8. INDEXAREA SI SORTAREA FISIERELOR

8.1.SORTARE SI INDEXARE

Crearea de noi inregistrari intr-un fisier se face la sfirsit (APPEND), sau prin inserari intre cele existente (INSERT), fara a tine in general cont de continutul cimpurilor inregistrarii. Se admite si concatenarea unor fisiere total sau partial prin:

APPEND FROM fis [FOR cond] [TYPE tip_fis]

Inregistrarile fisierului specificat, care indeplinesc conditia, sint adaugate la fisierul curent deschis. Daca fisierul nu este de tip DBF, se va specifica si tipul sau: SDF - ASCII, DIF - Visicalc, SYLK - Multiplan, WSK - Lotus, FW2, - Framework, RPD - Rapid file.

Pentru a realiza diferite situatii si rapoarte, articolele trebuie sa fie ordonate in fisier dupa anumite criterii: pe sectii, facultati, an grupa, cod de material sau piesa, conturi, cod salariat, etc.

Ordonarea completa sau partiala a unui fisier dupa unul sau mai multe cimpuri (fractiuni de cimp), se realizeaza prin comanda SORT care, genereaza un nou fisier sortat, cel curent raminind nemodificat.

SORT [domeniu] TO fis_sortat ON cimp1 [/A][/C][/D],
cimp2 [/A].....[FOR cond][ASCEN/DESC]

Sortarea in ordinea cheilor se face:
/A - crescator tinind cont de codul ASCII pentru siruri de caractere
/C - fara deosebire intre literele mari si mici
/D - ordine descrescatoare
ASCE/DESC -ordinea de sortare este aceeasi pentru toate cheile

USE STUD - deschide fisierul nesortat
LIST - listare fisier nesortat
SORT TO STUD_S ON NUME - sortare dupa nume
USE STUD-S - deschidere fisier sortat
LIST - afisare fisier sortat
Numarul maxim de chei este 10 si nu pot fi de tip logic,sau memo.

Sortarea necesita inca 2 fisiere de manevra si spatiu pentru fisierul sortat si nesortat pe disc. Pentru a putea parcurge fisierul in ordine dupa chei diferite se foloseste indexarea care este mai rapida si mai economica sub aspectul spatiului.

Un fisier poate fi indexat dupa maxim 47 de chei, care pot fi combinatii de cimpuri. Pentru fiecare cheie se creaza un fisier index (.NDX), care contine valorile sortate ale cimpului si numarul de ordine al inregistrarii corespunzatoare din fisier.

8.2. METODE DE INDEXARE

Pentru regãsirea rapidã în acces direct a informațiilor din fişierele unei baze de date (BD), pe baza unei chei simbolice, se folosesc metode de indexare sau randomizare a cheii. În majoritatea sistemelor de gestiune a bazelor de date (SGBD) se utilizează metode de indexare care s-au perfecționat mult. Indexarea multinivel permite o regãsire a unei înregistrări cu valoare datã a cheii prin 3-4 poziționări pe disc. Pentru un timp de acces disc care este în prezent sub 20 ms, aceasta înseamnã regãsirea unei înregistrări dintr-o BD cu zeci de mii de articole în mai puțin de 0,1 secunde. Avantajele indexării față de utilizarea hashingului constau în :

- posibilitatea citirii ordonate a înregistrãrilor, secvențial în ordinea cheii de indexare;
- permite accesul direct la o înregistrare şi parcurgerea în continuare a fişierului, secvențial din acel punct, pentru a citi înregistrări care au egalã o parte a cheii (studenți din aceeaşi secție şi an);
- posibilitatea creării mai multor fişiere index, pentru acelaşi fişier de date dupã diferite câmpuri care sunt chei secundare (există mai multe înregistrări cu aceeaşi valoare a câmpului index).

Pentru regãsirea informațiilor în acces direct, performanțele tuturor metodelor de indexare sunt bune, dar indexarea are şi dezavantaje care trebuie menționate:

- utilizarea fişierelor index multinivel (cele mai des folosite), consumã mult timp pentru actualizarea fişierelor index la adãugarea de noi înregistrări în BD mari, care limitează utilizarea lor mai ales când există mai multe fişiere index pentru un fişier de date;
- spațiul disc consumat de fişierele index este de acelaşi ordin de mãrime cu cel al fişierelor de date şi mult mai mare decât cel utilizat de directoarele de hashing.

Primul dezavantaj se eliminã prin utilizarea unor algoritmi de indexare dinamicã cum sunt cei cu arbori echilibrați B^+ cu încãrcare incompletã a nodurilor, care se vor prezenta în continuare. Această metodã complicã procedurile de cãutare şi actualizare a fişierului index. Spațiul disc folosit pentru fişierele index poate fi redus funcție de aplicație prin utilizarea unor structuri de BD care folosesc liste înlãnțuite cu pointeri sau fişiere de legãturi. Se eliminã în acest caz spațiul ocupat de cheie în înregistrarea de index.

8.2.1. Indexarea multinivel nedensã dupã cheia primarã

Fişierele secvențial indexate clasice au fost realizate pe un fişier secvențial sortat, format din blocuri grupate pe cilindri. Pentru fiecare cilindru s-a creat o tabelă index înregistrări pe cilindru formată din câte o intrare pentru fiecare bloc de date. O înregistrare de index conține cheia primară şi adresa blocului la care se referă. Tabela este plasată în unul sau mai multe blocuri la sfârşitul cilindrului (fig.5.1).

Pentru întregul fişier disc (volum) se realizează o tabelă index cilindri pe fişier, având câte o intrare pentru fiecare cilindru. Intrarea conține cheia maximã pe cilindru şi adresa tabelei index a cilindrului. Dacã o tabelă de index conține mai multe blocuri, se va crea o tabelă index rezumat urmatã de blocurile tabelei detaliu. O intrare din tabela rezumat conține cheia maximã dintr-un bloc detaliu şi adresa blocului respectiv. Tabela rezumat este separatã de tabela detaliu prin unul sau mai multe blocuri libere. La consultare se citeşte tabela rezumat şi se determinã blocul din tabela detaliu care conține cheia cãutatã. Acesta se poate citi în aceeaşi turã de pistã prin folosirea blocurilor libere, ce vor fi parcurse pe timpul cãutării în tabela rezumat. Citirea tabelei cilindru şi a blocului ce conține înregistrarea cu cheia cãutatã se va face printr-o singurã poziționare pe disc, deoarece ele se gãsesc pe acelaşi cilindru.

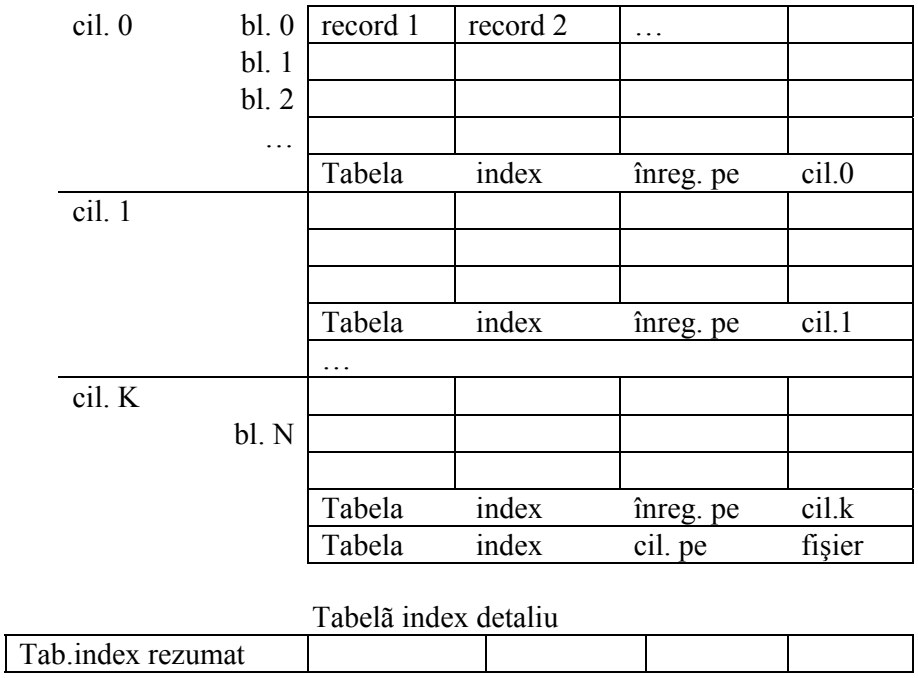


Fig.5.1. Indexare multinivel nedensã

Metoda foloseşte indexarea nedensã, deoarece numãrul de chei din tabela index este mult mai mic decât numãrul de chei din fişierul de date. Spaţiul ocupat de index este de cca 10 % din spaţiul ocupat de fişierul de date. Aceasta duce la un acces rapid la înregistrãri cu maxim 2 accese disc. Metoda a fost totuşi abandonatã, deoarece adãugãrile ulterioare de înregistrãri se fac într-o zonã de depãşire unde se înlãnţuie cu pointer în înregistrarea precedentã existentã în zona principalã a fişierului. Dupã crearea fişierului, tabelele index rãmân nemodificate, nu se admit valori duble pentru chei (cheie primarã).

