di Francesco Petroni

Archivi dati e archivi indice

Terza parte

Nella scorsa puntata del corso abbiamo visto come si costruisce un archivio, definendone prima la struttura (nome, tipo e lunghezza dei campi) e esaminando i vari comandi con i quali si inseriscono, modificano e cancellano i dati.

Abbiamo visto come il DB organizza fisicamente i record e cioè come li accoda l'uno all'altro attribuendogli numeri progressivi. Questo numero progressivo, identificato dalla variabile RECNO () è utilizzabile per identificare la posizione fisica del record all'interno del file. Esistono quindi alcuni comandi che accettano come parametro il numero record e permettono il movimento del puntatore sul record voluto (go <num. record>, skip +/ − num.).

Il numero record identifica l'organizzazione fisica del file, in quanto fisicamente i record sono accodati l'uno all'altro. Questa organizzazione è estremamente comoda in alcuni casi, ma assolutamente inadeguata in altri.

È comoda quando anche l'organizzazione logica dei record segue una progressione numerica, ma questa progressione non deve permettere eccezioni, come inserimenti forzati o cancellazioni fisiche. Queste due operazioni richiedono necessariamente una riattribuzione dei numeri progressivi, operazione ovviamente possibile, ma tanto più impegnativa e lenta quanto più voluminoso è l'archivio.

Accanto all'organizzazione fisica dei record all'interno del file, gestita in tutto e per tutto dal DB, esiste un'altra struttura, che permette di costruire una o più organizzazioni logiche dell'archivio, tramite le quali è possibile eseguire una serie di operazioni senza essere costretti a conoscerne l'organizzazione fisica. Questa struttura è l'ARCHIVIO INDICE.

La padronanza del concetto di archivio indice è fondamentale per la conoscenza del DB e per il suo sfruttamento, ma purtroppo è un concetto non immediato perché non trova esemplificazioni nella vita di tutti i giorni.

Facciamo un esempio. Abbiamo tutti chiaro il concetto di elenco telefonico, tale elenco ha una organizzazione per ordine alfabetico. Se conosciamo il cognome dell'abbonato siamo abituati a fare una ricerca secondo una modalità che si definisce binaria (anche se non lo sappiamo).

Supponiamo di dover cercare il cognome MANZONI, il ragionamento che facciamo è questo: la lettera «M» è quasi a metà dell'alfabeto e quindi MANZONI sarà quasi a metà dell'elenco. Apriamo l'elenco pressapoco a metà e vediamo quali sono i cognomi degli abbonati di quella pagina.

Se sono cognomi precedenti a Manzoni, valutiamo di quanto abbiamo sbagliato, se di poco avanziamo di poche pagine, se di tanto avanziamo di molte pagine. Viceversa se abbiamo aperto una pagina successiva a Manzoni arretriamo.

Questa tecnica di ricerca efficace ci è talmente familiare che in genere non riflettiamo sulla sua logica. In realtà si tratta di una logica perfettamente traducibile in un flow-chart e quindi in un programma, l'unica limitazione sta nel fatto che, poiché nel programma non possiamo inserire facilmente una logica di valutazione, ragioneremo spezzando a metà l'archivio, ovvero apriremo l'elenco a metà, se ci troviamo troppo avanti, apriremo a metà la prima metà e così via. È comunque una tecnica di ricerca rapida con la quale si raggiunge il record voluto eseguendo pochi confronti.

Questo metodo va bene solo nel caso che l'archivio sia organizzato fisicamente per ordine alfabetico, non può valere ad esempio nel caso di ricerca di un abbonato partendo dal suo indirizzo o dal suo numero di telefono. In questo caso poiché l'organizzazione fisica non segue la logica di ricerca che ci interessa saremo costretti a scorrere tutto l'archivio alla ricerca dell'abbonato di cui conosciamo solo l'indirizzo o il numero di telefono.

Oppure occorrerebbe avere tre elenchi differenti, uno per cognome uno per indirizzo e uno per numero di telefono. Ma in questo caso è evidente che il contenuto dei tre elenchi è assolutamente lo stesso, varia solo la sua organizzazione.

Torniamo al DB: l'ARCHIVIO DATI rimane organizzato per numero progressivo di immissione, l'ARCHIVIO INDICE crea una corrispondenza tra l'organizzazione logica che ci interessa e numero record. Se ci interessano più logiche, costruiremo più archivi indice, mentre, ed è questa la co-

```
. * PRGUNO
SET TALK OFF
* elimina echo dei comandi
USE PRGDAT
DISPLAY STRUCTURE
* visual. struttura del file
?
LIST
?
INDEX ON CODICE TO PRGND1
* crea primo indice
LIST
?
INDEX ON NOME TO PRGND2
* crea secondo indice
LIST
CLEAR ALL
```

Figura 1 - Listato Programma PRGUNO. Il programma serve per esemplificare la funzione degli ARCHIVI INDICE, che permettono di riorganizzare il modo di «vedere» un ARCHIVIO DATI, modificandone l'ordine LOGICO, ma lasciando inalterato il contenuto e l'ordine FISICO.

sa fondamentale, l'ARCHIVIO DATI rimane sempre lo stesso.

