Le basi del
Data Base

Data Base Management System: costruiamo un data base

di Andrea de Prisco

Ottava Parte

A conclusione del ciclo di articoli sulle basi di dati, questo mese e il prossimo parleremo di una piccola applicazione creata per il Commodore 64: il Galileo/J, interamente scritto in Basic.
Su questo numero vi verrà consegnato tutto il listato con una rapida spiegazione dei comandi per i più impazienti: il mese prossimo, con più calma, vedremo in pratica qualche semplice esempio d'uso.

Il Galileo/J

Il Galileo/J deve il suo nome al Galileo, un sistema di gestione per basi di dati tuttora in fase di realizzazione presso il Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Pisa, al quale è ispirato nei suoi lineamenti generali.

Si tratta di un data base seguente il modello semantico dei dati o, per essere più precisi, un subset dello stesso. Infatti il modello semantico dei dati, discusso nella terza puntata di questa serie, è il più avanzato dei modelli visti anche perché è il più nuovo, ancora oggetto di studio in tutto il mondo. Dire di averlo implementato completamente su un personal computer sarebbe una grossa eresia. Il Galileo/J è usato in questo contesto per mostrare qualcosa di soltanto prossimo a sistemi di gestione per basi di dati più seri.

Gioca tutte le sue carte sul meccanismo della classificazione, mediante il quale entità diverse vengono considerate omogenee, da inserire cioè in una stessa classe. Il concetto di entità è del resto assai intuitivo: sono le cose che ci interessano, appartenenti al mondo osservato.

Per fare qualche esempio il caro amico Accalappiagalli Gregorio, il libro "Software, violenza e vivisezione", una fattura di vendita sono entità. Se abbiamo a che fare con questo genere di oggetti, potremo descrivere la nostra base di dati creando tre classi: la prima denominata Persone, la seconda Libri, la terza Fatture. Posto che le fatture riguardino un negozio di libri e, conseguentemente, una serie di libri acquistati e la persona acquirente, potremo mettere le tre classi in associazione, permettendo facili operazioni di navigazione tra i dati, per risalire da un libro del nostro negozio all'insieme di persone che l'hanno acquistato, o roba simile.

Di questo però ci siamo già occupati sul n. 35 di MC, nell'articolo riguardante appunto il modello semantico al quale vi rimandiamo per eventuali chiarimenti.

Il Galileo/J dispone di alcuni operatori per definire le classi di una base, nonché per recuperare, editare o cancellare elementi dalla stessa. Il meccanismo di Aggregazione serve per definire la struttura di una classe, non senza stabilire le associazioni con altri dati. Ricordiamo che nel modello semantico le associazioni con altri elementi sono proprietà degli elementi stessi. Ritornando all'esempio precedente, una fattura avrà come proprietà un numero progressivo, una data, un Cliente, (associazione con la classe Persone) e l'elenco dei libri acquistati (associazione multipla con la classe Libri).

Nel nostro caso, definendo la struttura di una classe, oltre alle proprietà chiave e costanti, elencheremo l'insieme delle proprietà associazione, realizzate col meccanismo delle chiavi esterne proprie questa volta dei modelli relazionali.

Una chiave esterna è chiave per un'altra classe: identifica univocamente un altro elemento. Nessun vincolo è posto sull'esistenza dello stesso, in quanto eventuali associazioni pendenti rimangono tali. Se diciamo cioè che una fattura riguarda il cliente Arcibaldo Polizzi, il fatto che questo non sia presente nella classe Persone non vieta di inserire lo stesso la fattura in classe. È quando avremo completato l'inserimento di tutte le persone (compreso cioè il Polizzi) che avremo la correlazione desiderata.

Chiariremo in seguito con vari esempi. Per quanto riguarda gli operatori per la ricerca, possiamo chiedere al sistema di trovare tutti quelli, o soltanto di mostrarci il primo, che soddisfano una particolare condizione. È possibile poi muoverci in una classe chiedendo il successivo elemento che soddisfa la condizione data.