8.2.2. Fişiere index dense multinivel

SGBD relaţionale folosesc fişiere index "dense" aplicate pe fişiere de date neordonate. Se pot construi fişiere index pentru mai multe câmpuri din fişier, fãrã ca acestea sã fie câmpuri cheie (chei secundare). Fişierul index va conţine câte o intrare pentru fiecare înregistrare din fişier. Intrarea de index este de forma (K_i, P_i) unde K_i este valoarea cheii, iar P_i un pointer ce indicã adresa înregistrãrii ce conţine cheia în fişierul de date. Adresa este relativã în fişier şi poate fi:

- adresa în octeţi a înregistrãrii în fişier;
- adresa sector început bloc plus adresa octet în bloc a înregistrãrii (.NDX);
- numãrul înregistrãrii în fişier (fişiere .MDX în dBASE).

Ultima formã ocupã doar 4 octeţi şi presupune înregistrãri cu lungime fixã, condiţie impusã în BD relaţionale. Tabela index obţinutã se sortează în ordinea valorii cheilor şi face referiri la înregistrãrile din fişierul de date prin pointer.

Dacã tabela index conţine mai mult de un bloc se creazã o tabelã rezumat de nivel 2 (fig.1), în care pentru fiecare bloc din nivelul 1 se creazã o intrare ce conţine cheia maximã din acel bloc şi numãrul sectorului unde începe blocul. Fiecare bloc din tabela de nivel 2 se plasează dupã blocurile de nivel 1, la care se referã. În acelaşi mod se formează tabele index de nivel 3, ş.a.m.d.

Căutarea unei înregistrări pentru o cheie datã presupune citirea a câte un bloc din fiecare nivel, plus blocul din fişierul de date. Căutarea unei chei în blocurile index se face rapid în memoria centralã şi timpul se neglijeazã. Pentru un fişier de 30 de mii de articole cu lungimea înregistrării index de 16 octeti (12 cheie +4 adresã) rezultã 3 nivele de index, deci 4 poziționări pe disc. Timpul de acces este sub 0,1 secunde pentru hard-discuri cu timp de acces sub 20 ms.

Fişierele index multinivel necesitã mult timp pentru actualizare când sunt de mari dimensiuni. Adãugarea unei noi chei duce la o reorganizare a tabelelor index, prin inserare într-un tabel secvențial ordonat. Pentru a pãstra consistența BD pentru orice actualizare în fişierul de date, se actualizeazã toate fişierele index asociate, dacã în dBASE se utilizeazã fişiere .MDX.

La folosirea fişierelor de tip NDX, este posibil ca la adãugări şi ştergeri numeroase de înregistrări fişierele index sã nu fie deschise. Refacerea fişierelor index se va face la sfirşitul lucrării prin reindexare şi nu dupã fiecare înregistrare adaugatã. Dacã se folosesc grupări de fişiere index de tip MDX sau CDX, la deschiderea fişierului de date se deschid automat toate fişierele index asociate. Dupã orice actualizare a unei înregistrări din fişierului de date care afectează un câmp cheie, se restructurează întregul fişier index.

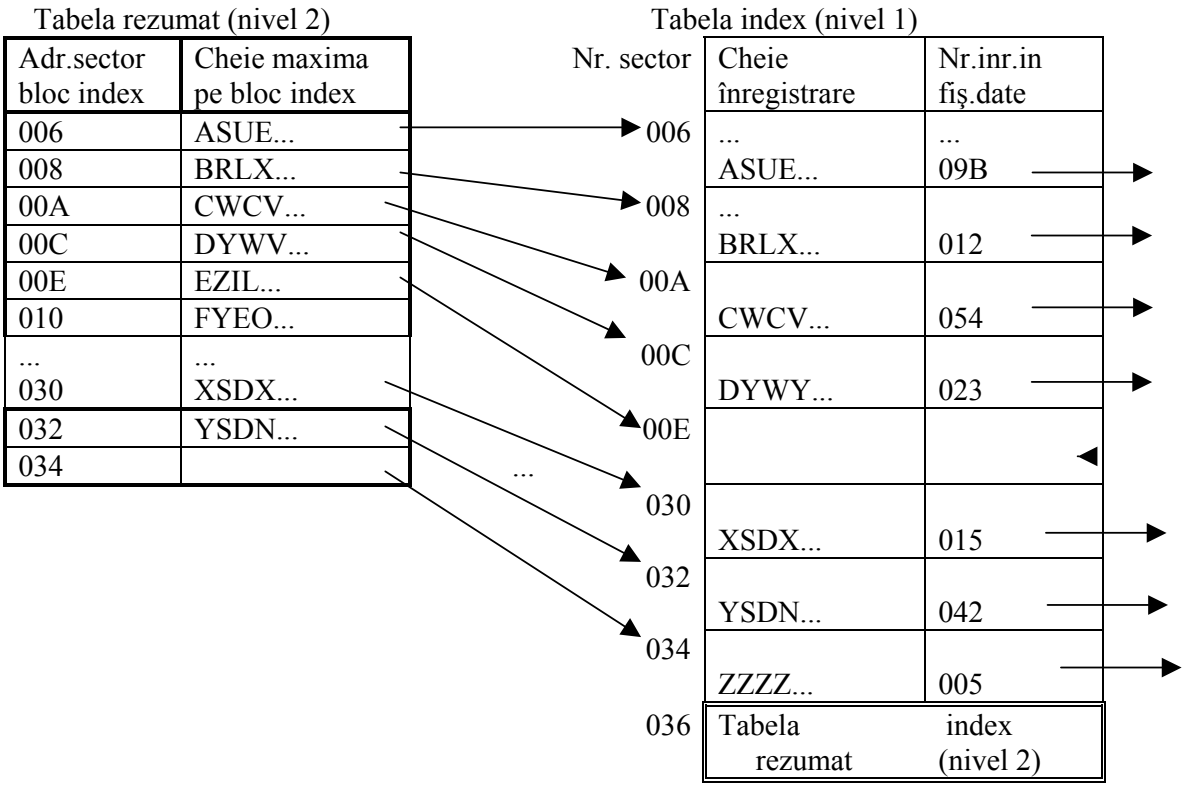


Fig. 5.2. Fişier index dens multinivel

8.2.3. Comenzi de indexare

Indexarea se face în dBase folosind comanda INDEX unde se specifică câmpul cheie după care se face indexarea şi numele fişierului index generat.

INDEX ON cheie TO fis_index [UNIQUE][DESCENDING]

Clauza UNIQUE nu accepta chei cu aceeasi valoare in mai multe inregistrari, iar DESC specifica sortarea fisierului index in ordine descrescatoare.

Daca fisierul index este deschis parcurgerea inregistrarilor din fisier secvential se va face in ordinea data de fisierul index (DISP, LIST, EDIT, BROWSE, SKIP, LOCATE). Deschiderea fisierelor index se poate face odata cu fisierul de date sau separat prin :

SET INDEX TO lista_fis_index

USE STUD - deschide fisierul
LIST - listeaza fisierul in ordine naturala
INDEX ON SUBSTR (NUME,1,7) TO INUME
* S-a indexat dupa primele 7 caractere din NUME
LIST - listare in ordinea numerelor
INDEX ON CODS TO ICODS - indexare dupa cod student
LIST - listare in ordinea codurilor (facultate,sectie)
USE STUD - deschidere fara fisiere index
LIST - listare in ordine naturala
SET INDEX TO ICODS - deschidere fisier index ICODS
LIST - listare in ordinea codurilor
USE - inchidere fisier de date si index asociate

La inchiderea unui fisier de date se inchid si toate fisierele index asociate lui. Un fisier DBF indexat poate fi exploatat si fara fisiere index sau numai cu unele din acestea. Daca se deschid simultan mai multe fisiere index, atunci primul din lista este index master si va determina ordinea de parcurgere a inregistrarilor la accesul secvential sau cheia de selectie in acces direct. Oricare fisier index poate fi specificat ca master prin comanda:

SET ORDER TO n -in care se specifica numarul de ordine n=l..7 din lista in care a fost deschis.
SET ORDER TO 0 - reprezinta ordinea naturala

Daca se fac modificari in fisierul de date (adaugari, stergeri, modificari inregistrari), se actualizeaza automat toate fisierele index deschise (APPEND, EDIT, GET, BROWSE, REPL).

USE STUD INDEX ICODS,INUME - deschide fisier DBF si index;
LIST - afisare in ordinea codurilor studentilor
SET ORDER TO 2 - fisier index master INUME
LIST - afisare in ordinea numelor.