Per tornare all'esempio dell'elenco telefonico è come avere un elenco completo di tutti i dati (cognome, indirizzo, numero) ciascuno dei quali individuato da un progressivo. L'archivio indice è in sostanza un elenco, di lunghezza uguale a quella dell'archivio dati, ma il cui contenuto è semplicemente dato da chiave (ad esempio Cognome, questa volta organizzata per ordine alfabetico) e numero record.

L'operazione di posizionamento viene svolta in due fasi, la prima, con il metodo binario e quindi pressochè istantanea, sull'archivio indice. Da questo viene prelevato il numero record tramite il quale ci si posiziona istantaneamente sul record voluto nell'archivio dati.

Di tutte queste fasi si occupa il DB in maniera trasparente per l'utente, suo unico compito essendo quello di definire archivi da usare e indicare il cognome da ricercare.

Se poi si inserisce un nuovo record, nell'archivio dati viene semplicemente accodato, mentre l'archivio indice viene riorganizzato in funzione della posizione che il nuovo arrivato prende nell'organizzazione, e la sua riorganizzazione è immediata.

Il primo vantaggio dell'utilizzazione degli archivi indice sta nel fatto che praticamente può essere del tutto ignorata l'organizzazione fisica dell'archivio, in quanto tutte le operazioni di ricerca, inserimento, modifica, ecc. possono avvenire attraverso l'organizzazione logica e nel fatto che l'organizzazione logica viene costantemente tenuta aggiornata.

L'altro vantaggio sta nel fatto che se di uno stesso archivio servissero più organizzazioni logiche non occorre fare duplicazione di dati, in quanto esisterebbero solo più indici che riferiscono a numeri record dello stesso archivio dati.

Al contrario nel caso di elenchi «cartacei» del telefono occorrerebbero tre elenchi, di uguale formato e contenuto, il primo per cognome, il secondo per indirizzo e l'ultimo per numero telefonico.

La gestione degli archivi indice avviene in DB tramite una nutrita serie di istruzioni specifiche che permettono, sia lavorando in comandi diretti che sotto programma, non solo di eseguire le funzioni comuni di gestione archivi, ma anche complesse operazioni che riguardano l'insieme dei record.

In figura 1 esemplifichiamo il concetto, usiamo e listiamo (comando LIST) un archivio, dapprima senza indice, e quindi l'elenco è per numero record. Poi costruiamo un archivio indice per cognome per cui il successivo comando di list elenca gli stessi dati secondo la nuova organizzazione, costruiamo un altro indice, per numero di telefono e eseguiamo di nuovo il comando LIST.

L'altro campo di utilizzazione degli archivi indice è nella costruzione della relazionalità tra vari archivi (punto di forza del DB) per cui dati più archivi differenti, ma che abbiano dei campi corrispondenti, ovvero in relazione tra di loro è possibile operare, contemporaneamente e senza penalizzazioni in termini di prestazioni, su tutti gli archivi, prelevando le varie informazioni che interessano là dove risiedono.

Prima di passare alle esemplificazione di quanto fin qui detto, riassumiamo i vari campi di utilizzazione degli archivi indici:

— per creare organizzazione degli archivi dati;

— per analizzare il contenuto di archivi dati;

— per creare chiavi di accesso all'archivio dati;

— per realizzare la relazionalità tra archivi;

— per creare tabelle di controllo dati.

In pratica sull'archivio indice si basa tutta la potenzialità dello strumento DB, anche perché molte delle sue istruzioni pretendono archivi indicizzati, per cui una sua utilizzazione senza il ricorso agli indici è sicuramente limitata.

E diremo anche che poichè la gestione degli archivi indici necessita di istruzioni con sintassi complessa diventa pressochè indispensabile ricorrere alla programmazione per poter effettivamente sfruttare le potenzialità dello strumento.

Come si crea un Archivio Indice

Una volta definita la struttura del file è subito possibile costruire un file indice, indipendentemente dal fatto che nel file siano stati inseriti o meno dei record.

