di richiamo (.) per mostrare che è pronto per nuove istruzioni.

Se, in questo momento, volete smettere, battete semplicemente ^QUIT^.

La battitura di ^QUIT^ deve essere effettuata *ogni volta* che terminate una sessione di lavoro con il dBASE II. Così facendo, si chiudono tutti i file automaticamente. Se non lo fate, la vostra base-dati può venire distrutta.

NOTA : Più avanti vi spiegheremo come potete preparare i FORMAT FILES (estensione.FMT) mediante i quali potete "formattare" (dotare del formato voluto) il vostro schermo.

Quando si usano questi files, i campi del record e le chiamate del programma verranno visualizzate dove volete voi e non secondo il listato automatico che vi è stato appena presentato. Con un file formattato, potete anche usare solamente i campi che voi selezionate, invece di tutti i campi della base-dati. Ciò si applica ai comandi CREATE, EDIT, INSERT e APPEND.

LA MODIFICA DEI DATI CON ^EDIT^ E ^BROWSE^

Se fate qualche errore durante l'inserimento dati, potete correggerlo rapidamente e con facilità con il sistema di correzione a Schermo Intero (Full Screen Edit). Battete:

^USE NOMI^
^EDIT <numero>^

dove il "numero" è quello di uno dei record della base-dati.

Il dBASE II vi presenterà il record completo e voi potrete usare i comandi della Composizione a Schermo Intero per modificare ciò che volete nel vostro record. Per avanzare al record seguente, usate ^ctl-C^. Per retrocedere di un record, usate ^ctl-R^. Provateci battendo ^EDIT 3^.

```
RECORD 00003

NOME       :BATTAGLIA,CESARE          :
INDIRIZZO  :V.Pitré 22                :
CAP        :00162                     :
CITTA:PROV:ROMA
```

Adesso, se volete segnare un record per poi cancellarlo, usate ^ctl-U^ e vi apparirà la parola "DELETED" in alto sullo schermo. Premendo ^ctl-U^ una seconda volta la parola scomparirà e il record sarà ripristinato. Se voi richiamate la vostra base-dati tramite i comandi ^LIST^ o ^DISPLAY^, vedrete un asterisco a fianco di ciascuno dei record segnati per la cancellazione.

Per terminare il comando di correzione a schermo intero senza salvare le modifiche inserite, usate ^ctl-Q^.

Nel caso vogliate terminare salvando le modifiche inserite, bisogna usare ^ctl-W^ (^ctl-O^ - la lettera "O" - con Superbrain).

BROWSE FIELDS [<Elenco dei campi>]

è uno dei comandi più potenti del dBASE II per correggere e visualizzare dati.

BROWSE (ripassare) vi può mostrare fino a 19 record, visualizzando tanti campi di ciascun record quanti ne possono stare nella vostra "finestra". ^ctl-B^ muove la finestra verso destra di un campo, e ^ctl-Z^ la muove verso sinistra.

I campi (non i record) con più di 80 caratteri "si avvolgono" intorno al vostro schermo, in modo che voi potete vedere sempre l'intero campo. Se specificate una lista di campi (separati da virgole), avrete la visualizzazione di quei campi solamente.

I comandi FULL SCREEN (schermo intero) e EDIT (correzione) muoveranno il vostro cursore all'interno dei campi, di campo in campo e di record in record. Potrete fare qualsiasi modifica vogliate all'interno della vostra base-dati, che saranno salvate con ^ctl-W^, oppure saranno ignorate, se uscite con ^ctl-Q^.

CARATTERISTICHE DELLO SCHERMO INTERO

ctl-X muove il cursore al campo seguente (o ctl-F).
ctl-E muove il cursore al campo precedente (o ctl-A).
ctl-D muove il cursore avanti di un carattere.
ctl-S muove il cursore indietro di un carattere.
ctl-V commuta tra il modo di riscrittura e quello di inserimento.
ctl-G sopprime i caratteri sotto il cursore.
<Rubout> sopprime il carattere a sinistra del cursore.
ctl-P accende e spegne la vostra stampante.
ctl-W salva tutte le modifiche fatte e riprende l'attività normale del dBASE II. Con Superbrain, il comando analogo è ^ctl-O^ (la lettera "O").
ctl-Q abbandona e torna alla attività normale del dBASE II senza effettuare modifiche, nemmeno se si trova nel modo MODIFY.

FUNZIONI "EDIT" E "BROWSE"

(da non usare quando ci si trova nel modo "APPEND")

ctl-C scrive il record sul disco e avanza al prossimo record.
ctl-R scrive il record sul disco e retrocede al record precedente.
ctl-U commuta il segno di cancellazione.

FUNZIONI DI "BROWSE":

ctl-B sposta la finestra verso destra di un campo.
ctl-Z sposta la finestra verso sinistra di un campo.

FUNZIONI DI "MODIFY":

ctl-T elimina il campo in cui ci troviamo e muove verso l'alto tutti i campi che ne sono al di sotto.
ctl-Y libera gli spazi a destra del cursore e lascia il resto dei campi come si trovano.
ctl-N muove i campi verso il basso di una posizione per permettere l'inserimento di un nuovo campo nel punto in cui si trova il cursore.

FUNZIONI DI "APPEND":

ctl-R scrive il record sul disco e si muove sul record seguente.
<Enter> permette di tornare alla attività normale del dBASE II, quando il cursore si trova sulla posizione iniziale di un nuovo record.
ctl-Q cancella il record e ritorna alla attività normale del dBASE II.

INTRODUZIONE AI COMANDI DI dBASE II E AL COLLOQUIO DI CORREZIONE DI ERRORI (USE, LIST, DISPLAY)

Generalmente i comandi del dBASE II sono verbi. Si possono scrivere all'apparire del punto (.) di richiamo.

Quando volete far sapere al dBASE II con quale file della base-dati volete lavorare, battete sulla tastiera ^USE nome del file ^.

Per vedere su quale record vi trovate, battete ^DISPLAY^.

Per vedere tutti i record immagazzinati nella base-dati, battete ^LIST^. (Per fermare e ricominciare il listato che scorre, usate ^ctl-S^).

I comandi del dBASE II si possono abbreviare fino a solo quattro lettere, ma se ne usate più di quattro ognuna di esse deve essere corretta (^DISPLAY^, ^DISP^ e ^DISPLA^ sono validi, ma ^DISPRAY^ non lo è).

Se, quando avete installato il dBASE II, avete scelto il colloquio di correzione di errori, la linea di comando viene controllata e ogni volta che si diagnostica un errore ne ricevete il messaggio. In questo modo, avete una seconda possibilità e potete correggere senza bisogno di riscrivere tutta la riga.

Battete ^EDUT 3^.

```
Edut 3
***UNKNOWN COMMAND
Edut 3
correct and retry (Y/N)? y
CHANGE FROM:u
CHANGE TO:i
Edit 3
MORE CORRECTIONS(Y/N)? n
```

Il dBASE II ripete i comandi che non conosce. Se decidete di cambiarli, non avete bisogno di ripetere tutto il comando.

In risposta a "CHANGE FROM:" si batte di nuovo un numero sufficiente di caratteri della parte errata così da evitare qualsiasi ambiguità, e poi si preme <enter>.

In risposta a "CHANGE TO:" si battono le lettere in sostituzione.

Nel nostro esempio abbiamo cambiato una sola lettera, ma troverete questa opzione particolarmente utile nel caso abbiate inserito un comando con più lettere.

NOTA: Il comando "ERASE" cancella lo schermo e posizione il punto di richiamo nell'angolo superiore sinistro in modo che possiate ripartire con nuovi comandi.

ESPANSIONE DEI COMANDI CON ESPRESSIONI E OPERATORI RELAZIONALI

Una delle più potenti caratteristiche del dBASE II è la possibilità di espandere "su misura" i suoi comandi.

Infatti, in questo modo potete aggiungere "frasi" ed espressioni alla maggior parte dei comandi in modo che si possa poi definirne esattamente la loro funzione. I comandi si possono introdurre sia in maiuscole che in minuscole ed ogni linea di comando può essere lungo fino a 254 caratteri.

Per estendere il comando su più di una riga, battete un punto e virgola (;) dopo l'ultimo carattere della riga (senza farlo seguire da alcuno spazio). Il dBASE II userà allora la riga seguente come parte del comando.

Dato che il dBASE II è un Sistema di Gestione di Base-Dati (DBMS) di tipo relazionale, vi troverete gli utilissimi operatori relazionali:

<	:	minore di
>	:	maggiore di
=	:	uguale a
<=	:	minore o uguale a
>=	:	maggiore o uguale a

Questi comandi vogliono dire esattamente ciò che indicano. Come risultato, generano un valore logico (Vero o Falso). Se l'espressione è Vera (True), il comando la esegue, mentre, se è Falsa, (False) non la esegue.

Poco fa abbiamo detto che il comando LIST ci mostra tutti i records contenuti nella base-dati (per partire o fermare lo scorrimento verticale, usate ^ctl-S^). La forma completa del comando è

^LIST [OFF] [FOR <espressione>]

Se si utilizza l'opzione OFF i numeri di record non verranno mostrati.

Se invece si usa l'opzione FOR, dBASE II mostrerà solamente i record che corrispondono all'espressione. Provate a battere le seguenti istruzioni, ricordandovi di usare la virgoletta semplice (apostrofo) prima e dopo i caratteri appartenenti al dato (nella Sezione II riparleremo dei tipi di dato):

```
^USE Nomi^
^LIST^
^LIST OFF^
^LIST FOR Cap='4'^
^LIST FOR Cap<'2'^
^LIST FOR Nome='MALDACEA'^
```

E' da notare che quando introduciamo una parte del contenuto di un campo, il dBASE II confronta solamente quella. Per esempio noi dovevamo scrivere il nome completo del Sig. Maldacea solo in caso ci fossero stati altri Maldacea nella nostra base-dati.

```
.use nomi
.list
00001 AROSIO,ANGELA          V.Induno 12        20092  Cinisello
00002 BANDINO,PAOLO          V.Savelli 30       40138  Bologna
00003 BATTAGLIA,CESARE       V.Pitré 22         00162  Roma
00004 DE PAOLIS,LIANA        C.so Garibaldi 8   40126  Bologna
00005 FRASSINETTI,MARIA PIA  V.Nannetti 15      40069  Zola Pedr
00006 GIANNESINI,ANDREA      V.Cesarea 89/19    16121  Genova
00007 GIORELLO,FRANCESCO     V.F.Filzi 37       35100  Padova
00008 MALDACEA,LUIGI         V.Notarbartolo 7   90141  Palermo

.list off
AROSIO,ANGELA          V.Induno 12          20092 Cinisello
BANDINO,PAOLO          V.Savelli 30         40138 Bologna
BATTAGLIA,CESARE       V.Pitré 22           00162 Roma
DE PAOLIS,LIANA        C.so Garibaldi 8     40126 Bologna
FRASSINETTI,MARIA PIA  V.Nannetti 15        40069 Zola Pedrosa
GIANNESINI,ANDREA      V.Cesarea 89/19      16121 Genova
GIORELLO,FRANCESCO     V.F.Filzi 37         35100 Padova
MALDACEA,LUIGI         V.Notarbartolo 7     90141 Palermo


.list for cap = '4'
00002 BANDINO,PAOLO          V.Savelli 30       40138 Bologna
00004 DEPAOLIS,LIANA         C.so Garibaldi 8   40126 Bologna
00005 FRASSINETTI,MARIAPIA   V.Nannetti 15      40069 Zola Pedr

.list for cap < '2'
00003 BATTAGLIA,CESARE       V.Pitré 22         00162 Roma
00006 GIANNESINI,ANDREA      V.Cesarea 89/19    16121 Genova

.list for nome ='MALDACEA'
00008 MALDACEA,LUIGI         V.Notarbartolo 7   90141 Palermo
```

Oltre ad utilizzare questo comando (LIST) per selezionare determinati dati, potete farlo anche per ottenere informazioni del sistema stesso.

"LIST STRUCTURE" vi mostrerà la struttura della base-dati in uso (quella su cui state lavorando).

"LIST FILES" vi mostra i nomi dei files (.DBF) esistenti nel disco in funzione.

"LIST FILES ON <drive>" vi mostra i files (.DBF) che si trovano su un altro drive (NON USATE i due punti abituali del CP/M).

```
.use nomi
.list structure
STRUCTURE FOR FILE:NOMI.DBF
NUMBER OF RECORDS:00008
DATE OF LAS UPDATE:00/00/00
PRIMARY USE DATABASE
FLD       NAME          TYPE          WIDTH          DEC
001       NOME           C             025
002       INDIRIZZO      C             030
003       CAP            C             005
004       CITTA:PROV     C             035
**TOTAL**                              00096
.list files
DATABASE  FILES         #RCDS          LAST UPDATE
NOMI      DBF           00008          00/00/00
```

VISUALIZZAZIONE DEI DATI (DISPLAY)

Il comando "DISPLAY" è simile al "LIST". La sua forma più completa è:

```
              [All]
DISPLAY       [Record n] [OFF] [FOR <espressione>]
              [Next n]
```

Ciò vi dà la possibilità di specificare lo "scopo" del comando "DISPLAY" (così come di "LIST").

La specifica "Record n" visualizza solo il record col numero indicato; "Next n" visualizza "n" record successivi, includendo quello su cui siamo posizionati; "DISPLAY ALL" è uguale a "LIST", con la differenza che quest'ultimo fa scorrere sullo schermo tutti i record esistenti nella base-dati mentre "DISPLAY ALL" li presenta a gruppi di 15 per volta (premendo un tasto qualsiasi si passa al gruppo seguente).
Battete:

"DISPLAY ALL"
"DISPLAY Record 3"
"DISPLAY Next 4"

```
.display all
00001 AROSIO,ANGELA           V.Induno 12        20092 Cinisello
00002 BANDINO,PAOLO           V.Savelli 30       40139 Bologna
00003 BATTAGLIA,CESARE        V.Pitré 22         00162 Roma
00004 DE PAOLIS,LIANA         C.so Garibaldi 8   40126 Bologna
00005 FRASSINETTI,MARIA PIA   V.Nannetti 15      40069 Zola Pedr.
00006 GIANNESINI,ANDREA       V.Cesarea 89/19    16121 Genova
00007 GIORELLO,FRANCESCO      V.F.Filzi 37       35100 Padova
00008 MALDACEA,LUIGI          V.Notarbartolo     90141 Palermo

.display record 3
00003 BATTAGLIA,CESARE        V.Pitré 22         00162 Roma

.display next 4
00003 BATTAGLIA,CESARE        V.Pitré 22         00162 Roma
00004 DE PAOLIS,LIANA         C.so Garibaldi 8   40126 Bologna
00005 FRASSINETTI,MARIA PIA   V.Nannetti 15      40060 Zola Pedr.
00006 GIANNESINI,ANDREA       V.Cesarea 89/19    16121 Genova
```

Così come per il comando "LIST", la clausola opzionale FOR si può usare per selezionare dei dati specifici servendosi di espressioni logiche.

Il comando DISPLAY si può usare, come LIST, per richiamare le funzioni del sistema:

DISPLAY STRUCTURE = LIST STRUCTURE
DISPLAY FILES = LIST FILES

Sia "LIST" sia "DISPLAY" possono presentarvi un tipo specifico di file nel drive, mediante le "maschere" CP/M. "DISPLAY FILES LIKE *.COM ON B", per esempio, visualizzerà tutti i files ".COM" che si trovano nel drive B. Se non siete sicuri, controllate sul vostro Manuale CP/M e poi usate questa forma:

"DISPLAY FILES LIKE <maschera>"

COMANDI DI POSIZIONAMENTO (GO, GOTO E SKIP)

Una volta costruita la vosta base-dati, potete perfettamente muovervi da record a record con rapidità.
SCRIVETE:

"USE Nomi"
"GO TOP"
"DISPLAY"
"GO BOTTOM"
"DISPLAY"
"GOTO 5"
"DISPLAY"
"8"
"DISPLAY"

```
.use nomi
.go top
.display
00001 AROSIO,ANGELA     V.Induno 12    20092 Cinisello

.go bottom
.display
00008 MALDACEA,LUIGI    V.Notarbartolo 7 90141 Palermo

.goto 5
.display
00005 FRASSINETTI,MARIA PIA  V.Nannetti 15  40069 Zola Pedr.

.8
.display
00008 MALDACEA,LUIGI    V.Notarbartolo 7 90141 Palermo
```

"GO TOP" (oppure "GOTO TOP") vi porta al primo record della base-dati. "GO BOTTOM" vi porta all'ultimo record. Inoltre potete andare ad un record determinato usando "GOTO <numero>" (oppure "GO <numero>"). Potete perfino eliminare il GO e specificare semplicemente il numero del record.

"SKIP" salta al record seguente. "SKIP± n" vi fa avanzare o retrocedere di "n" record. Potete anche usare "SKIP ± <variabile/espressione>", dove il valore della variabile o dell'espressione (che vi definiremo più avanti) determinano il numero di record da saltare (skip).
Scrivete:

```
"DISPLAY"
"SKIP-3"
"DISPLAY"
"SKIP"
"DISPLAY"
```

```
.display
00008 MALDACEA,LUIGI          V.Notarbartolo 7 90141 Palermo

.skip-3
RECORD:00005

.display
00005 FRASSINETTI,MARIA PIA   V.Nannetti 15    40069 Zola Pedr

.skip
RECORD:00006

.display
00006 GIANNESINI,ANDREA       V.Cesarea 89/19  16121 Genova
```

IL COMANDO INTERATTIVO "?"

Il comando "?" vi permette di usare dBASE II come se fosse una calcolatrice. Battete semplicemente il punto interrogativo, uno spazio e poi la quantità o la funzione matematica di cui avete bisogno e il dBASE II provvederà a rispondervi nella riga seguente. Se si usano due punti interrogativi _??) la risposta apparirà sulla stessa riga.

Inserite:

```
"? 73/3.0000"
"? 73.00/3"
"? 73/3"
```

```
.? 73/3.0000
   24.3333
.? 73.00/3
24.33
.? 73/3
24
.STORE 73/3 to s
24
```

Il comando "?" mostra il risultato di una operazione matematica con la stessa precisione del numero introdotto con più decimali.

Ma "?" può anche voler dire: "Che cosa è.. ", dove al posto dei puntini ci sia un'espressione, una variabile (il nome di un campo oppure una variabile di memoria), una funzione del dBASE II o una lista di tutti questi elementi separati da virgole. Scrivete:

```
"USE Nomi"
"6"
"? Cap"
"? Nome"
"SKIP"
"GO BOTTOM"
"? Citta:Prov"
```

```
.use nomi
.6
.? cap
16121
.? nome
GIANNESINI,ANDREA
.skip
RECORD:00007
.? nome
GIORELLO,FRANCESCO
.go bottom
.? citta:prov
PALERMO
```

Nella Sezione dedicata alle funzioni e ai comandi, vi mostreremo come si possa usare "?" anche per accedere ad altre funzioni del dBASE II e per rappresentare sul video le chiamate all'operatore in un file di comandi.

L'AGGIUNTA DI DATI TRAMITE APPEND e INSERT

Con dBASE II potete aggiungere dati a qualsiasi base-dati rapidamente e con facilità usando un comando di una sola parola. Prima

di tutto scegliete il file della base-dati in cui volete introdurre le nuove informazioni, battendo "USE <nome del file>", e poi battete "APPEND":

"USE Nomi"
"APPEND"

```
.use nomi
.append

RECORD # 00009

NOME      :
INDIRIZZO :
CAP       :     :
CITTA:PROV:
```

Il dBASE II risponde presentando il numero di record successivo all'ultimo record nel file e i suoi campi. Questo record sarà aggiunto alla fine del file (in coda) dopo che lo avrete compilato.

Sul video compaiono tutti i nomi dei campi, con i due punti che segnano la lunghezza di ciascuno. Il cursore si trova sulla prima posizione da cui potete cominciare a introdurre dati. Se riempite tutto il campo, automaticamente il cursore si posizionerà sul campo seguente. In caso contrario, battete <enter>.

Se in qualche campo non dovete introdurre alcun dato, usate <enter> ; così facendo portate il cursore al campo seguente. I campi di caratteri resteranno riempiti automaticamente da spazi in bianco, i campi numerici invece mostreranno uno zero. Quando si introducono dati numerici, se non vi sono cifre decimali non è necessario battere il punto decimale perché il dBASE II mette automaticamente il punto e gli zeri necessari.

I record possono venire inseriti in una determinata posizione nella base-dati (per esempio, in ordine alfabetico) battendo questo comando:

"INSERT [BEFORE] [BLANK]"

Se si usa solo la parola "INSERT" il record viene semplicemente aggiunto dopo il record su cui ci troviamo. Se specificate BEFORE verrà aggiunto immediatamente prima di questo. In ogni caso, riceverete le stesse richieste che si presentano con i comandi "APPEND" e "CREATE". Se specificate BLANK, viene inserito un record in bianco e non si presentano richieste.

Ed ora aggiungete i nomi seguenti in ordine alfabetico alla vostra base-dati <Nomi.DBF>:

DE CRESCENZO,PAOLO	V.le Regina Margherita 101	00198 Roma
MERLI,SILVANO	V.Imparato 30	80146 Napoli
RAVASIN,ROBERTO	V.Santa Caterina 23	24100 Bergamo

La sequenza di comandi è:

"USE Nomi"
"4"
"INSERT BEFORE" (introducete il dato del primo nome)
"APPEND" (introducete i dati degli altri due nomi)

Col comando "INSERT", appena riempito l'ultimo campo, il dBASE II ritornerà al punto di richiamo.

Per uscire dal comando "APPEND" battete <enter> quando vi trovate sul nuovo record in bianco.

In entrambi i modi si può uscire tramite il "ctl-W" ("ctl-O" con Superbrain). Con ciò salveremo i dati che abbiamo introdotto finora e torneremo al modo di comando.

Sia con i comandi di "APPEND" sia con quelli di "INSERT" impareremo più avanti a costruire un FORMAT FILE, ossia un file di formati (.FMT extension), con il quale potremo visualizzare solo i campi della base-dati che selezioneremo ed i messaggi speciali che vorremo includervi.

LA PULIZIA DI UNA BASE-DATI (DELETE, RECALL, PACK)

Le eliminazioni si possono fare direttamente con il dBASE II così come con il comando ^EDIT^.

Per eliminare il record in cui vi trovate, battete ^DELETE^.

Per eliminare più di un record, usate la forma ^DELETE [scopo] [FOR <espressione>]^, dove "espressione" è una condizione o una serie di condizioni che devono verificarsi. (Vi diremo di più su questo punto nella Sezione II).

Per eliminare un file, battete ^DELETE FILE <drive>:<nome del file>^. Ma ricordate che quando avrete fatto ciò i dati saranno scomparsi per sempre: state attenti.

A differenza dei file, i record segnati per l'eliminazione si possono recuperare. Infatti ^DELETE^, invece di cancellare i dati, marca ogni record con un asterisco. Mentre il dBASE II ignora questi record e non li usa in nessuna delle sue elaborazioni, voi vedrete il loro contenuto ogni volta che effettuerete i comandi ^LIST^ e ^DISPLAY^.

Per ripristinare questi record, usate il comando:

^RECALL [scopo] [FOR <espressione>]^.

Questa funzione è uguale a quella di ^DELETE^ e lo scopo e la condizione sono opzionali. Se si usa una espressione condizionale, non deve necessariamente essere la stessa che si è usata per marcare l'eliminazione.