Selectia in acces direct. In fisierele indexate se poate cauta o inregistrare dupa cheia din fisierul index master deschis utilizind comenzile:

SEEK expr - cauta inregistrarea cu cheia data prin expresie
FIND cheie - cauta inregistrarea cu cheia specificata direct

Daca inregistrarea cu cheia specificata exista se pozitioneaza functiile **FOUND()** pe.T. si **EOF()** pe .F. iar daca nu exista valorile vor fi invers. Dupa orice cautare se va verifica una din functii si daca EOF()=.F.,inseamna ca s-a pozitionat pe inregistrarea cautata. Compararea chei cu cimpul cheie se face pentru siruri functie de modul de comparatie SET EXACT ON sau OFF.

```
USE STUD INDEX INUME
SET EXACT OFF      - cautare dupa primele caractere din cheie
FIND POP           - se cauta primul care are primele litere POP
* SEEK 'POP'       -daca se utilizeaza seek
IF EOF ()          - verifica daca s-a gasit cheia
 ? ` Nu exista student cu numele cerut`
 RETURN
ENDIF
DO WHILE Nume =`POP` .AND..NOT. EOF().
DISP Nume, Adresa  - afiseaza toti studentii POP..
SKIP               - trece la urmatorul articol conform fisierului index
ENDDO
RETURN
```

Acelasi exemplu se poate generaliza pentru dialog:

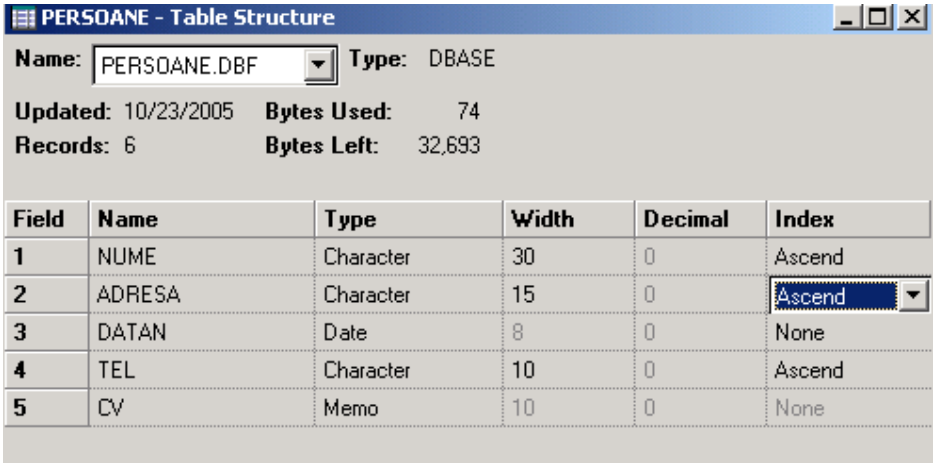
```
USE STUD INDEX INUME
ACCEPT `Nume student: ` TO VNUME      - cere nume student cautat
FIND &VNUME                - se specifica cheia indirect
*  SEEK VNUME              - in SEEK se specifica numele variabilei
IF EOF()                   - verificare daca studentul exista
 ? ` studentul  ` + VNUME + ` nu exista`
 RETURN
ENDIF
DO WHILE NUME=VNUME .AND..NOT. EOF()
DISP          - afiseaza toti studentii cu numele dat
SKIP
ENDDO         - urmatorul student conform fisierului index
```

S-a introdus si conditia .NOT.EOF() in DO pentru a se opri ciclul in cazul in care studentii sint la sfirsitul fisierului.
Comanda SEEK are exact aceeasi functie dar difera putin scrierea, pentru ultimul exemplu va fi:

```
SEEK VNUME        - iar restul ramine la fel.
```

Grup de fisiere index. In DBASE IV s-a introdus notiunea de grup de fisiere index, care are extensia .MDX si care poate contine,mai multe fisiere index, de obicei pentru acelasi fisier de date (DBF).

Fiecare fisier index se numeste TAG si are un numar propriu, Daca la crearea structurii fisierului DBF se completeaza coloana Index cu Ascending din dreptul unor cimpuri,ele devin automat chei (implicit valoarea este None). Pentru fisierul DBF se genereaza automat un fisier MDX cu acelasi nume si pentru fiecare cheie specificata un fisier index TAG cu acelasi nume. La orice deschidere a fisierului DBF se deschide automat si fisierul index multiplu MDX asociat, asigurind automat actualizarea fisierelor index componente.



Fisierul index (TAG) master se specifica prin:

SET ORDER TO TAG nume_tag - index din fisierul MDX activ
SET ORDER TO TAG nume_tag OF fis.MDX -daca indexul se gaseste inalt fisier MDX
La crearea ulterioara a unor fisiere index TAG se va folosi:
INDEX ON cheie TO TAG nume_tag OF fis. MDX.

In acest fel se pot genera tag_uri index prentru fisiere DBF diferite in acelasi fisier MDX.Deschiderea fisierului si a indexilor se face prin specificarea listei_Tag si a fisierului MDX.

USE fisier INDEX lista_tag OF fis.MDX [ORDER nume_tag].

OBSERVATIE: Utilizarea fisierelor index MDX este contraproductiva, deoarece toate fisierele TAG se deschid la deschiderea fisierului DBF si vor duce la actualizarea "ON LINE" a fisierelor index, care consuma mult timp. Daca se adauga multe inregistrari este preferabil ca fisierele index sa nu fie deschise si sa se reindexeze o singura data la sfirsit.

Reindexarea fisierului se poate face pentru **toate fisierele index deschise** prin comanda **REINDEX**. Reindexarea este obligatorie si dupa comanda PACK, care modifica numarul de ordine al inregistrarilor dupa eliminarea celor marcate.

USE STUD
PACK - eliminare inregistrari marcate
SET INDEX TO INUME,ICOD - deschide fisiere index
REINDEX - reindexeaza fisierele index deschise
USE - inchide fisierul

In versiunea dBASE IV Borland la comenzile de actualizare s-a introdus clauza REINDEX, care cere ca reindexarea sa se faca doar la terminarea comenzii si nu pentru fiecare inregistrare modificata(APPEND,INSERT,REPLACE).

8.3. UTILIZAREA SIMULTANA A MAI MULTOR FISIERE

Pina in prezent Baza de date (BD) a fost considerata dintr-un singur fisier deschis, la care se refereau toate comenzile.

Deschiderea altui fisier implica inchiderea celui deschis. Toate datele despre acest fisier se pastrau intr-o zona de memorie pe care o vom numi zona 1 sau A. In dBASE IV se pot utiliza maxim 10 zone si 40 in versiunea 1.5 BORLAND. Fiecare zona pastreaza aceleasi informatii despre :

- numele fisierului DBF deschis
- numele fisierelor index, report, format asociate.
- contorul de inregistrari si inregistrarea curenta.
- zona tampon pentru citirea din fisier
- adresa blocului curent din fisier citit
- informatiile privind structura inregistrarilor

Orice fisier poate fi deschis in orice zona care se selecteaza cu **SELECT nr_zona**. Zona in care se deschide fisierul poate fi specificata prin comanda USE utilizind **clauza IN:**

```
SELECT 3          - selectie zona 3
USE STUD INDEX INUME          - deschide fisier studenti in zona 3
SELECT l          - selectie zona 1
USE CURS INDEX ICODC          - deschide fisier cursuri
USE MASINI IN 4 INDEX INRM    - deschide fisier masini in zona 4
LIST                  - afiseaza lista cursuri
SELECT 3              - selectare zona 3
LIST                  - afisare lista studenti
SELECT 4              - selectare zona 4
LIST                  - afiseaza lista masini
```

Prelucrari ale unui fisier intr-o zona(SKIP,GO,FIND, APPEND...) nu afecteaza fisierele din alte zone, care ramin la forma si pozitia ultimei prelucrari.

Forma generala a comanzii USE este:

USE fisier [INDEX lista fisiere_index][IN nr_zona][ALIAS alias]

Clauza ALIAS specifica un nume prescurtat al fisierului ce poate fi folosit pentru referire in program la fisier sau la zona in care este deschis. Nu se permite deschiderea simultana a unui fisier in 2 zone. Inchiderea fisierelor se poate face prin:

USE - inchide fisierul DBF si cele index din zona curenta
USE IN nr_zona - inchide fisierul din zona specificata
CLOSE DATABASE - inchide toate fisierele DBF din toate zonele
CLOSE ALL - inchide toate fisierele DBF, NDX, FMT, FRM, din toate zonele.