La chiave su cui costruire l'indice può essere un campo, o l'insieme di più campi, o una qualsiasi espressione matematica o di stringa che compren-

```
. do pr
Struttura del database: B:PRGDAT.dbf
Totale record inseriti:        8
Ultima revisione       : 01/01/80
Campo  Nome campo  Tipo campo    Dim    Dec
    1  CODICE      Carattere      4
    2  NOME        Carattere     16
    3  CITTA       Carattere     12
    4  PROV        Carattere      2
    5  DATISCR     Data           8
    6  QUOTA       Numerico       7
Totale:                          50

 Record  CODICE NOME            CITTA        PROV DATISCR    QUOTA
      1  0123   ROSSI           ROMA         RM   03/12/84  600000
      2  0342   GIGLI           ROVIGO       RO   23/11/84  456000
      3  8123   LUCIDI          GENOVA       GE   12/03/82  500000
      4  4134   DONINI          MONZA        MI   11/07/82  560000
      5  4356   VALLI           FIESOLE      FI   01/08/81  500000
      6  4009   SANTI           BOLOGNA      BO   05/05/79  350000
      7  3302   WILLIAMS        ROMA         RM   07/12/83  350000
      8  0012   VERDI           ROMA         RM   04/03/83  500000

Record  CODICE NOME            CITTA        PROV DATISCR    QUOTA
     8  0012   VERDI           ROMA         RM   04/03/83  500000
     1  0123   ROSSI           ROMA         RM   03/12/84  600000
     2  0342   GIGLI           ROVIGO       RO   23/11/84  456000
     7  3302   WILLIAMS        ROMA         RM   07/12/83  350000
     6  4009   SANTI           BOLOGNA      BO   05/05/79  350000
     4  4134   DONINI          MONZA        MI   11/07/82  560000
     5  4356   VALLI           FIESOLE      FI   01/08/81  500000
     3  8123   LUCIDI          GENOVA       GE   12/03/82  500000

Record  CODICE NOME            CITTA        PROV DATISCR    QUOTA
     4  4134   DONINI          MONZA        MI   11/07/82  560000
     2  0342   GIGLI           ROVIGO       RO   23/11/84  456000
     3  8123   LUCIDI          GENOVA       GE   12/03/82  500000
     1  0123   ROSSI           ROMA         RM   03/12/84  600000
     6  4009   SANTI           BOLOGNA      BO   05/05/79  350000
     5  4356   VALLI           FIESOLE      FI   01/08/81  500000
     8  0012   VERDI           ROMA         RM   04/03/83  500000
     7  3302   WILLIAMS        ROMA         RM   07/12/83  350000
```

Figura 2 - Output Programma PRGUNO. L'ordine FISICO è dato dal progressivo di immissione, gestito totalmente dal DB e identificabile con una variabile (# in DB II e RECNO () in DB III). L'ordine logico non è altro che l'ordine alfabetico prodotto dalla chiave.

da uno o più campi. Insomma si può costruire una modalità di lettura di un archivio, comunque complessa.

La sintassi del comando è la seguente:

INDEX ON <chiave> TO <nome del file indice>

Una volta costruito un file indice, o più file indice, per richiamarli il comando è:

USE <file dati> INDEX <file indice 1>, <file indice 2>, ecc.

Questo se si vogliono aprire i file indici contemporaneamente al file dati, altrimenti si possono aprire successivamente, ma solo se è già attivo il file dati. Il comando è:

SET INDEX TO <file indice 1>, <file indice 2>, ecc.

Se di uno stesso archivio dati si aprono più archivi indici il DB garantisce il loro allineamento in caso di inserimenti, modifiche, cancellazioni. Ma di tutti quelli aperti quello usato in quel momento (su cui per intendersi lavora il comando LIST) è il primo. Se si vuole utilizzare il secondo, senza però perdere la garanzia dell'allineamento, basta invertirli con un comando:

SET INDEX TO <file indice 2>, <file indice 1>, ecc.

Tanti più archivi DATI e tanti più archivi INDICI sono aperti contemporaneamente tanto più è difficile trovarsi nel momento giusto con in uso l'archivio dati e l'indice giusti. Risulta molto utile in fase di prova dei programmi l'uso del DISPLAY STATUS.

Esempi di Archivi Indice

Facciamo ora alcuni esempi di come si costruiscono file indici. Supponiamo di aver realizzato un archivio per inserire tutti i dischi della nostra discoteca, la cui struttura sia:

TITOLO
GENERE
AUTORE
INTERPRETE
MARCA
DATA INCISIONE

Da un file dati comprendente 6 campi si potrebbero creare decine di file indici. Ad esempio:

INDEX ON titolo TO indice 1

utile per ricercare un disco dato il titolo

INDEX ON autore+interprete TO indice 2
INDEX ON interprete+autore TO indice 3

raggruppamenti incrociati, il primo mette insieme tutti i dischi di BEETHOVEN, e all'interno di questo insieme vengono raggruppati per interprete, il secondo fa il contrario, raggruppa gli interpreti, ad esempio VON KARAYAN, e all'interno li raggruppa per autore.

Sia chiaro che il file dati è sempre lo stesso, anzi i dischi sono sempre gli stessi, cambia solo l'organizzazione secondo la quale vengono visti.

INDEX ON str (year (data incisione), 2)+marca TO indice 4

L'organizzazione è per anno di incisione e all'interno di questo è per marca.

Quindi dal campo «data di incisione», viene estratto, per mezzo di due funzioni di stringa, presenti nella sintassi del DB III, l'anno che viene accoppiato alla marca. Per cui generalizzando si può dire che la chiave dell'archivio indice può essere costruita anche in maniera complessa, utilizzando uno o più campi.

In definitiva la chiave non si identifica con un campo, e in molti ambiti applicativi, questo risulta essere un grosso vantaggio.

Quando conviene usare gli Archivi Indice modalità interattiva e modalità batch

Come detto gli archivi indice risolvono parecchi dei problemi che si presentano nella gestione di grossi archivi di dati. Le modalità di uso degli archivi indice sono sostanzialmente due:

MODALITÀ INTERATTIVA: quando, in DB, si apre un archivio con il comando USE <nome dell'archivio>, si possono aprire contemporaneamente più di un archivio indice, relativi a quello stesso file. Se l'operazione di apertura è contestuale il DB garantisce l'aggiornamento e l'allineamento degli archivi (dati e indici). Il comando relativo è:

USE <nome archivio> INDEX <nomè indice 1>, <nome indice 2>, ecc.