Ad ogni modo, se a un certo punto volete ripulire i vostri file per ottenere visualizzazioni più chiare o per far posto sul dischetto, battete:

^PACK^

Tutti i record segnati verranno cancellati e saprete quanti ne rimangono nella vostra base-dati.

NOTA: Ricordate che una volta usato questo comando i record saranno perduti per sempre.

rovate voi stessi come funzionano questi comandi. Battete:

```
USE Nomi
LIST
DELETE RECORD 2
DELETE RECORD 5
LIST
RECALL RECORD 5
LIST
PACK
LIST
```

co come vedremo sullo schermo i primi record del nostro file <Nomi.DBF>:

```
. use nomi
. list
00001 AROSIO,ANGELA            V.Induno 12              20092 Cinisello
00002 BANDINO,PAOLO            V.Savelli 30             40138 Bologna
00003 BATTAGLIA,CESARE         V.Pitré 22               00162 Roma
00004 DE CRESCENZO,PAOLO       V.le R.Margherita 101    00198 Roma
00005 DE PAOLIS,LIANA          C.so Garibaldi 8         40126 Bologna
00006 FRASSINETTI,PARIA PIA    V.Nannetti 15            40069 Zola P.
00007 GIANNESINI,ANDREA        V.Cesarea 89/19          16121 Genova
00008 GIORELLO,FRANCESCO       V.F.Filzi 37             35100 Padova
00009 MALDACEA,LUIGI           V.Notarbartolo 7         90141 Palermo
00010 MERLI,SILVANO            V.Imparato 30            80146 Napoli
00011 RAVASIN,ROBERTO          V.Santa Caterina 23      24100 Bergamo

. delete record 2
00001 DELETION(S)
. delete record 5
00001 DELETION(S)
. list
00001 AROSIO,ANGELA            V.Induno 12              20092 Cinisello
00002*BANDINO,PAOLO            V.Savelli 30             40138 Bologna
00003 BATTAGLIA,CESARE         V.Pitré 22               00162 Roma
00004 DE CRESCENZO,PAOLO       V.le R.Margherita 101    00198 Roma
00005*DE PAOLIS,LIANA          C.so Garibaldi 8         40126 Bologna
00006 FRASSINETTI,MARIA PIA    V.Nannetti 15            40069 Zola P.
00007 GIANNESINI,ANDREA        V.Cesarea 89/19          16121 Genova
00008 GIORELLO,FRANCESCO       V.F.Filzi 37             35100 Padova
00009 MALDACEA,LUIGI           V.Notarbartolo 7         90141 Palermo
00010 MERLI,SILVANO            V.Imparato 30            80146 Napoli
00011 RAVASIN,ROBERTO          V.Santa Caterina 23      24100 Bergamo
```

```
  .recall record 5
00001 RECALL(S)
  .list
00001 AROSIO,ANGELA              V.Induno 12              20092 Cinisello
00002*BANDINO.PAOLO              V.Savelli 30             40138 Bologna
00003 BATTAGLIA?CESARE           V.Pitré 22               00162 Roma
00004 DE CRESCENZO,PAOLO         V.le R.Margherita 101    00198 Roma
00005 DE PAOLIS,LIANA            C.so Garibaldi 8         40126 Bologna
00006 FRASSINETTI,MARIA PIA      V.Nannetti 15            40069 Zola P.
00007 GIANNESINI,ANDREA          V.Cesarea 89/19          16121 Genova
00008 GIORELLO,FRANCESCO         V.F.Filzi 37             35100 Padova
00009 MALDACEA,LUIGI             V.Notarbartolo 7         90141 Palermo
00010 MERLI,SILVANO              V.Imparato 30            80146 Napoli
00011 RAVASIN,ROBERTO            V.Santa Caterina 23      24100 Bergamo

  .pack
PACK COMPLETE,00010RECORDS COPIED
  .list
00001 AROSIO,ANGELA              V.Induno 12              20092 CInisello
00002 BATTAGLIA,CESARE           V.Pitré 22               00162 Roma
00003 DE CRESCENZO PAOLO         V.le R.Margherita 101    00198 Roma
00004 DE PAOLIS,LIANA            C.so Garibaldi 8         40126 Bologna
00005 FRASSINETTI,MARIA PIA      V.Nannetti 15            40069 Zola P.
00006 GIANNESINI,ANDREA          V.Cesarea 89/19          16121 Genova
00007 GIORELLO,FRANCESCO         V.F.Filzi 37             35100 Padova
00008 MALDACEA,LUIGI             V.Notarbartolo 7         90141 Palermo
00009 MERLI,SILVANO              V.Imparato 30            80146 Napoli
00010 RAVASIN,ROBERTO            V.Santa Caterina 23      24100 Bergamo
```

RIASSUNTO DELLA SEZIONE I

A questo punto vi è chiara la potenza che un sistema di gestione relazionale di base-dati come il dBASE II può fornire.

Potete, ora, creare (^CREATE^) una nuova base-dati in pochi minuti e cominciare a introdurvi i vostri dati.

Se poi volete effettuare dei cambi, lo farete facilmente con ^EDIT^, ^DELETE^, ^RECALL^ e ^PACK^.

Se volete, potete aggiungere (^APPEND^) o inserire (^INSERT^) altri dati che vi sono necessari, e così pure listare (^LIST^) e visualizzare (^DISPLAY^) interi file oppure determinati record. Potete anche posizionarvi (^GO^, ^GOTO^, ^GO BOTTOM^) e saltare su e giù per la vostra base-dati, in modo rapido e facile.

Inoltre, potete usare il dBASE II in modo interattivo e farne una potente calcolatrice (e qualcosa di più) mediante il comando ^?^.

Vi abbiamo anche presentato le espressioni e il modo con cui si possono utilizzare per potenziare i comandi del dBASE II. Nella prossima Sezione approfondiremo questa parte e vi mostreremo come estrarre facilmente dalla vostra base-dati le informazioni che vi servono.

Prima, però, dovete "creare" altri due file perché ne avremo bisogno per produrre nuovi esempi:

S E Z I O N E II

```
.create
ENTER FILENAME:Pagamenti
ENTER RECORD STRUCTURE
AS FOLLOWS:
FIELD          NAME,TYPE,WIDTH,DECIMALPLACES
001            Ass.Data,C,7
002            Ass.Num,C5
003            Cliente,C,3
004            Serv:Num,N,3
005            Nome,C,20
006            Descr,C,20
007            Importo,N,9,2
008            Fatt:Data,C,7
009            Fatt:Num,C,7
010            Ore,N,6,2
011            Imp:Num,N3
012
```

```
.create
ENTER FILENAME:Ordini
ENTER RECORD STRUCTURE
AS FOLLOWS:

FIELD          NAPE,TYPE,WIDTH,DECIMAL PLACES
001            Num:Client,C,9
002            Articolo,C,20
003            Quantita,N,4
004            Prezzo,N,7,2
005            Importo,N,9,2
006            Ord:Data,C,6
007
```

In questa Sezione sviluppiamo l'uso delle espressioni per modificare i comandi del dBASE II. E' forse la parte più importante dell'insieme di nozioni che dovete apprendere per utilizzare efficacemente il dBASE II.

I comandi del dBASE II si possono imparare con facilità perché si basano sul linguaggio naturale e quotidiano: praticamente, imparare un nuovo comando è come accrescere il vostro vocabolario (e il vostro repertorio gestionale) di un altra parola.

Le espressioni, combinate con i comandi, vi forniscono il perfetto controllo necessario per maneggiare i dati allo scopo di realizzare obiettivi specifici. Quando avrete imparato a manipolare le espressioni, vi restano da imparare altre due cose sulla programmazione, che vi permetteranno di scrivere programmi applicativi. (Queste sono: come prendere decisioni e come ripetere una sequenza di comandi. Le troverete nella Sezione III).

USO DELLE ESPRESSIONI PER LA SELEZIONE E IL CONTROLLO

Nella Sezione I avete ricevuto una breve introduzione alle espressioni che si possono usare con i comandi del dBASE II.

Come abbiamo visto, si tratta di una potente maniera di estendere i comandi e di maneggiare facilmente i vostri dati con rapidità. Molti comandi del dBASE II si possono modificare con questa formula:

```
<COMANDO> [FOR <espressione>]
```

Questa estensione vi dà una flessibilità che semplicemente non è possibile ottenere con altri sistemi di gestione di base-dati. Programmatori esperti ci hanno riferito che un programma in dBASE II per una applicazione si può sviluppare in una decima parte del tempo che ci vorrebbe con il BASIC e perfino con linguaggi di livello più alto come COBOL, FORTRAN e PL/1.

Ma, per fare ciò, è necessario che voi impariate come lavorare con le espressioni e gli operatori e come combinare i comandi nei file di comandi perché le stesse operazioni si ripetano tante volte quante sono necessarie.

Le prossime pagine di questo manuale vi avvieranno su questa strada e, in definitiva, l'esperienza che andrete acquistando sarà la vostra migliore maestra.

Ricordate: Mentre vi presentiamo i comandi, cercheremo di spiegarvene un aspetto particolare che vi può permettere la realizzazione di qualche altra operazione nella vostra base-dati. Ma queste particolarità vengono spiegate singolarmente. Per trovare riunite tutte le prerogative di un comando, ricorrete al sommario che conclude la Parte I e le definizioni contenute nella Parte II.

NOTA: Se, quando avrete finito di studiare questa Sezione, vi troverete ancora incerti su come scrivere le espressioni in modo tale che i comandi di dBASE II eseguano esattamente ciò che volete, vi potrà essere utile prendere visione di qualche testo di programmazione, che sicuramente troverete nella biblioteca più vicina. La maggior parte di questi testi spiegano le espressioni nei primi capitoli.

COSTANTI E VARIABILI (STORE)

In dBASE II le espressioni si usano come ausilio per la selezione e la manipolazione dei dati nella vostra base-dati (Vedi "DISPLAY"). La quantità che state trattando può essere sia una costante sia una variabile.

Le costanti sono quei dati che non cambiano, senza che importi in che luogo della base-dati o del computer si trovino. Queste costanti sono valori letterali perché sono esattamente ciò che rappresentano. Per esempio, una cifra come il 3 e valori logici come la T e la F sono delle costanti.

I caratteri e le stringhe di caratteri (tutti i caratteri di stampa più gli spazi) possono essere delle costanti ma devono essere trattate in modo un po' diverso.

Le "stringhe" sono semplicemente un insieme di caratteri (compresi gli spazi, i numeri e i simboli) che si trattano, si modificano o si usano in qualsiasi modo, come dati. Una "sottostringa" è una porzione di una qualsiasi stringa.

Se un carattere o un insieme di caratteri deve essere trattata come una stringa costante, bisogna chiuderla fra virgolette semplici o doppie, oppure fra parentesi quadre, in modo che il computer comprenda che ha a che fare con dei caratteri come tali. Vediamo cosa significa questo. Battete:

```
"dBASE"
"USE Nomi"
"? 'Nome'"
"? Nome"
```

Al primo "Che cos'è ..." (il comando "?"), il computer risponde NOME perché questo è il valore della costante. Quando voi eliminate le virgolette, il computer per prima cosa controlla, per vedere se la parola era un comando. Non lo era, e allora controlla se era il nome di una variabile.

Le variabili sono elementi che possono cambiare. Spesso sono i nomi dei campi della base-dati, il cui contenuto può cambiare. In questo caso, il computer scopre che la nostra base-dati ha un campo chiamato <Nome> e ci dà il dato che si trova in quel campo in questo momento. Battete:

"SKIP 2"
"? Nome"

```
.use nomi
.? 'Nome'
Nome
.? Nome
AROSIO,ANGELA
.skip 2
RECORD: 00003
.? Nome
DE CRESCENZO,PAOLO
```

Ed ora battete "USE". Dato che non specifichiamo un nome di file, il computer semplicemente chiude tutti i file.

Se scriviamo di nuovo "? Nome", il computer risponde che abbiamo fatto un errore. In questo caso, quel che abbiamo fatto è cercare di usare una variabile che non esiste, dato che non stiamo più usando un file che ha un campo con quel nome.

Le variabili possono anche essere variabili di memoria invece che nomi di campi. Il dBASE II, infatti, riserva un'area della memoria per immagazzinare fino a 64 variabili, ognuna con una lunghezza massima di 254 caratteri ma con un totale massimo di 1.536 caratteri per tutte le variabili.

Si può pensare a questa area come ad una serie di 64 caselle a vostra disposizione per immagazzinarvi provvisoriamente dei dati mentre risolvete un altro problema.

Il nome di una variabile può essere un qualsiasi legittimo identificatore in dBASE II e, come i nomi dei campi, deve cominciare con una lettera, può essere lungo fino a dieci caratteri, e può contenere, se si vuole, i due punti e dei numeri, ma non può contenere spazi.

Si può usare una variabile di memoria per immagazzinare provvisoriamente dei dati o per mantenere separati dei dati dalle variabili di campo. In una fase continua del lavoro, per esempio, noi possiamo mettere un dato in una casella di una variabile di memo-

ria, che si chiami < Data >. Durante questa stessa fase possiamo ricuperare il dato chiedendo <Data > e collocarlo in qualsiasi campo-data di qualsiasi base-dati senza doverlo introdurre di nuovo.

Per immagazzinare dei dati (alfanumerici, numerici o logici) in una variabile di memoria, ricorriamo al comando "STORE". La formula completa è:

"STORE" <espressione > TO <variabile di memoria >

Battete:

```
"STORE" "Com'è andata finora?" TO Messaggio
"STORE 10 TO Ore"
"STORE 1.735 TO Salario"
"? Salario*Ore"
"? Messaggio"
```

```
.STORE "Com'è andata finora?" TO Messaggio
Com'è andata finora?
.STORE 10 TO Ore
10
.STORE 1.735 TO Salario
.? Salario*Ore
                17.350
.? Messaggio
Com'è andata finora?
```

E' da notare che abbiamo usato le virgolette doppie per la stringa alfanumerica (una costante) della prima riga perché dovevamo usare un apostrofo all'interno della stringa.

Per chiarire questo punto, provate a sperimentare con le virgolette e senza le virgolette e capirete meglio la distinzione fra costanti e variabili. Vi diamo il via noi, battete:

```
^STORE 99 TO Variabile^
^STORE 33 TO Altra^
^STORE Variabile/Altra TO Terza^
^STORE '99' TO Costante^
^? Variabile/Altra^
^? Variabile/3^
^? Costante/3^
^DISPLAY MEMORY^
```

```
.Store 99 to Variabile
99
.Store 33 to Altra
33
.STORE Variabile/Altra TO Terza
3
.STORE '99' To Costante
99
.? Variabile/Altra
                3
.? Variabile/3
               33
.? Costante/3'
***SYNTAX ERROR***
?
? COSTANTE/3.

.DISPLAY MEMORY
MESSAGGIO                       (C)       Com'è andata finora
ORE                             (N)              10
SALARIO                         (N)           1.735
VARIABILE                       (N)              99
ALTRA                           (N)              33
COSTANTE                        (C)              99
**TOTAL**           07 VARIABLES  USED       00084 BYTES USED
```

Introdurre un valore in una variabile indica immediatamente al dBASE II di che tipo di dato si tratta. D'ora in avanti, non potrete mescolare più tipi, cercando, ad esempio, di dividere una stringa alfanumerica per un numero.

Regole: Le stringhe alfanumeriche (di caratteri) che fanno parte di espressioni devono venir chiuse fra virgolette semplici o doppie o fra parentesi quadre. Le stringhe alfanumeriche possono contenere qualsiasi carattere di stampa, compreso lo spazio. Se volete usare l'"ampersand" (&) come carattere, deve andare tra due spazi perché serve ancne come macrofunzione del dBASE II che descriveremo più avanti.

L'ultimo comando che appare nella visualizzazione che vi abbiamo appena presentato è un'altra forma del ^DISPLAY^ che troverete utile. (Potete anche usare ^LIST MEMORY^).

Una variabile di memoria si elimina con ^RELEASE nome ^, ma potete anche eliminare tutte le variabili di memoria in una volta inserendo ^RELEASE ALL^.

Ora potreste usare ^ERASE^ per cancellare tutto ciò che avete sullo schermo e poi battere:

```
^DISPLAY MEMORY^
^RELEASE Altra^
^DISPLAY MEMORY^
^RELEASE ALL^
^DISPLAY MEMORY^
```

Nota: Quando date un nome a qualche variabile cercate di usare tutti i caratteri necessari perché sia comprensibile oltre che al computer anche alle persone.

Avvertenza: Se usate meno di dieci caratteri per i nomi di campo della vostra base-dati e avete bisogno dello stesso nome per una variabile di memoria, potete usarlo aggiungendovi una "M" al principio. Ciò che rappresenta vi sarà più chiaro, quando rimetterete ordine nei vostri programmi, più che se inventaste un nome completamente nuovo e diverso.

GLI OPERATORI DEL dBASE II

Gli operatori servono al dBASE II per manipolare i vostri dati. Alcuni di essi vi sono già familiari, ma per altri dovete prendere un po' di pratica.

Gli operatori aritmetici sono forse i più comuni. Essi generano risultati aritmetici.

```
()    : parentesi per raggruppare
*     : moltiplicazione
/     : divisione
+     : addizione
-     : sottrazione
```

Questi operatori vengono considerati in ordine di precedenza. L'ordine è quello esposto qui sopra: parentesi, moltiplicazione e divisione, addizione e sottrazione. Quando si trovano ad avere uguale precedenza, ha precedenza l'operatore più a sinistra. Ecco qualche esempio:

```
17/33*72.00+8=45.09 (dividere, moltiplicare e sommare)
17/(33*72.00000+8)=0.00644 (moltiplicare, sommare e dividere)
17/33*(72.00+8)=41.21 (dividere, sommare e moltiplicare)
```

Gli operatori relazionali effettuano paragoni e generano risultati logici. Agiscono in base al principio del paragone, che può essere Vero o Falso.

```
<   : minore di
>   : maggiore di
=   : uguale a
<>  : non uguale a
<   : minore o uguale a
>   : maggiore o uguale a
```

Battete:

```
^USE.Nomi^
^LIST FOR Cap <='40000'^
^LIST FOR Nome ='FRASSINETTI'
```

```
.LIST FOR Cap<='40000'
00001 AROSIO,ANGELA       V.Induno 12       20092 Cinisello
00002 BATTAGLIA,CESARE    V.Pitré 22        00162 Roma
00006 GIANNESINI,ANDREA   V.Cesarea 89/19   16121 Genova
00007 GIORELLO,FRANCESCO  V.F.Filzi 37       35100 Padova
00010 RAVASIN,ROBERTO     V.Santa Caterina 23 24100 Bergamo
```

```
   .LIST FOR Nome = 'FRASSINETTI'
00005   FRASSINETTI,MARIA PIA  V.Nannetti15 40069 Zola Pedrosa
```

Gli operatori logici espandono notevolmente la possibilità di ritoccare i dati e di manipolare i record. Un ulteriore approfondimento di questo argomento va oltre lo scopo di questo manuale, ma se non basta potete ricorrere a qualche testo di informatica; quasi tutti hanno un capitolo, tra i primi, dedicato agli operatori logici. Come abbiamo detto, essi generano risultati logici (Vero o Falso). Qui di seguito ve li presentiamo nell'ordine di precedenza che hanno all'interno di un'espressione (.NOT. si applica prima di .AND., ecc.):

```
()    :  parentesi per raggruppare
.NOT.:  inversione di Boole (negazione logica): operatore u-
        nario
.AND.:  congiunzione di Boole (moltiplicazione logica)
.OR. :  disgiunzione di Boole (addizione logica)
$    :  operatore logico di sottostringa (ricerca di sotto-
        stringa)
```

Osservate:

```
.LIST FOR (Serv:Num=730.OR.Serv:Num=731);
        .AND.(Fatt:Data>='791001'.AND.;
             Fatt:Data<='791031')
```

visualizza tutti i record dei servizi prestati con i numeri 730 731 e con la data di fatturazione compresa in ottobre. Notate che il comando si è esteso a una seconda e una terza riga tramite il punto e virgola.

Se non avete dimestichezza con gli operatori logici, pensate ad essi come operatori che vi diranno cosa è Vero e cosa è Falso.Nel nostro esempio, il dBASE II fa i seguenti controlli per ciascun record:

1) Il Serv:Num è uguale a 730 (T o F)?
2) Il Serv:Num è uguale a 731 (T o F)?
3) La Fatt:Data è maggiore o uguale a '791001' (T o F)?
4) La Fatt:Data è minore o uguale a '791031' (T o F)?

Subito dopo, il dBASE II esegue tre prove logiche (.OR., .AND., .AND.) prima di decidere se si deve visualizzare il record o no.

Le parentesi si usano, come nelle operazioni aritmetiche, per rendere più chiare operazioni e relazioni. A causa del primo .AND., il dBASE II presenterà i record solo quando le condizioni di ambedue le proposizioni fra parentesi sono vere.

Quando considera la prima espressione, comincia per controllare il campo < Serv:Num >. Se nel campo trova il valore 730 o 731, questa espressione risulta Vera. Se il campo contiene qualche altro valore, l'espressione è falsa e il record non sarà visualizzato.

Ma se la prima espressione è vera, il dBASE II deve ancora controllare il contenuto del campo < Fatt:Data> per poter considerare la seconda espressione. Se i contenuti dei campi sono fra '791001'e 791031' compreso, questa espressione è vera e, solamente allora, il record sarà visualizzato. Altrimenti, tutta l'espressione (nelle sue due parti) è falsa e il dBASE II salterà al record seguente, dove procederà nello stesso modo.

Provate a far qualcosa con < Nomi.DBF>. Battete:

```
^USE Nomi^
^DISPLAY ALL FOR Cap > '3' .AND. Cap<'5'^
^DISPLAY ALL FOR Nome < 'F'^
^DISPLAY ALL FOR Citta:Prov >'B' .AND. Citta:Prov < 'Z'^
^DISPLAY ALL FOR Citta:Prov <'B' .OR. Citta:Prov < 'Z'^
```

```
.USE Nomi
.DISPLAY all FOR Cap >  '3' .AND. Cap < '5'
00004 DE PACLIS,LIANA       C.so Garibaldi 15   40126 Bologna
00005 FRASSINETTI,MARIAPIA  V.Nannetti 15       40069 Zola P.
00007 GIORELLO,FRANCESCO    V.F.Filzi 37        35100 Padova

.DISPLAY all FOR Nome < 'F'
00001 AROSIO,ANGELA         V.Induno            20091 Cinisello
00002 BATTAGLIA,CESARE      V.Pitré 22          00162 Roma
00003 DE CRESCENZO,PAOLO    V.Regina M.101      00198 Roma
00004 DE PAOLIS,LIANA       C.so Garibaldi 3    40126 Bologna
```

```
   .DISPLAY all FOR Citta:Prov > 'B' .AND. Citta:Prov < 'Z'
 00001 AROSIO,ANGELA             V.Induno 12             20092 Cinisello
 00002 BATTAGLIA,CESARE          V.Pitré 22              00162 Roma
 00003 DE CRESCENZO,PAOLO        V.R.Margherita 101      00198 Roma
 00006 GIANNESINI,ANDREA         V.Cesarea 89/19         16121 Genova
 00007 GIORELLO,FRANCESCO        V.F.Filzi 37            35100 Padova
 00008 MALDACEA,LUIGI            V.Notarbartolo 7        90141 Palermo
 00009 MERLI,SILVANO             V.Imparato 30           80146 Napoli

   .DISPLAY all FOR Citta:Prov < 'B'  .OR. Citta:Prov < 'Z'
 00001 AROSIO,ANGELA             V.Induno                20092 Cinisello
 00003 BATTAGLIA,CESARE          V.Pitré 22              00162 Roma
 00004 DE PAOLIS,LIANA           C.so Garibaldi 8        40126 Bologna
 00005 FRASSINETTI,MARIA PIA     V.Nannetti 15           40069 Zola P.
 00006 GIANNESINI,ANDREA         V.Cesarea 89/19         16121 Genova
 00007 GIORELLO,FRANCESCO        V.F.Filzi 37            35100 Padova
 00008 MALDACEA,LUIGI            V.Notarbartolo 7        90141 Palermo
 00009 MERLI,SILVANO             V.IMparato 30           80146 Napoli
 00010 RAVASIN,ROBERTO           V.Santa Caterina 23     24100 Bergamo
```

Notate cosa è accaduto con l'ultimo comando: abbiamo visualizzato tutti i record del file. Se non avete dimestichezza con gli operatori logici, fate attenzione a queste selezioni che in effetti non sono selettive...!