Intr-o zona se pot utiliza cimpuri ale inregistrarii curente din alta zona prin **prefixare** cu numele fisierului sau un nume alias astfel:

```
USE STUD INDEX INUME ALIAS S1     - fisier studenti indexat dupa nume
USE MASINI IN 2 INDEX INRM        - fisier masini indexate dupa NRM
```

ACCEPT `Nume student` TO VNUME
SEEK VNUME - cauta studentul
* Presupunem ca in fisierul STUD exista un numar masina NRM
SELE 2
SEEK S1->NRM - cauta dupa NRM din fisierul STUD in fisierul MASINI
? S1->NUME,S1->ADR,NRM,TIP_M - afisare date student si masina

In fisiere diferite pot exista nume de cimpuri identice (NRM).
Intre doua fisiere deschise in zone diferite se poate stabili o relatie incit la pozitionarea pe inregistrarea din fisierul master sa se realizeze automat pozitionarea pe o inregistrare corespunzatoare din fisierul 2. Relatia se poate specifica prin:
- numar de inregistrare ce poate fi data printr-o expresie
- nume de cimp comun celor doua fisiere, care in fisierul 2 este cheie pentru un fisier index deschis (NRM)

USE STUD IN 1 - deschide fisierul studenti in zona 1
USE MASINI IN 2 - deschide fisierul studenti in zona 2
SET RELATION TO RECNO() INTO MASINI
GO 5 - pozitioneaza pe aticolul 5 in STUD si MASINI
In exemplul dat pentru orice pozitionare in fisierul master studenti, se cere automat pozitionarea pe inregistrarea cu acelasi numar din fisierul masini.

USE STUD IN 1
USE MASINI INDEX INRM IN 2 -deschide fis.masini indexat dupa NRM
SET RELATION TO NRM INTO MASINI
Cere ca la orice pozitionare pe o inregistrare din master sa se selecteze automat masina cu numarul NRM.Cimpul NRM trebuie sa existe in ambele fisiere.
In clauza TO se poate specifica si un pointer.

Pentru a putea exploata mai usor mai multe fisiere cu relatii intre ele au fost modificate si anumite comenzi si functii.

EOF(zona) - permite testarea lui EOF din zona specificata
FOUND(zona) -indicatorul logic FOUND() din zona indicata
SKIP n IN alias - pozitionare in fisier din alta zona fata de inregistrarea crt.
GO n IN alias - pozitionare pe inregistrarea n din fisierul indicat
RECNO(zona) - numarul inregistrarii curente din zona
RECSIZE(zona) - lungime inregistrare fisier din zona
RECCOUNT(zona) - numar de inregistrari in fis. din zona

Folosind SET RELATION se pot crea BD complexe, formate din mai multe fisiere, care au relatii intre ele si pot fi consultate
simultan fara complicatii de programare.
Atentie:Se interzice folosirea literelor ca alias,sau variabile deoarece sint implicit considerate nume de zone.
Forma generala a comenzii SET RELATION mai legaturi cu mai multe fisiere este:
SET RELATION TO exp1 INTO alias1, exp2 INTO alias2,....
La un moment dat dintr-o zona numai ultima relatie definita este operationala.

8.4.EXEMPLE DE PROGRAME

Consideram fisierele care contin cimpurile specificate:
STUD: NUME, ADRESA, DATA_N, CODS, BURSA, NRM
MASINI: NRM, TIP, AN_F, CAP_C ,PUTERE
Fisierul STUD il presupunem indexat dupa NUME, CODS si NRM iar MASINI dupa numar masina(NRM). Programul va permite cautarea unui student dupa oricare din chei si afisarea datelor personale si ale masinii:

```
*=========================================
* PREL * Program pentru exemplificare legaturi intre fisiere
*=========================================
SET TALK OFF
USE STUD INDEX  INUME, ICODS, INRM        - deschis in zona 1
USE MASINI IN 2 INDEX INR ALIAS M l       - deschis in zona 2
SET RELATION TO NRM INTO M1               - precizare relatie
DO WHILE .T.          -ciclu infinit cu iesire conditionata
CLEAR
TEXT
        Cautarea se poate face dupa:
        1- Nume student
        2- Cod student
        3- Numar masina
        4- terminare program
ENDTEXT
WAIT TO R
DO CASE
  CASE R=`1`
     SET ORDER TO 1                 - selectare fisier index master INUME
     C1=`NUME STUDENT: `            - C1 mesaj parametric de dialog
     C2=`NUME`                      - C2 nume cimp utilizat in DO WHILE
  CASE R=`2`
     C1='COD STUDENT: `
     C2=`CODS`                      - C2 numele cimpului cod student
     SET ORDER TO 2                 - selectare fisier index  ICODS
  CASE R=`3`
     C1=`NR. MASINA: `
     C2=`NRM`                       - cimp numar masina
     SET ORDER TO 3                  - selectare fisier index INRM ca master
  CASE R=`4`
     CLOSE ALL                      -inchide fisiere si sfirsit program
     RETURN
  OTHER
   ? `Functia` + R + ` inexistenta`
   WAIT
   LOOP                         - reluare dialog
ENDCASE
ACCEPT C1 TO CH                 - introducere cheie (C1 - mesaj corespunzator)
SEEK CH                         - cautare student dupa cheie
IF EOF()                        - student negasit
 ? `Studentul precizat prin `+ C1 + CH + `nu exista`
```

```
  WAIT
  LOOP   -reluare dialog
ENDIF
DO WHILE &C2=CH          - cautare toti studentii cu cheia data
    DISP                 - afisare date despre student
  IF EOF(2)              - verificare daca s-a gasit masina cu NRM in zona 2
    ? `Studentul nu are masina
   ELSE
  ? M1->NRM, M1->AN_F, M1->CAP_C, M1->PUTERE  - date despre masina
 ENDIF
?
WAIT
SKIP                     - trece la studentul urmator cu aceeasi cheie
ENDDO
   ENDOO
```

Secventa DO CASE s-ar fi putut inlocui cu:

```
IF R $ `l234`
N=VAL(R)     - valoarea numerica a lui R
SET ORDER TO N  - setarea cheie utilizind parametrul R
ELSE
? `Functia ` +R+ `nu exista`
WAIT
LOOP
ENDIF
```

Dupa incercarea exemplului prezentat incercati modificarea lui pentru a se selecta masina dupa NRM si apoi datele studentului inversind relatia.

Exemplu 2 de program care utilizează 3 fişiere legate între ele STUD, MASINI şi ACCIDENT pentru care s-au definit 3 ferestre. Pentru fiecare student selectat se afişează folosind SAY şi GET, în fereastra corespunzătoare datele studentului, ale maşinii şi accidendele maşinii respective.

```
set talk off
set date to dmy
use stud index inume, icods, inrm alias st
use masini in 2 index inm alias m1
set relation to nrm into m1
use accident in 3 index inr alias ac
select 2
set relation to nrm into ac
select 1
defi wind f1 from 1,1 to 15,37
defi wind f2 from 1,40 to 10, 70
defi wind f3 from 11,40 to 18,70
do while .t.
activ wind f1
clear
text
  Cautarea se face dupa:
    1. Nume student
```

```
      2. Cod student
      3. Numar masina
      4. Terminare
endtext
wait 'Dati optiunea: ' to r
do case
   case r='1'
      set order to 1
      c1='Nume student:'
      c2='Nume'
   case r='2'
      set order to 2
      c1='cod student:'
      c2='Cods
   case r='3'
      set order to 3
      c1='Nr. masina:'
      c2='Nrm'
   case r='4'
      close all
      clear all
      return
   other
   ? 'Functia '+r+' inexistenta'
      wait
      loop
endcase
clear    && sterge f1
accept c1 to ch
seek trim(ch)
if eof()
  @ 1,1 say 'Studentul precizat prin '+c1+ch+' nu exista'
   wait
   loop
endif
 do while &c2=ch
   clear            && sterge f1
 @ 1,1 say 'Student: ' +nume
 @ 2,1 say 'Adresa: ' + adresa
 @ 3,1 say 'Nr. masina: ' + nrm
 @ 4,1 say 'Cods: ' +cods
   activ wind f2
   clear            && sterge f2
if eof(2)
   @ 1,1 say 'Studentul nu are masina'
   else
   * ? m1->nrm, m1->tip, m1->an_f, m1 ->cap_c, m1 ->putere
   @ 1,1 say 'Nr. masina: ' +m1->nrm
   @ 2,1 say ' Tip masina: '+ m1->tip
   @ 3,1 say  'An fabricatie: ' + str(m1->an_f,5)
```

```
  @ 4,1 say ' Putere: '
  @ 4,10 say m1->putere
  activ wind f3
  clear
        if .not. eof(3)
  @ 1,1 say 'Masina nr: ' +ac->nrm
  @ 2,1 say 'Accident: ' +ac->cauza
  @ 3,1 say 'Data: '
  @ 3,10 say ac->data
  @ 4,1 say 'Valoare repar: ' +str(ac->valoare,8)
         endif
  endif
  ?
  activ wind f1
  c1=' '
  @ 9,1 Say 'continuati' get c1
  read
  clear            && sterge f1
  skip
        enddo
enddo
cancel
```

Structura Fisier STUD.DBF

Field	Field Name	Type	Length	Dec	Index
1	NUME	CHARACTER	10		N
2	ADRESA	CHARACTER	10		N
3	BURSA	NUMERIC	6		N
4	DATA	DATE	8		N
5	CODS	CHARACTER	10		N
6	NRM	CHARACTER	10		N