Poiché il tenere aperto un archivio, di qualsiasi tipo, è comunque un «costo» in termini di prestazioni, risulta conveniente aprire solo quelli operativi, la cui utilizzazione è prevista nelle procedure operative.

MODALITÀ BATCH: a volte occorre utilizzare una organizzazione del file solo saltuariamente. Si pensi ad una elaborazione mensile di un archivio fatture, ad esempio per conteggiare le operazioni provincia per provincia. In questo caso è più opportuno costruire li per li l'archivio indice,

```
* 27/12/85 PRGDUE
set talk off
delete file prgnd3.ndx
use PRGDAT index PRGND1
index on PROV to PRGND3
go top
totg=0
numt=0
clear
? "Provincia  Num.     Quote"
do while .not. eof()
 totp=0
 nump=0
 var1=prov
 do while var1=prov
  totp=totp+quota
  nump=nump+1
  totg=totg+quota
  numt=numt+1
  skip
 enddo
 ? var1+"       ",str(nump,4),str(totp,9)
enddo
 ? "Totale ",str(numt,5),str(totg,9)
clear all
*
```

Figura 3 - Listato Programma PRGDUE. Viene esemplificato un uso dell'indicizzazione per organizzare l'archivio in modo da raggruppare i dati secondo una logica che permetta di eseguire dei calcoli. In questo caso vengono raggruppati per provincia e l'obiettivo è quello di eseguire i totali di un certo campo numerico per provincia.

Provincia	Num.	Quote
BO	1	350000
FI	1	500000
GE	1	500000
MI	1	560000
RM	3	1450000
RO	1	456000
Totale	8	3816000

Figura 4 - Output Programma PRGDUE. Vengono eseguiti conteggi e totali parziali, e un totale generale per provincia. Vedremo nelle prossime puntate comandi specifici per eseguire in maniera più diretta questi calcoli, ma che richiedono anch'essi una preventiva indicizzazione.

usarlo per la elaborazione e poi distruggerlo.

Questa operazione per archivi di una certa consistenza richiede qualche minuto, ma è sicuramente conveniente rispetto al costo di tenere costantemente aperto e aggiornato un indice che viene usato saltuariamente.

Nel corso della puntata esemplificheremo i vari aspetti degli archivi indice, che peraltro useremo ancora nelle prossime puntate. Cominceremo presentando un programma per la gestione di un semplice archivio la cui chiave di accesso sia un codice, poi vedremo come utilizzare un indice per analizzare dati di un file e infine realizzeremo un piccolo archivio da utilizzare come scodifica di una tabella.

Programma PRGUNO

La prima esemplificazione, listato del programma in figura 1 e output prodotto in figura 2, serve per far capire il modo di lavorare degli archivi indici. Supponiamo di avere un archivio

qualsiasi, per conoscere la sua struttura eseguiamo il DISPLAY STRUCTURE. Se eseguiamo il LIST, senza condizioni, viene visualizzato l'intero file e i vari record sono messi in ordine di immissione e cioè secondo il numero record.

Questo numero record, viene visualizzato dal comando LIST, e può essere considerato a tutti gli effetti come un campo numerico, generato automaticamente in fase di creazione della struttura. Non possono esistere record che abbiano il RECNO () uguale o buchi nella numerazione.

Le operazioni di cancellazione record, che come abbiamo visto la puntata scorsa si esegue a due livelli (DELETE e PACK), e l'operazione di inserimento (INSERT) provocano una riattribuzione dei numeri record, e sono quindi delle operazioni lente.

Il comando INDEX ON <campo> TO <nome file indice> produce un file di tipo *.NDX che comprende tanti elementi quanti sono i record dell'archivio dati, in ciascun elemento è presente la chiave (ovvero il contenuto del <campo>) e il numero record che quella chiave occupa nell'archivio dati.

L'archivio indice viene tenuto ordinato secondo la chiave dal DB, in modo che le operazioni di posizionamento possano essere eseguite con il sistema binario.

Quando si apre un file e si apre contemporaneamente un suo indice, i comandi di movimento nel file dati e di visualizzazione seguono l'ordine imposto dall'indice. Nella figura 2 vediamo l'effetto del comando LIST dapprima con l'indice CODICE e poi con l'indice COGNOME, ma anche i comandi di GO TOP, SKIP, ecc. seguono l'ordine dell'indice.

In caso di chiavi uguali, ad esempio due cognomi uguali, i record vengono raggruppati in sequenza e il primo è quello che ha il RECNO () più basso. Se si vuole evitare l'uguaglianza della chiave se ne può complicare la composizione, ad esempio COGNOME + NOME + ANNO DI NASCITA. In questo caso si riducono pericoli di omonimie di chiave, ma quando si imposta una chiave di ricerca bisogna immettere tutti i dati.

Il pericolo maggiore che si corre quando si usano archivi dati con uno o più indici è quello di disallinearli, ovvero un dato è presente in uno ma non in un altro. Se capita un fatto del genere la colpa è vostra e non del DB. Si può rimediare aprendo l'archivio dati e tutti gli indici e dando un comando REINDEX, ma questa deve essere considerata una procedura di emergenza.