L'operatore logico di sottostringa è estremamente utile per la sua capacità di ricerca. Ecco il suo formato:

```
<sottostringa> $ <stringa>
```

Ciò che fa l'operatore è cercare la sottostringa di sinistra nella stringa di destra. Uno o ambedue i termini possono essere sia variabili sia costanti di stringa. Per vedere come funzionano provate a battere:

```
USE Nomi
LIST FOR 'AO'$Nome
LIST FOR '2'$Indirizzo
LIST FOR 'Roma'$Citta:Prov
? '00'$'Roma'
GO 2
.DISPLAY
? "Roma"$Citta:Prov
```

```
   .USE Nomi
   .LIST FOR 'AO'$Nome
00003 DE CRESCENZO,PAOLO        V.le R.Margherita 101 00198 Roma
00004 DE PAOLIS,LIANA           C.so Garibaldi 8      40126 Bologna

   .LIST FOR '2'$Indirizzo
00001 AROSIO,ANGELA             V.Induno 12           20092 Cinisello
00002 BATTAGLIA,CESARE          V.Pitré 22            00162 Roma
00010 RAVASIN,ROBERTO           V.Santa Caterina 23   24100 Bergamo

   .LIST FOR 'Roma'$Citta:Prov
00002 BATTAGLIA,CESARE          V.Pitré 22            00162 Roma
00003 DE CRESCENZO,PAOLO        V.le R.Margherita 101 00198 Roma

   .? '00'$'Roma'
   .F.
   .go 2
   .display
00002 BATTAGLIA,CESARE          V.Pitré 22            00162 Roma

   .? "Roma"$Citta:Prov
   .T.
```

Con questa funzione, per esempio, potreste semplificare la struttura del vostro indirizzario. Le città potrebbero entrare come parte dell'indirizzo. Per recuperare i nomi di una data città vi basterebbe battere la seguente formula, in cui XX simbolizza la città che state cercando:

"<COMANDO> FOR 'XX' $ Indirizzo"

Gli operatori di stringa generano stringhe.

+ = concatenazione di stringhe (esatta)
- = concatenazione di stringhe (che muove gli spazi in bianco)

La concatenazione è un'altra affascinante prerogativa del dBASE II, che "attacca" una stringa all'altra.

Scrivete:

```
JSE Nomi^
^? Nome+Indirizzo^
 ? Nome-Indirizzo^
 ? 'Il nome in questo record è '+ Nome - ' e l'indirizzo è ' +;
Indirizzo^
```

```
.USE Nomi
.? Nome + Indirizzo
AROSIO,ANGELA                V.Induno 12
.? Nome - Indirizzo
AROSIO,ANGELA V.Induno 12
.? 'Il nome in questo record è ' + Nome - ' e l'indirizzo è ' + ;
Indirizzo
Il nome in questo record è AROSIO,ANGELA e l'indirizzo è
V.Induno 12
```

segni ^+^ e ^-^ congiungono due stringhe. Ma, mentre il segno "più" le unisce lasciandole inalterate, il segno "meno" sposta tutti i caratteri rimasti in bianco alla fine della nuova stringa. In questo modo, gli spazi li ritroviamo raggruppati alla fine piuttosto che a metà stringa.

Se poi volete eliminare gli spazi alla fine dovete usare la funione TRIM^, battendo sulla tastiera ^STORE TRIM (<variabile>) TO < variabile >^. Per esempio, avremmo potuto battere ^STORE TRIM (Nome) TO Nome)^ ed avremmo eliminato gli spazi che seguivano i caratteri del nome. Nell'esempio che vi abbiamo portato poco fa, avremmo potuto battere ^STORE TRIM (nome - Indirizzo) TO Campione^.

Ora che vi abbiamo spiegato le espressioni e gli operatori del dBASE II, procederemo con altri comandi. Cercheremo di farvi impratichire con quanto fatto introducendo piano piano il concetto del file di comandi (command file).

CAMBIO DI STRUTTURA DI UNA BASE-DATI VUOTA (MODIFY)

Attenzione: Il comando MODIFY può distruggere la vostra base-dati. Seguite attentamente le istruzioni. Quando nella vostra base-dati non

vi sono informazioni, il comando "MODIFY" costituisce il modo più rapido e facile per aggiungere, eliminare, cambiare i nomi dei campi o qualsiasi altro elemento nella sua struttura. Ciò distrugge assolutamente tutti i dati della base-dati, per cui non lo dovete usare dopo averli introdotti. (Più avanti vi mostreremo una maniera per farlo senza pericolo).

<Pagamenti.DBF> non contiene ancora dati, perciò possiamo rielaborarlo. Una modifica utile può essere quella di cambiare il nome di Serviz:Num con quello di Serv:Num in modo che l'abbreviazione sia analoga a quella degli altri campi (Ass:Num , Fatt:Num e Imp:Num).

Inserite dunque:

"USE Pagamenti"
"LIST STRUCTURE
"MODIFY STRUCTURE"
"y" (in risposta alla domanda visualizzata)

```
   .use Pagamenti
   .list structure
STRUCTURE FOR FILE: Pagamenti.DBF
NUMBER OF RECORDS: 00000
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD         NOME              TYPE          WIDTH     DEC
001         CLIENTE           C               004
002         SERVIZ:NUM        C               003
003         FATT:DATA         C               006
004         FORNITORE         C               028
005         DESCR             C               010
006         ORE               N               006       002
007         IMP:NUM           C               002
008         IMPORTO           N               009       002
009         FATT:NUM          C               006
010         ASS:NUM           C               005
011         ASS:DATA          C               006
**TOTAL**                                     00085

  .modify structure
 MODIFY ERASES ALL DATA RECORDS...PROCEED?(Y/N)'y'
```

Il dBASE II pulisce lo schermo e presenta una lista dei primi sedici campi (o meno) della base-dati. Posizionate il cursore al principio del campo da correggere e semplicemente scrivete il nuovo nome sopra quello vecchio, usando la barra spaziatrice per eliminare le lettere che avanzano.

Potete uscire dal comando "MODIFY" con uno di questi due modi : "ctl-W" cambia la struttura sul disco e poi torna alla normale attività del dBASE II (per Superbrain, "ctl-O"); "ctl-Q" abbandona e torna anch'esso alla normale attività del dBASE II, però non effettua alcuna modifica. In realtà, vi riporta semplicemente al punto di partenza senza distruggere la base-dati, ma usatelo con prudenza e ricordatevi di tenere una copia del file su disco.

COPIA DELLE BASE-DATI E DELLE STRUTTURE (COPY)

Copiare un file senza tornare al sistema operativo del vostro computer è semplice. Battete:

"USE Nomi"
"COPY TO Provv"
"DISPLAY STRUCTURE"

"LIST"

```
  .use nomi
  .copy to provv
 00010 RECORDS COPIED

  .use provv
  .display structure
STRUCTURE FOR FILE:PROVV.DBF
NUMBER OF RECORDS:00010
DATE OF LAST UPDATE:00/00/00
PRIMARY USE DATABASE
 FLD        NAME              TYPE          WIDTH      DEC
 001        NOME              C               025
 002        INDIRIZZO         C               030
 003        CAP               C               005
 004        CITTA:PROV        C               035
 **TOTAL**                                  00096
```

```
  .list
00001 AROSIO,ANGELA              V Induno 12              20092 Cinisello
00002 BATTAGLIA,CESARE           V.Pitré 22               00162 Roma
00003 DE CRESCENZO,PAOLO         V.le R.Margherita        00198 Roma
00004 DE PAOLIS,LIANA            C.so Garibaldi 8         40126 Bologna
00005 FRASSINETTI,MARIA PIA      V.Nannetti 15            40069 Zola P.
00006 GIANNESINI,ANDREA          V.Casarea 89/19          16121 Genova
00007 GIORELLO,FRANCESCO         V.F.Filzi 37             35100 Padova
00008 MALDACEA,LUIGI             V.Notarbartolo 7         90141 Palermo
00009 MERLI,SILVANO              V.IMparato 30            80146 Napoli
00010 RAVASIN,ROBERTO            V.Santa Caterina 23      24100 Bergamo
```

Attenzione: Se il destinatario della copia è un file il cui nome già esiste, il file da copiare cancellerà quello esistente.

"COPY TO Provv" ha creato una nuova base-dati chiamata <Provv.DBF>, identica a <Nomi.DBF>, con la stessa struttura e gli stessi dati. Il comando può venire espanso ulteriormente:

"COPY TO <nome del file> [STRUCTURE] [FIELD lista]

Con questo comando si può copiare su un altro file solamente la struttura o parte di essa. Battete:

```
"USE Nomi"
"COPY TO Provv STRUCTURE"
"USE Provv"
"DISPLAY STRUCTURE"
```

```
  .use nomi
  .copy structure to provv

  .use provv
  .display structure
STRUCTURE FOR FILE:PROVV.DBF
NUMBER OF RECORDS:00000
DATE OF LAST UPDATE:00/00/00
PRIMARY USE DATABASE
FLD          NAME              TYPE          WIDTH        DEC
001          NOME               C             025
002          INDIRIZZO          C             030
003          CAP                C             005
004          CITTA:PROV         C             035
**TOTAL**                                     00096
```

Per copiare una parte della struttura, facciamo una lista dei campi che vogliamo avere nel nuovo file:

```
^USE Nomi^
^COPY TO Provv STRUCTURE FIELDS Nome,Citta:Prov^
^USE Provv^
^DISPLAY STRUCTURE^
```

```
.use nomi
.copy structure to provv field nome,citta:prov
.use provv
.display structure
STRUCTURE FOR FILE:PROVV.DBF
NUMBER OF RECORDS:00000
DATE OF LAST UPDATE:00/00/00
PRIMARY USE DATABASE
FLD         NAME            TYPE        WIDTH        DEC
001         NOME             C           025
002         CITTA:PROV       C           035
**TOTAL**                               00061
```

Per programmatori esperti: COPY si può usare anche per far accedere i vostri programmi alla struttura di una base-dati. Ecco come:

```
^USE Nomi^
^COPY TO Nuova STRUCTURE EXTENDED^
^USE Nuova^
^LIST^
```

```
.use Nomi
.copy to Nuova structure extended
00004 RECORDS COPIED
.Use Nuova
.display structure
STRUCTURE FOR FILE:NUOVA.DBF
NUMBER OF RECORDS:00004
DATE OF LAST UPDATE:00/00/00
```

```
PRIMARY USE DATABASE
FLD         NAME            TYPE        WIDTH      DEC
001         FIELD:NAME      C             010
002         FIELD:TYPE      C             001
003         FIELD:LEN       N             003
004         FIELD:DEC       N             003
**TOTAL**                                 00018

  .list
00001       NOME            C              25        0
00002       INDIRIZZO       C              30        0
00003       CAP             C              05        0
00004       CITTA:PROV      C              35        0
```

I record della nuova base-dati <Nuova.DBF> descrivono la struttura della base-dati <Nomi> e un programma applicativo ha accesso diretto a questa informazione.

Abbiamo anche l'alternativa di inserire un file dalla struttura uguale a <Nuova.DBF> in un altro programma in modo che un utente diverso possa introdurre questa struttura in un file senza bisogno d'imparare il dBASE II. Il programma creerebbe per lui la base-dati con il seguente comando:

"CREATE <file di dati> FROM <file di struttura>"

AGGIUNTA E SOPPRESSIONE DI CAMPI GIA' RIEMPITI

Andando avanti con le applicazioni del dBASE II, arriverà probabilmente il momento in cui vorrete aggiungere o sopprimere dei campi nella vostra base-dati.

Se usate solamente "MODIFY STRUCTURE" distruggerete tutti i dati che avete introdotto, ma se lo usate insieme a "COPY" o "APPEND" potete aggiungere o togliere campi a volontà.

La procedura necessaria consiste nel copiare la struttura della base-dati in un file provvisorio e poi fare le vostre modifiche su di esso. Fatto questo, trasportate i dati dal vecchio file al nuovo modificato.

er fare un esempio, useremo i nostri file Nomi e Ordini . A n certo punto, può essere utile fare una lista delle ordinazioni di un dato cliente. Lo farete facilmente aggiungendo un campo er il numero di cliente al file Nomi , che coincide con il campo corrispondente del file Ordini . Per riuscirvi senza distruggere i record introdotti, battete:

```
 USE Nomi^
^COPY TO Provv STRUCTURE^
 USE Provv^
 MODIFY STRUCTURE^
^y^  (in risposta alla domanda visualizzata)
```

Utilizzate il modo di Composizione a Schermo Intero (Full Screen diting) per scendere fino al primo campo vuoto (in bianco) e batete i vostri cambi nelle colonne appropriate (il nome è "Num: Client", il tipo di dato è "C" e la lunghezza è 9). Fatto questo, sate ^ctl-W^ (^ctl-O^ con Superbrain) per salvare i campi e usci e direttamente sul punto di richiamo del dBASE II.

attete ^DISPLAY STRUCTURE^, ovvero visualizzate la struttura, er essere sicuri che sia corretta e, se lo è, potete ora aggiungere i dati prendendoli dal file Nomi, scrivendo:

```
^APPEND FROM Nomi^
```

otremmo anche aver cambiato le dimensioni del campo: il comando ^APPEND^ scrive i dati nei campi che hanno lo stesso nome.

```
   .display structure
 STRUCTURE FOR FILE:PROVV.DBF
NUMBER OF RECORDS:00010
DATE OF LAST UPDATE:00/00/00
 PRIMARY USE DATABASE
 FLD          NAME            TYPE            WIDTH      DEC
 001          NOME             C               025
 002          INDIRIZZO        C               030
 003          CAP              C               005
 004          CITTA:PROV       C               035
 005          NUM:CLIENT       C               009
 **TOTAL*                                      105
```

Il nostro nuovo file <Provv> dovrebbe avere adesso il nuovo campo che vogliamo aggiungere e anche tutti i dati che corrispondono al file <Nomi>. Per assicurarvene, inserite "DISPLAY STRUCTURE" e poi "LIST", perché può sempre accadere che qualche disturbo di rete oppure una irregolarità del disco abbiano sciupato tutto.

Se i dati sono stati trasportati correttamente, possiamo concludere:

"COPY TO Nomi"
"USE Nomi"

E il comando "COPY" scriverà sulla vostra struttura i dati relativi con la nuova struttura. Dopo aver visualizzato la struttura e la lista del nuovo file <Nomi>, potete cancellare il file PROVV col comando "DELETE FILE Provv"

Riassumendo, la procedura da seguire per aggiungere o sopprimere campi in una base-dati è, in sequenza:

"USE <file vecchio>"
"COPY TO <file nuovo> STRUCTURE"
"USE <file nuovo>"
"MODIFY STRUCTURE"
"APPEND FROM <file vecchio>"
"COPY TO <file vecchio>"

```
   .use nomi
   .copy to provv structure
   .use provv
   .modify structure
MODIFY ERASES ALL DATA RECORDS...PROCEED? (Y/N) 'y'

   .append from nomi
00010 records added

   .copy to nomi
00010 RECORDS COPIED

   .use nomi
   .display structure
STRUCTURE FOR FILE:NOMI.DBF
NUMBER OF RECORDS:00010
DATE OF LAST UPDATE:00/00/00
```

```
PRIMARY USE DATABASE
FLD         NAME            TYPE            WIDTH      DEC
001         NOME            C                 025
002         INDIRIZZO       C                 030
003         CAP             C                 005
004         CITTA:PROV      C                 035
005         NUM:CLIENT      C                 009
**TOTAL**                                   00105
```

TRATTAMENTO DI FILE DI DATI CP/M ED ALTRI (ANCORA CON COPY E APPEND)

Le informazioni del dBASE II possono venire trasformate in modo da essere compatibili con altri elaboratori e linguaggi (BASIC, PASCAL, FORTRAN, PL/1, ecc.). Similmente, il dBASE II può leggere i file di dati creati da quei processori.

Nel CP/M, il Formato dei Dati del Sistema (abbreviato in dBASE II a SDF) comprende un ritorno del carrello e un salto-riga dopo ogni linea del testo. Per creare un file di dati compatibile (per videoscrittura, per esempio) dall'una all'altra base-dati, userete un'altra forma del comando COPY :

```
^USE Nomi^
^COPY TO Dati SDF^
```

Questo comando realizza un file chiamato Dati.TXT . Ora potete inserire ^QUIT^ e usare il vostro programma di videoscrittura per osservare il file. Vedrete che si può lavorare su di esso esattamente come se fosse stato creato tramite CP/M.

Inoltre, il Formato di Dati del Sistema (SDF) permette al dBASE II di lavorare con i dati di file CP/M. Ma attenzione: i dati devono coincidere con la struttura della base-dati che si dovrà utilizzare.

Se abbiamo usato un programma di videoscrittura per creare il file <Nuovi Dati.TXT>, per esempio, possiamo aggiungerlo a < Nomi. DBF > mediante questo comando. Ma notate: la spaziatura dei dati deve coincidere con la struttura della base-dati. Se il file < Nuovi Dati.TXT > conteneva queste informazioni:

```
MICALIZZI,OLIVIA    V.Montenapoleone 11     20121 Milano
REDA,VINCENZO       V.Vicenza 39            36063 Marostica, Vicenza
ROMANI,MARCELLO     V.Garibaldi 2           46043 Castiglione Stiv.
VALLICELLI,PIERA    C.so Peschiere 192      10139 Torino

    (25)                (30)                 (5)        (35)
```

potremmo aggiungerla al file <Nomi.DBF> battendo:

```
^USE Nomi^
^APPEND FROM NuoviDati.TXT SDF^
```

Come abbiamo visto, aggiungere dati a un file esistente estraendoli da un file "esterno" richiede solo pochi secondi.

La procedura è molto simile se i file "esterni" usano differenti delimitatori. Un normale formato di file di dati utilizza la virgola tra campo e campo e le virgolette semplici intorno alle stringhe, in modo da delimitare i dati. Per creare o per usare questi tipi di file di dati, servitevi della parola DELIMITED invece di SDF.
Vediamo come funziona, scrivendo.

```
^COPY TO Provv DELIMITED^
```

e poi tornando al vostro sistema operativo per osservare i dati.

Se il vostro sistema ha delimitatori differenti, potete specificarlo nel comando: ^DELIMITED WITH <delimitatore>^ senza scrivere i simboli "<" e ">". Se per esempio il vostro sistema usa solamente le virgole e non chiude le stringhe, usate ^DELIMITED WITH ,^.

I comandi completi per lavorare con ^COPY^ e ^APPEND^ su file di dati del sistema sono:

```
                                                        [SDF]
COPY [ scopo ] TO <nome del file> [ listaFIELD] [STRUCTURE ]
                                                        [FOR <espressione>]
                                    [ DELIMITED] [WITH <delimitatore>]]

APPEND FROM <nome del file>.TXT [ SDF] [FOR <espressione>]
                                [DELIMITED] [ FOR <espressione>]
```

Tutti e due questi comandi possono diventare selettivi mediante un'espressione condizionale, e lo scopo di ^COPY^ può venire specificato come nel caso di altri comandi di dBASE II.

Attenzione: Mentre il dBASE II genera automaticamente le estensioni per i file che costruisce, quando fate delle aggiunte mediante APPEND da un file di dati "esterno", dovete assolutamente specificare l'estensione ".TXT" del file.

Notate anche che con il comando APPEND, qualsiasi campo usato all'interno dell'<espressione> deve esistere precedentemente nella base-dati alla quale vengono trasportati i dati.

IL CAMBIO DI NOME DEI CAMPI DI UNA BASE-DATI MEDIANTE COPY E APPEND

Come abbiamo detto più sopra, "APPEND" trasporta i dati da un file all'altro in base alla coincidenza dei campi. Se un nome di campo del file di provenienza (FROM) non si trova nel file su cui lavoriamo (USE), i dati non verranno trasportati.

Eppure la forma completa ci permette di trasportate precisamente i dati, se usiamo questa possibilità per cambiare i nomi nella base-dati. Se ora vogliamo cambiare <Num:Client> in <Cod:Client> nel nostro <Nomi.DBF>, scriveremo:

```
"USE Nomi"
"COPY TO Provv SDF"          (per Provv.TXT, solo i dati)
"MODIFY STRUCTURE"
"APPEND FROM Provv.TXT SDF"  (dopo aver cambiato il nome del campo)
```

Quando tornerete a visualizzare il risultato tramite "DISPLAY STRUCTURE", l'ultimo campo si chiamerà <Cod:Client>. Non dimenticate di cambiare questo nome anche nel nostro file <Ordini>, in modo che tutti i campi coincidano.

```
.use nomi

.copy to provv sdf
00014 RECORDS COPIED
.modify structure
MODIFY ERASES ALL DATA RECORDS...PROCEED? (Y/N) Y

.append from provv.TXT sdf
00014 RECORDS ADDED
```

I dati di un file <.TXT>, creato mediante l'opzione SDF (o DELIMITED) si trovano ordinati in colonne con la stessa spaziatura che avevano nel file originale.

Attenzione: Non cambiate le posizioni o le dimensioni dei campi: i dati sono stati salvati in base alla posizione e non per il loro nome. Se cambiate le dimensioni dei campi quando modificate la struttura, distruggerete la vostra base-dati nel momento in cui vorrete riportarvi l'informazione.

Quando copiate ("COPY") i dati in un file <.TXT>, potete usare il comando completo per specificare lo scopo, i campi e le condizioni (ripassate la spiegazione già data).

LA MODIFICA RAPIDA DEI DATI (REPLACE, CHANGE)

I cambi rapidi di uno o di tutti i records vengono effettuati con questo comando:

"REPLACE [scopo] <campo> WITH <espressione> [,<campo> WITH <espressione>,...] [FOR <espressione>]"

La potenza di questo comando consiste nel "rimpiazzare" (REPLACE) un "<campo-che-voi-nominate> con (WITH) <qualcosa-che-voi-scrivete-qui>". Voi potete così sostituire, o rimpiazzare, anche più di un campo mettendo una virgola dopo la prima combinazione, elencando poi i nuovi campi e i nuovi dati come segnalato nelle parentesi quadre.

I "dati" possono essere sia una nuova informazione (compresi gli spazi in bianco), sia un'operazione, come potrebbe essere quella di dedurre l'IVA sulla vendita da tutte le vostre fatture (REPLACE ALL Importo WITH Importo/1.06).

Questa sostituzione si può fare anche condizionalmente mediante FOR, sempre specificando le vostre condizioni in un'espressione.