Ogni associazione con altri elementi (di altre classi o della stessa) viene esplicitata al primo livello: se un elemento della classe A è in associazione con un elemento della classe B, e gli elementi della classe B sono in associazione con elementi della classe C, accade che accedendo alla classe A otterremo anche gli elementi correlati in classe B (e non quelli in classe C); accedendo di contro in classe B, otterremo anche gli elementi di C. Ciò per evitare cicli infiniti: infatti, se le classi sono 2, A e B, e ogni elemento di A è in relazione con un elemento di B e viceversa, il sistema comincerebbe a saltare da una classe all'altra, all'infinito, mostrando sempre gli stessi 2 elementi. Cerchiamo di essere un po' più chiari: abbiamo due classi, Maschi e Femmine, con la doppia associazione SposatoCon. È chiaro che se Ermenegildo è sposato con Ermengarda, deve essere anche vero il viceversa. Se l'esplicitazione delle associazioni, cioè il mostrare oltre all'elemento cercato anche quelli a lui correlati, fosse ricorsiva accadrebbe che chiedendo notizie su Renzo sapremo che questo è sposato con Lucia, la quale è sposata con Renzo, il quale è sposato con Lucia, la quale è sposata con Renzo... until crash (fino a quando non si spacca tutto).

```
10 DIM CO$(50),CL$(7),TP$(7,8),TU$(7,8),K1$(7),AU$(7,1)
,KE$(7,60),LE%(7)
15 DIM A1$(7,60),A2$(120)
17 FORI=0TO7:KE$(I,0)="*":NEXT
20 GOSUB7000:PRINTCHR$(8)
21 REM ++++++++++++++++++++++++++++++++++++++++++++
22 REM +                                          +
23 REM +  @@    @@  @    @@@ @    @@@@  @@        +
24 REM + @  @ @  @ @     @  @    @     @  @        +
25 REM + @    @  @ @     @  @    @     @  @   @    +
26 REM + @ @@ @@@@ @     @  @    @@    @  @   @    +
27 REM + @  @ @  @ @     @  @    @     @  @   @ @@ +
28 REM +  @@@ @  @ @@@@ @@@ @@@@ @@@@  @@  @    @  +
29 REM +                                    @      @  +
30 REM +     ------------------------       @   @  @  +
31 REM +     (C)  1985  ADP SOFTWARE            @@   +
32 REM +     ------------------------               +
33 REM +                                          +
34 REM ++++++++++++++++++++++++++++++++++++++++++++
35 REM
50 OPEN10,0:T2$="A;":CC$=CHR$(13)
60 FORI=1TO40:BS$=BS$+CHR$(157):NEXT
100 QQ$=CHR$(14)+"¬:":B$="":PRINT"□";:Q$=QQ$
:F$="FAILURE : ":MI$="MISSING "
110 PRINTCHR$(13)+Q$;:A$="":INPUT#10,A$:B$=B$+A$+" "
120 IFRIGHT$(B$,2)<>";" THENQ$="  ":GOTO110
130 PRINTCHR$(13)+"■"
135 T1$=T2$:T2$=B$:GOSUB500:B$="":Q$=QQ$:GOTO110
500 II=0:JJ=0:FL=0
510 II=II+1:A$=MID$(B$,II,1):IFA$=CHR$(34)THEN950
520 IFA$<>" "ANDA$<>"("ANDA$<>")"ANDA$<>";"ANDA$<
>"="ANDA$<>"←"THEN510
522 CO$(JJ)=LEFT$(B$,II-1):IFCO$(JJ)<>""THENJJ=JJ+1
525 IFA$<>" "THENCO$(JJ)=A$:JJ=JJ+1
530 B$=MID$(B$,II+1):II=0
540 IFB$<>""THEN510
550 A$=CO$(0):IFA$=";"THENB$=T1$:GOTO135
552 IFA$<>"MAKE"ANDMA=1THENMA=0:GOSUB8000
555 IFA$<>"NEXT"THENTS$=""
560 IFA$="CLASS"THENGOSUB1000:RETURN
570 IFA$="FIND"THENGOSUB2000:RETURN
580 IFA$="NEXT"THENGOSUB3000:RETURN
582 IFA$="DEF"THENGOSUB12000:RETURN
590 IFA$="MAKE"THENGOSUB4000:RETURN
600 IFA$="DESTROY"THENGOSUB5000:RETURN
610 IFA$="QUIT"THENCLOSE10:GOTO6000
620 IFA$="EDIT"THENGOSUB9000:RETURN
630 IFA$="PRINTER"ANDCO$(1)=";"THENPP=1:RETURN
640 IFA$="DISPLAY"ANDCO$(1)=";"THENPP=0:RETURN
650 IFA$="ALL"THENAL=1:GOSUB3100:AL=0:RETURN
900 ER=10:GOTO15000
950 II=II+1:IFMID$(B$,II,1)<>CHR$(34)THEN950
960 A$=" ":B$=MID$(B$,2):II=II-1:GOTO522
1000 I=0:CL$=CO$(1)
1005 IFI>7THENER=3:GOTO15000
1010 IFCL$=CL$(I)THENER=0:GOTO15000
1020 IFCL$(I)<>""THENI=I+1:GOTO1005
1040 IFCO$(2)<>"(-)"THENER=1:GOTO15000
1050 IFCO$(3)<>"("THENER=2:GOTO15000
1055 CL=I
1060 FORK=0TO7:FORKK=0TO1:AU$(K,KK)="":NEXT:NEXT
1070 KK=4:II=0
1080 IFCO$(KK+1)<>"←"THENER=4:GOTO15000