Il DB, una volta che si aprono correttamente archivi dati e relativi indici, ne garantisce l'allineamento qualsiasi operazione si faccia o comando si usi, per cui, ripeto, se succede qualcosa, per cui vi sembra di aver perso dei dati, è sicuramente per una vostra disattenzione, probabilmente avete modificato un archivio senza che abbiate anche aperto gli indici ad esso collegabili.

```
* 27/12/85 prgqua
clear
set delimiter on
use PRGDAT index PRGND1
@  1,10 say "MC Microcomputer                    Corso DB 2 DB 3"
@  3,10 say "GESTIONE SEMPLICE ARCHIVIO ORGANIZZATO PER CHIAVE"
@  5,10 say "-------------------------------------------------"
*                       MAIN PROGRAM
do while .t.
@  8,0  clear
@ 18,10 say "-------------------------------------------------"
@ 20,10 say "       Per Qualsiasi Operazione Immetti il Codice"
@ 22,10 say "                              per Finire Premi >RETURN"
var1="  "
@  8,10 say "Immetti il Codice  " get var1 pict "!!!!!"
read
@ 20,0 clear
*                       FINE LAVORO
if var1=" "
  clear
  return
endif
*                       RICERCA
find "&var1"
*                       NON  C'E'
IF eof()
  var2=" "
  do while .not. var2$"SN"
    @ 20,10 say "Non Presente   Vuoi Immetterlo   S/N  " get var2 pict "!"
    read
  enddo
  @ 20,0 clear
  if var2="S"
   append blank
   replace codice with var1
   @ 10,10 say "Nominativo  "    get nome  pict "!!!!!!!!!!!!!!!!!"
   @ 12,10 say "Citta       "    get citta pict "!!!!!!!!!!!!!"
   @ 12,40 say "Provincia      " get prov  pict "!!"
   @ 14,10 say "Data Iscr.  "    get datiscr
   @ 14,40 say "Quota   "        get quota pict "#,###,###"
   read
  endif
*                       C'E'
ELSE
   @ 10,10 say "Nominativo  :"+nome+":"
   @ 12,10 say "Citta       :"+citta+":"
   @ 12,40 say "Provincia      :"+prov+":"
   @ 14,10 say "Data Iscr.  :"+dtoc(datiscr)+":"
   @ 14,40 say "Quota   : ,   ,   :"
   @ 14,49 say quota   pict "#,###,###"
   var3=" "
   do while .not. var3$"MCP"
    @ 20,10 say "M-modifica  C-cancella  P-prosegui   " get var3 pict "!"
    read
   enddo
   @ 20,0 clear
   if var3="M"
    @ 10,22 get nome  pict "!!!!!!!!!!!!!!!!!"
    @ 12,22 get citta pict "!!!!!!!!!!!!!"
    @ 12,55 get prov  pict "!!"
    @ 14,22 get datiscr
    @ 14,48 get quota pict "#,###,###"
    read
   else
     if var3="C"
       delete
       pack
     endif
   endif
ENDIF
ENDDO
```

Figura 5 - Listato Programma PRGQUA. Viene esemplificato un uso dell'indice per realizzare una chiave di accesso univoca ad un archivio dati. Il programma, pur non avendo nessun controllo, è completo, in quanto permette le funzioni di RICERCA, IMMISSIONE, MODIFICA e CANCELLAZIONE.

Programma PRGDUE

In questo esempio (listato di fig. 3 e output di fig. 4) viene creato un file indice dello stesso archivio usato prima, semplicemente per raggruppare o meglio per mettere in sequenza i record secondo una certa logica.

Il nostro obiettivo è quello di eseguire i totali del campo «quota» per ciascuna provincia. Se i dati fossero sparsi dovremmo ricorrere a tante variabili quante sono le province, e cosa ancor più pesante dovremmo creare nel programma un algoritmo che data la provincia identifichi direttamente quale è la sua variabile.

Viceversa, se riusciamo a raggruppare le province tutte assieme, possiamo eseguire lo stesso calcolo mentre facciamo scorrere l'archivio e possiamo usare una sola variabile.

Con il comando DO WHILE... ENDDO si creano due loop nestati. Quel-

lo esterno fa scorrere tutto l'archivio ed è quindi chiuso quando si verifica la condizione di EOF ().

Quello interno crea una cosiddetta condizione di rottura, ovvero si pone la variabile VAR1 uguale alla prima provincia, si va avanti finché la VAR1, corrispondente alla prima provincia, è uguale alle province dei record che scorrono (e per ogni ciclo si totalizza la «quota»). Quando la provincia cambia, si scrivono i totali, si azzera la variabile dei totali e si assegna alla VAR1 la nuova provincia.

Questo è il sistema più diffuso per eseguire calcoli di totali e, in ambiente DB, è permesso dalla organizzazione realizzata con gli archivi indice.

Nel listato vanno notate oltre alle istruzioni DO WHILE... ENDDO, anche le istruzioni di print, che sono? e ??, e che rappresentano la maniera più brutale per visualizzare dati, in quanto non accettano parametri di formato.