Per mostrarvi come funziona tutto ciò, dobbiamo aggiungere qual-

che dato ai nostri due file della base-dati, < Nomi > e < Ordini > .

Per prima cosa, inserite "USE Nomi" seguito da "EDIT 1". Poi introducete "1001" nel campo "Cod:Client" usando la composizione a schermo intero per posizionarvi. Quando avete finito passate al record seguente servendovi di "ctl-C". I codici cliente si dovrebbero introdurre come numeri di quattro cifre, delle quali le ultime due sarebbero il numero del record (1001, 1002, 1003, ecc.).

Ora passate a "USE Ordini" e aggiungete ("APPEND") le seguenti informazioni, senza scrivere i titoli delle colonne:

(Cliente)	(Articolo)	(Quantità)	(Prezzo)
1012	38567	5	830
1003	83899	34	1200
1009	12829	7	1700
1012	73833	23	1470

seguite da:

"USE Ordini"
"REPLACE ALL Importo WITH Quantità * Prezzo"
"LIST"

```
    .use Ordini
    .replace  all importo with quantità * prezzo
 00004 REPLACEMENT(S)
    .list
 00001        1012          38567        5        830      4150
 00002        1003          83899       34       1200     40800
 00003        1009          12829        7       1700     11900
 00004        1012          73833       23       1470     33810
```

Il comando di "REPLACE" è molto utile nei file-comandi per riempire un record vuoto già aggiunto ad un file, come si fa spesso richiamando i dati dalle variabili di memoria.

Anche effettuare cambi in pochi campi in un archivio di molti records si può fare rapidamente:

"CHANGE [scopo] FIELD <lista>[FOR < espressione >]

Lo "scopo" è lo stesso che si indica usando gli altri comandi del dBASE II. Bisogna elencare per lo meno un campo, ma mettendone più di uno è necessario separarli per mezzo di virgole. Questo comando agisce cominciando col trovare il primo record che coincide con le condizioni indicate nell'"espressione", poi visualizza il nome del campo e il suo contenuto con un punto di domanda. In risposta, scrivete la nuova informazione al posto della vecchia, e avrete così cambiato il vostro dato. Quando non lo volete cambiare, battete enter . Se avete un campo in bianco e volete riempirlo, battete prima uno spazio.

Dopo aver sostituito tutti i campi di un record, vi verrà presentato il record successivo che coincide con le condizioni da voi stabilite, permettendovi di continuare a effettuare i vostri cambi.

Per tornare al dBASE II, battete il tasto "ESCAPE"

```
.use nomi
.change field cod:client

RECORD: 00001

COD:CLIENT
CHANGE? ' '     (PER CAMBIARE UN CAMPO VUOTO, BATTERE   UNO
TO      1001                                            SPAZIO)

COD:CLIENT: 1001
CHANGE?  enter

record: 00002
CHANGE?
```

Ricordate: Se volete introdurre sostituzioni in un numero di record relativamente basso, il comando BROWSE può essere ciò che fa per voi.

COME ORDINARE LA VOSTRA BASE-DATI (SORT, INDEX)

I dati vengono introdotti spesso in modo casuale, così come è avvenuto nella nostra base-dati Nomi . Non sempre ciò corrisponde a quello che voi volete, ma il dBASE II dispone di comandi per aiutarvi a ordinare (SORT) e indicizzare (INDEX) la vostra base-dati.

Gli archivi indicizzati vi permettono di ricercare i vostri record velocemente (di solito su diskette, in due secondi).

Gli archivi si possono ordinare in ordine ascendente o discendente. Il comando completo è:

`SORT ON <nome del campo> TO <nome del file> [DESCENDING]`

Il nome del campo, alfanumerico o numerico (ma non logico), determina la chiave secondo la quale viene ordinato il file. Se non si specifica l'opzione discendente, la classificazione verrà effettuata in ordine ascendente.

Per ordinare secondo più di una chiave, cominciate dalla chiave meno importante e proseguite attraverso una serie di chiavi che conducano a quella principale. Durante questo processo, il dBASE II muoverà solamente i record necessari all'operazione.

```
.USE Nomi
.SORT ON Nomi TO Provv
.USE Provv
.LIST
.COPY TO Nomi
```

```
.use nomi
.sort on nomi to provv
.SORT COMPLETE
.use provv
.list
```

00001	ARCSIO,ANGELA	V.Induno 12	20097	Cinisello	1001
00002	BATTAGLIA,CESARE	V.Pitré	00162	Roma	1002
00003	DE CRESCENZO,PAOLO	V.R.Margherita 101	00198	Roma	1004
00004	DE PAOLIS,LIANA	C.so Garibaldi 8	40126	Bologna	1005
00005	FRASSINETTI,MARIA PIA	V.Nannetti 15	40069	Zola Pedr.	1006
00006	GIANNESINI,ANDREA	V.Casarea 89/19	16121	Genova	1011
00007	GIORELLO,FRANCESCO	V.F.Filzi 37	35100	Padova	1012
00008	MALDACEA,LUIGI	V.Notarbartolo 7	90141	Palermo	1011
00009	MERLI,SILVANO	V.Imparato 30	80146	Napoli	1007
00010	MICALIZZI,OLIVIA	V.Montenapolecne	20121	Milano	1008
00011	RAVASIN,ROBERTO	V.S.Caterina 23	24100	Bergamo	1009
00012	REDA,VINCENZO	V.Vicenza 39	36063	Marostica	1010
00013	ROMANI,MARCELLO	V.Garibaldi 2	46043	Castiglione	1015
00014	VALLICELLI,PIERA	C.so Peschera	10139	Rorino	1115

NOTA: Come avrete notato, l'ordinamento del file non è stao effettuato sul file stesso ma su un altro <temporaneo>, da dove, dopo aver controllato i dati, li abbiamo ricopiati sull'originale.

La classificazione mediante "INDEX" si effettua secondo il seguente comando completo:

"INDEX ON < chiave (variabile/espressione) > TO < file indice>"

Con ciò si crea un nuovo file chiamato < Indice>, con l'estensione < NDX >. Soltanto i dati riferibili alla chiave vengono ordinati anche se l'intera base-dati apparirà classificata. La chiave può essere il nome di una variabile oppure un'espresisone complessa che può avere fino a 100 caratteri. Però non potrà trattarsi di un campo logico.

Vediamo come organizzare la nostra base-dati < Nomi> secondo il codice postale:

"USE Nomi"
"INDEX ON Cap TO Codice"
"LIST"

Ma possiamo anche ordinare la nostra base-dati secondo tre chiavi inserendo:

"INDEX ON Nome + Cod:Client + Citta:Prov TO Compos"

Se vogliamo includere dei campi numerici in questa operazione, dobbiamo convertirli in caratteri. Se Cod:Client fosse un campo numerico di cinque cifre, la funzione ^STR^ (che descriveremo più avanti) usata come segue, eseguirebbe la conversione:

^INDEX ON Nome + STR (Cliente,5) + Citta:Prov TO Compos^

Per approfittare della velocità implicita di un file Indice dovete aggiungerlo come parte del comando ^USE^:

^USE <nome del file >INDEX <Indice>^

I comandi di posizionamento o indirizzo (GO, GO BOTTOM, ecc.), quando si lavora con un file <Indice>, si muovono su quest'ultimo e non sulla base-dati. ^GO BOTTOM^, per esempio, si posizionerà sull'ultimo record nell'indice e non su quello del file corrente.

Le sostituzioni che si effettuano sui campi-chiave della base-dati mediante ^APPEND^, ^EDIT^, ^REPLACE^ o ^PACK^, si riflettono nell'indice che si sta utilizzando.

Se avete altri indici della vostra base-dati, aggiornateli inserendo: ^SET INDEX TO <Indice 1>, <Indice 2>, <Indice n>...^. Poi eseguite ^APPEND^, ^EDIT^, ecc. In questo modo, tutti gli indici nominati saranno attivati.

Il vantaggio principale di un file che disponga di < Indici> è che vi permette di usare il comando ^FIND^ per localizzare i records in pochi secondi, anche in base-dati molto grandi.

LA RICERCA DELL'INFORMAZIONE (FIND, LOCATE)

Se sapete che dato state cercando, potete usare ^FIND^ (trovare) ma solo se la vostra base-dati dispone di indici e se il file < Indice > è in funzione. Con un sistema a floppy disk, questa operazione non richiede in genere più di due secondi.

Battete semplicemente ^FIND^ < stringa alfanumerica>^ (senza nessun tipo di virgolette). Questa "stringa" può corrispondere a tutto il contenuto di un campo o solamente a una parte di esso. Può essere anche cortissima, ma deve avere la lunghezza sufficiente a evitare di essere confusa con quella di un altro record. Per esem

pio, troviamo le lettere "te" in un gran numero di parole, mentre "teat" è notevolmente meno frequente.

Inserite:

```
^USE Nomi INDEX Codice^
^FIND 10^
^DISPLAY^
^FIND 2^
^DISPLAY^
^DISPLAY Next 3^
```

```
.use nomi index cap

.find 10
.display
00014 VALLICELLI,PIERA     C.so Peschiera     10139 Torino     1115

.find 2
.display
00001 AROSIO,ANGELA        V.Induno 12        20092 Cinisello  1001

.display next 3
00001 AROSIO,ANGELA        V.Induno 12        20092 Cinisello  1001
00010 MICALIZZI,OLIVIA     V.Montenapoleone   20121 Milano     1008
00011 RADVASIN,ROBERTO     V.S.Caterina 23    24100 Bergamo    1009
```

Se la chiave non è unica, il dBASE II si ferma al primo record che soddisfa la richiesta e che può essere quello che voi cercate oppure no. Se non esiste alcun record con una chiave identica a quella che voi indicate, il dBASE II visualizza "NO FIND".

^FIND^ si può usare anche per file che dispongono di indici con chiavi multiple. Lo svantaggio di una chiave multipla (che nel vostro caso può non essere uno svantaggio) è che si deve usare partendo da sinsitra al momento di accedere ai dati. Il che vuol dire che mediante FIND potete accedere al dato dando il Nome, oppure il Nome e il Cod:Client, o il Nome, il Cod:Client e l'Indirizzo ma non, per esempio, l'Indirizzo solamente. Per far ciò dovrete ricorrere al comando ^LOCATE^ o disporre di un altro file in cui il campo dell'Indirizzo sia la prima chiave.

In conseguenza, quando state cercando uno specifico dato usate:

^LOCATE [scopo] [FOR <espressione>] ^

Questo comando si usa quando si cerca un dato specifico in un file i cui indici non sono ordinati secondo la chiave che vi interessa.

Se quello che volete è esplorare tutta la base-dati dal vostro puntatore fino alla fine del file, non avete bisogno di specificare lo scopo. Se volete cercare per tutto il file, o specificate " ALL" o vi posizionate al principio del file (GO TOP). Se poi state cercando dei dati in un campo alfanumerico, il dato dev'essere racchiuso fra virgolette semplici. Inserite i seguenti comandi:

```
^USE Nomi^
^LOCATE FOR Nome = 'MICA'^
^DISPLAY^
^LOCATE FOR Cap > '3' .AND. Nome < 'G'^
^DISPLAY Nome,Cap
```

Se viene trovato un record che coincide con la vostra <espressione>, il dBASE II ve lo segnala presentandovi il messaggio RECORD n . Sta a voi decidere se volete visualizzarlo o rielaborarlo per mezzo di ^EDIT^, o se vi basta che sia stato localizzato con il suo numero di record.

Se credete che possa esserci più di un record soddisfacente alle vostre condizioni, battete ^CONTINUE^ tante volte quante lo ritenete necessario:

```
^CONTINUE^
^CONTINUE^
^CONTINUE^
```

Se il dBASE II NON RIESCE A TROVARE IL RECORD CHE VI INTERESSA NEI LIMITI DELLO "SCOPO", VI MOSTRERA': END OF LOCATE oppure END OF FILE ENCOUNTERED.

```
    .use nomi
    .locate for Nome = 'MICA'
RECORD:00010
    .display
00010 MICALIZZI,OLIVIA  V. Montenapoleone    20121 Milano 10008

    .locate for Cap > '3' .AND. Nome < 'G'
RECORD:00004
    .display Nome,Cap
00004 DE PACLIS,LIANA                  40126
    .continue
RECORD:00005
    .continue
RECORD:00007
    .continue
END OF FILE ENCOUNTERED
```

ESTRARRE INFORMAZIONI DALLA BASE-DATI
(IL COMANDO REPORT)

Nonostante l'efficacia dei comandi "FIND" e "LOCATE" per localizzare singoli record e determinati dati, il più delle volte abbiamo bisogno di un sommario di dati, con certe caratteristiche. Per ottenerlo celermente e con facilità abbiamo il comando "REPORT".

Se state usando fogli singoli nella vostra stampante, battete prima di tutto "SET EJECT OFF" per spegnere il "formfeed". Dopo aver selezionato la base-dati dalla quale volete estrarre le informazioni inserite:

"SET EJECT OFF"
"USE <nome del file>"
"REPORT"

Il dBASE II vi guiderà, attraverso una serie di chiamate, nella costruzione di un formato personalizzato per il vostro rapporto o relazione. Tocca a voi specificare quali campi della vostra base-dati volete, quali sono i titoli adatti al rapporto e alle co-

lonne, quali di queste vanno totalizzate, ecc. I valori standard (default) sono di 8 colonne a partire dal margine sinistro della carta per la pagina stampata, di 56 righe per pagina e di 80 caratteri per ciascuna riga.

Provate ad eseguire questa operazione sul disco dimostrativo, con i file creati fin qui. Ma le base-dati <Nomi> e <Ordini> non contengono dati sufficienti per dimostrarvi tutta la potenza del dBASE II. Di qui in avanti useremo <Pagamenti.DBF> ed altri file contabili esistenti sul disco.

E' opportuno creare una struttura di base-dati che possiate realmente usare nella vostra azienda. Introducetevi i vostri dati e sostituitela al file <Pagamenti> che continueremo a porre come esempio.

```
.use Pagamenti
.report
ENTER REPORT FORM NAME:CostoSer
ENTER OPTIONS,M = LEFT MARGIN, L = LINES/PAGE, W = PAGE WIDTH
PAGE HEADING? (Y/N) y
ENTER PAGE HEADING: RIASSUNTIVO DEI COSTI
DOUBLE SPACE REPORT? (Y/N) n
ARE TOTALS REQUIRED (Y/N) y
SUBTOTALS IN REPORT? (Y/N)n
COL        WIDTH,CONTENTS
001        001,ASS:Data
ENTER HEADING:DATA
002        22,Nome
ENTER HEADING:FORNITORE
003        22,Descr
ENTER HEADING:DESCRIZIONE
004        12,Importo
ENTER HEADING: IMPORTO
ARE TOTALS REQUIRED? (Y/N) y
005        "<enter>"
```

Dopo aver definito il contenuto del rapporto, premete < enter> subito dopo l'ultimo campo che avete riempito.

Il dBASE II entrerà immediatamente in azione cominciando col presentarvi la vostra maschera di stampa e elaborando tutta la base-dati. Se volete fermarlo, premete il tasto <escape>.

Contemporaneamente, il dBASE II salva il formato stabilito per il rapporto in un file con l'estensione .FRM permettendovi di usarlo nuovamente senza bisogno di ridefinirlo. Ecco la forma completa del comando

"REPORT FORM <nome del rapporto> [scopo] [FOR < espressione>] [TO PRINT]"

Ciò significa che, battendo

"REPORT FORM CostoSer FOR Serv:Num = '770'"

otterremo una lista di tutti i costi del servizio N.770 senza bisogno di ridefinire il formato.

```
.REPORT FORM CostoSer FOR Serv:Num = 770

Page NO 00001

                        SOMMARIO DEI COSTI

DATA        FORNITORE           DESCRIZIONE          IMPORTO
830113      Lettera SRL         Linotipo              17.700
830113      Mercurio CA         Spedizione             8.500
830113      Longoni,Franco      Fotografia            53.700
830113      Testa,Livia         Impaginazione         20.000
830113      3Design SRL         Titolazione           56.500
830113      Dani,Mario          Copia                 15.600
**TOTAL**                                            172.000
```

Per cambiare il titolo si inserisce "SET HEADING TO stringa alfanumerica" (fino a 60 caratteri compresi gli spazi e, come qui, senza virgolette).

Lo scopo, se non viene specificato, è ALL (il file completo).

Le espressioni si possono espandere con altre condizioni e tutto il rapporto può essere stampato su carta aggiungendo TO PRINT alla fine del comando.

CONTEGGIO E SOMMA AUTOMATICA (COUNT, SUM)

Vi saranno applicazioni in cui ciò di cui avete bisogno non sarà esaminare dei records, ma sapere quanti o quali di essi soddisfano a certe caratteristiche e il totale di quelli che si trovano in una determinata condizione (per esempio: Quanti articoli abbiamo in magazzino? Quanti ne sono stati ordinati? Qual'è l'importo totale delle ordinazioni?).

Per contare dovete usare:

^COUNT [scopo] [FOR condizioni] [TO variabile di memoria]^

Questo comando si può usare sia solo sia con tutte le varianti. Se non si specifica altrimenti, conta tutti i record della base-dati. Lo "scopo" può essere limitato a uno o ad un certo numero di records. La "condizione" può essere qualsiasi espressione logica complessa (Vedi Sezione I). Il risultato del conteggio può essere assegnato ad una variabile di memoria che, se non esiste previamente, viene creata al momento di esecuzione del comando.

Per ottenere un totale userete:

^SUM campo (o campi) [scopo] [FOR condizione][TO variabile (o variabili) di memoria]^

Potete elencare fino a 5 campi numerici da totalizzare all'interno della base-dati in USE. Quando si tratta di più di un campo, i nomi rispettivi verranno separati da virgole. Per i records, i limiti si fissano mediante lo "scopo" e/o le espressioni condizionali collocate dopo il FOR (Cliente < >'SEM' .AND. Importo >10, per esempio).

Se si usano variabili di memoria (separate da virgole), ricordate che i totali vengono immagazzinati secondo la loro posizione. Se non volete immagazzinare gli ultimi campi nelle variabili di memo

ria ma volete vedere qual'è il loro importo, assegnate semplicemente le prime variabili che desiderate. Se operate in base a una scelta di campi (per es., volete salvare il primo, il terzo e il quarto campo di una serie di records che ne hanno sei), nominate le variabili di memoria corrispondenti ai primi quattro campi, fate la somma (SUM) e poi scartate (RELEASE) la seconda.

```
    .USE Pagamenti
    .Count FOR Importo > 100 TO Piccolo
COUNT = 00060
    .SUM Importo FOR Serv:Num = 770 TO Costo
172.000
    .display memory
PICCOLO                          (N)                    60
COSTO                            (N)               172.000
**TOTAL**              02 VARIABLES  USED      00012 BYTES USED
```

RIASSUNTIVO DEI DATI ED ELIMINAZIONE DEI DETTAGLI (TOTAL)

"TOTAL" agisce in modo simile alla funzione di "sottototale" che ha il comando "REPORT", eccetto che i risultati, invece di venire stampati, sono collocati in una base-dati:

TOTAL ON <chiave> TO <nome del file> [FIELDS lista] [FOR condizioni]

NOTA : La base-dati da cui proviene l'informazione deve essere stata ordinata o indicizzata previamente secondo la chiave usata in questo comando.

Si tratta anche di un comando particolarmente utile per eliminare dettagli e fornire relazioni. Sullo schermo vi apparirà cosa accade con la nostra base-dati <Pagamenti>:

```
"USE Pagamenti"
"INDEX ON Serv:Num TO Servizi"
"USE Pagamenti INDEX Servizi"
"TOTAL ON Serv:Num TO Provv FIELDS Importo FOR Serv:Num > 699;
      .AND.Serv:Num <800"
"USE Provv"
"LIST"
```

La nuova base-dati ha un'entrata per ogni numero di servizio e un totale dei costi relativi. Il problema di questa nuova base-dati è quello di avere solo due campi con informazione utile. Ma anche questo viene risolto, mediante un'altra linea di comando.

Il comando "TOTAL" trasferisce tutti i campi se la base-dati nominata non esiste, ma se esiste ne utilizza la struttura. Nel comando precedente, avremmo potuto limitare i campi della nuova base-dati creandola previamente, ossia prima di usare il comando "TOTAL".

```
"COPY TO Provv FIELDS Serv:Num,Importo"
```

In questo modo, quando eseguiamo "TOTAL TO Provv" troviamo che la nuova base-dati contiene solo i numeri dei servizi e i totali. Provate a farlo con la base-dati che avete appena creato.

Questa stessa tecnica può essere usata per riassumere qualsiasi altra informazione ordinata (elaborata attraverso SORT o INDEX).

```
    .USE Pagamenti
    .INDEX ON Serv:Num TO Lavori
0093 RECORDS INDEXED
    .USE PAGAMENTI INDEX Lavori
    .TOTAL ON Serv:Num TO Provv FIELDS IMPORTO FOR Serv:Num>699;
                                                .AND. Serv:Num<800

00028 RECORDS COPIED
    .USE Provv
    .LIST
00011   290181        3145      SML     778 LETTERA SRL  LINOTIPO
00012   290181        3146      SMM     769 TESTA,LIVIA  TITOLI
00013   010281        3148      SMN     770 MERCURIO CA  DOGANA
00014   010281        3149      SML     773 DANI,DANILO  COPIA
```

Lista parziale

SOMMARIO DELLA SEZIONE II

In questa Sezione abbiamo allargato il campo delle possibilità di gestione di dati con il dBASE II.

Vi abbiamo mostrato come potete usare differenti operatori (aritmetici, relazionali e di stringa) per modificare i comandi del dBASE II per ottenere un maggior grado di controllo sui vostri dati.

Siccome strutturare i dati costituisce la base dei sistemi di base-dati, abbiamo passato in rassegna una serie di modi diversi che vi consentono di alterare le strutture dei file, con o senza dati nella vostra base-dati.

Vi abbiamo anche mostrato come introdurre, alterare e recuperare l'informazione specifica che state cercando. Inoltre, vi abbiamo presentato altre istruzioni, che vi permettono di trasformare tutti i vostri dati in informazione con un solo comando (COUNT, SUM, REPORT, TOTAL).

Nella prossima Sezione vi insegneremo a preparare i file di comandi (programmi), in modo che possiate automatizzare l'elaborazione della vostra informazione.

S E Z I O N E I I I

Se avete già imparato a scrivere le espressioni, siete ormai molto vicini a saper scrivere un programma.

Vi sono quattro strutture di programma basilari che possono portare il computer a fare ciò che si vuole:

- Sequenza
- Scelta/decisione
- Ripetizione
- Procedure

Avete già visto che dBASE II elabora i vostri comandi in sequenza, ovvero secondo l'ordine in cui voi li avete dati. In questa parte vi si spiegherà come ripetere una sequenza di comandi (DO WHILE...) e come usare i sotto-file comandi, ossia le procedure.

Poi vi mostreremo come usare questi semplici strumenti per scrivere i file comandi (programmi) che risolveranno i vostri problemi applicativi.

PREPARAZIONE DI UN FILE COMANDI
(PER SCRIVERE IL VOSTRO PRIMO PROGRAMMA)

I comandi che vi abbiamo presentato sono molto potenti e possono compiere una gran quantità di funzioni, eppure non hanno messo in piena luce la potenza del dBASE II. La sua vera potenza comincia nel momento stesso in cui voi preparate il vostro file di comandi e scoprite che potete utilizzare i comandi in forma ripetitiva e permanente.