1090 C$=CO$(KK+2)
1100 IFC$="STRING"ORC$="INT"ORC$="PAGE"THEN1110
1105 IFC$="EXTKEY"ORC$="EXTKEYS"THEN1600
1108 ER=5:GOTO15000
1110 AU$(II,0)=CO$(KK):AU$(II,1)=LEFT$(C$,1)
1120 D$=CO$(KK+3):IFII<7ANDD$="AND"THENII=II+1:KK=KK+4
:GOTO1080
1125 IFII=7ANDD$="AND"THENER=23:GOTO15000
1130 IFD$<>")"THENER=6:GOTO15000
1140 KK=KK+4:IFCO$(KK)<>"KEY"THENER=8:GOTO15000
1150 KK=KK+1:IFCO$(KK)<>"("THENER=2:GOTO15000
1160 KK=KK+1:KE$=CO$(KK):A=0
1170 FORKL=0TOII:A=AOR(KE$=AU$(KL,0)):NEXT
1180 IFA=0THENER=9:GOTO15000
1190 KK=KK+1:IFCO$(KK)<>")"THENER=6:GOTO15000
1300 LE=0:IFCO$(KK+1)<>"LEN"THEN1340
1310 KK=KK+2:IFCO$(KK)<>"("THENER=2:GOTO15000
1320 KK=KK+1:LE=VAL(CO$(KK)):IFLE<1ORLE>254THENER=11
:GOTO15000
1330 KK=KK+1:IFCO$(KK)<>")"THENER=6:GOTO15000
1340 KK=KK+1:IFCO$(KK)<>";"THENER=10:GOTO15000
1350 CL$(I)=CL$
1360 FORK=0TOII:TP$(I,K)=AU$(K,1):TU$(I,K)=AU$(K,0)
:NEXT:IFLE=0THENLE=254
1365 LE%(I)=LE:K1$(I)=KE$
1370 FORK=0TOII:IFKE$=TU$(CL,K)ORLEFT$(TP$(CL,K)
,2)="ES"THEN1378
1373 K2$=LEFT$(CL$,3)+"."+TU$(CL,K)
1375 OPEN7,8,7,K2$+",S,W":PRINT#7,"*":CLOSE7
1378 NEXT
1380 OPEN1,8,15:OPEN4,8,4,CL$+",L,"+CHR$(LE)
1390 INPUT#1,A,A$,B,C:CLOSE1:CLOSE4
1500 RETURN
1600 IFCO$(KK+3)<>"IN"THENER=7:GOTO15000
1605 C$=C$+" ":AU$(II,1)=LEFT$(C$,1)+MID$(C$,7,1)
1610 AU$(II,0)=CO$(KK):AU$(II,1)=AU$(II,1)+CO$(KK+4)
1620 KK=KK+2:GOTO1120
2000 GOSUB2500:IFER>0THEN15000
2003 IFKE$="TRUE"THEN2400
2005 IFCO$(4)<>"="THENER=20
2010 IFER>0THEN15000
2015 K=5:KE$=CO$(5)
2020 IFK>49THENER=10:GOTO15000
2022 IFCO$(K+1)<>";"THENK=K+1:KE$=KE$+" "+CO$(K)
:GOTO2020
2030 K=0:IFFA=0THEN2100
2040 IFKE$(CL,K)=KE$THEN2070
2050 IFKE$(CL,K)<>"*"THENK=K+1:GOTO2040
2060 ER=9:GOTO15000
2070 GOSUB3700:A=CL:GOSUB3500:TS$="KEY"
2080 RETURN
2100 AA=0:K$=CO$(3):M$=LEFT$(CL$,3)+".*K$:KE$="*"+KE$
2110 OPEN3,8,3,M$+",S,R":OPEN1,8,15
2120 INPUT#1,A1,A1$,B1,B1:IFA1<>0THENCLOSE3:CLOSE1
:ER=18:GOTO15000
2130 INPUT#3,N$:IFN$<>KE$ANDN$<>"*"THEN2130
2140 SL=0:IFN$="*"THENER=9:KE$=MID$(KE$,2):GOTO15000
2150 INPUT#3,N$:A2$(SL)=N$:IFLEFT$(N$,1)<
>"*"THENSL=SL+1:GOTO2150
2160 A2$(SL)="*":CLOSE3:CLOSE1
2170 N$=A2$(0):IFN$="*"THENER=17:TS$="":GOTO15000
2180 K=VAL(N$):SL=1
2190 N$=A2$(SL):A2$(SL-1)=N$:IFN$<>"*"THENSL=SL+1
:GOTO2190
2200 GOSUB3700:A=CL:GOSUB3500:TS$="ALTRI":RETURN
2400 K=0
2405 IFKE$(CL,K)="-"THENK=K+1:GOTO2405
2410 IFKE$(CL,K)<>"*"THENGOSUB3700:A=CL:GOSUB3500:K=K+1
:TS$="TRUE":RETURN
2420 ER=17:TS$="":GOTO15000
2500 GOSUB4700:IFER>0THENRETURN
2510 CL=J:FA=0:KE$=""
2515 IFCO$(2)=";"THENKE$="TRUE":RETURN
2520 IFCO$(2)<>"WITH"THENER=16:RETURN
2530 KE$=CO$(3):IFKE$=K1$(CL)THENFA=-1
2560 RETURN
3000 IFTS$="EDIT"THENTS$=""
3005 IFTS$=""THENER=19:GOTO15000
3010 IFTS$="ALTRI"THEN2170
3020 IFTS$="KEY"THENER=17:TS$="":GOTO15000
3030 IFTS$="TRUE"THEN2405
3040 IFTS$="DEF"THEN12005
3100 GOSUB2000
3102 IFPPTHENOPEN6,4,7:CMD6
3105 IFTS$<>""THENPRINT"-------------------------------
---":PRINT
3107 IFPPTHENPRINT#6:CLOSE6
3110 IFTS$="ALTRI"THENGOSUB2170:GOTO3102
3120 IFTS$="TRUE"THENGOSUB2405:GOTO3102
3140 RETURN
3500 IFPPTHENOPEN6,4,7:CMD6
3501 KK=0:AA$="":CC$=CHR$(13):J=A:GOSUB4800:A=J:J=AA
3505 AA$=""
3510 C$=LEFT$(A$,1):IFC$<>CC$THENAA$=AA$+C$:A$=MID$(A$
,2):GOTO3510
3515 IFLEFT$(TP$(A,KK),1)<>"E"THEN3517
3516 PRINT:PRINTTU$(A,KK):PRINT:GOSUB3600:KK=KK+1
:GOTO3530
3517 IFTP$(A,KK)="P"THENGOSUB4400:KK=KK+1:GOTO3530
3520 PRINTLEFT$(TU$(A,KK)+"                    ",J)": "AA$
:KK=KK+1
3530 A$=MID$(A$,2):IFTU$(A,KK)<>""THEN3505
3540 PRINT:IFPPTHENPRINT#6:CLOSE6
3545 RETURN
3600 LC$=MID$(TP$(A,KK),3):AZ=0:FW=0
3610 FORFK=0TO7:IFLC$=CL$(FK)THENAZ=1:AW=FK
3620 NEXT:IFAZ=0THENRETURN
3625 FW=FW+1:FW$=MID$(AA$,FW,1):IFFW$<>""ANDFW$<
>"|"THEN3625
3630 FJ$=LEFT$(AA$,FW-1):AA$=MID$(AA$,FW+1):FW=0
3635 FK=0
3640 AZ$=KE$(AW,FK):IFAZ$<>"*"ANDAZ$<>FJ$THENFK=FK+1
```