	NUME	ADRESA	BURSA	DATA	CODS	NRM
1	adi	tm	2592	03	ac231	tm1
2	deni	TM	4074	12	ac427	tm5
3	dan	tm	32442	02	ac163	tm3
4	virgil	TM	3241	12	ac523	tm4
5	pop	ar	57438	12	ac234	ar1
6	vlad	ar	342	12	ac235	ar2
7	liviu	DEVA	657	02	ac123	ar6
8	stefan	TM	3402	10	ac124	tm5
12	radu	tm	12345	11	ac125	tm2
13	gelu	tim	500		ac126	ar5
14	asul	tim	2345	12	ac127	tm10
16	ralu	ar	9876	05	ac236	ar3
17	vasile	ar	10123	12	ac238	ar4

Structura fisier MASINI.DBF

Field	Field Name	Type	Length	Dec	Index
1	NRM	CHARACTER	10		N
2	TIP	CHARACTER	10		N
3	AN_F	NUMERIC	4		N
4	CAP_C	NUMERIC	7	2	N
5	PUTERE	NUMERIC	10		N

	NRM	TIP	AN_F	CAP_C	PUTERE
1	tm5	opel	1993	1575	85
2	tm4	golf	2000	1600	90
3	tm3	logan	2004	1500	75
4	tm2	audi	2001	1700	85
5	tm1	dacia	1996	1400	70
6	ar1	bmw	2000	2200	110
7	ar2	mercedes	2001	2300	120
8	ar3	dacia	2002	1400	70
9	ar4	ford fiest	2004	1400	120
10	ar5	nissan	1999	1600	130
11	tm10	ford musta	1988	1480	210

Structura fisier ACCIDENT.DBF

	Field Name	Type	Length	Dec	Inde
1	NRM	CHARACTER	10		
2	DATA	DATE	8		
3	CAUZA	CHARACTER	20		
4	VALOARE	NUMERIC	10		

	NRM	DATA	CAUZA	VALOAR
1	tm3	01	derapaj	1000
2	tm1	10	coliziune	2500
3	ar1	02	viteza 150	3100
4				

COMENZI PREZENTATE

APPEND FROM	SEEK	CLOSE ALL	USE..IN
SORT	FIND	CLOSE DATAB	SELECT
INDEX ON	REINDEX	SET ORDER	SET RELATION TO
SET INDEX TO			

9.UTILIZARE PROCEDURI SI FUNCTII

9.1.PROCEDURI SI FUNCTII UTILIZATOR

Pentru a realiza o programare modulara , programe inteligibile, usor de depanat si dezvoltat, se recomanda scrierea unui program principal din care se cheama proceduri. Fiecare procedura va realiza functii bine determinate si va avea parametrii propri de intrare si iesire.

In proceduri se utilizeaza parametrii formali specificati prin comanda PARAMETER, variabile locale, nume de cimpuri din orice zona (utilizind prefixarea), variabile globale.

Variabilele globate se declara in sectiunile unde se folosesc prin:

PUBLIC lista _var

Variabilele locale (PRIVATE) se pot utiliza numai in procedura unde sint definite, sint alocate in stiva si dispar dupa executia procedurii. Se pot realiza copii private ale unor variabile publice prin

PRIVATE lista_var ALL [LIKE/EXCEPT generic]

Parametrii formali ai unei proceduri sau functii se dau prin:

PARAMETER lista_var_formale

O procedura are structura:

PROCEDURE nume_proced
PARAMETER lista_var_formale
PUBLIC lista_var_globale
PRIVATE lista_var_locale
---------- - secventa comenzi definitie procedura

RETURN - sfirsit de procedura

Sfirsitul de procedura se marcheaza prin inceputul altei proceduri sau functie.

O procedura se cheama prin:

DO nume_proced WITH lista_param

Parametrii de apel trebuie sa fie de acelasi tip cu cei formali din definitie.

Variabilele publice nu se pot sterge cu RELEASE lista_var ci numai cu CLEAR ALL

Functia poate avea mai multi parametrii de intrare dar numai o valoare transmisa la iesire prin numele functie.

Structura unei functii este asemanatoare cu a procedurii,dar valoarea de iesire se transmite in argumentul comenzii RETURN:

FUNCTION nume_functie
PARAMETER lista_var_formale

PUBLIC lista_var_globale
---------- - secventa comenzi definitie functie

RETURN (exp) - iesire cu transmitere valoare calculata.

O functie poate fi utilizata in expresii cu operanzi de acelasi tip cu ea.
Tipul functiei e dat de tipul expresie din RETURN.
Utilizarea procedurilor permite refolosirea unor secvente de program. Procedurile pot apela proceduri dar nu recursiv.
Procedurile se pot scrie in program la sfirsit sau se pot grupa intr-un fisier de proceduri ce se va declara cu:

SET PROCEDURE TO fis_proc

La un moment dat poate exista un singur fisier de proceduri activ. Inchiderea unui fisier de proceduri se face prin:
SET PROCEDURE TO sau
CLOSE PROCEDURE

9.2.PROIECTARE APLICATIE SIMPLA DE BAZA DE DATE

La proiectarea unei aplicatii se recomanda urmatoarele faze:

-Analiza aplicatiei si alegerea fisierelor utilizate
-Stabilirea structurii fisierelor componente ale BD si a legaturilor dintre ele (cimpuri, fisiere index, relatii)
-Stabilirea functiilor care trebuie realizate (cerute de beneficiar)
-Stabilirea procedurilor ce realizeaza fiecare functie si a algoritmilor folositi
-Stabilirea meniurilor de selectie si a ierarhiei lor
-Stabilirea ierarhiei de chemare a procedurilor
-Scrierea programului principal care cuprinde meniul principal si secventa de selectie.
-Scrierea si testarea fiecarei proceduri si a programului in ansamblu.

Vom prezenta in continuare un program care realizeaza gestiunea unei BD de studenti, ce realizeaza urmatoarele functii:
- crearea structurii BD utilizind o structura de referinta
- adaugarea inregistrari - singulare(deschise fisierele index)
-masive cu reindexare la sfirsit
- afisarea unor inregistrari selectate in acces direct
- modificarea datelor unor studenti selectati prin nume
- stergerea unor studenti din BD

Ultimele trei proceduri cuprind secventa de selectare a studentului si afisarea lui.Ele difera putin, motiv pentru care s-a scris o singura procedura avind ca parametru codul functiei.
Listarea studentilor din BD s-a realizat tot cu o procedura parametrica pentru a realiza 3 functii distincte utilizind un meniu propriu.
Procedura PLIST realizeaza paginare, cu cap tabel si numerotare pagini. Se permite listarea totala in ordinea alfabetica, sau partiala dupa CODS si eventual indexarea dupa acest cimp.