Creare un file comandi significa programmare il computer, ma dato che i comandi del dBASE II sono simili al linguaggio naturale, ciò è molto più semplice di quello che sembra. Inoltre, essendo il dBASE II un sistema di gestione di dati relazionale, voi lavorerete sui dati e sulle informazioni più che sui bit e sul byte.

Per preparare un file comandi, fate un elenco dei comandi che volete far eseguire in un file, con estensione <.CMD> o <.PRG> a seconda che usiate come sistema operativo il CP/M o MS-DOS. Farete ciò tramite qualsiasi programma di video scrittura.

Il dBASE II comincia al principio e elabora i comandi uno per volta fino a giungere al termine dell'elenco.

Gli altri linguaggi operano esattamente allo stesso modo. Con BASIC la sequenza è molto visibile perché tutte le righe del programma sono numerate. Con altri linguaggi (tra cui quello del dBASE II) la sequenza è implicita e il computer elaborerà la prima riga della pagina, poi la seconda, e così via. Alcuni linguaggi utilizzano separatori (come i due punti) tra una frase di comando e l'altra; il dBASE II usa semplicemente il ritorno carrello (CR), per terminare la linea di comando.

L'unico caso in cui non viene seguita la sequenza è quello in cui il computer riceve l'istruzione specifica di fare qualcos'altro. In genere questo avviene quando qualche condizione o espressione che voi avete preparato all'interno del file comandi fa sì che il computer deva prendere una decisione. Di ciò parleremo più avanti.

Per il momento, create un file comandi chiamato <Prova>.

Potete farlo utilizzando un programma di video scrittura, ma il dBASE II vi facilita il compito; battete:

MODIFY COMMAND Prova

Vi si presenta così uno schermo vuoto sul quale potete scrivere usando i comandi di composizione a schermo intero, già descritti. Introducete il breve programma che vi proponiamo (non battete i simboli "^"), usando gli stessi criteri del "Full screen editor".

Il termine di una riga indica il termine di un comando (a meno che usiate un punto e virgola). Scrivete:

```
^USE Nomi^
^COPY STRUCTURE TO Provv FIELDS Nome,Citta Prov^
^USE Provv^
^APPEND FROM Nomi^
^COUNT FOR Nomi = 'G' TO G^
^DISPLAY MEMORY^
^? Abbiamo completato con successo il nostro primo file comandi'.
```

Quando avrete finito, usate ^ctl-W^ (^ctl-O^ con Superbrain) per tornare al punto di richiamo del dBASE II. Adesso scrivete:

```
^DO Prova^
```

Se avete battuto il programma esattamente come compare qui, non funzionerà. Dovrete battere ^MODIFY COMMAND Prova^ e inserire i due punti per correggere il nome del campo Citta:Prov.

Quando inizierete a preparare per contro vostro file comandi più complessi, vi renderete conto che questo "editor incorporato" del dBASE II è una delle sue caratteristiche più convenienti, dato che vi consente di scrivere, correggere e cambiare i programmi senza dover uscire dall'ambiente dBASE. In genere, questo "editor" incorporato può manipolare soltanto 5.000 caratteri circa, per cui in caso di file più grandi sarà necessario pianificare questa fase.

Il file comandi che abbiamo appena preparato è di per sè semplice, ma ci permette di dimostrare come sia possibile eseguire una sequenza di comandi collezionati all'interno di un file. Si tratta di un procedimento simile a quello che vi consente di usare i file .COM all'interno del sistema operativo.

Suggerimento: Se volete, potete cambiare il nome del file principale del dBASE II da dBASE.COM in DO.COM. In questo modo potrete eseguire i vostri programmi battendo "DO <nome del file>" sia che vi troviate in dBASE II, sia nel sistema operativo. Potete farlo attraverso il comando di sistema operativo.

```
A >  "REN  DO.COM = DBASE.COM"
```

SCELTE E DECISIONI (IF ... ELSE)

Per scegliere e per decidere, il dBASE II dispone del comando "IF ...ELSE...ENDIF", che agisce in gran parte secondo il suo significato letterale: SE (IF) ho fame, mangio; ALTRIMENTI (ELSE) o anche OPPURE (OR), non mangio. Con il computer si applica questa stessa costruzione, facendo attenzione di usare queste parole esattamente secondo il loro significato.

Decisioni semplici: Se la decisione da prendere deve essere una sola, mettete da parte ELSE e usate questa forma:

```
IF condizione  [.AND.cond.2 OR.cond.3...]
     esegui questo comando
     [comando 2]
     [ ... ]
ENDIF
```

La "condizione" può essere una serie di espressioni (fino a un massimo di 254 caratteri) valutabili logicamente come vere o come false. Usate gli operatori logici per collegarle fra di loro. Se usiamo Pagamenti possiamo preparare la seguente richiesta:

```
IF Serv:Num = '730'.AND.Importo > 99.99;
        .OR.Fornitore = '3DESIGNSRL';
        .OR.Fatt:Data > '821231'
esegui questo comando
[comando 2 ]
[...]
ENDIF
```

Se si verificano tutte le condizioni indicate, il computer eseguirà i comandi elencati fra IF e ENDIF (in sequenza), e poi passe

rà alla istruzione successiva ad ENDIF. Se non trova nessun record che soddisfi a quelle condizioni, salterà direttamente al comando successivo ad ENDIF.

Scelta fra due alternative: Se invece bisogna scegliere fra due possibili azioni, secondo la condizione posta, usate IF...ELSE:

```
IF (condizione (o condizioni)
    esegui comando 1 (o serie di comandi 1)
ELSE
    esegui comando 2 (o serie di comandi 2)
ENDIF
```

Il computer eseguirà allora la prima serie di comandi oppure la seconda serie, prima di passare alla istruzione seguente a ENDIF.

Scelta fra varie alternative: Spesso è necessario scegliere da una lista di alternative. Un buon esempio potrebbe essere l'utilizzo di un "menù" che appare sullo schermo da cui selezionare una delle diverse procedure. In questo caso, userete la costruzione IF...ELSE...ENDIF.

Questa costruzione è la stessa di IF...ELSE, ma si deve usare a vari livelli (chiamati "annidamenti"), come vi mostriamo qui di seguito:

```
IF condizioni 1
   esegui comandi 1
ELSE
   IF condizioni 2
      esegui comandi 2
   ELSE
      IF condizioni 3
         esegui comandi 3
      ELSE
      .
      .
      .
      ENDIF 3
   ENDIF 2
ENDIF 1
```

Questa struttura deve venire "annidata" (nested) in questo modo tante volte quante siano le alternative tra le quali bisogna sce-

gliere. Vi consigliamo comunque, prima di scrivere un programma di "menù", di consultare il comando DO CASE, nella seconda parte del manuale; faciliterà il vostro compito.

Notate che ciascun IF deve avere il suo corrispondente ENDIF. Se non si osserva questa regola, il vostro programma non funzionerà correttamente.

Suggerimento: Il dBASE II non legge il resto della riga dopo END-IF, perciò potete perfettamente aggiungervi le identificazioni o i riferimenti che volete, così come lo abbiamo fatto nel nostro esempio. Questa abitudine, infatti, aiuta a mantenere leggibile il vostro programma mentre lo scrivete.

LA RIPETIZIONE DI UN'ESECUZIONE (DO WHILE)

La possibilità di fare un insieme di istruzione è uno dei maggiori vantaggi di un computer. Esso può continuare ad eseguire lo stesso compito innumerevoli volte senza annoiarsi né sbagliarsi a causa della monotonia di un'operazione; Nella maggior parte dei linguaggi il comando adatto a questo scopo è:

```
DO WHILE condizioni
     esegui questi comandi
ENDDO
```

Mentre (WHILE), ossia fino a quando, le condizioni specificate continueranno a risultare logicamente vere, i comandi verranno eseguiti.

Suggerimento: Ricordate che arriverà un momento in cui questi stessi comandi dovranno cambiare le condizioni, altrimenti questo ciclo (loop) continuerà all'infinito.

Se sapete a priori quante volte l'esecuzione debba essere ripetuta, usate questa scrittura:

```
STORE 1 TO Indice                    * Inizializza il contatore
DO WHILE Indice  < 11                * Elabora 10 record
     IF Articolo = ''                * Se non vi sono dati
        SKIP                         * salta il record, e
        LOOP                         * torna al DO WHILE
     ENDIF  spazio                   * senza eseguire il ProcessoA
     DO ProcessoA                    * Esegue il file ProcessoA.CMD
     STORE Indice + 1 TO Indice      * Aumenta il contatore di 1
ENDDO  dieci volte
```

In questo esempio, se vi sono dati nel campo < Articolo > il computer esegue tutte le istruzioni contenute nel file di comandi < ProcessoA.CMD > e poi torna al punto del file di comandi in cui si trovava. Aumenta di 1 il valore della variabile indice e subito dopo controlla se questo valore è minore di 11, nel qual caso procede a seguire di nuovo le istruzioni del DO WHILE. Quando il contatore supera il 10, il computer salta direttamente ad eseguire le istruzioni che seguono ENDDO.

L'istruzione LOOP si usa per fermare una sequenza e far tornare il computer all'inizio del ciclo DO WHILE che la contiene.

In questo caso, se il campo Articolo è vuoto, il record non viene elaborato perché il comando LOOP rimanda il computer al DO WHILE Indice < 11. Il record vuoto non viene contato dato che non viene eseguita l'istruzione che aggiunge 1 al contatore.

L'inconveniente di LOOP è che altera il flusso del programma, rendendo estremamente difficile seguire la logica del programma stesso. Perciò vi consigliamo di utilizzarla con la massima attenzione.

LE PROCEDURE (FILE SUSSIDIARI DI COMANDI)

La possibilità di creare procedure standard in un linguaggio semplifica in gran misura la programmaizone di un computer.

In BASIC, queste procedure si chiamano subroutine. In PASCAL e in PL/1, si chiamano procedure. In dBASE II, sono considerate file di comandi, richiamabili da un programma sviluppato da voi.

Nell'esempio precedente abbiamo richiamato una procedura quando abbiamo indicato DO ProcessoA. "ProcessoA" è un altro file di comandi (con l'estensione .CMD). Il contenuto di questo file di comandi potrebbe essere questo:

```
IF Stato = S
      DO Sposato
ELSE
      IF Stato = L
            DO Libero
      ELSE
            IF Stato = CF
                  DO Capofam
            ENDIF
      ENDIF
ENDIF
RETURN
```

Ancora, possiamo richiamare altre procedure che a loro volta richiamano altri file comandi. Si possono mantenere aperti contemporaneamente al più 16 file; quindi, se uno di essi si trova in USE, se ne possono aprire altri 15. Alcuni comandi utilizzano sempre dei file aggiuntivi. REPORT, INSERT, COPY, SAVE, RESTORE e PACK ne utilizzano uno; SORT utilizza due file aggiuntivi.

Questa è raramente una limitazione; comunque, se i files una volta utilizzati vengono chiusi e non se ne aprono più di 16 alla volta, se ne possono usare un numero infinito.

I file vengono chiusi quando si arriva al termine del file comandi o si esegue il comando RETURN; così facendo, si restituisce il controllo al file comandi che lo ha attivato o alla tastiera del computer se il file è stato lanciato direttamente da essa.

Benché il comando RETURN non sia sempre strettamente necessario (come accade nel caso citato, in cui il raggiungimento della fine di un file compie automaticamente questa funzione), inserirlo d'abitudine quando si arriva al termine di un file è senz'altro una buona norma di programmazione.

Suggerimento importante: Avrete notato che nei nostri esempi le righe di comando vengono "sfalsate". Ciò non è indispensabile, ma specialmente quando abbiamo annidamenti di strutture accresce notevolmente la chiarezza del file comandi. Anche l'uso di maiuscole e minuscole per i comandi è di molto aiuto.

L'INTRODUZIONE DI DATI DURANTE L'ESECUZIONE DI UN PROGRAMMA (WAIT, INPUT, ACCEPT)

In molte applicazioni i file comandi non si potranno limitare all'elaborazione delle informazioni contenute nella base-dati, ma avranno bisogno di ricevere altri dati dall'utente.

In questo senso, è possibile preparare i file comandi in modo che l'utente possa inserire di volta in volta le informazioni necessarie. Come esempio, pensiamo alla scelta di una delle procedure di un "menù". Oppure, alla possibilità di assicurarsi che dei dati siano stati inseriti correttamente.

I comandi appropriati per questi casi sono:

^WAIT [TO variabile di memoria]^.

interrompe l'elaborazione del file comandi e attende (wait) l'introduzione di un carattere dalla tastiera, presentando un richiamo di attesa. L'elaborazione riprende appena si batte un tasto qualsiasi (come con il comando di BASE II ^DISPLAY^).

Se, in più, si specifica una variabile, il carattere introdotto viene immagazzinato. Se il tasto che abbiamo battuto non corrisponde a un carattere di stampa (enter , ctl , ecc.) nella variabile viene introdotto uno spazio in bianco.

^INPUT ['messaggio'] TO variabile di memoria^

trasferisce qualunque tipo di dato nella variabile di memoria nominata, oppure crea la variabile nel caso che non esista.

Se si usa il messaggio di richiamo opzionale (tra virgolet-

te semplici o doppie, ma sempre uguali), esso apparirà sul terminale seguito dai due punti che indicano dove bisogna scrivere il dato. Il tipo di dato della variabile (alfanumerico, numerico o logico) viene determinato dal tipo di dato che è stato introdotto. Le stringhe di caratteri si devono inserire fra virgolette o parentesi quadre.

^ACCEPT ['messaggio'] TO variabile di memoria^

accetta dati alfanumerici senza bisogno di delimitatori. E' molto utile per introdurre stringhe lunghe.

Raccomandazioni:

Il comando di WAIT si può usare per introduzioni rapide (reagisce istantaneamente a un "input") ma non si deve utilizzare se un inserimento errato può danneggiare seriamente la vostra base-dati.

Il comando di ACCEPT è utile per le stringhe alfanumeriche lunghe perché non richiede virgolette né parentesi. Si può usare anche per battere un solo carattere quando la necessità di battere il tasto < enter > dopo l'immissione può salvaguardare l'integrità dei dati.

INPUT accetta sia dati numerici, logici che alfanumerici, e si usa come ACCEPT.

LA COLLOCAZIONE DI DATI E MESSAGGI IN DETERMINATE POSIZIONI (TEXT, @ ...SAY...GET, .FMT)

I comandi ^?^, ^ACCEPT^ e ^INPUT^, come abbiamo visto, servono a posizionare sullo schermo i messaggi diretti all'utilizzatore.

Lo svantaggio che presentano queste funzioni è che i messaggi appaiono immediatamente al di sotto dell'ultima riga visualizzata. Ciò non limita la sua utilità, ma esiste un modo più elegante.

Se il vostro terminale possiede la gestione di cursore X-Y, esiste un altro comando del dBASE II che vi consentirà di collocare

i vostri messaggi o di ottenere i vostri dati in una qualsiasi posizione sullo schermo:

```
@ < coordinate > [SAY  < 'messaggio' >]
```

Il messaggio (che avrete introdotto fra virgolette o parentesi quadre) apparirà alle coordinate che avete indicato. Le coordinate sono la riga e la colonna del CRT, essendo 0,0 la posizione superiore sinistra sul video (punto d'origine). Se specifichiamo le coordinate "9,34", il nostro messaggio comincerà sulla decima riga della trentacinquesima colonna.

NOTA: Se avete installato la media intensità o il video inverso il messaggio vi sarà presentato sotto una di queste due forme di visualizzazione. Per disabilitare ciò dovrete ripetere la procedura d'installazione e usare l'opzione "MODIFY/Change".

La frase SAY... è opzionale; infatti il comando ... si utilizza anche per cancellare sullo schermo una riga qualunque (o una parte di una riga).

Servendovi del dBASE II, battete:

```
ERASE
@ 20,30 SAY 'Come?'
@  5,67 SAY "c'è..."
@ 11,11 SAY 'Basta così'
@ 20,0
@  5,0
@ 11,16
```

Ma oltre a mostrare semplicemente un messaggio, questo comando serve per presentare il valore di un'espressione con una o più variabili. La sua formula è:

```
    < coordinate > [SAY  < espressione >]
```

Scriviamo in dBASE II:

```
USE Nomi
@  13,9 SAY Cap
@  13,6 SAY Citta:Prov
SKIP 3
@  23,5 SAY Nome + ',' + Indirizzo
```

Il comando può venire espanso ulteriormente per mostrarvi i valori delle variabili che si stanno utilizzando (variabili di memoria o nomi di campi di una base-dati) in qualsiasi punto dello schermo che voi scegliete. (Le variabili utilizzate da GET e SAY devono esistere antecedentemente alla vostra chiamata: in caso contrario vi sarà segnalato un errore).

^ @ <coordinate> [SAY <espressione>] [GET <variabile>]

Per vedere come funziona, battete ciò che segue (non uscite dal dBASE II dopo aver finito, perché dobbiamo continuare).

```
^ERASE
^USE Nomi^
^ @ 15,5 SAY 'Citta:Prov' GET Citta:Prov^
^ @ 10,17 GET Cap^
^ @  5,0 SAY 'Nome' GET Nome
```

(RESTARE IN dBASE)

Questa sequenza posiziona i valori delle variabili (con messaggi e senza messaggi) in diversi punti dello schermo. Con questo strumento potete disegnare la maschera per l'introduzione di dati, in modo che l'operatore veda lo schermo esattamente come i vecchi moduli di carta a cui era abituato.

Per introdurre dati nelle variabili situate sullo schermo secondo il vostro schema, inserite:

^READ^

Il cursore si posiziona da solo sul primo campo che avete inserito, e voi potete scrivere in quel punto dei nuovi dati oppure lasciarlo così battendo < enter>. Quando lasciate questo campo, il cursore si posiziona sulla seconda variabile.

Cambiate i dati nei due campi restanti. Quando avrete finito, vi troverete di nuovo in dBASE II. Inserite allora ^DISPLAY^. Il record conterrà i nuovi dati da voi introdotti.

Abbiamo visto che GET funziona in modo simile ai comandi INPUT e ACCEPT, ma è molto più potente perché vi permette di introdurre molte variabili.

Il record di una base-dati può avere fino a 32 campi, ma a voi

può interessare riempirne solo alcuni. In questo caso, invece di utilizzare ^APPEND^, che vi presenterebbe la lista di tutti i campi esistenti nel record, è più efficiente usare ^APPEND BLANK^ per creare un record di campi vuoti, riempiendo poi quelli che vi interessano mediante GET.

Il nostro file <Nomi> non è l'esempio migliore, ma può servire per dimostrarvi con quale flessibilità si possono collocare i dati in una base-dati con struttura complessa.

Continuate ad esercitarvi con i file comandi creando un file chiamato < Prova.CMD > che contiene i seguenti comandi:

```
^ERASE^
^?^ 'Questa procedura permette di aggiungere nuovi record a'^
^?  'Nomi.DBF in modo selettivo. Ora aggiungiamo'^
^?  'solamente il nome e il Cap.'^
^?^
^?  'Battete S per fermare la procedura,'^
^?  '<enter > per continuare'^
^WAIT TO Continua^
^USE Nomi^
^DO WHILE Continua < > 'S' .AND. Continua < >'s'^
^APPEND BLANK^
^ERASE^
^  @ 10,0 SAY "NOME" GET Nome^
^  @ 10,30 SAY "CAP" GET Cap^
^READ^
^?  'S per fermare la procedura'^
^?  '<enter > per continuare'^
^WAIT TO Continua^
^ENDDO^
^RETURN^
```

Quando vi trovate di nuovo in CP/M o MS/DOS, inserite ^dBASE Prova^ (oppure ^DO Prova^ se avete cambiato il nome del file dBASE.COM secondo i nostri suggerimenti). Ora potete introdurre dati nei records. Quando avrete finito, battete ^LIST^ per controllare cosa avete fatto.

Come vedete, l'introduzione dati è stata semplice e non ha provocato nessuna confusione mentre voi avete personalizzato il vostro schermo collocando messaggi e campi di variabili dove desideravate.

NOTA: Dovete usare "ERASE" o "CLEAR GETS" dopo ogni 64 "GET". Usate il secondo comando se non volete variare la maschera del videc.

UN FILE COMANDI CHE RIASSUME CIO' CHE ABBIAMO IMPARATO

Prima di continuare a leggere, fate eseguire il seguente programma per vedere il suo comportamento. Scrivete <dBASE Campione> se vi trovate in CP/M o MS/DOS oppure "DO Campione" se siete in dBASEII. Rispondete ai messaggi. Dopo averlo eseguito, potete tornare indietro. Scoprirete che riassume la maggior parte di ciò che abbiamo spiegato finora e che comprende numerose osservazioni e commenti.

```
************************** CAMPIONE.CMD **************************
*
* Questo file comandi richiama l'attenzione dell'utente mediante
* messaggi visualizzati e accetta dati in una variabile di memo-
* ria per poi eseguire la procedura selezionata. Questo è solo u-
* na parte di programma, ma funziona. Non abbiamo ancora scrit-
* to le procedure richiamate dal menù, perciò possiamo solo far
* eseguire al computer una serie di azioni che ci mostreranno
* ciò che fa e quali sono i suoi passi consecutivi (stubbing).
*
* Normalmente, dBASE II mostra i risultati dei comandi su CRT. Da
* to che in questo caso potrebbero confondere, lavoreremo con SET
* TALK OFF.

SET TALK OFF
USE Pagamenti
ERASE

* E' buona norma cancellare (erase) lo schermo prima di visualiz-
* zare dei dati nuovi.
* Invece di usare il comando DISPLAY, le informazioni possono es-
* sere rappresentate sullo schermo CRT in questo modo:
```

```
?
?
?
?
? '             MENU' USCITE DI CASSA'
?
?
? '             0 = Esci'
? '             1 = Sommario di Contabilità
? '             2 = Introdurre Nuove Fatture'
? '             3 = Introdurre Pagamenti Effettuati'
?
? '             Scegliere il Numero'
WAIT TO Scegliere
ERASE

* Dato che non abbiamo ancora sviluppato le procedure per le ulti
* me tre voci, chiederemo al computer di visualizzare     diverse
* osservazioni,  secondo le alternative che saranno   selezionate
* dal menù.

IF Scegliere = '1'
   @  0,20 SAY 'Uno'
ELSE
   IF Scegliere = '2'
      @  1,20 SAY 'Due'
   ELSE
      IF Scegliere = '3'
         @  2,20 SAY 'Tre'
      ELSE
         @  7,20
         @  8,20 SAY 'TUTTI I CARATTERI ECCETTO 1,2,3'
         @  9,20 SAY 'FANNO TERMINARE QUESTO FILE COMANDI'
         @ 10,20 SAY 'DOPO LA STAMPA DI QUESTO MESSAGGIO.NOTATE'
         @ 11,20 SAY 'CHE I NUMERI CHE SEGUONO IL COMANDO "IF" SONO'
         @ 12,20 SAY 'TRA VIRGOLETTE PERCHE  IL COMANDO'
         @ 13,20 SAY '"WAIT" ACCETTA SOLO CARATTERI'
         @ 14,20 SAY '                                '
      ENDIF 3
   ENDIF 2
ENDIF 1

* Ciascun IF deve avere il suo corrispondente ENDIF. Abbiamo an-
* che numerato gli ENDIF per indicare a quali IF   corrispondono,
* in modo da essere sicuri di aver chiuso tutti i blocchi.
```

```
?
?
?
?
INPUT 'Volete continuare (Si o No)?'  TO Decisione
ERASE
IF Decisione
   INPUT 'Va bene, dammi subito un numero.' TO Numero
ELSE
   @  10,20 SAY "PERCHE' NO?"
   WAIT
ENDIF
ERASE
 @ 10,20 SAY 'NON SONO ANCORA PRONTO. ARRIVEDERCI'

* Il prossimo ciclo DO WHILE prevede una sosta di pochi  secondi
* per mantenere l'ultimo messaggio su CRT abbastanza a lungo da
* permettere la sua lettura prima che termini il programma. Vi sa
* rà utile includerlo nel file comandi che state scrivendo.  Per
* alterare il tempo di sosta si può cambiare il limite (100) oppu
* re il passo (+1).

STORE 1 TO X
DO WHILE X  < 100
   STORE X + 1 TO X
ENDDO
ERASE
RETURN
*************************************************************************
```

Se volete, avviate il nuovo programma. Provate tutte le alternati ve e provate anche a introdurre elementi sicuramente sbagliati.Ve drete così come si comporta il programma e come il dBASE.II ma- neggia gli errori.