(continua a pag. 108)

(segue da pagina 107)

```
:GOTO3640
3645 IFA2$<>FJ$THEN3670
3648 OPEN1,8,15:OPEN4,8,4,LC$
3650 PRINT#1,"P"CHR$(4)CHR$(FK+1)CHR$(0)CHR$(1)
3652 A9=0:A9$=""
3654 IFTU$(AW,A9)=""THEN3657
3655 INPUT#4,C9$:A9$=A9$+C9$:IFRIGHT$(A9$
,1)="|"THEN3655
3656 A9$=A9$+CC$:A9=A9+1:GOTO3654
3657 CLOSE1:CLOSE4:IFPPTHENCMD6
3658 KQ=0:AQ$="":CC$=CHR$(13):JQ=A:JW=J:J=AW:GOSUB4800
:A=JQ:JQ=AA:J=JW
3660 AQ$=""
3662 C0$=LEFT$(A9$,1):IFC0$<>CC$THENAQ$=AQ$+C0$
:A9$=MID$(A9$,2):GOTO3662
3665 IFJ>JQTHENJQ=J
3666 IFTP$(AW,KQ)="P"THEN3668
3667 PRINTLEFT$(TU$(AW,KQ)+"                    ",JQ)": "AQ$
3668 KQ=KQ+1:A9$=MID$(A9$,2):IFTU$(AW,KQ)<
>""THEN3660RETURN
3670 PRINT:IFAA$<>""THEN3625
3675 RETURN
3700 OPEN1,8,15:OPEN4,8,4,CL$
3710 PRINT#1,"P"CHR$(4)CHR$(K+1)CHR$(0)CHR$(1)
3720 A=0:A$=""
3730 IFTU$(CL,A)=""THEN3740
3733 INPUT#4,C$:A$=A$+C$:IFRIGHT$(A$,1)="|"THEN3733
3735 A$=A$+CC$:A=A+1:GOTO3730
3740 CLOSE1:CLOSE4:RETURN
4000 GOSUB4700:CL=J:JR=0
4003 IFER>=0THEN15000
4005 FORE1=0TO7:FORE2=0TO1:AU$(E1,E2)="":NEXT:NEXT
4006 KK$=KE$(CL,JR):IFKK$<>"-"ANDKK$<>"*"THENJR=JR+1
:GOTO4006
4007 IFJR=60THENER=22:GOTO15000
4010 GOSUB4800
4030 KL=0:B$=""
4040 IFTS$="EDIT"THEN9200
4042 A$="":E$=TU$(J,KL):PRINTLEFT$(E$+"                ",AA);
:INPUTA$:IFA$=""THEN4040
4045 IFE$=K1$(J)THENKE$=A$
4050 IFLEFT$(TP$(J,KL),2)="ES"THENGOSUB 4600
4054 IFTP$(J,KL)="P"ANDA$<>"Y"ANDA$<>"N"THENER=14
:GOSUB15000:GOTO4040
4056 IFTP$(J,KL)="P"ANDA$="Y"THENGOSUB4500
4070 IFTP$(J,KL)<>"I"THEN4110
4080 C$=STR$(VAL(A$)):C=(A$=C$)OR(A$=MID$(C$,2))
4090 IFC=0THENER=14:GOSUB15000:GOTO4040
4110 B$=B$+A$+CHR$(13):AU$(KL,0)=E$:AU$(KL,1)=A$
:KL=KL+1
4120 IFKL<8THENIFTU$(J,KL)<>""THEN4040
4130 IFMA=1ANDMC$<>CL$THENGOSUB8000
4140 IFMA=0ORMC$<>CL$THENMA=1:MC$=CL$:MX=CL:GOSUB8500
4150 JK=0:IFLEN(B$)>LE%(CL)THENER=24:PRINT:GOTO15000
4160 KK$=KE$(CL,JK):IFKE$<>KK$ANDKK$<>"*"THENJK=JK+1
:GOTO4160
4165 IFKE$=KK$THENER=15:PRINT:GOTO15000
4170 JK=JR
4180 N$=KE$(CL,JK):KE$(CL,JK)=KE$:IFN$<>"-"THENKE$(CL
,JK+1)="*"
4190 OPEN1,8,15:OPEN4,8,4,CL$
4200 PRINT#1,"P"CHR$(4)CHR$(JK+1)CHR$(0)CHR$(1)
4210 PRINT#4,B$:INPUT#1,A,A$,A,A:CLOSE4:CLOSE1
4220 FORKL=0TO7
4230 IFAU$(KL,0)=K1$(CL)THEN 4290
4240 IFAU$(KL,0)="" THEN 4290
4250 IFLEFT$(TP$(CL,KL),2)="ES"THEN 4290
4260 JL=0
4270 IFA1$(KL,JL)<>""THENJL=JL+1:GOTO4270
4280 A1$(KL,JL)=AU$(KL,1)+"|"+STR$(JK)
4290 NEXT
4300 RETURN
4400 IFAA$="N"ORPP=1THENRETURN
4405 PRINT:PRINTLEFT$(TU$(A,KK)+"                    ",J)": "
:PRINT
4410 OPEN3,8,3,"P."+LEFT$(CL$,3)+"."+KE$(CL,K)+",S,R"
4415 GETT$:GET#3,A$:IFST=64ORT$<>""THENCLOSE3:RETURN
4420 POKEPEEK(209)+PEEK(211)+PEEK(210)*256,ASC(A$)
:PRINT"■";  :GOTO4415
4500 PRINT"□";:OPEN9,0
4505 Z$="":INPUT#9,Z$:PRINT:IFLEFT$(Z$,1)<
>"@"ORLEN(Z$)<>1THEN4505
4510 CLOSE9:Z=1024:OPEN3,8,3,"@:P."+LEFT$(CL$
,3)+"."+KE$+",S,W"
4520 IFPEEK(Z)<>0THENPRINT#3,CHR$(PEEK(Z));:Z=Z+1
:GOTO4520
4530 CLOSE3:RETURN
4600 IFTS$="EDIT"THEN9300
4605 AA$="":PRINTLEFT$(E$+"                ",AA);
4610 INPUTAA$:IFAA$<>""THENA$=A$+"|"+CC$+AA$:GOTO4605
4620 RETURN
4700 ER=-1:FL=0:CL$=CO$(1):A=0:A1=0:IFCL$=";"THENER=12
:RETURN
4710 FORKL=0TO7:IFCL$=CL$(KL)THENA=1:J=KL
4720 NEXT:IFA=0THENER=13:RETURN
4730 RETURN
4800 A=0:FORKL=0TO7:AA=LEN(TU$(J,KL)+" ")
:IFA<AATHENA=AA
4810 NEXT:AA=A:RETURN
5000 GOSUB4700
5010 CL=J:KE$="":K=0
5015 IFCO$(4)<>"="THENER=20
5020 IFCO$(2)<>"WITH"THENER=16
5025 IFER>=0THEN15000
5030 KE$=CO$(3):IFKE$<>K1$(CL)THENER=21:GOTO15000
5040 KE$=CO$(5):K2=5
5042 IFK2>49THENER=10:GOTO15000
5044 IFCO$(K2+1)<>";"THENK2=K2+1:KE$=KE$+" "+CO$(K2)
:GOTO5042
5050 N$=KE$(CL,K):IFN$<>KE$ANDN$<>"*"THENK=K+1:GOTO5050
5060 IFN$="*"THENER=9:GOTO15000
5065 OPEN1,8,15,"S:P."+LEFT$(CL$,3)+"."+KE$:CLOSE1
5070 KE$(CL,K)="-":K$=MID$(STR$(K),2)
5080 FORKL=0TO7:M$=LEFT$(CL$,3)+"."+TU$(CL,KL)
5090 OPEN3,8,3,M$+",S,R":OPEN1,8,15:JL=0
5100 INPUT#1,A1,A1$,B1,B1:IFA1<>0THENCLOSE1:CLOSE3
:GOTO5150
5110 INPUT#3,N$:IFN$=K$THEN5110
5120 A2$(JL)=N$:IFN$<>"*"THENJL=JL+1:GOTO5110
5130 CLOSE1:CLOSE3:OPEN3,8,3,"@:"+M$+",S,W":JL=0
5140 PRINT#3,A2$(JL):IFA2$(JL)<>"*"THENJL=JL+1:GOTO5140
5150 CLOSE3:NEXT:RETURN
6000 I=0:C$=CHR$(13)
6010 IFCL$(I)<>""THENI=I+1:GOTO6010
6020 I=I-1:OPEN3,8,8,"@:SERVIZIO,S,W"
6030 PRINT#3,STR$(I)+C$;
6040 FORK=0TOI
6050 PRINT#3,CL$(K)+C$;
6060 PRINT#3,K1$(K)+C$;
6070 PRINT#3,STR$(LE%(K))+C$;
6090 FORJ=0TO7