Questa è un'applicazione che chiamiamo BATCH, perché eseguendo delle elaborazioni su tutti i dati dell'archivio, richiede necessariamente molto tempo e quindi non può essere richiesta in modo interattivo. Analogamente l'organizzazione per province dell'archivio, non ha nessuna utilità operativa e quindi il relativo archivio indice è del tipo «usa e getta», conviene crearlo quando si usa e poi cancellarlo.

Programma PRGQUA

Il terzo esempio presentato in questa puntata del corso dedicata agli archivi indice, è rappresentato da un programma completo di gestione archivio, in cui esista una chiave «CODICE» che identifichi biunivocamente il record.

In molte applicazioni è necessario identificare un record tramite un codice, ad esempio in una procedura di magazzino con il CODICE ARTICOLO, o in una procedura di fatturazione con il CODICE CLIENTE, o nelle procedure personali di una azienda con la MATRICOLA del dipendente.

Scopo di tutti questi codici è proprio quello di evitare duplicazioni che potrebbero ingenerare equivoci, infatti nessuno può garantire che in una azienda non esistano due persone con lo stesso cognome o in un magazzino sia possibile identificare un articolo con una semplice descrizione.

Se nel nostro ARCHIVIO DATI abbiamo un campo CODICE e realizziamo un ARCHIVIO INDICE la cui chiave sia proprio il codice, possiamo far gestire del tutto dal DB il sistema di movimentazione dell'archivio.

Nel programma presentato, listato in figura 5, l'unica modalità di accesso all'archivio è tramite il campo CODICE.

Se non immettiamo un codice perché diamo un >RETURN< a vuoto significa che vogliamo finire il lavoro. Se viceversa immettiamo un codice esistono due possibilità, che tale codice non sia presente in archivio o che lo sia.

Se non è presente esistono due sottocasi, lo vogliamo immettere, oppure non lo vogliamo immettere semplicemente perché abbiamo sbagliato a digitarlo. Se invece è presente, esistono tre sottocasi, lo vogliamo solo consultare, oppure lo vogliamo modificare o, infine, lo vogliamo cancellare.

nella definizione dei REPORT, ma questo lo vedremo la prossima volta.

Tornando al listato va notato che nel DB non ci sono istruzioni di salto (il GOTO del BASIC) né di loop (FOR ... NEXT), quindi si utilizza molto il DO WHILE <condizione>, che crea un loop dal quale si esce solo se si verifica la condizione. Inoltre esistono altri due comandi, molto utilizzati in programmazione e cioè IF .. ELSE .. ENDIF che può essere nidificato, e il DO CASE.

C'è un tipo particolare di loop che si usa moltissimo il DO WHILE .T., che genera un loop eterno, dal quale si esce solo con un RETURN, per il rien-

```
MC Microcomputer                          Corso DB 2 DB 3
GESTIONE SEMPLICE ARCHIVIO ORGANIZZATO PER CHIAVE
------------------------------------------------

Immetti il Codice   :0123:

Nominativo  :ROSSI            :

Citta       :ROMA        :    Provincia      :RM:

Data Iscr.  :03/12/84:        Quota   :  600,000:


------------------------------------------------

M-modifica  C-cancella  P-prosegui    : :
```

Figura 6
Output Programma PRGQUA. È una Hard Copy della videata «piena» di dati. Ovvero se si immette un codice già presente in archivio, viene visualizzato, e viene richiesto un codice per Modificare, Cancellare e Proseguire. Se il codice non fosse stato presente sarebbe apparsa una maschera vuota.

Il programma presentato ha uno scheletro che identifica tutti questi casi e sottocasi, per cui si può considerare completo. È quindi abbastanza complesso da richiedere una certa attenzione.

Dopo una inizializzazione che prevede la pulizia dello schermo (clear), la definizione di delimitatori dei campi in input (set delimiter on) e l'apertura del file e relativo indice (sono i file usati nell'esempio di fig. 1, per cui use PRGDAT index PRGND1), ci sono le prime istruzioni di output.

Il comando più potente di print è il:

@ V,O SAY <nome della variabile o del campo> PICTURE <specifiche>

@ e SAY sono previsti dalla sintassi, V,O sono le coordinate verticale e orizzontale della device prevista in out (o video o stampante), la SAY può essere fatta di variabili o di nomi di campi dei file attivi in quel momento.

La PICTURE definisce il formato di visualizzazione a seconda delle specifiche che gli si indicano. Le specifiche possibili sono molto numerose e quindi risolvono la maggior parte dei casi.

L'unica carenza del comando PICTURE è che non è accettato con gli altri comandi di print (come ? e ??) e cosa ancora più grave, non è accettato

tro al programma chiamante o per finire, nel caso che il chiamante non ci sia. Nel nostro caso vogliamo lavorare con il file fin quando non diamo un RETURN a vuoto, che attiva l'esecuzione dell'istruzione «RETURN».

L'istruzione di input più evoluta è GET, che ha una sintassi identica alla SAY, e che può essere legata, in una stessa istruzione, ad una SAY. È una istruzione passiva, nel senso che entra in funzione solo per mezzo di una successiva istruzione di READ. Una READ può attivare una o più GET. In generale però quando della variabile in input si fa un uso immediato, o perché il dato deve essere controllato o perché il dato serve per proseguire, occorre dedicare una READ specifica alla GET.