****

```
** PRINC ** Program principal de selectie  proceduri   *
*********************************************
SET TALK OFF
SET PROCED TO FPRO              - specificare fisier proceduri
SET CLOCK TO 1,50               - afisare ceas pe linia 1 coloana 50
DO WHILE .T.
CLEAR
@ 1,1 SAY 'CAT.CALCULAT'
@ 2,5 SAY 'PROIECT DIDACTIC'
@ 2,60 SAY 'IONESCU BOGDAN'
@ 5,10 SAY 'BAZA DE DATE EVIDENTA STUDENTI'
@ 6,10 SAY REPL('=',30)
@ 7,5 SAY 'FUNCTII OFERITE:'
@ 9,10
TEXT
        -1 CREARE INITIALA BD
        -2 ADAUGARI MASIVE
        -3 ADAUGARI SINGULARE
        -4 INTEROGARE BD
        -5 MODIFICARI IN BD
        -6 STERGERI INREGISTRARI
        -7 LISTARE BD
        -8 TERMINARE PROGRAM
ENDTEXT
WAIT TO R
IF .NOT. R$ '12345678'          && verificare cod functie
? 'FUNCTIA '+R+'  NU EXISTA'
WAIT
LOOP
ENDIF
DO CASE
 CASE R='1'                     && creare structura BD
DO PCREA
CASE R='2' .OR. R='3'           && adaugari
DO PAD WITH R
CASE R $ '456'                  && afisari,modificari,stergeri
DO PIMS WITH R
CASE R='7'
DO PLIST                        && listari parametrice
CASE R='8'
? 'PROGRAMUL S-A TERMINAT * La revedere!  '  STYL 'U'
SET CLOCK OFF                   && oprire ceas
CANCEL
ENDCASE
ENDDO
*****************************
** FPRO ** Fiser de proceduri    **
*****************************************
** PCREA ** Procedura creare baza de date      *
*****************************************
```

```
PROCEDURE PCREA
CLEAR
@10,10 SAY 'CREARE INITIALA BD'
USE RSTUD                  && deachidere fisier de referinta
COPY STRU TO STUD          && copiere structura
USE STUD                   && deschide fisierul creat
DISP STRU                  && afisare structura
INDEX ON SUBSTR(NUME,1,7)  TO INUME      - indexare fisier vid
USE                        && inchidere fisier
@13,10 SAY 'S-A CREAT FISIERUL STUDENT' style 'i'
WAIT
RETURN
****************************************
*  PAD   ** Procedura adaugari                **
*  R=2 Adaugari masive  R=3 Adaugari singulare    **
****************************************
PROCEDURE PAD
PARAM R
USE STUD
IF R='3'                   && adaugari singulare
SET INDEX TO INUME         && deschidere fisier index
ENDIF
R1='D'
DO WHILE R1 $ 'DdYy'
CLEAR
@3,10 SAY 'PROCEDURA ADAUGARE'
APPEND BLANK               && adaugare inregistrare alba
@5,10 SAY 'NUME STUDENT  '  GET NUME
@6,10 SAY 'ADRESA  '  GET ADRESA
@7,10 SAY 'BURSA  '  GET BURSA
@8,10 SAY 'COD STUDENT  '  GET CODS
READ                       && activare geturi pentru completare cimpuri
? 'MAI EFECTUATI ADAUGARI D/N?'
WAIT TO R1                 && confirmare continuare adaugari
ENDDO
IF R='2'                   && au fost adaugari masive?
INDEX ON SUBSTR(NUME,1,7) TO INUME              && reindexare
ENDIF
USE
RETURN
*SFIRSIT PROCEDURA ADAUGARI
************************************************
**PIMS Procedura interogari,modificari,stergeri          *
*R=4 INTEROGARE,R=5 MODIFICARE,R=6 STERGERE  *
************************************************
PROCED PIMS
PARAM R
USE STUD INDEX INUME
STORE 'D' TO R1
DO WHILE R1 $ 'DdYy'
```

```
clear
@3,10 say 'PROCEDURA INTEROGARI,MODIFICARI,STERGERI'
*     secventa comuna de selectie si afisare
VNUME=SPACE(20)
@ 5,10 SAY 'NUMELE STUDENTULUI  ' GET VNUME
READ                                  && nume student selectat
VN=SUBSTR(TRIM(VNUME),1,7)
SEEK VN
IF EOF()
?'STUDENTUL CU NUMELE  ' +VNUME+ '  NU EXISTA'
WAIT
LOOP
ENDIF
*** ciclu de afisare grup de studenti
DO WHILE .NOT. EOF() .AND. NUME=VN
@ 7,5 SAY 'NUME STUDENT'GET NUME
@ 8,5 SAY 'ADRESA'GET ADRESA
@ 9,5 SAY 'BURSA' GET BURSA
@ 10,5 SAY 'COD STUDENT' GET CODS
IF R='5'
READ                  && se trece in editare
ENDIF
IF R='6'
DELETE                && se marcheaza pentru stergere
WAIT
ENDIF
IF R='4'
WAIT                  && se asteapta dupa afisare un student
ENDIF
SKIP                  && se trece la inregistrarea urmatoare
ENDDO
?'CONTINUATI D/N?'
WAIT TO R1
ENDDO
IF R='6'
PACK                  && compactare fisier daca s-au cerut stergeri
INDEX ON SUBST(NUME,1,7)TO INUME          && reindexare
ENDIF
USE
RETURN
************************************
** PLIST *Procedura listari parametrice  **
************************************
PROCEDURE PLIST
R1='D'
DO WHILE R1 $ 'DdYy'
CLEAR
@ 3,10 SAY 'PROCEDURA LISTARE'
@ 6,5 SAY 'FUNCTII OFERITE' STYLE 'I'
@ 8,10 SAY '1-LISTARE IN ORDINE ALFABETICA'
```

```
@ 9,10 SAY '2-LISTARE TOTALA'
@ 10,10 SAY '3-INDEXARE DUPA CODS'
@ 11,10 SAY '4-TERMINARE PROCEDURA'
WAIT TO R2
IF .NOT. R2 $ '1234'
? 'FUNCTIE INEXISTENTA'
WAIT
LOOP
ENDIF
DO CASE
CASE R2='1'
USE STUD INDEX INUME
CL='.T.'                    && afisare toti studentii in ordine alfabetica
CASE R2='2'
CL='CODS=VCOD'              && parametru conditie
ACCEPT 'CODURI DE LISTAT' TO VCOD
VCOD=TRIM(VCOD)             && codul grupului de studenti
USE STUD INDEX ICOD         && fisier index master ICODS
SEEK VCOD                   && cautare primul student cu codul dat
IF EOF()
? 'CODUL '+VCOD+' NU EXISTA'
WAIT  'Continuati D/N  '  TO  R1
LOOP
ENDIF
CASE R2='3'
USE STUD                    && indexare dupa CODS daca nu exista index
INDEX ON CODS TO ICOD
USE
LOOP
CASE R2='4'
RETURN                      && terminare program
ENDCASE
** urmeaza secventa comuna cu parametrul CL
CLEAR
NRP=0                 && contor de pagini
NL=100                && contor de linii ,fortare afisare cap tabel
* SET DEVICE TO PRINT                 && pentru imprimanta
DO WHILE .NOT. EOF() .AND. &CL        && ciclu de afisare
IF NL>23              && nr linii ecran (65 la imprimant)
DO CAPT               && afisare cap tabel
ENDIF
** afisare un rind pe o pozitie variabila
@NL,1 SAY NUME+'     I '+ADRESA+'    I  '+STR(BURSA,7,2)+' I  '+CODS
@NL+1,1 SAY REPLICATE ('_',75)          && linie subliniere
NL=NL+2               && incrementare contor linii
SKIP                  && inregistrarea urmatoare
ENDDO
?'LISTAREA S-A TERMINAT. CONTINUATI D/N ?'
WAIT TO R1
* SET DEVICE TO SCREEN                 && daca s-a afisat la imprimanta
```

```
ENDDO
RETURN
* SFIRSIT PROCEDURA LISTARE
*****************************
** CAPT ** Procedura cap tabel *
*****************************
PROCEDURE CAPT
WAIT                    && asteptare inainte de afisare pagina noua
CLEAR                   && daca se lucreaza pe ecran
*EJECT                  && pagina noua la imprimanta
NRP=NRP+1               && incrementare nr.pagina
@ 0,1 SAY 'FAC CALCULATOARE'
NL=5                    && primul rind de informatii dupa cap tabel
@ 1,20 SAY 'TABEL STUDENTI'             && titlul tabelului
@ 1,70 SAY 'PAG:'+STR(NRP,2)            && afisare nr.pagina
@ 2,1 SAY REPL('=',75)                  && linie cap tabel
@ 3,1 SAY '     NUME SI PRENUME      I      ADRESA       I';
  + ' BURSA     I   CODS '              && text in cap tabel
@ 4,1 SAY REPLICATE('=',75)
RETURN
=========================================
```

11. UTILIZARE MENIURI BARA SI POPUP

11.1. DEFINIRE SI UTILIZARE MENIURI BARA (PAD)

Definirea unor meniuri in care selectia sa se faca cu sageti si ENTER (ca in NORTON) se poate realiza si in dBASE IV utilizind:

DEFINE MENU - definire meniu orizontal (bara)

DEFINE POPUP- definire meniu vertical intr-o fereastra.

Elementele unui astfel de meniu se specifica in mai multe comenzi. Vom considera un meniu orizontal de forma:

DIRECTOR	**STRUCTURA BD**	**LISTARE**	**ACTUALIZARE**	**EXIT**
0,1	0,15	0,30	0,45	0,60

Acest meniu urmeaza sa realizeze functiile:

- afisare DIRECTOR curent
- afisare STRUCTURA fisier DBF activ
- CREARE structura pentru un fisier DBF
- ACTUALIZARE fisier
- LISTARE fisier
- terminare program si stergere meniu (EXIT)

Fiecare element al meniului numit PAD se caracterizeaza prin:

- textul continut ce se va specifica prin clauza PROMPT
- coordonatele primului caracter al textului
- functia ce se executa la selectia pad-ului, care poate fi o comanda dBASE, inclusiv DO procedura
- mesajul explicativ afisat pe ultima linie a ecranului

Selectia PAD se face cu sageti orizontale (stinga, dreapta), sau cu prima litera. Pad-ul selectat va fi afisat in video invers si comanda asociata se lanseaza la apasarea tastei ENTER.

Definirea unui meniu ca grup de pad-uri se face prin comanda:

DEFINE MENU nume_m [MESSAGE expC]

unde: nume_m - este numele meniului.

expC - contine un mesaj explicativ care se afiseaza pe ultima linie a ecranului cind pad-ul nu are propriul mesaj

ACTIVATE MENU nume_m [PAD nume_p]

Afiseaza meniul nume_m anterior definit pozitionat pe nume_p.

La un moment dat poate exista un singur meniu activ.Dezactivarea meniului activ se face fara ai preciza numele prin:

DEACTIV MENU - se sterge meniul de pe ecran.

Un menu deactivat nu se sterge din memorie, el putind fi reactivat ulterior.