6090 PRINT#3,TP$(K,J)+"*"+C$+TU$(K,J)+"*"+C$;:NEXT
6100 SL=0
6110 A$=KE$(K,SL):PRINT#3,A$+"*"+C$;:IFA$<
>"*"THENSL=SL+1:GOTO6110
6120 NEXT:CLOSE3:CLOSE10:PRINTCHR$(9):END
7000 OPEN3,8,8,"SERVIZIO,S,R"
7020 OPEN1,8,15:INPUT#1,A,A$,B,C:IFA<>0THENCLOSE1
:CLOSE3:RETURN
7030 INPUT#3,I$:I=VAL(I$)
7040 FORK=0TOI
7050 INPUT#3,CL$(K)
7060 INPUT#3,K1$(K)
7070 INPUT#3,S$:LE%(K)=VAL(S$)
7080 FORJ=0TO7
7090 INPUT#3,T$:TP$(K,J)=LEFT$(T$,LEN(T$)-1)
7095 INPUT#3,T$:TU$(K,J)=LEFT$(T$,LEN(T$)-1):NEXT
7100 SL=0
7110 INPUT#3,K$:KE$(K,SL)=LEFT$(K$,LEN(K$)-1):IFK$<
>"**"THENSL=SL+1:GOTO7110
7120 NEXT:CLOSE3:CLOSE1:RETURN
8000 FORKL=0TO7:IFA1$(KL,0)=""THEN 8200
8010 M$=LEFT$(MC$,3)+"."+TU$(MX,KL)
8020 OPEN3,8,3,M$+",S,R":JL=0
8025 INPUT#3,N$:A2$(JL)=N$:IFN$<>"*"THENJL=JL+1
:GOTO8025
8030 CLOSE3:JL=0
8040 N$=A1$(KL,JL):H=0:IFN$=""THEN 8120
8050 H=H+1:IFMID$(N$,H,1)<>"|"THEN8050
8060 N1$="*"+LEFT$(N$,H-1):N2$=MID$(N$,H+1)
8070 H=0
8080 IFA2$(H)<>N1$ANDA2$(H)<>"*"THENH=H+1:GOTO8080
8090 IFA2$(H)="*"THENA2$(H)=N1$:A2$(H+1)=N2$
:A2$(H+2)="*":JL=JL+1:GOTO8040
8100 FORII=120TOH+2STEP-1:A2$(II)=A2$(II-1):NEXT
8110 A2$(H+1)=N2$:JL=JL+1:GOTO8040
8120 OPEN3,8,3,"@:"+M$+",S,W":JL=0
8130 PRINT#3,A2$(JL):IFA2$(JL)<>"*"THENJL=JL+1:GOTO8130
8140 CLOSE3
8200 NEXT:RETURN
8500 FORE1=0TO7:FORE2=0TO60:A1$(E1,E2)="":NEXT:NEXT
:RETURN
9000 GOSUB4700
9010 CL=J:KE$="":K=0
9015 IFCO$(4)<>"="THENER=20
```

```
9020 IFCO$(2)<>"WITH"THENER=16
9025 IFER>=0THEN15000
9030 KE$=CO$(3):IFKE$<>K1$(CL)THENER=21:GOTO15000
9040 KE$=CO$(5):K2=5
9042 IFK2>49THENER=10:GOTO15000
9044 IFCO$(K2+1)<>";"THENK2=K2+1:KE$=KE$+" "+CO$(K2)
:GOTO9042
9050 N$=KE$(CL,K):IFN$<>KE$ANDN$<>"*"THENK=K+1:GOTO9050
9060 IFN$="*"THENER=9:GOTO15000
9061 GOSUB3700:A2$=A$
9065 OPEN1,8,15,"S:P."+LEFT$(CL$,3)+"."+KE$:CLOSE1
9070 KE$(CL,K)="-":K$=MID$(STR$(K),2)
9080 FORKL=0TO7:M$=LEFT$(CL$,3)+"."+TU$(CL,KL)
9090 OPEN3,8,3,M$+",S,R":OPEN1,8,15:JL=0
9100 INPUT#1,A1,A1$,B1,B1$:IFA1<>0THENCLOSE1:CLOSE3
:GOTO9150
9110 INPUT#3,N$:IFN$=K$THEN9110
9120 A2$(JL)=N$:IFN$<>"*"THENJL=JL+1:GOTO9110
9130 CLOSE1:CLOSE3:OPEN3,8,3,"@:"+M$+",S,W":JL=0
9140 PRINT#3,A2$(JL):IFA2$(JL)<>"*"THENJL=JL+1:GOTO9140
9150 CLOSE3:NEXT
9160 TS$="EDIT":GOTO4005
9200 V$=""
9210 N2$=LEFT$(A2$,1)
9215 IFN2$<>CC$ANDN2$<>"|"THENV$=V$+N2$:A2$=MID$(A2$,2)
:GOTO9210
9220 A2$=MID$(A2$,2):V=LEN(V$)+2
9230 A$="":E$=TU$(J,KL):PRINTLEFT$(E$+"                    "
,AA+2);V$;LEFT$(BS$,V);
9240 INPUTA$:IFA$=""THEN9240
9245 GOTO4045
9300 V$="":IFN2$<>"|"THEN4605
9310 V$=""
9320 N2$=LEFT$(A2$,1)
9325 IFN2$<>CC$ANDN2$<>"|"THENV$=V$+N2$:A2$=MID$(A2$,2)
:GOTO9320
9330 A2$=MID$(A2$,2):V=LEN(V$)+2
9340 PRINTLEFT$(E$+"                    "
,AA+2);V$;LEFT$(BS$,V);
9350 INPUTAA$:IFAA$<>""THENA$=A$+"|"+AA$:GOTO9300
9360 RETURN
12000 CL=0
12005 IFCL$(CL)=""THENER=17:TS$="":GOTO15000
12008 IFPPTHENOPEN6,4,7:CMD6
12010 PRINT:PRINT"-:CLASS ";CL$(CL);" <-> (":A1=0
12020 A$=TP$(CL,A1)
12021 IFLEFT$(A$,1)="E"THENA$="EXTKEY"+MID$(A$,2
,1)+" IN "+MID$(A$,3)
12022 IFA$="S"THENA$="STRING"
12023 IFA$="I"THENA$="INT"
12025 IFA$="P"THENA$="PAGE"
12030 PRINT"  "TU$(CL,A1)"+"A$;
12040 A1=A1+1:IFA1<8THENIFTU$(CL,A1)<>""THENPRINT" AND"
:GOTO12020
12050 PRINT")":PRINT"  KEY("K1$(CL)")";:A1=0
12100 PRINT"LEN("MID$(STR$(LEN(CL)),2)")";"
12200 CL=CL+1:TS$="DEF":IFPPTHENPRINT#6:CLOSE6
12210 RETURN
15000 CLOSE1:CLOSE3:CLOSE6
15005 IFER=0THENPRINTF$"CLASS EXISTS"
15010 IFER=1THENPRINTF$MI$"(-)"
15020 IFER=2THENPRINTF$MI$"("
15030 IFER=3THENPRINTF$"TOO MANY CLASSES"
15040 IFER=4THENPRINTF$MI$"+"
15050 IFER=5THENPRINTF$+C$+" = ILLEGAL TYPE"
15060 IFER=6THENPRINTF$MI$")"
15070 IFER=7THENPRINTF$MI$"'IN' NEAR EXTKEY"
15080 IFER=8THENPRINTF$MI$"KEY"
15090 IFER=9THENPRINTF$+KE$+" = UNKNOWN KEY"
15100 IFER=10THENPRINTF$"SYNTAX ERROR"
15110 IFER=11THENPRINTF$;LE;" = INVALID LEN"
15120 IFER=12THENPRINTF$MI$"CLASS NAME"
15130 IFER=13THENPRINTF$"CLASS UNKNOWN"
15140 IFER=14THENPRINTF$"TYPE MISMATCH"
15150 IFER=15THENPRINTF$"KEY EXISTS"
15160 IFER=16THENPRINTF$MI$"WITH"
15170 IFER=17ANDAL=0THENPRINTF$"NEXT NOT FOUND"
15180 IFER=18THENPRINTF$"KEY NOT INDEXED"
15190 IFER=19THENPRINTF$"NEXT WITHOUT FIND"
15200 IFER=20THENPRINTF$MI$"="
15210 IFER=21THENPRINTF$"INVALID KEY"
15220 IFER=22THENPRINTF$"CLASS FULL"
15230 IFER=23THENPRINTF$"TOO MANY FIELDS"
15240 IFER=24THENPRINTF$"RECORD TOO LONG"
15999 RETURN
```