Benché < Campione.CMD > sia solo una parte di un programma e non realizzi alcun lavoro utile, ci serve a chiarire varie cose:

1. - L'uso frequente di ERASE è una buona norma di lavoro.

2. - L'uso delle rientranze del testo chiarisce il flusso del programma. Per la stessa ragione usiamo lettere maiuscole e lettere minuscole. Il computer le interpreta come se fosse ro tutte maiuscole, ma per l'utente risulta più semplice la comprensione delle operazioni.

3. - Il "?" si può usare per ottenere le spaziature fra le righe, e per rappresentare le stringhe alfanumeriche (tra virgolette o parentesi quadre).

4. - Il comando WAIT attende un solo carattere per far procedere il programma. Questo carattere di input deve essere trattato come un carattere alfanumerico, come abbiamo fatto nei blocchi "IF" quando abbiamo posto fra virgolette i valori che stavamo attendendo.

5. - Il comando INPUT attende e accetta qualsiasi tipo di dato, ma i caratteri e le stringhe alfanumeriche si devono scrivere fra virgolette semplici o doppie, o fra parentesi quadre. Quando abbiamo un apostrofo nel messaggio, dobbiamo usare le virgolette doppie o le parentesi quadre per definire la stringa evitando che il computer si confonda.

6. - Non è necessario definire le variabili in anticipo. Potete usare un nuovo nome ogni qualvolta ne abbiate bisogno (fino a un massimo di 64 contemporaneamente).

7. - I valori logici si possono trattare in forma abbreviata o ridotta. "IF Decisione" funziona come si avessimo scritto: "IF Decisione = T".

8. - Terminare il programma con il comando RETURN non è necessario, ma l'abbiamo utilizzato perché sarebbe obbligatorio se il programma fosse richiamato da un altro file comandi. E' attraverso questo comando che il computer sa che deve tornare dove si trovava in precedenza e non semplicemente abbandonare la procedura e ricedere il controllo alla tastiera (.).

LA GESTIONE DI BASE-DATI MULTIPLE (PRIMARY, SECONDARY SELECT)

Come abbiamo visto, quando cominciate a lavorare con il dBASE II per prima cosa battete sulla tastiera ^USE < nome del file>^ per far sapere al dBASE II qual'è il file che vi interessa, in secondo luogo procedete a introdurre dati, a "correggere" (edit), ecc.

Per lavorare su una base-dati diversa, dovete scrivere ^USE < nuovo file>^. Allora il dBASE II chiude il primo file e apre il secondo, senza che voi ve ne preoccupiate. In questo modo potete u-

sare il numero di file che volete, sia dal vostro terminale sia con un file comandi. Potete anche chiudere un file senza aprirne un altro battendo soltanto ^USE^.

Quando aprite un file tramite ^USE^, il dBASE lo apre posizionandosi sul primo record. Nella maggior parte dei casi questo è ciò che desideravate. In certe applicazioni, però, vorreste accedere ad un altro file (o a vari) senza "perdere il puntatore" nel primo file.

Il dBASE II ha una caratteristica eccezionalmente avanzata che vi consente di lavorare allo stesso tempo in due aree attive e separate: PRIMARY e SECONDARY. Andrete dall'una all'altra con il comando ^SELECT^.

Quando iniziate, vi trovate automaticamente nell'area PRIMARY. Per lavorare con un'altra base-dati senza perdere la vostra posizione nella prima, battete ^SELECT SECONDARY^ e poi ^USE < nuovo file >^. Per tornare nell'area di origine, battete ^SELECT PRIMARY^ e continuate a lavorare con la prima base-dati.

Le due aree di lavoro si possono gestire indipendentemente: qualsiasi comando che maneggia dati opera soltanto nell'area in USE.

Tuttavia, l'informazione può venire trasportata da un'area all'altra utilizzando i prefissi P. e S. davanti ai nomi dei campi. Se vi trovate nell'area PRIMARY, utilizzando la S. otterrete i dati che avete nell'area SECONDARY; allo stesso modo, userete il prefisso P. davanti al nome di qualsiasi campo dell'area PRIMARY di cui avete bisogno, quando vi trovate in SECONDARY.

Per esempio, questi comandi si usano per il sistema di contabilità del file < VerifNomi.CMD > alla fine di questa Parte del manuale. Ciascun record nella partizione primaria viene confrontato con i record di quella secondaria. Questo stesso comando viene usato anche con i files < CalTempi.CMD >, < RepTrasp.CMD > e <LibroPag.CMD>.

Anche se per ora non vi viene in mente nessuna applicazione del genere, non dimenticate questo comando perché ad un certo punto lo troverete utile.

COMANDI E FUNZIONI DEL SISTEMA

COMANDI

MODIFY COMMAND < nome del file >
vi permette di modificare il contenuto del file comandi indicato operando direttamente dal dBASE II, tramite le normali caratteristiche della composizione a schermo intero (full screen editing).

BROWSE
visualizza fino a 19 record e tanti campi quanti ne contiene lo schermo, consentendo di far scorrere i campi fuori schermo sul lato destro tramite ctl-B e sul lato sinistro tramite ctl-Z.

CLEAR
inizializza di nuovo il dBASE II, cancellando tutte le variabili e chiudendo tutti i file.

RESET
si deve usare dopo un cambio di disco per predisporre nuovamente la mappa di I/O del sistema. Prima dell'utilizzo di questo comando leggete con attenzione la sua descrizione nella Parte II di questo Manuale

*
permette di scrivere osservazioni o commenti in un file comandi, senza che vengano visualizzati durante l'esecuzione del programma; questo permette al programmatore di effettuare note senza confondere l'utilizzatore. Tra il simbolo e il commento deve esserci per lo meno uno spazio e altresì il commento non si può trovare nella stessa riga di un comando: ossia comandi e commenti devono andare su due righe separate.

REMARK
consente di immagazzinare i commenti in un file comandi, in modo che possano essere visualizzati come messaggi quando di esegue il programma. Si deve lasciare almeno uno spazio tra la parola e il comando REMARK, che non può stare sulla stessa riga di un comando.

RENAME < file vecchio > TO < file nuovo >
cambia i nomi dei file nel direttorio del CP/M o MS/DOS. NON CERCATE DI CAMBIARE I NOMI DEI FILE APERTI.

QUIT TO < lista file estensi.COM >
vi permette di uscire dal dBASE II e cominciare automaticamente l'esecuzione del CP/M o MS/DOS ed altri file .COM. Il nome di ciascuno di questi file deve essere posto fra virgolette semplici e separato con una virgola dai nomi degli altri file (anch'essi tra virgolette).

FUNZIONI

Il comando "?" attiva le seguenti funzioni:

= è la funzione del numero di record. Quando viene chiamata, presenta il valore del numero del record corrente (quello su cui stiamo lavorando).

* è la funzione di record soppresso e serve per controllare se effettivamente un record è stato eliminato (DELETED). In caso affermativo, questa funzione presenta il valore True (Vero) abbreviato .T.; in caso negativo, ossia se il record non è stato cancellato, presenta il valore False (Falso) abbreviato .F.

EOF è la funzione di fine di file. Indica .T. se abbiamo raggiunto la fine del file in USE, .F. nel caso contrario.

RACCOMANDAZIONI SULLA PROGRAMMAZIONE E LA PIANIFICAZIONE DEI FILE COMANDI

La prima cosa da fare quando volete organizzare un file comandi, è quella di spegnere il computer.

Proprio così: è questo un punto su cui si sbagliano molti programmatori, che cominciano immediatamente a "codificare" una soluzione prima di avere un'idea chiara di cosa devono fare.

Numerosi testi di programmazione strutturata e qualcuno dei linguaggi strutturati presentano una soluzione molto migliore a questo problema. Come riferimento, vi possiamo indicare il Capi-

olo 2 di "An Introduction to Programming and Problem Solving in Pascal" di Schneider, Weingart e Perlman; i Capitoli 1 e 4 e le prime pagine del Capitolo 7 di "Pascal Programming Structures" di Cherry; per approfondirle, "PL/1 Structured Programming" di Joan Hugnes.

In breve, ecco come affrontare la programmazione:

Cominciate col definire il problema in linguaggio naturale. Trasformatelo in un enunciato generale.

Definite ulteriormente il problema. Che tipo di dati dovete introdurre? In che forma dovrete richiamarli e organizzarli in rapporti?

Determinate le eccezioni. Quali sono le condizioni iniziali? Che succederebbe se si perdesse un record?

Descrivete i dettagli in linguaggio modificato. E' ciò che i testi di programmazione chiamano "pseudocodice". Il che vuol dire che state adattando il linguaggio naturale a quello inteso dal computer.

Ecco come potrebbe essere lo schema iniziale del vostro programma:

```
Usare il file Costi
Stampare le fatture da pagare del mese scorso
Fare un assegno da pagare per ogni fattura da pagare
```

Se entriamo in dettagli, scriveremo:

```
Use Costi
Stampare le fatture da pagare del mese scorso usando
                    il file SOMMARIO.FRM
Cominciare dall'inizio della base-dati
Continuare fino alla fine:
se una fattura non è stata pagata
                    Pagare la fattura
        e introdurre il pagamento nella base-dati
Fare questo per ogni record
```

Probabilmente, passando per un paio di altre fasi intermedie, possiamo arrivare alla seguente traduzione:

```
USE Costi
* Stampare il sommario su carta per Dicembre 1982
REPORT FORM Sommario FOR Fatt:Data >  = '821201'.AND.;
      Fatt:Data <  = '821201' TO PRINT
GOTO TOP                          *Andare al primo record
DO    WHILE.NOT.EOF               *Ripetere fino alla fine del file
      IF Ass:Num = '              *Se la fattura non è pagata
         DO FareAss               *fare un assegno e
         DO Aggiorn               * aggiornare i record
      ENDIF
SKIP
ENDDO
```

In realtà, la nostra è stata soltanto una maniera di affrontare sensatamente la soluzione della maggior parte dei problemi.

Primo, enunciare il problema generale; secondo, cercare di definire cosa è e cosa non è; terzo, approfondire gradualmente i dettagli, risolvere quelli più facili da risolvere e mettere da parte i più complicati per una soluzione posteriore (che forse sarà anch'essa parziale).

A questo punto, nel nostro esempio non abbiamo realizzato nessuno dei file che abbiamo nominato: né Sommario.FRM , né Fare Ass.CMD , né Aggiorn.CMD .

Ma non importa, perché probabilmente sarà più opportuno non preoccuparci ancora dei dettagli e invece concentrarci sulla soluzione generale del problema. Dopo aver verificato con calma questa soluzione potremo perfettametne tornare indietro e rifinire le procedure.

NOTA: Potete anche verificare un programma parziale come questo realizzando quello che i programmatori chiamano "stubbing". Preparate i file comandi citati nel programma e introducete tre elementi: un messaggio che vi lasci capire che il programma è arrivato fin lì; WAIT; RETURN. Il dBASE II eseguirà questi file di procedura e darà il vostro messaggio, quando voi premerete un tasto qualsiasi.

S E Z I O N E I V

COME POTETE MIGLIORARE IL CONTROLLO MEDIANTE FUNZIONI

Le funzioni sono operazioni con uno scopo ben preciso, che si usano all'interno delle espressioni per eseguire cose difficili o impossibili da realizzare mediante regolari operazioni aritmetiche, logiche o di stringa. Le funzioni del dBASE II restano comunque comprese in queste tre categorie, in base ai risultati che generano.

Il richiamo delle funzioni si effettua battendo "?", poi uno spazio e di seguito la funzione. Questa richiesta si può fare sia dal terminale sia dai file comandi.

Nota: Le parentesi fanno parte della sintassi della funzione e devono essere usate come descritto più avanti. (Ricordate che le "stringhe" sono semplici collezioni di caratteri - compresi spazi, cifre e simboli - che vengono manipolate e gestite in qualsiasi modo come dati. Una "sottostringa" è una porzione di una stringa).

Non vi preoccupate di ricordarvi adesso queste funzioni, ma leggete attentamente la descrizione che ve ne diamo in modo da sapere dove trovarle quando ne avrete bisogno in un file comandi.

! (<variabile/stringa>)

è la funzione minuscola-maiuscola. Mette in maiuscolo tutti i caratteri dall'"a" alla "z" di una stringa o di una variabile di stringa. Il resto dei caratteri della stringa rimangono inalterati. Vedrete che si usa spesso nei sistemi di contabilità per convertire a un formato standard le entrate effettuate dalla tastiera. Ciò serve a semplificare la ricerca dei dati, se stabilite che tutti i dati vengano immagazzinati in maiuscole, senza che importi in qual modo sono stati introdotti.

TYPE (<espressione>)

è la funzione tipo di dato e restituisce come risultato le lettere C, N, L o U in base al tipo di dato dell'espressione, a seconda che sia Carattere (alfanumerico), Numerico, Logico o Indefinito.

INT (<variabile/espressione>)

è la funzione numero intero. "Arrotonda" i numeri con decimali, ma lo fa eliminando tutto quello che si trova a destra del punto decimale. Il termine tra parentesi (ricordate che dovete usare le parentesi) può essere un numero, il nome di una variabile o un'espressione complessa. In quest'ultimo caso, per prima cosa viene considerata l'espressione, dopo di che viene formato un numero intero in base al risultato.

E' da notare che INT (123,86) produce 123, così come INT (-123,86) produce -123. La chiamata di una variabile restituirà un numero intero formato dal valore corrente di quella variabile. Se ci fossimo trovati sul record 7 di Pagamenti. DBF, la chiamata a INT (Importo) avrebbe prodotto 2333, ossia la parte intera di $ 2333.75.

Per approssimare al numero intero più prossimo (invece di troncare) usate questa forma: INT (valore + 0.5). Per prima cosa viene determinato il valore tra parentesi e poi viene eseguita la funzione.

La funzione di numero intero si utilizza anche per arrotondare il valore dei numeri decimali. INT (valore*10 + 0/5 / 10 arrotonda il valore al decimale più prossimo seguendo l'ordine di precedenza delle operazioni (parentesi, intero, divisione). Per arrotondare le prime cifre decimali bisogna usare "100" invece di "10". Per tre cifre, usate "1000", ecc.

VAL (<variabile/stringa/sottostringa>)

è la funzione da stringa a intero. Converte una stringa alfanumerica, o una sottostringa di numeri, compreso il segno e punto, nella quantità numerica equivalente. VAL('123') restituisce il numero 123. VAL(Serv:Num) restituisce il valore numerico del contenuto del campo Serv:Num nella nostra base-dati <Pagamenti>, che avevamo immagazzinato come caratteri. Si può usare anche con l'operatore di sottostringa: VAL ($(stringa)).

STR(<espressione/variabile/numero>,<lunghezza>,<decimali>)

è la funzione da intero a stringa. Converte un numero o il

contenuto di una variabile numerica in stringa, di lunghezza e quantità di cifre decimali specificate. La lunghezza data deve essere sufficiente a comprendere per lo meno tutte le cifre più il punto decimale. Se il valore numerico è più corto del campo specificato, la parte restante viene riempita da spazi in bianco. Se non specifichiamo il numero dei decimali, la funzione assume "0".

Spesso si usa questa funzione nei sistemi di contabilità, per semplificare la rappresentazione sul video. Così i numeri vengono convertiti e poi concatenati ad altre stringhe di caratteri prima della visualizzazione.

LEN (<variabile/stringa>)

è la funzione lunghezza di stringa, che ci dice quanti sono i caratteri contenuti in una stringa. E' utile quanto il programma deve decidere lo spazio per salvare l'informazione senza l'intervento dell'utente. Se si indica come variabile un campo alfanumerico, la funzione ci da la lunghezza del campo e non quella del suo contenuto (dato che le posizioni non utilizzate sono riempite da spazi bianchi dal dBASE II).

$ (<espressione/variabile/stringa>,<inizio>,<lunghezza>)

è la funzione sottostringa. Seleziona i caratteri in una stringa o in una variabile di caratteri, cominciando dalla posizione stabilita e continuando per tutta la lunghezza specificata.

Per esempio, se abbiamo una variabile chiamata Data il cui valore sia '810823', la funzione "$(Data,5,2)" ci darà '23'. Per convertire la stringa in numero, useremo "VAL ($(Data,5,2))".

Attenzione a non confondere questa funzione con l'operatore logico di sottostringa descritto nella Sezione II.

@ (<variabile/stringa1>,<variabile2/stringa2>)

è la funzione ricerca di sottostringa. E' come se dicessimo: "In che punto della stringa 2 si trova la strainga 1?". Questa funzione indica, infatti, la posizione in cui inizia

la prima stringa o variabile di caratteri all'interno della seconda stringa o variabile di caratteri. Se la prima stringa non è presente, il valore indicato dalla risposta è "0".

Uno dei suoi impieghi è la ricerca dell'inizio di una stringa in modo da poter usare la funzione di sottostringa spiegata sopra. Un'altra utilizzazione possibile è quella di sapere se una stringa specifica si trova o no dove noi supponiamo.

(Se voi volete sapere soltanto se una stringa si trova in un'altra, potete usare l'operatore relazionale di stringa String1$String2, spiegato nella Sezione II).

CHR (<numero>)

dà come risultato l'equivalente alfanumerico ASCII di un numero. A seconda di come il vostro terminale usa il codice standard ASCII, ? CHR(12) pulirà il costro schermo, ? CHR(14) produrrà il video inverso e ? CHR(15) lo annullerà. In altri casi potrà controllare dispositivi "hardware" come per esempio la stampante. Cercate nel vostro manuale: troverete probabilmente delle caratteristiche interessanti di questa funzione.

Per esempio, per ottenere la sottolineatura sulla vostra stampante, provate a costruire una stringa di caratteri come segue ? 'stringa' + CHR(13) + '___'. Potete anche preparare un file comandi che usi la funzione LEN per sapere la lunghezza della stringa e ottenere numerose battiture di sottolineatura.

è la funzione di macrosostituzione. Quando si usa questo simbolo davanti al nome di una variabile di memoria, il dBASE II sostituisce il nome con il valore della variabile di memoria (devono essere sempre dati alfanumerici). Ciò serve quando si deve usare con frequenza un'espressione complessa per passare dei parametri fra i files comandi, oppure in un file comandi quando il valore del parametro deve essere passato mentre il programma è in esecuzione.

Si usa anche per abbreviare un comando. Per esempio, se so-

stituiamo "STORE 'Delete Record' TO D" con "&D5" il record numero 5 verrà eliminato."

Se questo "macrocomando" non è seguito da una variabile stringa corretta, non viene eseguito. Il che significa che potete usarlo con il suo valore di simbolo tipografico come parte di una stringa senza che vi venga presentata un'indicazione di errore.

FILE (<"nome del file"/variabile/espressione>)

genera il valore True (Vero) se il file si trova sul disco e False (Falso) in caso contrario. Se si specifica il nome del file si devono usare le virgolette doppie, mentre il nome delle variabili di stringa non ne hanno bisogno. Si può usare anche un'espressione di stringa valida: "FILE ("B:" + Base-dati)" vi permetterà di sapere se il nome del file immagazzinato nella variabile di memoria Base-dati si trova nel drive B.

TRIM (<stringa>)

elimina gli spazi in coda di una variabile di stringa. Ciò si esegue battendo:

"STORE TRIM (<variabile>) TO <nuova variabile>"

RANK <stringa>)

restituisce il valore decimale del primo carattere di una stringa. Corrisponde alla funzione ASC, comune a molte versioni di BASIC.

LA MODIFICA DELLE CARATTERISTICHE (DEFAULTS) DEL dBASE II

Il dBASE II possiede una serie di comandi che controllano la sua interazione con il sistema. Se volete, potete modificare a vostro piacere questi parametri, cioè secondo la necessità del momento, oppure fissarli all'inizio del vostro file comandi. In molte applicazioni i defaults saranno proprio quelli di cui avete bisogno.

I parametri si cambiano, tanto nei file comandi come in maniera interattiva, mediante il comando SET. Nella lista che vi presentiamo, i normali valori di default sono sottolineati.

Anche qui, non è necessario ricordarli tutti. A mano a mano che esaminerete i valori di default, potrete decidere se volete cambiarli o no.

SET TALK	ON	Visualizza i risultati dei comandi.
	OFF	Non visualizza i risultati dei comandi.
SET PRINT	ON	Invia tutto ciò che compare su video anche alla stampante.
	OFF	Non lo invia alla stampante.
SET CONSOLE	ON	Invia i messaggi al terminale.
	OFF	Non compare nessun messaggio sul terminale.
SET SCREEN	ON	Attiva l'operazione a schermo intero per i comandi APPEND, EDIT, INSERT e CREATE.
	OFF	Disattiva l'operazione a schermo intero.
SET FORMAT TO SCREEN		Invia le uscite del comando al video.
SET FORMAT TO PRINT		Invia le uscite del comando alla stampante.
SET FORMAT TO <file.FMT>		Usa formati creati precedentemente per i comandi APPEND, EDIT, INSERT e CREATE.
SET MARGIN TO <nnn>		Fissa il margine sinistro della vostra stampante ("nnn" < = 254).
SET RAW	ON	Visualizza i records secondo i modi DISPLAY e LIST (ma senza lasciare spazi fra i diversi campi).
	OFF	Visualizza i records secondo i modi DISPLAY e LIST, con uno spazio in più fra i diversi campi.
SET HEADING TO <stringa>		Cambia il titolo nel comando REPORT.

SET ECHO	ON	Visualizza sullo schermo tutti i comandi di un file comandi via via che vengono eseguiti.
	OFF	I comandi in esecuzione non vengono visualizzati.
SET EJECT	ON	Attiva il salto pagina della stampante nel comando REPORT.
	OFF	Lo disattiva.
SET STEP	ON	Si ferma a compimento avvenuto di ogni comando. Serve per eliminare errori (debugging) nel file comandi.
	OFF	Permette l'esecuzione continua dei comandi.
SET DEBUG	ON	Invia le uscite dei comandi ECHO e STEP alla stampante soltanto.
	OFF	Invia le uscite dei comandi ECHO e STEP sul video.
SET BELL	ON	Attiva la suoneria quando il campo è stato riempito.
	OFF	Disattiva la suoneria.
SET COLON	ON	Colloca i due punti per delimitare le variabili sullo schermo.
	OFF	Non colloca i due punti.
SET CONFIRM	ON	Attende un comando < enter > prima di passare alla variabile successiva durante la composizione a schermo intero (full screen editing).
	OFF	Salta alla variabile successiva al riempimento del campo.
SET CARRY	ON	Trasporta i dati dal record precedente a un nuovo record nel comando APPEND.
	OFF	Presenta un record in bianco nel comando APPEND.