Nel nostro caso dobbiamo immettere il codice e utilizziamo la variabile di comodo VAR1. Se VAR1 è vuota, ovvero abbiamo dato un RETURN, finiamo il programma altrimenti proseguiamo.

Viene eseguita la ricerca con la istruzione FIND «&VAR1», che significa: cerca nel file indice il record cui corrisponde il codice immesso nella variabile VAR1. La ricerca ha due possibili effetti, il codice non c'è, ovvero si verifica la condizione logica di EOF (), oppure c'è.

```
* PRGTRE Immissione e Controllo di Un Campo
set talk off
clea
use PRGDAT index PRGND1
select 2
use PRVDAT index PRVND1
list
wait
select 1
@ 11,10 say "MC microcomputer      corso DB2 & DB3"
@ 13,10 say "programma per il controllo di un dato"
@ 14,10 say "                in una tabella esterna"
@ 15,10 say "--------------------------------------"
do while .t.
@ 17,0  clear
var1="  "
f1=1
 do while f1=1
    @ 17,10 say "Immetti Cod. Prov. " get var1 pict "!!"
    read
    @ 17,35 say "                                "
    if var1=" "
       clear
       return
    else
       select 2
       find "&var1"
       if eof()
          @ 17,35 say "Inesistente"
       else
          @ 17,35 say citta
          f1=0
       endif
    endif
 enddo
 @ 19,10 say "--------------------------------------"
 wait
 select 1
 * posizione del comando REPLACE
 * sul campo archivio 1 della variabile var1
enddo
```

Figura 7 - Listato Programma PRGTRE. Viene esemplificato il controllo di presenza in una tabella esterna, di un dato in immissione. Supponiamo di voler controllare che la sigla di provincia immessa sia presente in un archivio secondario. Se non è presente il controllo non permette la prosecuzione, viceversa scrive la scodifica della sigla.

Se non c'è il programma chiede «Vuoi Immetterlo S/N». Per controllare la risposta a questa domanda viene realizzato un piccolo loop DO WHILE, dal quale si esce solo se la risposta, inserita nella variabile VAR2, è o S o N. Per controllare l'uscita dal loop si usa la funzione logica $, di ricerca di sottostringa, che dà un True se è verificata o altrimenti un False.

Se si è immesso S, viene eseguito un comando di APPEND BLANK, che esegue l'accodamento, di un nuovo record, per ora vuoto, e che riempiamo subito nel campo codice con la variabile di comodo VAR1, e poi nei vari campi con le GET direttamente eseguite con il nome dei campi.

Se invece si è immesso N si esce dalle varie condizioni e sottocondizioni e si rientra al loop principale di immissione codice.

Nel caso il codice immesso fosse presente viene visualizzato l'intero record e viene richiesto che cosa si intende fare (vedi fig. 6). Le possibilità sono Prosegui, che fa tornare alla richiesta codice, Cancella che fa il Delete del record e il Pack dell'archivio. L'ultima opzione è la Modifica che riattiva i GET, sovrapponendoli agli output precedenti, e quindi questa volta a campi pieni.

Come detto la struttura del programma è completa e quindi risulta abbastanza complesso per un principiante. Ad un esperto viceversa risultano evidenti delle carenze che preferiamo denunciare direttamente, ma che abbiamo dovuto accettare per non complicare ulteriormente la comprensione ai meno esperti.

Mancano del tutto controlli sui campi, e questa è una situazione che nessuna applicazione reale presenta.

Non è curata l'estetica della maschera, mancano caratteri speciali e trucchetti per abbellire.

Manca una richiesta di OK alla fine sia dell'immissione che della modifica.

Il pack è fatto al volo, e abbiamo sempre detto che non può essere fatto durante una procedura interattiva.

Queste carenze dichiarate possono diventare vostro argomento di sperimentazione.

Programma PRGTRE

Una delle carenze dichiarate è quella di mancanza di controlli dei dati immessi. Come ulteriore esemplificazione dell'uso degli archivi indici supponiamo quindi di voler controllare al momento dell'immissione il campo provincia e per far questo vogliamo utilizzare un archivio in cui sono memorizzate tutte le sigle delle province e il nome della città cui corrisponde la sigla stessa.

Questo è un tipo di controllo frequente, e che può essere inserito anche nel programma PRGQUA presentato prima.

Il concetto è questo: inserito il campo provincia nel primo archivio, si attiva il secondo, si va a cercare se esiste una provincia con quella sigla. Se non c'è si chiede di nuovo, e si invia un messaggio di non trovato. Se c'è si visualizza il nome della città cui corrisponde la sigla e si prosegue tranquillamente.

```
 Record  SIGLA CITTA
      4  BO    BOLOGNA
      2  FI    FIRENZE
      5  GE    GENOVA
      1  RM    ROMA
      3  TO    TORINO
Premere un tasto qualsiasi per continuare.


MC microcomputer      corso DB2 & DB3

programma per il controllo di un dato
             in una tabella esterna
--------------------------------------

Immetti Cod. Prov.  TO   TORINO
--------------------------------------
Premere un tasto qualsiasi per continuare.
```

Figura 8 - Output del Programma PRGTRE. L'elenco delle province viene visualizzato semplicemente per mostrare le strutture dell'archivio. L'obiettivo è quello di controllare, scodificandola, la sigla immessa. Se la sigla è errata appare un messaggio «inesistente».