Definiamo una base

Per definire un base di dati in Galileo/J, dopo aver stabilito quali classi e quali associazioni tra dati utilizzare, occorre dichiarare le varie classi e, per ognuna di esse, la struttura degli elementi. Solo dopo si potrà procedere all'inserimento delle registrazioni che ci interessano.

Per dichiarare una classe si usa l'operatore class. Senza dilungarci in parole, cominciamo a fare un piccolo esempio: immaginiamo di dover memorizzare alcuni nominativi accompagnati da relativo recapito telefonico. Scriveremo così:

class Rubrica <—> (Nome ← string and Telefono ← int)

Rubrica è il nome della classe, racchiusa tra parentesi il tipo dell'ennupla utilizzata, indicando per ogni campo il tipo. In questo caso abbiamo usato un campo di tipo stringa (il nome) e (and) un campo di tipo intero (il telefono). L'operatore <—> serve solo come delimitatore tra nome della classe e ennupla di definizione. Oltre a questo dovremo scegliere un campo chiave (in questo caso la scelta è solo tra nome e telefono) tramite il quale sarà assicurato l'accesso univoco alla registrazione. Ciò implica (e il sistema avviserà con un messaggio l'eventuale violazione) che non si potranno inserire due registrazioni con uguale campo chiave. Nel nostro caso o sceglieremo di registrare nominativi tutti diversi o con diverso numero telefonico.

La chiave si specifica subito dopo la descrizione di ennupla, nell'esempio precedente, scegliendo come chiave il nominativo, la dichiarazione completa sarà:

class Rubrica <—> (Nome ← string and Telefono ← int) Key (Nome)

Abbiamo detto che tra parentesi si elencano i vari campi dell'ennupla, specificando per ognuno il tipo. Vediamo quali sono i tipi utilizzabili. Non ci soffermeremo sui tipi string e int che, come facile intuire, hanno un comportamento assai familiare. I campi string possono contenere stringhe di caratteri, i campi int, solo valori numerici. Chiaramente il tipo stringa, come in Basic, non permette di inserire stringhe molto lunghe, proprio per questo è disponibile anche il tipo page che offre ben 1000 posizioni (un'intera schermata) per annotare più roba. Un altro tipo di dato molto importante è l'extkey disponibile anche al plurale (extkeys). Tramite questo è possibile correlare dati di classi diverse o della stessa classe. Al singolare costruisce una associazione univoca, al plurale un'associazione multipla. Per rendere meglio il concetto, andiamo avanti con i nostri esempi: questa volta vogliamo costruire un indirizzario ottimizzato, specialmente per quanto riguarda le informazioni riguardo le città abitate dalle persone che memorizzeremo.

Come già più volte descritto, spesso capita che la ridondanza di alcune informazioni appesantisca notevolmente (e inutilmente) tutta la base di dati: occorre sempre modellare bene la struttura delle varie classi. Se ad esempio conosciamo trenta persone a Benevento, è inutile memorizzare trenta volte le informazioni riguardo la città (Nome, Cap, Prefisso). Meglio è memorizzare le città interessate in una classe, e gli indirizzi comprensivi di numero telefonico in un'altra. Un'opportuna associazione, come vedremo, ricostruirà l'indirizzo completo ad ogni interrogazione della base. La prima delle due classi sarà definita così:

class Città <—> (Località ← string and Sigla ← string and CAP ← string and Prefisso ← string) Key (Sigla)

Sigla identifica univocamente la località: quando inseriremo gli elementi in questa classe, indicheremo per ogni città la sigla automobilistica se questa è capoluogo di provincia, due o tre caratteri del nome altrimenti. L'importante è usare una codifica univoca.