Stergerea meniului de pe ecran si din memorie se face cu:

RELEASE MENU nume_m

Se pot sterge toate meniurile de pe ecran si din memorie cu:

CLEAR MENU

Afisarea simpla fara activare a unui meniu se realizeaza cu:

SHOW MENU nume_m

Pad-urile componente ale unui meniu se definesc prin:

DEFINE PAD nume_p OF nume_m PROMPT expC AT x1,y1 [MESS expC]

- clauza OF precizeaza numele meniului din care face parte padul
- PROMPT specifica textul continut in PAD ce poate fi o expresie
- AT indica coordonatele primului caracter din PAD linie,coloana
- MESS da mesajul explicativ afisat la selectia pad-ului

Asocierea unei comenzi pentru un pad definit se face prin:

ON [SELECTION] PAD nume-P OF nume-m comanda

Daca lipseste clauza SELECTION atunci activarea comenzii (sau procedurii lansate cu DO) asociata se face la pozitionarea pe pad, fara a se cere apasarea tastei ENTER. Comanda asociata pad-ului poate activa un meniu popup prin: ACTIV POPUP nume_p.

Pentru meniul discutat vom scrie un scurt program, care realizeaza functiile in modul cel mai simplu utilizind comenzile ! DIR, DISP STRU, CREATE fis, LIST.

Pad-ul ACTUALIZARE va activa un meniu popup.

***MENU* Program simplu cu meniu orizontal**

```
*=======================================
SET TALK OFF
USE STUD INDEX INUME
DEFI MENU M 1
DEFI PAD DIR1 OF M1 PROMPT 'DIRECTOR` AT 0,1 MESS `director crt.`
DEFI PAD STR1 OF M1 PROMPT `STRUCTURA BD` AT 0,15;
        MESS ' afisare structura fisier curent`
DEFI PAD LIST1 OF M1 PROMPT `LISTARE` AT 0,30;
        MESS `afisare continut fisier`
DEFI PAD ACT1 OF M1 PRPMPT `ACTUALIZARE` AT 0,45;
        MESS  `actualizari fisier`
DEFI PAD TERM OF M1 PROMPT `EXIT` AT 0,60 MESS`terminare program`
ON SELE PAD DIR1 OF M1 !DIR                    &&` afisare director`
ON SELE PAD STR1 OF M1 DISP STRU        && - afisare structura
ON SELE PAD LIST1 OF M1 LIST            && - listare fisier
ON SELE PAD ACT1 OF M1 ACTIV POPUP ACT_P    && -activare popup
ON SELE PAD TERM OF M1 DEACT MENU    && - stergere meniu
ACTIVATE MENU M1            && - activare meniu
RETURN                      && - terminare dupa deactivare
```

11.2. DEFINIRE SI UTILIZARE MENIURI POPUP

Meniurile POPUP sau verticale afiseaza pe ecran o fereastra cu texte de tip meniu, care se pot selecta cu sageti sau prima litera din rind. Fiecarei linii (bara) ii va corespunde o functie de executat, stabilita intr-o procedura de selectie asociata meniului. Linia selectata in meniu se indica in procedura de selectie asociata prin numarul ei (l..n), dat de functia BAR().

In cazul utilizarii **clauzei PROMPT** anumite informatii se afiseaza in meniu. Informatia selectata se transmite procedurii de selectie prin functia PROMT() care ia valoarea respectiva.

Pentru utilizarea unui POPUP se folosesc mai multe comenzi;

- de definire meniu (DEFINE POPUP), care se pastreaza in memorie

- definire procedura de selectie asociata (ON SELECT POPUP) activata la apasarea tastei ENTER, dupa pozitionarea pe linia dorita.
- activare meniu (ACTIVATE POPUP), care afiseaza meniul pe ecran;
- deactivare meniu (DEACTIV POPUP), care sterge de pe ecran meniul fara al sterge din memorie; - stergere meniuri de pe ecran si din memorie (CLEAR MENU)
- afiseaza meniul pentru verificare (SHOW MENU)

DEFINE POPUP nume_p FROM x1,y1 TO x2,y2 [MESSAGE expC]

Defineste meniul popup cu numele **nume_p** intr-o fereastra definita prin coordonatele x1,y1 si x2,y2,care afiseaza pe ultima linie de ecran mesajul dat de expC.

DEFINE BAR n OF nume_p PROMPT expC [MESS expC] [SKIP [FOR cond]]

Defineste **linia n** a meniului nume_p,care va contine sirul dat de expC din PROMPT si va afisa pe ultima linie a ecranului mesajul indicat. Se poate interzice selectarea unor linii de menu prin SKIP, sistematic (comentarii,titluri), sau cind conditia din FOR e indeplinita (nu exista proceduri pentru acele functii).

Daca se definesc mai multe linii de meniu decit dimensiunea ferestrei, atunci se face o baleere in limitele ferestrei.

Varianta a 2-a de definire se face fara linii definite.
Continutul ferestrei va fi format din:

- valoarile unui cimp din fisierul curent,
- numele cimpurilor fisierului curent,
- numele fisierelor de date (DBF) din directorul curent.

Valoarea selectata se transmite prin functia PROMPT(),spre procedura de selectie si prelucrare. Sintaxa comenzii este atunci:

DEFINE POPUP nume_p FROM x1,y1 TO x2,y2 [MESSAGE expC];
PROMPT FIELD nume_cimp - afisare valori cimp
FILE LIKE *.DBF - afisare nume fisiere
STRUCTURE - afisare nume cimpuri fisier

Procedura de selectie asociata se activeaza dupa selectia rindului din fereastra si apasarea tastei ENTER.

Numele procedurii asociate unui popup se specifica prin:
ON SELECTION POPUP nume_p DO proced_selectie
Unde numele meniului e dat de nume_p iar procedura de selectie asociate se da in DO.

Dupa definirea meniului si a barelor (daca e cazul),afisarea sa pe ecran se face prin activare specificind numele meniului:
ACTIVATE POPUP nume_p
Stergerea de pe ecran al unui meniu se face prin deactivare, unde nu se specifica numele, fiindca la un moment dat un singur popup poate fi activ. Meniul deactivat ramine in memorie si poate fi ulterior reactivat.
DEACTIVATE POPUP
Pentru a verifica forma unui meniu definit el poate fi afisat fara activare prin:
SHOW POPUP nume_p
Stergerea din memorie a meniurilor popup se face cu:
RELEASE POPUP nume_p-sterge meniul specificat

CLEAR POPUP -sterge toate meniurile popup din memorie

CLEAR ALL -sterge toate meniurile popup si bara,toate variabilele (inclusiv cele publice si tablouri) ,toate ferestrele, inchide toate fisierele DBF, index, report, format din toate zonele.

O procedura de selectie are in principiu structura.

```
* PSELECT  *Procedura de selectie
* Functia BAR() da numarul liniei selectate din meniul popup
*==================================
PROCED PSELECT
DO CASE
 CASE BAR() = 2
   DO PCREA          && - procedura asociata liniei 2 din meniu
 CASE BAR() = 3.OR.BAR() = 4
   DO PAD            && -procedura asociata liniei 3 si 4
CASE BAR() = 8       && -linia 8 indica iesire din popup (TERM)
   DEACTIV  POPUP          && -stergere de pe ecran meniul popup
   RETURN                  && -revenire in meniul anterior
ENDCASE
          RETURN           && -revenire in meniul popup
```

Se prezinta in continuare un exemplu de utilizare a meniului popup cu PROMPT pentru afisare informatii dintr-un fisier oarecare de date.

```
* Definire meniu popup ce afiseaza fisierele de date din director
clear
defi popup pop1 from 2,3 to 10,30 prompt file like *.DBF
on select popup pop1 do afis          && -asociere procedura meniului
activate popup pop1                   && -activare si afisare nume fisiere
return                                && -terminare program
* Procedura de selectie si prelucrare activata de popup
PROCEDURE AFIS
v=prompt()            && -memorare nume fisier selectat
use &v                && -deschidere fisier selectat
clear
disp stru             && -afisare structura fisier
wait
list                  && -afisare continut fisier
wait
return                && -revenire in meniul popup
```

Urmatorul exemplu afiseaza in popup numele cimpurilor fisierului activ.Valorile existente in fisier pentru cimpul selectat se afiseaza.

```
clear
use stud              && -deschidere fisier studenti
* Definire meniu popup ce afiseaza cimpurile fisierului activ
defi popup pop1 from 2,3 to 10,30 prompt stru
on select popup pop1 do afis          && -asociere procedura meniului
activate popup pop1                   && -activare si afisare nume cimpuri
return                                && -terminare program
```

```
* Procedura de selectie si prelucrare activata de popup
PROCEDURE AFIS
v=prompt()              && -memorare nume cimp selectat
clear
list &v                 && -afisare valori cimp selectat
wait
return                  && -revenire in meniul popup
```

O sa exemplificam acum modul de selectie si afisare a unei valori dintr-un cimp al fisierului.

```
clear
use stud
* Definire meniu popup ce afiseaza toate numele studentilor
defi popup pop1 from 2,3 to 10,30 prompt field nume
on select popup pop1 do afis        && -asociere procedura meniului
activate popup pop1                 && -activare si afisare nume nume
return                              && -terminare program
* Procedura de selectie si prelucrare activata de popup
PROCEDURE AFIS
clear
? PROMT()               && -afisare numele studentului selectat
wait
return                  && -revenire in meniul popup
```