SET INTENSITY	ON	Attiva la doppia intensità per le operazioni a schermo intero.
	OFF	Disattiva la doppia intensità.
SET LINKAGE	ON	Unisce due base-dati per permettere la presentazione sullo schermo fino a 64 campi e 2.000 byte per ciascun record. Quando i nomi dei campi delle due base-dati sono simili fra loro, si devono usare i prefissi P. e S.
	OFF	Non permette di unire i file.
SET ESCAPE	ON	Permette che il tasto < escape > tronchi la esecuzione di un file comandi.
	OFF	Disattiva il tasto < escape>.

"SET ALTERNATE TO <nome del file>"

Crea un file con l'estensione .TXT per salvare tutto ciò che appare sullo schermo intero (full screen display). Per iniziare a far ciò battete sulla tastiera il comando "SET ALTERNATE ON".

Potete cambiare il file che state usando, inserendo "SET ALTERNATE TO < nuovo file >".

Per fermare l'immagazzinamento nel file, inserite "SET ALTERNATE OFF" oppure "QUIT".

L'AGGIORNAMENTO DI RECORDS DIRETTAMENTE DA UNA BASE-DATI AD UN'ALTRA (UPDATE)

I dati si possono trasportare da un file all'altro tramite il seguente comando:

UPDATE FROM <nome del file> ON <chiave > [ADD <lista di campi>] [REPLACE < lista di campi> [RANDOM] [REPLACE < campo> WITH <campo >]

Nota: Ambedue le base-dati devono essere già state ordinate o indicizzate (index) sul campo-chiave, a meno che non si usi l'opzione RANDOM (a caso).

Senza il comando RANDOM, i due file vengono posizionati all'inizio e poi si effettua il confronto dei campi-chiave. Se questi sono identici, i dati del file FROM sono aggiunti numericamente ai dati del file USE, oppure rimpiazzano i dati del file in USE in quei campi che siano stati specificati nel comando. Quando i campi "chiave" non coincidono, i record relativi vengono saltati. Questo comando è adatto per esempio all'aggiornamento di inventari.

Se i campi dei due file hanno gli stessi nomi, basta dare la lista dei campi nei quali vogliamo cambiare i dati. Se invece i file hanno nomi di campi diversi, bisogna usare la terza forma dell'opzione REPLACE per specificare in quale campo del file in USE si deve collocare un determinato campo del file FROM.

Se si utilizza l'opzione RANDOM, la base-dati che si deve aggiornare dev'essere indicizzata sul campo-chiave, ma i record del file da cui si prelevano i dati (FROM <file>) possono trovarsi in qualsiasi ordine. Quando ciascun record del file FROM viene letto, un comando FIND localizza immediatamente il record giusto nel file da aggiornare.

LA CONGIUNZIONE DI FILE

Il comando JOIN è fra i più potenti del dBASE II. Esso può, infatti, combinare due base-dati, poste nelle due aree PRIMARY e SECONDARY, per crearne una terza. Ecco la forma del comando:

JOIN TO <file nuovo> ON < espressione> [FIELD <lista >]

Il comando opera posizionando il dBASE II sul primo record del file primario (PRIMARY) e considera uno per uno i record del file secondario (SECONDARY). Ogni volta che l'"espressione" produce un risultato logico positivo (True, Vero), il record relativo viene aggiunto al "file nuovo". Se quando lanciate il comando JOIN siete nell'area primaria, usate il prefisso S. davanti ai nomi delle variabili dell'area secondaria. Se, al contrario, vi trovate in quest'ultima, procedete allo stesso modo usando però il prefisso P. (Vedi l'esempio più sotto).

Quando ciascun record del file secondario in USE è stato confrontato con il primo record del file primario (in USE), il dBASE II avanza al secondo record del file primario e torna a confrontare con questo tutti i record del file secondario. Ciò si ripete finché tutti i record dei due file sono stati confrontati fra loro.

Nota: Questa operazione può essere molto lunga quando i file contengono molti records, e rischia di non potersi completare se si perdono di vista alcune delle condizioni al contorno. Per esempio, due file che abbiano 1.000 record ciascuno creerebbero un terzo file di 1.000.000 di records in caso che l'espressione contenuta nel comando JOIN risultasse sempre vera; bisogna ricordare che dBASE II limita il numero di record per ogni base-dati a 65.535.

Per utilizzare questo comando, applicate la sequenza di istruzioni:

```
USE Invent
SELECT SECONDARY
USE Ordini
JOIN TO Nuovo FOR P.Ricamb:Num = Ricamb:Num;
        FIELD Cliente,Articolo,Importo,Costo
```

Si crea in questo modo una nuova base-dati chiamata <Nuovo.DBF> con quattro campi: Cliente, Articolo, Importo e Costo. La struttura di questi campi (tipo di dato, dimensioni) sarà la stessa delle due basi-dati congiunte. Notate nell'esempio che il prefisso "P." serve a chiamare una variabile dell'area che non è in USE).

CONFIGURAZIONE A SCHERMO INTERO

Per evidenziare o per stampare un lungo testo, questo può essere compreso fra le due direttive:

TEXT/ENDTEXT

Un testo qualsiasi, introdotto in questo modo, verrà inviato direttamente sullo schermo o alla stampante senza ulteriori elaborazioni. Fanno eccezione le righe che continuano tramite il punto

e virgola. Il programma riprende dopo il comando: ENDTEXT.

Per un preciso controllo del formato, il dBASE II possiede una serie di comandi potenti che vi consentono di collocare l'informazione esattamente dove volete. Lo avete già visto nel nostro esempio di programma <Campione.CMD>, dove abbiamo usato:

<coordinate > SAY ['messaggio'] GET <variabile>

Con questo comando abbiamo potuto posizionare messaggi e variabili (insieme al loro valore) in qualsiasi punto specifico dello schermo. Quando lo si inserisce in una serie di comandi seguito da READ, si può controllare il formato (format) di tutto lo schermo. Come ripasso create e fate eseguire il seguente file comandi:

```
STORE "                    " TO MData
STORE "                    " TO MBilancio
STORE "                    " TO MEstConto
   5,5 SAY "Inserire la data GG/MM/AA/" GET MData
  10,5 SAY "Qual'è il bilancio?" GET MBilancio
  15,5 SAY "Quanto è l'estratto conto?" GET MEstConto
READ
ERASE
   5,5 SAY "Dobbiamo fare un confronto?" GET MConfr
READ
```

Il comando si può usare anche senza il SAY se inseriamo <coordinate> GET <variabile > seguito da READ nel programma. Con ciò avremo soltanto la visualizzazione dei due punti che delimitano la lunghezza del campo per la variabile.

Suggerimento: Nel modo SCREEN i numeri delle righe possono anche non essere in ordine, ma è una buona norma scriverle in ordine fin d'ora, dato che più tardi sarà indispensabile per l'operazione di stampa (FORMAT TO PRINT).

La sintassi completa del comando è:

<coordinate > SAY [espressione] GET <variabile > [PICTURE] <formato >

L'opzione PICTURE si riempie usando i simboli che elenchiamo qui sotto. Se ora inseriamo:

```
5,1 SAY "La data di oggi è" GET Data PICTURE '99/99/99'
```

ci verrà presentato:

```
La data di oggi è:    /  /   :
```

tenendo conto che la variabile Data è in bianco. In questo esempio si possono introdurre solamente cifre.

I simboli associati al comando GET sono:

```
9 oppure #  : accetta solo l'entrata di cifre
A           : accetta solo l'entrata di caratteri alfabetici
!           : converte tutti i caratteri introdotti in maiuscole
X           : accetta qualsiasi carattere
$           : presenta '$' sullo schermo
*           : presenta '*' sullo schermo
```

Con questo comando, potete creare le maschere necessarie per i vostri menù e le vostre tabelle ogni volta che volete, velocemente e con facilità.

Se usate soltanto ...SAY...GET e i commenti (preceduti da un asterisco), ciò che state elaborando può venir salvato come file formato con l'estensione .FMT. Questi file formato rappresentano una visualizzazione personalizzata della vostra base-dati sostituendosi al comando di default del dBASE II, tramite cui ottenete solamente una lista dei campi della base-dati. Il file formato .FMT può comprendere istruzioni speciali e qualsiasi altro aiuto all'utente perché introduca correttamente i dati e solamente i nomi dei campi sui quali si vuole lavorare.

Per usare un file formato .FMT battete:

```
SET FORMAT TO <nome del file>
```

Così facendo, al momento di usare APPEND, EDIT o INSERT il formato del file indicato vi sarà presentato sullo schermo.

LA CONFIGURAZIONE DI UNA PAGINA DI STAMPA

Quando usate il comando SET FORMAT TO PRINT, il comando invia l'informazione alla stampante invece che allo schermo.

Le opzioni GET e PICTURE vengono ignorate e il comando READ non si deve usare.

I dati che vogliamo stampare su assegni, ordini d'acquisto, fatture ed altri moduli standard possono essere prima organizzati sullo schermo perché siano poi stampati esattamente come vengono rappresentati. Ecco il comando:

coordinate SAY [variabile/espressione/'stringa'] [USING formato]

Per stampare, le coordinate devono essere in ordine. Le righe si devono trovare in ordine crescente (ossia bisogna stampare la riga 7 prima della 9, ecc.). Inoltre, su ciascuna linea anche le colonne devono essere in ordine (stampare la colonna 15 prima della colonna 63, ecc.).

Come nel modo SCREEN, il comando SAY si può usare per stampare il valore di una variabile di memoria, il risultato di un'espressione oppure un messaggio o la richiesta di stringa letterale.

L'opzione USING serve a specificare sia il tipo di caratteri da stampare sia il punto in cui devono apparire sulla pagina. I simboli da usare sono:

9 o #	stampa soltanto cifre
A	stampa soltanto caratteri alfabetici
X	stampa qualsiasi carattere stampabile
$	stampa una cifra o "$" al posto di uno zero iniziale
*	stampa una cifra o "*" al posto di uno zero iniziale

Per esempio, il comando 10,50 SAY Ore*Salario USING '$$$$$$.99' si può usare sia per lo schermo sia per la stampante, perché non contiene l'opzione GET. Se Ore = 8 e salario = 1.273, verrà stampato o rappresentato $$$$10.184, molto utile per gli assegni che divengono così difficilmente alterabili.

PREPARAZIONE E STAMPA DI UN MODULO

Per preparare una maschera di stampa, le misure devono basarsi sul tipo di spaziatura della stampante (la nostra stampa, orizzontalmente, 10 caratteri per pollice, e verticalmente 6 righe per pollice).

Al "Menù Uscite Cassa" che abbiamo usato nel nostro file "Campione.CMD" (Sezione III) possiamo aggiungere un'altra chiamata "4 = Fare assegni" Supponiamo che stiamo sviluppando il programma relativo a questa chiamata.

Per cominciare, dobbiamo introdurre la data. Le seguenti linee di comando accettano la data in una variabile chiamata MData e controllano se è (probabilmente) giusta:

```
ERASE
SET TALK OFF
STORE "            " TO MData
STORE T TO NoData
DO WHILE NoData
     5,5 SAY "Introdurre la data GG/MM/AA" GET MData PICTURE "99/99/99"
   READ
   IF VAL ($(MData,4,2))   < 1;
      .OR.VAL ($(MData,4,2))   > 12;
      .OR.VAL ($(MData,1,2))   < 1;
      .OR.VAL ($(MData,1,2))   > 31;
      .OR.VAL ($(MData,7,2)) < > 81
      STORE "                " TO MData
        7,5 SAY "****DATA SBAGLIATA,RIPETERE.****"
      STORE T TO NoData
    ELSE
      STORE F TO NoData
    ENDIF
ENDDO   *   perché ora abbiamo una data valida
ERASE
```

Quseto programma prima fissa il valore di MData in 8 spazi in bianco, poi, tramite il comando ...SAY visualizza:

```
Introdurre la data GG/MM/AA:   /   /   :
```

Quando la data è stata introdotta, viene confrontata dall'IF per vedere se il mese si trova fra 1 e 12, il giorno fra 1 e 31 e l'anno è = 81. Questa operazione viene effettuata in tre fasi:

:: La funzione di sottostringa $ prende i due caratteri che rappresentano il giorno, il mese e l'anno (p.es., per il primo comincia sulla prima posizione e prende due caratteri).

:: La funzione VAL converte questi due caratteri in un numero intero.

:: Questo numero intero viene confrontato con i valori limiti.

Se il valore esce dai limiti dati, MData viene ridefinita con 8 spazi in bianco e viene indicato un errore. Quando invece la data si trova entro i limiti permessi, il programma continua.

La parte seguente del programma consisterà nel comando per effettuare la stampa. Nell'esempio usiamo le dimensioni dei nostri assegni:

```
8,3 SAY Scritto * Una variabile di caratteri che stampa l'im
                * porto in scrittura e viene riempita un'al
                * tra procedura chiamata Chng2Scrpt.
                * Per ora, possiamo eseguire la nostra prova
                * così:
                * STORE 'ProvaScrit' TO Scritto
                * RETURN
11,08 SAY Vend_Num
11,50 SAY MData .
11,65 SAY Importo
13,10 SAY Venditore
14,10 SAY Indirizzo
15,10 SAY Cap
15,35 Say Citta_Pr
17,10 SAY Chi
```

Prima di stampare gli assegni potete controllarli sullo schermo con l'opzione SCREEN; il comando SET FORMAT TO PRINT vi riporterà alla stampa. I valori per le variabili provengono da altre istruzioni del vostro file comandi.

Moduli più lunghi non costituiscono un problema, dato che una pagina di stampante può avere fino a 255 righe. Per azzerare il contatore delle linee, inserite 'EJECT' con la stampante già selezionata.

RACCOMANDAZIONI RIASSUNTIVE

Essendo il dBASE II un sistema molto potente, esso possiede un gran numero di comandi e di accorgimenti per rispondere alle vostre esigenze e per darvi modo di ottenere le informazioni più facilmente di qualsiasi altro pacchetto equivalente che giri su microcalcolatori.

La maniera più facile per imparare queste tecniche è quella di osservare come si usano in un programma. Abbiamo cercato di mantenere gli stessi nomi dei campi e le stesse strutture individuali per permettere la combinazione tra file. I dati di una base-dati trovano posto nei campi corrispondenti di un'altra base-dati e il loro trasporto dall'una all'altra risulta semplice e diretto.

Potete anche provare i file comandi nei programmi posti nel dischetto di esempi. La maggior parte di essi sono stati già usati e i file funzionano secondo il modo in cui sono stati preparati. Ciascun programma è stato pensato perché avesse una utilità ed è stato pensato in modo che possiate comprenderne il significato. Il primo file comandi è abitualmente il menù del sistema e i sotto-file (o file secondari) possono essere selezionati premendo il tasto numerico corrispondente.

Per scrivere questi file comandi abbiamo usato esattamente il metodo che vi abbiamo raccomandato: prima definire il senso generale del problema; scendere gradualmente al livello dei dettagli; usare al principio il linguaggio naturale, poi lo pseudocodice e arrivare finalmente ai codici operativi del dBASE II.

Quando siamo arrivati a certe operazioni che bisognava fare ma non sapevamo con sicurezza come, le abbiamo semplicemente chiamate con il nome di una procedura, per tornare su di esse più tardi.

Gli annidamenti delle linee e la combinazione di lettere maiuscole e minuscole non sono stati scelti solamente per questo manuale ma costituiscono il nostro normale modo di lavorare. Il loro scopo è quello di evidenziare costantemente i diversi raggruppamenti che corrispondono alle strutture che vengono usate.

Abbiamo anche inserito commenti in tutti i files, sebbene molte volte i programmi si commentino da sè, data la quantità di comandi del dBASE che corrispondono al loro significato letterale inglese.

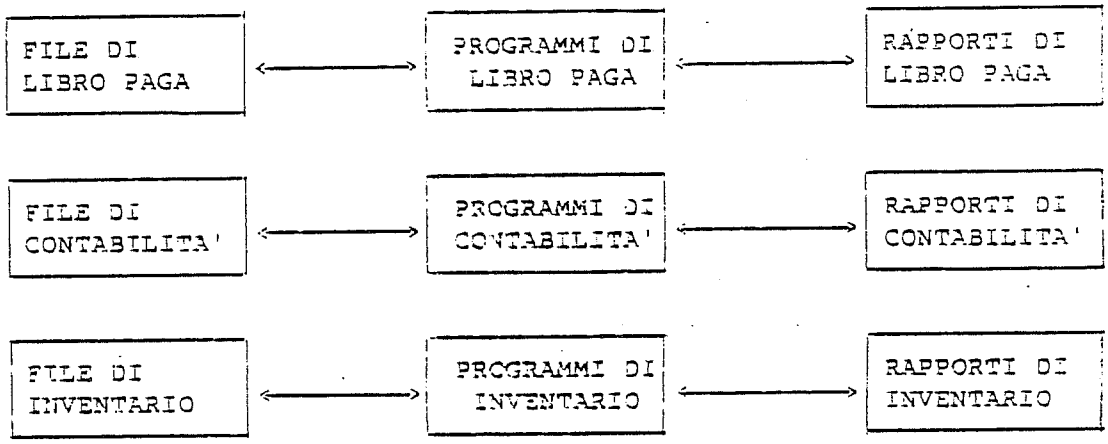
Vi invitiamo a sperimentare a vostro agio questi programmi, utilizzando le regole e le tecniche che abbiamo già discusso.

S E Z I O N E V

PRINCIPI GENERALI DI UNA BASE-DATI

Un sistema gestionale di base-dati (DBMS) come il dBASE II si differenzia in modo considerevole da un sistema di gestione di file in generale.

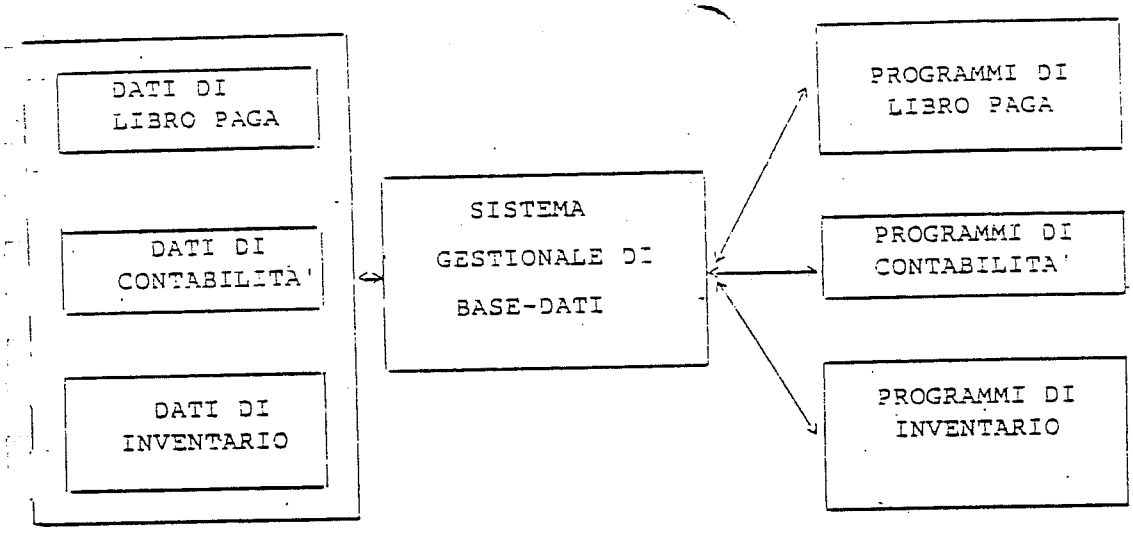
Un sistema di gestione di file presenta abitualmente il seguente schema:



I programmi di libro paga elaborano i file di libro paga, quelli di contabilità i file di contabilità e quelli d'inventario i file d'inventario. Per ottenere rapporti che siano il risultato della combinazione di file diversi, bisognerebbe scrivere un nuovo programma che, d'altra parte, non necessariamente funzionerebbe: i dati potrebbero risultare incompatibili da file a file o si troverebbero talmente "nascosti" che estrarli presenterebbe più complicazioni che vantaggi.

Un sistema gestionale di base-dati integra i dati fra loro e rende molto più facile il recupero della informazione utile direttamente dal record invece di fornire delle risme di dati.

Concettualmente, un DBMS si presenta in questo modo:



(base-dati)

I dati vengono controllati e maneggiati dal DBSM e non dai programmi applicativi individuali. Tutti i programmi applicativi hanno accesso a tutti i dati. Nel sistema di gestione di file, ciò richiederebbe una quantità enorme di dati duplicati. Oltre alla probabilità di errori di entrata, l'integrità stessa dei dati è estremamente difficile da mantenere quando gli stessi dati sono stati duplicati in differenti file: in effetti, è praticamente impossibile.

Per generare un nuovo sistema di elaborazione all'interno di un sistema del genere, si devono preparare un nuovo programma e dei nuovi file. Con il DBMS si scrive un nuovo programma d'accesso ma i dati non hanno bisogno di essere ristrutturati: di ciò si occupa il DBMS.

Se si aggiunge un nuovo tipo di dati a un record (per esempio l'andamento del salario in un file di personale), i programmi di un sistema di gestione di file devono essere modificati. Con il DBMS, le aggiunte e i cambi non hanno alcun effetto sui programmi, che non hanno bisogno di questa nuova informazione: essi non la vedono e non sanno nemmeno che esiste.

Il sistema gestionale di base-dati può essere del tipo gerarchico oppure relazionale: questi termini definiscono il rapporto tra i dati.

Il sistema gerarchico tende a farsi estremamente complesso e difficile da mantenere aggiornato, perché le relazioni fra i vari elementi passano attraverso preparazioni, collegamenti di liste e puntatori che chiedono in che direzione devono procedere. Molto presto si può arrivare ad avere liste di liste di liste e puntatori a puntatori a puntatori.

Un sistema di gestione relazionale di base-dati come il dBASE II è veramente molto semplice. I dati vengono rappresentati come sono e la relazione fra i loro elementi si può considerare come una matrice a due dimensioni:

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
Fattura Numero	Fornitore	Descrizione	Importo	Numero
2386	La Grafica SRL	Stampa	230.000	BBQ-747
78622	Incisioni Rossi	Litografia	97.400	TFS-90L
M1883	Alvolo SPA	Spedizione	9.700	SPT-233

Possiamo chiamare ogni riga record e ogni colonna campo del record. Ogni entrata di questa tavola deve avere un valore individuale (non una serie o un gruppo). Tutte le entrate di una colonna devono essere dello stesso tipo. Ogni record sarà unico e l'ordine dei record non avrà importanza.

Quando faremo, più avanti, degli esempi più realistici, vedrete che i record potranno essere più grandi ma non più complicati.

BREVE INTRODUZIONE ALL'ORGANIZZAZIONE DI UNA BASE-DATI

Appena avete preparato la vostra base-dati vorrete accedere ai dati in maniera ordinata.

In alcune base-dati, l'ordine in cui avete introdotto i dati è lo

stesso di quello in cui volete estrarre l'informazione. Ma nella maggior parte dei casi vi sarà necessario organizzarla in un altro modo.

Per organizzare i dati con il dBASE II si utilizzano i comandi SORT e INDEX (descritti in dettaglio nella Sezione II).

Il comando SORT riorganizza i record in modo che la base-dati venga ordinata in ordine ascendente o discendente in base al campo che voi specificate (nome, codice postale, ecc.). Questo campo si chiama chiave.