La classe delle persone sarà definita così:

class Amici <—> (NomeCognome ← string and Recapito ← string and Residenza ← extkey in Città and Telefono ← int and Varie ← page) Key (NomeCognome)

Amici è il nome della classe, Residenza il puntatore (chiave esterna) nella classe Città, Varie una paginata supplementare (fa-

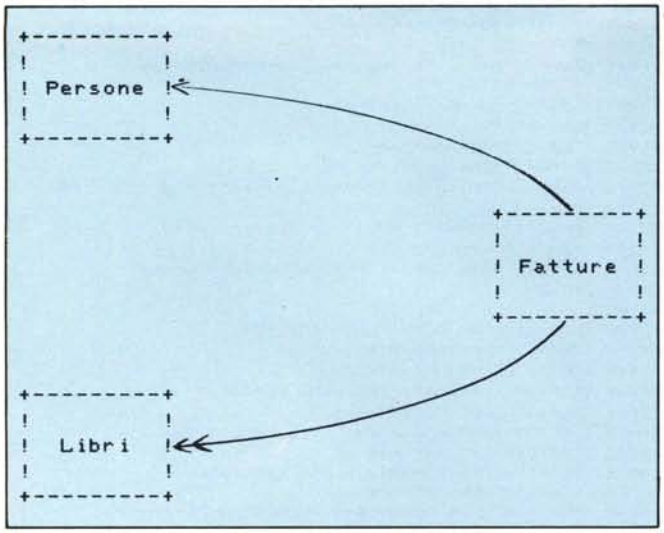

Figura 1 - In Galileo/J le associazioni si rappresentano graficamente tramite archi orientati (semplici o doppi).

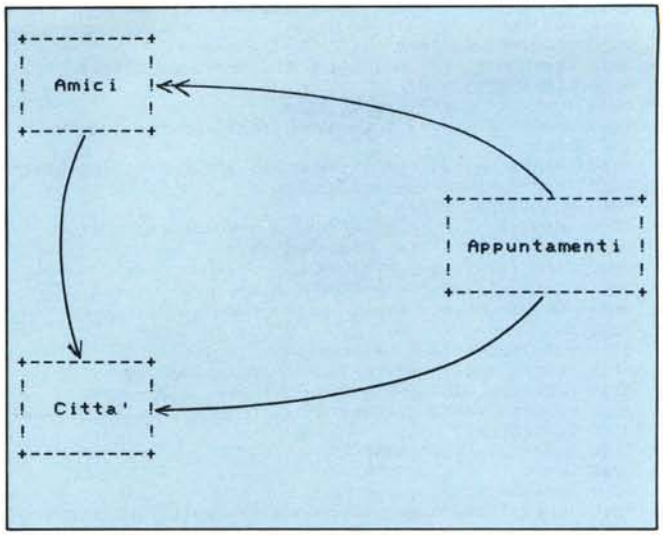

Figura 2 - Coi meccanismi propri del Galileo/J è possibile anche formare strutture a rete.

coltativa) per introdurre note sull'individuo da noi "schedato".

Inseriamo una città: il comando è make, seguito dal nome della classe da incrementare. Quindi:

```
make città
```

alle domande Località, Sigla, Cap e Prefisso, risponderemo secondo le nostre intenzioni: inseriremo la città nel nostro primo indirizzo. Con:

```
make Amici
```

potremo introdurre il resto.

Vi rimandiamo all'apposito paragrafo per chiarirvi maggiormente le idee; in questa sede, staremo ancora lontani dai particolari per non confondervi troppo.

Foto 1

Foto 2

Vediamo ora l'uso del tipo extkeys per costruire associazioni multiple. In questo caso associeremo ad ogni elemento di una classe, più elementi di un'altra classe, o della stessa. Immaginiamo di aver inserito nominativi e città delle persone che conosciamo. Per curare le nostre public relation costruiamo la classe Appuntamenti, nella quale registreremo per ognuno di essi una data, un luogo (la città) e le persone che dovremo incontrare. Inutile dire che sfrutteremo al massimo l'informazione già presente: ci limiteremo a definire la nostra nuova classe così:

```
class Appuntamenti <—> (Data ← string
and Luogo ← extkey in Città and Persone ←
extkeys in Amici) Key (Data)
```

per semplicità assumiamo di non avere mai più di un appuntamento al giorno.

Quando dovremo inserire un appuntamento, il sistema ci chiederà solo la data, la sigla del luogo e una sequenza di nominativi.

Al momento della ricerca, ossia quando saremo interessati a un determinato appuntamento, dando al sistema la data relativa, otterremo una marea di informazioni: luogo, prefisso e CAP della sede d'appuntamento; nome, cognome, indirizzo, residenza e telefono di tutte le persone partecipanti.

Il Galileo/J e il 64

Dopo aver descritto il Gaileo/J in generale, vediamo come questo sia stato implementato sul Commodore 64. Per intenderci meglio, caricate il programma e mandatelo in esecuzione: potremo così darvi passo-passo le istruzioni per costruire la nostra base di dati. Preferibilmente lasciate sul dischetto solo il programma Galileo/J, in quanto lo spazio-disco richiesto per ogni applicazione non è indifferente.

Per prima cosa occorre familiarizzare con l'interfaccia utente: i meccanismi del programma che permettono l'interazione uomo-macchina. Dato che il Galileo/J è scritto interamente in Basic, tale interfaccia è (ovviamente) realizzata con delle istruzioni di INPUT per i comandi da impartire, e da istruzioni di PRINT per gli output su video e stampante.

Subito dopo il Run, appare in alto a sinistra una E seguita dai due punti e dal cursore lampeggiante. Il sistema è pronto per eseguire un comando: corrisponde più o meno al Ready del Basic.

Anche se non vediamo punti interrogativi, occorre ricordare che siamo sempre in ambiente di INPUT (nel senso Basic che conosciamo) e ciò comporta alcune limitazioni, imposte dal 64: non è possibile digitare input per più di due linee di schermo e non è disponibile un editor sofisticato. Per il primo problema nessuna preoccupazione: se il nostro comando supera le due linee possiamo battere [RETURN] per disporre di due nuove linee e, eventualmente, anche ripetere il trucchetto quante volte si vuole. Chiaramente occorre non spezzare in due una parola, ma agire sul [RETURN] solo in sostituzione di uno spazio. A proposito di questi, sono obbligatori tra un parola e l'altra tranne il caso in cui sia presente un altro carattere separatore: ")", "—", "(", "=". L'editor di schermo è limitato alla linea che stiamo introducendo. Come terminatore di linea si usa il carattere ";" (punto e virgola): serve per far capire al sistema che il comando è stato introdotto e può essere eseguito. Sembra difficile, ma non lo è: basta non farsi prendere dal panico.