Questo è uno dei controlli che si possono eseguire sui dati in immissione, oltre a quelli eseguiti automaticamente del DB III in dipendenza delle tipologie della definizione dei campi.

Per fare un esempio il campo Data Iscr. viene controllato come DATA del DB III per cui non è possibile ad esempio immettere 29/02/85 se l'85 non è un anno bisestile. Un controllo che la Data Iscr. sia seguente la data di nascita va eseguito da programma.

Quando si costruisce un archivio e un programma per la sua gestione, occorre stabilire quali controlli far eseguire alla definizione, quali controlli eseguire da programma e quali invece non eseguire affatto.

Ad esempio, se dobbiamo costruire un archivio di indirizzi e vogliamo inserire il Codice Postale, dobbiamo decidere se e a qual livello controllarlo automaticamente. Se lo vogliamo controllare completamente dovremmo disporre di un archivio con tutte le località e per tutte le città grandi con tutte le strade. Ovvero dovremmo utilizzare un archivio ben più voluminoso dell'archivio dati che vogliamo realizzare.

Per tornare al programma presentato, la difficoltà principale sta nel passaggio, da eseguire al momento giusto, dall'archivio principale a quello secondario e viceversa, e soprattutto nell'uso della variabile di comodo VAR1, che viene controllata e, se il controllo viene superato, viene trasferita sul campo dell'archivio principale tramite il comando REPLACE.

La logica della nidificazione degli IF .. ELSE .. ENDIF e del DO WHILE .. ENDDO è sempre la stessa. Il controllo sul campo viene gestito tramite un flag FL. Viene posto uguale a 1 per poter entrare nel loop. Se il controllo viene superato viene posto uguale a 0, e si può uscire dal loop.

MC

Un LITHIUS® per tutte le professioni.

Un PC garantito dai suoi componenti D.O.C.

Nessuna sorpresa può capitarvi durante l'impiego di un PC LITHIUS. Nessuna incompatibilità Hardware o Software, malfunzionamenti, guasti improvvisi.

I componenti adottati nell'assemblaggio del PC LITHIUS sono tutti D.O.C.! Ciascuno di essi, infatti, è prodotto da grandi Marche, da noi scelto dopo accurata selezione e fornito sempre dai medesimi produttori, il cui nome è dichiarato nella garanzia che accompagna ogni PC LITHIUS.

Come ulteriore garanzia, dopo l'assemblaggio ciascun PC viene sottoposto ad una prova d'uso che dura un'intera settimana, vale a dire 170 ore di funzionamento ininterrotto!

SISTEMA BASE - (PC/1 - E.D.) L.2.160.000 + IVA

- Piastra madre con microprocessore INTEL 8088/4,77 MHz (opz. clock 4,77/8 MHz).
- 8 slots (IBM/PC hard-soft compatibili)
- Memoria RAM fornita su piastra: 256 kRAM (espandibile a 640 kB, direttamente 'on board')
- Memoria ROM 8 kB (BIOS) espandibile 64 kB
- 4 canali DMA - 8 livelli interrupts
- Scheda interfaccia video-grafica monocromatica (a scelta RGB colore) alta risoluzione (720 × 350 pixel)
- Porta per collegamento stampante parallela
- Video Philips od opzionale ADI:
  a) Monocromatico TTL verde alta risoluzione, 12'', 920 × 350 pixel.
  b) oppure Monocromatico videocomposito (verde o ambra).
- Tastiera ergonomica ASCII con tasti funzione e operativi (84 tasti) ben visibili, LED di caps lock, e numerical lock.
- Un driver slim chiusura a levetta, TEAC, fra i migliori sul mercato.
- Alimentazione 130 Watt 220 Volt alto rendimento, switching con ventola di raffreddamento silenziosa.
- CPU compatibile con sistemi operativi PC DOS, MS DOS, CCPM 86, CPM 86.
- Completo di cavi e manualistica in italiano.

SISTEMA DOPPIO DRIVER (PC/2 - E.D.) L. 2.440.000 + IVA

SISTEMA CON HARD DISK 10 MByte e 1 DRIVER (PC/XT - E.D.)

- Come PC/1 ma con aggiunta di un Hard Disk slim TEAC da 10 MByte formattati.
- **COSTO SISTEMA PC/XT E.D. L. 3.500.000 + IVA**

electronic devices srl

00173 Roma — Via Ubaldo Comandini, 49
Tel. 613.23.94-613.26.19 - Tlx. 620570 ELDEV-I

Rivenditori autorizzati

Sardegna: ASSOVEL - Via Sassari 57
09100 Cagliari - t. 070/665849

Sicilia: DATAMAX - via Campolo 39
90145 Palermo - t. 091/575369
HARDWARE SOFTWARE
SERVICE - 98100 Messina
Via Cernaia 11 - t. 090/775912