Uno svantaggio del SORT è che probabilmente vorrete accedere alla base-dati su un determinato campo per una certa applicazione e su un altro campo per un'applicazione differente. Un secondo svantaggio si avverte con l'aggiunta di nuovi records; in questo caso è necessario richiedere un ordinamento (SORT) per ogni nuovo dato introdotto affinché l'ordine sia mantenuto.

Anche la ricerca dei dati è relativamente lenta, poiché si deve effettuare in maniera sequenziale.

L'uso del comando INDEX è una maniera brillante di aggirare questi ostacoli.

La indicizzazione (indexing) è il modo di preparare un file indice usando solamente le chiavi che vi interessano, invece di usare tutta la base-dati. La chiave è un campo della base-dati (o una combinazione di campi) che costituisce il "contenuto" del record. In un sistema d'inventario, questo contenuto può essere il codice, e l'importo, il costo e la collocazione sarebbero campi descrittivi. In una base-dati di personale, il nome o il codice dell'impiegato sarebbero probabilmente le chiavi migliori.

In una base-dati indicizzata solamente le chiavi vengono organizzate, insieme ai puntatori del record a cui appartengono. Il sistema è simile a quello di un albero binario, ma può utilizzare l'immagazzinamento in modo molto più efficiente e molto più veloce. Il comando FIND (descritto nella Sezione II) impiega abitualmente due secondi a ricercare un dato in una base-dati, sia media che grande.

Se, d'altra parte, voi avete bisogno di organizzare la vostra base-dati per applicazioni diverse, potete preparare vari file indi

ce (uno per ogni campo) e usare il file indice più adatto a secondo del momento. Potete avere file indice ordinati per il nome del fornitore, per numero di cliente, per codice postale e qualsiasi altra chiave, tutti riferiti alla stessa base-dati.

Con questo sistema, le nuove entrate vengono aggiunte automaticamente al file indice in uso.

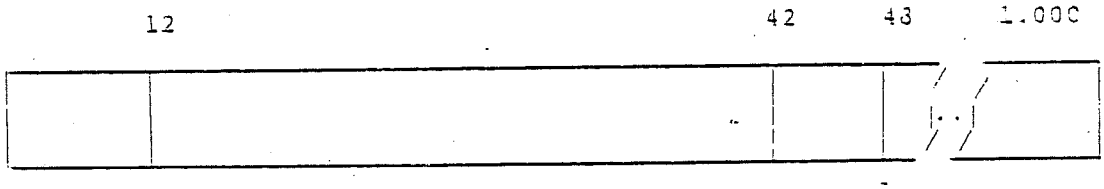
Un altro vantaggio delle base-dati indicizzate (indexed) è la rapida localizzazione dei dati che vi interessano.

RECORD, FILE E TIPI DI DATI NEL dBASE II

Il dBASE II è stato progettato per funzionare sul vostro microcalcolatore, per cui le sue capacità non sono infinite, ma vi sarà duro scoprire il limite massimo delle sue prestazioni.

Il dBASE II vi limita a 65.535 record per file, ma considerando la memoria del sistema e la "memoria di massa" di un micro, si può concludere che non esiste limitazione alcuna.

Un record del dBASE II può avere fino a 32 campi e una lunghezza di 1.000 caratteri (secondo la condizione che viene prima raggiunta):



Immaginate questo record come una striscia lunga 1.000 caratteri che voi potete segmentare a modo vostro in tutta la sua lunghezza, oppure accorciare se non avete bisogno di utilizzarla tutta. Potreste avere quattro campi che utilizzano tutti i caratteri (ricordando che ogni campo dispone di un massimo di 254 caratteri), oppure un record con un solo campo di un solo carattere o, naturalmente, qualsiasi record che si trovi fra questi due estremi.

Nell'esempio dato in precedenza, ogni record presentava cinque campi e la sua lunghezza totale era di 58 caratteri:

Fattura Numero	Fornitore	Descrizione	Importo	Servizio Numero
1 9	10 28	29 43	44 51	52 58

Tipi di dato

Come abbiamo già detto, ogni campo può contenere un solo tipo di dato, che nel dBASE II vuol dire:

* Carattere

  Tutti i caratteri stampabili ASCII, compresi i numeri interi, i simboli e gli spazi.

* Numerico

  Numeri positivi e negativi, grandi fino a 1,8 x 10< 63 > o piccoli fino a 1,0 x 10 <-63 >. La precisione arriva fino a dieci cifre.

* Logico

  Sono valori Vero/falso (Si/No), che occupano un campo di un solo carattere. Il dBASE II riconosce T, t, Y e y come Vero (True, Yes), e F, f, N e n come Falso (False, No).

I nomi dei campi

Ogni campo ha un nome, affinché il dBASE II lo possa riconoscere quando voi lo state cercando. I nomi dei campi possono arrivare fino a 10 caratteri (senza spazi); devono cominciare per una lettera e possono comprendere cifre e i due punti intercalati).

```
A             (valido)
A123456789    (valido)
Serv:Num      (valido)
A123,B456     (virgola: illegale)
Cliente:      (due punti non intercalati: illegali)
```

Suggerimento: Usate sempre un numero di caratteri sufficiente a rendere comprensibile il nome del campo. 'Serv:Num' è molto meglio di 'No.' e ancora di più di 'S'. L'utilizzo di un massimo di nove caratteri renderà l'impiego delle variabili di memoria molto più facile (come vedremo più avanti).

Quando vi metterete a preparare file comandi, troverete utile usare le lettere maiuscole per i comandi del dBASE II e le lettere minuscole e maiuscole per quelle che denotano i campi, le variabili e gli altri elementi che controllate voi. Apprezzerete questo metodo quando dovrete tornare su un file comandi per effettuare delle modifiche.

Tipi di file del dBASE II

I nomi dei file sono limitati a 8 caratteri più 3 dopo un punto per l'estensione. Potete usare i due punti in mezzo al nome, ma in questo caso potrete gestire i file solo tramite il dBASE II : il CP/M memorizza e richiama bene i nomi ma poi non li riconosce quando gli chiedete di eseguire una funzione come PIP. Un nome di dieci caratteri non costituisce un problema: semplicemente, il CP/M lo tronca riducendolo a otto. Se usate lettere maiuscole e minuscole per i nomi dei file, il CP/M le trasforma tutte in maiuscole, tuttavia sarà sempre meglio mantenerle miste nei vostri file comandi.

Un file di dBASE II è una semplice collezione di informazioni dello stesso tipo sotto un solo nome, qualcosa di simile a una gigantesca cartella d'archivio. Il dBASE opera con i sei differenti tipi di file che descriviamo qui di seguito:

.DBF File di base-dati:

Sono quelli in cui vengono conservati i vostri dati. Quando create (CREATE) un nuovo file, il dBASE II si incarica di contrassegnarlo con l'estensione. Ogni file .DBF può immagaz

zinare fino a 65.535 record. Non usate un sistema di video-scrittura su questi file.

.FRM File di configurazione di rapporti:

Questi file vengono creati automaticamente dal dBASE II quando voi eseguite il comando REPORT. Essi contengono titoli, totali, contenuti di colonne, ecc. Possono essere modificati mediante sistemi di video scrittura, ma noi vi raccomandiamo di non farlo: effettuate le vostre variazioni usando il dBASE II.

.CMD File di comandi (con la versione a 16 bit: .PRG):

Sono file che contengono una sequenza di istruzioni o comandi del dBASE II per eseguire funzioni usate con frequenza; possono essere complessi come potrebbe esserlo un sistema completo di contabilità. Questi file si creano mediante un sistema di video scrittura o con il comando MODIFY.

.NDX File indice:

Vengono creati automaticamente dal comando INDEX, e provvedono a una localizzazione molto rapida delle informazioni in base-dati assai grandi.

.MEM File di memoria:

Vengono creati automaticamente quando salvate con il comando SAVE il risultato di operazioni, costanti o variabili che volete recuperare più tardi. In essi potete salvare fino a 64 elementi, ciascuno dei quali può essere lungo fino a 254 caratteri, che potete poi recuperare con il comando RESTORE.

.TXT File di uscita di testi:

Questo tipo di file viene creato quando usate il comando SET ALTERNATE per immagazzinare sul disco tutto ciò che va al CRT (lo schermo del vostro terminale). E' una caratteristica che si usa per seguire cronologicamente ciò che avviene nel sistema. L'informazione così prodotta può essere poi "editata", ossia diversamente composta e organizzata, e anche

stampata o salvata. Il file .TXT viene creato anche quando si usa il comando COPY..SDF.

.FMT File di formato:

Sono file che possono contenere soltanto le istruzioni ... SAY e i commenti preceduti da un asterisco. Si usano per formattare lo schermo con i comandi APPEND, INSERT e EDIT.

SOMMARIO DEGLI OPERATORI DEL dBASE II

Operatori aritmetici (che generano risultati aritmetici)

```
( )     : parentesi per raggruppare
*       : moltiplicazione
/       : divisione
+       : addizione
-       : sottrazione
```

Operatori relazionali (che generano risultati logici)

```
<       : minore di
>       : maggiore di
=       : uguale a
# oppure < >  :  diverso da
< =     : minore o uguale a
> =     : maggiore o uguale a
```

Operatori logici (che generano risultati logici T/F)

```
( )     : parentesi per raggruppare
.NOT.   : inversione di Boole (operatore unitario)
.AND.   : congiunzione di Boole
.OR.    : disgiunzione di Boole
$       : operatore logico di sottostringa
          (trova la stringa 1 nella stringa 2)
```

Operatori di stringa (che generano risultati di stringa)

```
+       : concatenazione di stringhe
-       : concatenazione di stringhe con eliminazione di spazi
```

SOMMARIO DELLE FUNZIONI DEL dBASE II

```
#       : numero del record
*       : eliminazione del record
EOF     : termine del file
! (variabile/stringa): converte le maiuscole in minuscole
TYPE (<espressione>) : tipo di dato
INT (<variabile/espressione>): numero intero
VAL (<variabile/stringa/sottostringa>): stringa a intero
STR (<espressione/variabile/numero>,<lunghezza>,<decimali>):
      intero a stringa
LEN ( variabile/stringa ): lunghezza di stringa
$ (<espressione/variabile/stringa>,<inizio>,<lunghezza>):
    sottostringa
@ (<variabile1/stringa1>),(<variabile2/stringa2>):
    ricerca di sottostringa
CHR (<numero>): equivalente alfanumerico ASCII
&       : macrosostituzione
FILE ( <"nome del file"/variabile/espressione>):
         esiste questo file?
TRIM (<stringa>):  eliminazione di spazi
RANK (<stringa>):  valore ASCII del primo carattere
```

SOMMARIO DEI COMANDI DEL dBASE II

In questo sommario si useranno le seguenti abbreviazioni:

```
<esp>    = espressione
<var>    = variabile
<str>    = stringa
<coord> = coordinate
```

I simboli < ... > racchiudono elementi che devono essere specificati dall'utente. Le parentesi quadre [...] racchiudono le opzioni. Nel caso in cui queste abbiano a loro volta altre opzioni, vengono annidate.

?<esp [,lista]>

visualizza un'espressione o varie espressioni, separate da virgole.

<coord>[SAY <esp> [USING 'formato']] [GET <var> [PICTURE 'formato']]

Compone un formato (schema) per lo schermo o per la stampante.

ACCEPT ['richiamo'] TO < var >

Accetta l'entrata di una stringa di caratteri dal terminale senza virgolette.

APPEND [BLANK]
APPEND FROM < nome del file > [SDF] [FOR <esp>]
[DELIMITED] [FOR <esp>]

Aggiunge records alla base-dati.

CANCEL

Interrompe l'esecuzione di un file comandi.

CHANGE [scopo] FIELD <lista > [FOR < esp>]

Effettua variazione sui campi di una base-dati.

CLEAR

Riinizializza i file del dBASE e le variabili di memoria utilizzate.

CONTINUE

Continua nella ricerca con il comando LOCATE

COPY [scopo] TO < nome del file > { SDF } { STRUCTURE } [FIELD < lista >] [FOR < esp >] { DELIMITED [WITH < delimitatore >] }

Copia dati da una base-dati a un altro file.

COPY TO <nome del file> STRUCTURE EXTENDED

Crea un nuovo file .DBF i cui records mostrano la struttura del file vecchio (Vedi anche CREATE< file nuovo > FROM < file vecchio>).

COUNT [scopo] [FOR < esp >][TO < var >]

Conta i record che soddisfano una determinata condizione.

CREATE [<nome del file >]

Definisce una nuova base-dati.

CREATE <nuovo file > FROM <vecchio file >

Crea un < nuovo file > con la struttura determinata dai dati contenuti nei record del < vecchio file > (Vedi anche COPY STRUCTURE EXTENDED).

DELETE [scopo] [FOR <esp>]

Evidenzia i records specificati per un'eventuale cancellazione.

DELETE FILE < nome del file >

Cancella un file dal sistema.

DISPLAY <scopo> [FOR < esp>][OFF]

Visualizza i dati in base a una richiesta.

DISPLAY [scopo] [campo [,lista]]

Presenta soltanto il campo o i campi selezionati.

DISPLAY STRUCTURE

Presenta la struttura della base-dati in USE.

DISPLAY MEMORY

Presenta il contenuto delle variabili di memoria.

DISPLAY FILES [ON disco]

Elenca il contenuto di un disco.

DISPLAY STATUS

Presenta i file attivi aperti, i file indice, le "chiavi" e tutti i parametri "SET".

DO < nome del file >

Esegue un file comando.

DO WHILE < esp >

Ripete l'esecuzione di un gruppo di comandi.

EDIT

Modifica i dati i una base-dati.

EDIT [numero]

Presenta il record indicato per consentirne la modifica.

EJECT

Aziona un salto di pagina della stampante.

ELSE

Presenta l'esecuzione alternativa di un comando IF.

ENDDO

Termina il comando DO WHILE.

ENDIF

Termina il comando IF.

ENDTEXT
Termina il comando TEXT.

ERASE
Cancella lo schermo

FIND < chiave >
Localizza un record in una base-dati indicizzata (indexed) basandosi sul valore di una chiave (non si richiedono virgolette per le chiavi alfanumeriche).

GO oppure GOTO [RECORD], oppure [TOP], oppure [BOTTOM], n
Posiziona in un determinato punto della base-dati.

HELP [<verbo di comando >]
Presenta una breve spiegazione di un comando del dBASE II.

IF < esp >
Comando di esecuzione condizionale.

INDEX ON <chiave > TO < nome del file >
Crea un file indice per la base-dati in USE.

INPUT ['richiesta'] TO < var >
Accetta le entrate dell'utente nelle variabili di memoria. La stringa della richiesta dell'utente è opzionale.

INSERT [BEFORE] oppure [BLANK]
Inserisce un nuovo record tra gli altri records di una base-dati.

JOIN TO < nome del file> FOR <esp> [FIELD <lista>]
Crea una base-dati composta dai records coincidenti di altre due base-dati.

LIST
Presenta tutti i records.

LOCATE [scopo] [FOR <esp>]
Localizza il record che corrisponde a una condizione data.

LOOP
Operazione che termina il ciclo stabilito dal comando DO WHILE.

NOTE (oppure) *
Commento a un file comandi. Non viene visualizzato durante l'esecuzione del programma corrispondente.

MODIFY COMMAND <nome del file>
Permette la modifica di un file direttamente dal dBASE II.

MODIFY STRUCTURE
Modifica la struttura di una base-dati. Distrugge tutti i dati contenuti nella base-dati.

PACK
Elimina i records precedentemente marcati dal comando DELETE.

QUIT [TO lista dei comandi CP/M oppure dei file .COM]
Abbandona il dBASE ed esegue una serie di programmi o comandi, i quali devono venir specificati tra virgolette e separati da virgole.

READ
Attiva la funzione a schermo intero di uno schermo formattato. Accetta dati sotto le indicazioni del comando GET.

RECALL [scopo] [FOR <esp>]
Toglie il segno di cancellazione (*) ai records sottoposti al comando DELETE.

RELEASE [<var>[,lista]] oppure [ALL]
Elimina le variabili di memoria che non vengono più utilizzate.

REMARK
Commento che viene visualizzato durante l'esecuzione del programma.

RENAME <file vecchio> TO <file nuovo>
Assegna un nuovo nome a un file.

REPLACE [scopo] <campo> WITH <esp>[,<campo> WITH <esp>...]
[FOR <esp>]
Cambia i dati di una base-dati. Assicuratevi di avere una copia su disco perché il dBASE II farà precisamente quello che gli direte di fare, anche se non corrisponde sempre a ciò che realmente volete.

REPORT [scopo] [FORM <nome del file>] [TO PRINT] [FOR<esp>]
Genera un rapporto o compendio di informazioni.

RESET
Comunica al CP/M che può aver luogo un cambio di disco.

RESTORE FROM <nome del file>
Ripristina le variabili di memoria salvate con il comando SAVE e distrugge le variabili attuali.

RETURN
Termina un file comandi e torna al file che lo aveva richiamato con il comando DO.

SAVE TO <nome del file>
Scrive le variabili di memoria in un file per un uso successivo.

SELECT [PRIMARY] oppure [SECONDARY]
Seleziona le due aree di lavoro.

SET parametro [ON], [OFF], [TO <condizione, nome del file>]
Riconfigura le operazioni del dBASE II.

SKIP [-] <esp/numero>
Muove il puntatore avanti e indietro nella base-dati.

SORT ON <chiave> TO <nome del file> [ASCENDING] [DESCENDING]
Genera una base-dati indicizzata in base a un campo.

STORE < esp > TO < var >
Colloca un valore in una variabile di memoria.

SUM [scopo] < campo [,lista]> TO < var [,lista]> [FOR< esp >]
Totalizza i campi di una base-dati.

TEXT
Visualizza un blocco di testo non formattato fino a trovare il comando ENDTEXT che lo termina.

TOTAL TO < nome del file > ON < chiave> [FIELDS < campo [,lista]>]
Genera una base-dati con sottototali per records.

UPDATE FROM < nome del file> ON <chiave> [ADD < campo [,lista]>]
[REPLACE < campo [,lista]>]
Modifica una base-dati con i dati di un'altra.

USE <nome del file> [INDEX< nome del file>]
Apre una base-dati.

USE
Chiude tutte le base-dati aperte precedentemente.

WAIT [TO <var>]
Il programma sospende l'esecuzione, in attesa che venga battutto un carattere.

RAGGRUPPAMENTO FUNZIONALE DEI COMANDI DEL dBASE II

--- PER STRUTTURARE I FILE

CREATE
Definisce una struttura completamente nuova.

CREATE < file nuovo > FROM < file vecchio >
Crea un nuovo file la cui struttura è descritta nei record di un file già esistente.

USE <file vecchio>
COPY TO <file nuovo> STRUCTURE EXTENDED
Crea un nuovo file che contiene una struttura uguale a quella del file in USE.

CREATE <file nuovo> FROM <file vecchio>
Crea un nuovo file la cui struttura viene definita dai record di un file esistente.

DISPLAY STRUCTURE oppure LIST STRUCTURE
Presentano la struttura del file in USE.

MODIFY STRUCTURE
Cambia il nome dei campi, le dimensioni e qualsiasi altra caratteristica di una struttura, ma distrugge il contenuto della base-dati.

--- PER CAMBIARE UNA STRUTTURA CONSERVANDO I DATI

```
USE <file vecchio>
COPY TO <file nuovo>
USE <file nuovo>
MODIFY STRUCTURE
APPEND FROM <file vecchio>
COPY TO <file vecchio>
USE <file vecchio>
DELETE FILE <file nuovo>
```

--- PER CAMBIARE IL NOME DEI CAMPI CONSERVANDO I DATI

```
USE <file vecchio>
COPY TO <file nuovo> SDF
MODIFY STRUCTURE
APPEND FROM <file nuovo>. TXT SDF
DELETE <file nuovo>
```

--- PER OPERARE CON I FILE

USE <nome del file>
Apre un file

USE <file nuovo>
Chiude il file vecchio e apre il nuovo.

USE
Chiude tutti i file.

RENAME <nome esistente> TO <nome nuovo>
Non si deve cambiare il nome di un file aperto.

COPY TO <nome del file>
Crea una copia del file su disco.

CLEAR
Chiude tutti i file e cancella tutte le variabili di memoria.

SELECT [PRIMARY] [SECONDARY]
Consente l'apertura indipendente di due file nello stesso tempo. I dati vengono trasportati dall'uno all'altro mediante i prefissi P. e S.

DISPLAY FILES [ON <d>]
Presenta la lista delle base-dati che si trovano nel disco in uso oppure nel disco specificato. Si può usare anche il comando LIST al suo posto.

DISPLAY FILES LIKE <maschera> [ON <d>]
Presenta altri tipi di file nel disco.

QUIT
Chiude tutte le aree active e tutti i file e abbandona il dBASE II.

--- PER ORGANIZZARE UNA BASE-DATI

SORT ON <chiave> TO <file nuovo>
INDEX ON <chiave> TO <file nuovo>
Con tutti e due questi comandi si possono usare chiavi multiple.

INSERT [BEFORE] [BLANK]
Inserisce un record fra altri all'interno di una base-dati.

UPDATE FROM <altro file> ON < chiave >
Addiziona nuove cifre ai totali esistenti oppure sostituisce dati nel file in USE partendo da un altro file.

MODIFY COMMAND < nome del file>
Consente di effettuare cambi nel file comandi senza bisogno di utilizzare un programma di video scrittura.

--- PER USARE LE VARIABILI

(E' lecito usare fino a 64 variabili di memoria, oltre a qualsiasi numero di nomi di campo).

LIST MEMORY, DISPLAY MEMORY
Tutti e due questi comandi presentano sullo schermo le variabili, il tipo di dato e il contenuto.

STORE < valore > TO < var >
Assegna o cambia il valore delle variabili.

RELEASE <var>
Annulla la variabile indicata.

SAVE MEMORY TO < nome del file >
Immagazzina le variabili di memoria nel file indicato, con l'estensione .MEM.

RESTORE FROM < nome del file >
Legge le variabili di memoria contenute nel file e le ripristina distruggendo le altre variabili che si trovano nella memoria.

--- ENTRATA INTERATTIVA

WAIT
Interrompe l'attività sul video e permette che venga ripresa battendo qualsiasi tasto.

REPORT [FORM < nome del modulo >]
Crea uno schema personalizzato per le uscite e quando gli viene richiesto presenta i dati ordinati secondo quello schema.

@ <coord> SAY < var/esp/str >
Emette le uscite sullo schermo o sulla stampante, formattate secondo l'indicazione ricevuta. Per formattare ulteriormente la stampa, si può aggiungere [USING <formato>].

TEXT
Presenta un blocco di testo non formattato dal comando ? oppure da < coord > SAY.

--- PER PROGRAMMARE

(I programmi si immagazzinano nei file comandi con l'estensione .CMD o .PRG per i sistemi a 16 bit).

DO <nome del file>
Inizia l'esecuzione di un programma.

IF <condizioni >
Esegue comandi.
- Realizza scelte singole oppure multiple, quando è annidato con ELSE.

ELSE
Esegue altri comandi.

ENDIF
Termina i comandi IF e ELSE.

DO WHILE < condizioni >
Esegue comandi.
- A un certo punto le <condizioni> devono risultare cambiante all'interno del ciclo.

ENDDO
Termina il comando DO WHILE.

```
DO CASE                          - Effettua varie scelte.
      CASE <esp>
        <comandi>
      CASE<esp>
        <comandi>
      OTHERWISE
        <comandi>
ENDCASE
```