Proviamo a definire l'associazione Indirizzi-città già ampiamente discussa in quest'articolo e nei precedenti. Dobbiamo dichiarare le due classi: tenendo sott'occhio la foto 1 digitate la prima linea del comando cioè "class Amici <—> (". Potremmo continuare a digitare sulla stessa linea, ma per questioni di eleganza battiamo [RETURN] e continuiamo sulla riga successiva definendo la struttura dei vari campi, come mostrato nella foto 1, ricordandovi di premere [RETURN] per passare a nuova linea.

La specifica len (120) serve per dimensionare lo spazio occupato dal file Relativo

che utilizzeremo; 120 indica la lunghezza massima di ogni registrazione. Se nulla è specificato si assume la massima possibile: 254 caratteri per registrazione. Ricordatevi, comunque, che è sempre meglio abbondare.

Al termine il punto e virgola terminerà il comando: se tutto è andato per il verso giusto sentirete il drive ruotare; se qualcosa non va sarà segnalato un fallimento dell'operazione, con relativa causa.

Quando il sistema è pronto per un nuovo comando appare nuovamente la E seguita dai due punti. Introduciamo (fedelmente) la definizione della seconda classe, come mostrato nella foto 2, seguendo gli stessi accorgimenti di prima. Al termine il drive inizializzerà questo nuovo file relativo. Da questo momento in poi, ad ogni comando "make Amici" potremo inserire

Foto 3

Foto 4

Foto 5

nome e indirizzo di un nostro conoscente; digitando "make Città" possiamo inserire le città dei nostri conoscenti. Nelle foto 3, 4 e 5, è mostrato l'inserimento di un elemento in ciascuna classe e un esempio di comando di ricerca.

Prima di continuare con la descrizione dei comandi disponibili, fermiamoci un attimo per alcune considerazioni. Il tipo page, come detto in precedenza, permette di disporre di una intera paginata per introdurre informazioni circa la registrazione che stiamo memorizzando. Nel caso nostro, se alla domanda "Varie?" rispondiamo y vedremo ripulire il video: questa è la nostra paginata libera. Possiamo scrivere quel che vogliamo: al termine basterà digitare la sequenza [RETURN] [RETURN]. <Chiocciolina>, per intenderci, è il carattere del tasto tra la P e l'asterisco della tastiera del 64.

Come vedremo anche in seguito, il tipo page ha alcune limitazioni. Ma continuiamo ora con le considerazioni di carattere generale.

Altro punto da tenere sempre ben presente, è il fatto che tutte le parole chiave del Galileo/J sono scritte in minuscolo, mentre per quanto riguarda nomi di campi e classi sono disponibili sia le minuscole che le maiuscole. Non è possibile usare lo spazio all'interno di un nome (di una classe o di un campo), ma bisogna saldare eventuali doppie denominazioni: nella nostra prima classe definita, abbiamo scritto NomeCognome e non Nome Cognome. Chiaramente ciò non vale per il contenuto delle registrazioni: come è visibile nella 3ª foto, abbiamo scritto Vincenzo Tuccillo e non VincenzoTuccillo (tutto attaccato).

I comandi del Galileo/J

Il primo comando visto è class e serve per definire la struttura di una classe dati; la sua sintassi è:

```
class NomeClasse <—> (Struttura)
Key (ChiavePrimaria) [len(LunghezzaElementi)];
```

tra parentesi quadre la specifica len, come visto opzionale. Il secondo comando costruisce elementi in una classe; la sua sintassi è semplicissima:

```
make NomeClasse
```

costruisce un elemento nella classe specificata, chiedendo i valori dei vari campi.

```
find NomeClasse with Attributo = Costante
```

trova il primo elemento che soddisfa la condizione specificata dopo il with. Attributo è il nome di un campo degli elementi della classe, Costante è un suo possibile valore. Ad esempio potremmo scrivere find Città with Prefisso = 081 o find Amici with Città = NA o roba simile.

Per trovare eventuali altri elementi che soddisfano la medesima condizione, basta usare il comando:

```
next;
```

se nessun altro elemento in memoria è trovato o se il next è dato prima di un find, un messaggio di fallimento avviserà l'operatore.

Per trovare in un sol colpo tutti gli elementi che soddisfano una condizione (ossia senza prima trovare il primo e poi chiedere gli altri a forza di next) è disponibile il comando all con identica sintassi del find. Scriveremo cioè qualcosa del tipo:

```
all NomeClasse with Attributo = Costante;
```

Sia il find che l'all hanno un particolare formato per scorrere tutti gli elementi di una classe: basta non digitare affatto la condizione, scrivendo solo find NomeClasse; o all NomeClasse;.

Per dirottare gli output su stampante si usa il comando

```
printer;
```

e resta attivo fino a quando non si digita il comando complementare:

```
display;
```

Le registrazioni trovate saranno stampate su carta, eccezion fatta per le informazioni di tipo page (ecco la prima limitazione).

Per conoscere la definizione delle varie classi (se dimentichiamo qualche nome o qualche attributo) è disponibile il comando:

```
def;
```

che mostra la prima classe. Come per il find, per conoscere le altre, si usa il comando next più volte. Il comando:

```
quit;
```

serve per terminare una sessione di lavoro: è obbligatorio digitarlo tutte le volte che si smette di usare Galileo/J. Salva su disco lo stato interno del sistema, in modo da ritrovarsi nello stesso punto la prossima volta. Se non si termina sempre con quit (siete avvisati, è una minaccia) la base di dati può facilmente andare in uno stato inconsistente e occorre ricominciare tutto daccapo.

Gli ultimi due comandi servono per togliere o modificare un elemento di una classe. In questi soli due casi è obbligatorio usare la chiave primaria per identificare l'oggetto. Scriveremo così:

```
destroy NomeClasse with Chiave = costante;
```

se vogliamo togliere un elemento o

```
edit NomeClasse with Chiave = costante;
```

se vogliamo editare una registrazione. I vari campi sono mostrati nuovamente su video e sarà sufficiente battere [RETURN] per conservare l'informazione o digitare la sostituzione opportuna. Unica eccezione per il campo page (seconda limitazione) che dovrà interamente essere reintrodotto.

Per ora ci fermiamo qui: sul prossimo numero faremo qualche esempio un po' più significativo (specialmente per la comprensione). Arrivederci. MC

Foto 6 - Ogni sessione di lavoro deve terminare col comando quit.

Questo programma è disponibile su disco presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 155.