Pentru programul de evidenta a studentilor ,prezentat anterior se va utiliza acum un program principal pe baza de meniuri orizontale si popup care sa activeze procedurile existente.

```
*=====================================
* MENIU *Program principal BD studenti cu selectie prin meniu
*=====================================
SET TALK OFF
clear
*    Definire meniu orizontal    *
defi menu m1 mess 'selectati cu sageti si ENTER '
defi pad dir1 of m1 prompt 'DIRECTOR' at 0,1 mess 'Directorul curent'
defi pad stru of m1 prompt 'STRUCTURA BD'  at 0,10 mess 'Structura fisier studenti'
defi pad crea  of m1 prompt 'CREARE BD' at 0,25 mess 'Creare initiala structura BD'
defi pad actual of m1 prompt 'ACTUALIZARE BD' at 0,40 ;
        mess 'Adaugari,Modificari,Stergeri '
defi pad list of m1  prompt 'LISTARI'  at 0,60 mess 'Listari parametrice'
defi pad term of m1 prompt 'EXIT' at 0,73
*    Definire proceduri pentru fiecare pad     *
on select pad dir1 of m1 !DIR                   && -afisare director
on select pad stru of m1 DISP STRU                  && -afisare structura
on select pad crea of m1 DO PCREA               && -creare structura
on select pad actual of m1 ACTIV POPUP A_POP
on select pad list of m1 DO PLIST               && -listari fisier
on select pad term of m1 DEACT MENU
*    La EXIT se sterge meniul bara principal  *
*   Definire meniu popup pentru actualizari     *
```

```
defi popup a_pop from 5,5 to 15,35  mess 'selectia se face cu bare sau prima litera'
SET PROCEDURE TO FPRO                  && -specificare fisier proceduri
defi bar 1 of a_pop prompt 'ACTUALIZARI BD' SKIP
defi bar 2 of a_pop prompt repl('=',20) SKIP
defi bar 3 of a_pop prompt space(20) SKIP
defi bar 4 of a_pop prompt 'ADAUGARI MASIVE'
defi bar 5 of a_pop prompt 'ADAUGARI SINGULARE'
defi bar 6 of a_pop prompt 'INTEROGARE BD'
defi bar 7 of a_pop prompt 'MODIFICARI IN BD'
defi bar 8 of a_pop prompt 'STERGERI INREGISTRARI'
defi bar 9 of a_pop prompt 'IESIRE'
*     Activare procedura selectie popup A_POP
ON SELECTION POPUP a_pop DO SELECT
*   Definire fereastra pentru cimp memo
defi wind FM from 9,6 to 21,79
*     Activare meniu principal bara M1
ACTIVATE MENU M1
clear all
return
*==================================
*   SELECT  * Procedura selectie pentru POPUP a_pop *
*==================================
PROCED SELECT
r=str(bar()-2,1)                  && -parametru pentru proceduri
clear
do case
  case bar()=4 .or.bar()=5
          do pad with r           && -adaugari inregistrari
  case r $ '456'                  && -corespunde liniilor 6,7si 8
          do pims with r          && -interogare,modificare,stergere
  case  bar()=9                   && -iesire din meniul popup
          deactiv popup           && -stergere meniu popup de pe ecran
          return                  && -revenire in meniul superior
endcase
clear
RETURN                            && -revenire in popup
```

COMENZI PREZENTATE

DEFI MENU	DEFI POPUP	RELEASE MENU
DEFI PAD	DEFI BAR	RELEASE POPUP
ON SELECT PAD	ON SELECT POPUP	CLEAR MENU
SHOW MENU	SHOW POPUP	CLEAR POPUP
CLEAR WINDOW	CLEAR ALL	ACTIV/DEACT MENU
ACTIV/DEACT POPUP		

12.IMPLEMENTAREA STRUCTURILOR IERARHICE

Daca un cod se regaseste in mai multe inregistrari si aceste inregistrari au o parte din cimpuri identice,(determinate de cod) pentru a reduce redondanta,se vor folsi doua fisiere legate intre ele.Primul va contine setul de cimpuri ce au valori identice pentru un grup de inregistrari,iar al doilea cimpurile care difera.Legatura dintre ele se poate realiza in doua moduri:

a) -Adaugind in fisierul 2 codul inregistrarii parinte din fi- sierul 1 si indexind fisierul 2 dupa acest cod;

b) -Utilizind in fisierul 1 (master) un cimp pointer PT,care va indica numarul primei inregistrari asociate din fisierul 2. Toate inregistrarile din fisierul 2 asociate aceleasi inregistrari parinte vor fi inlantuite intre ele printr-un pointer UR,care specifica numarul de ordine in fisier a urmatoarei inregistrari. Ultima inregistrare din lant va avea UR=-1.

Ultima metoda este foarte rapida,eliminind cautarea prin fisierul index a articolelor asociate unui articol master,care poate dubla sau tripla timpul de acces la informatii.

Pentru exemplificare vom considera o baza de date pentru evidenta comenzi,care va contine fisierele COMENZI,PROD_C(produse comandate), CAT_P (catalog produse) si FURNIZOR avind structura:

COMENZI

NRC	DATA	CODF	PT
124	140793	345	25

PROD_C -produse comandate

	CODP	CANT	DATAL	PRET	UR
25	456	960.25	150394	500.0	51
...					
51	110	5.00	030694	4256.0	75
...					
75	123	23.10	010494	38578.0	-1

FURNIZOR -catalog furnizori

CODF	NUMEF	ADRESA
345	ELECTRO	PARVAN 2

CAT_P -catalog produse

CODP	DEN produs	UM
110	roata dintata	buc
...		
123	anvelopa 155*SR 13	buc
456	benzina CO-98	litri

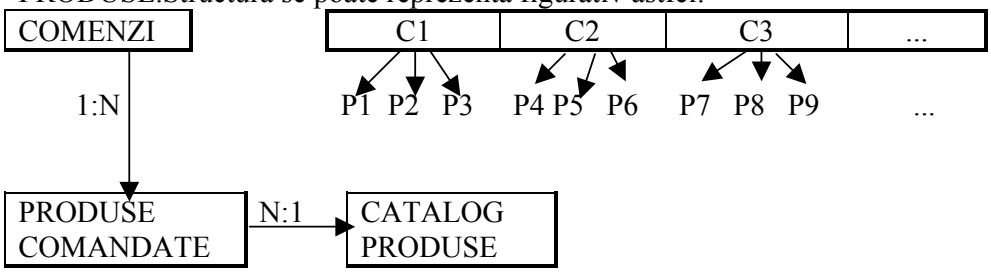
S-a notat: **NRC**-nr.comanda,
DATA-data lansarii comenzii,
CODF-cod furnizor, **CODP**-cod produs, **CANT**-cantitate,
DATAL-data livrarii, **PRET**-pret unitar
DEN-denumire produs, **UM**-unitate de masura

Intre fisierul **PROD_C** si **CAT_P** se realizeaza o legatura automata prin cimpul **CODP** utilizind comanda SET RELATION TO, iar intre **COMENZI** si **FURNIZOR** prin cimpul **CODF**.

Pentru fisierul PROD_C vom realiza recuperarea spatiului eliberat prin stergerea inregistrarilor si alocarea dinamica a acestui spatiu noilor inregistrari adaugate.Pentru aceasta utilizam pointerul UR din prima inregistrare,care va fi cap de lant pentru inregistrarile eliberate prin stergere.Pentru toti pointerii folosim -1 ca indicator sfirsit de lista.Inregistrarile corespunzatoare produselor sterse,vor fi eliminate din lantul asociat comenzii respective si vor fi adaugate in lantul de libere.Adaugarile de noi produse pentru o comanda se fac la inceputul listei dupa algoritmul :

```
SELECT PROD_C            -selectie fisier produse
APPEND BLANK             -adaugare produs nou la sfirsit
* -se memoreaza pointerul cap de lant din comanda in UR din
*    inregistrarea adaugata (sistem stiva)
REPLACE UR WITH COMENZI->PT
* -pointerul PT din COMENZI va indica ultimul produs adaugat
REPLACE COMENZI->PT WITH RECNO()
```

Aceste BD sint de tip ierarhic (arborescent) cu legaturi 1:N intre COMENZI si PRODUSE_COMANDATE si de tip referinta (N:1) prin cheie spre CATALOGUL PRODUSE.Structura se poate reprezenta figurativ astfel:



Programul va contine :

-Programul principal,care defineste si activeaza meniurile

-Procedurile: COMAND -selecteaza comanda indicata prin numar (NRC)
ADAUGP -adauga produse la comanda selectata reutilizind spatiul lantului inregistrarilor sterse (LIBER)
AFISP -afiseaza produsele de pe comanda selectata
STERGP -sterge produsele indicate prin CODP de pe c-da recuperind spatiul eliberat in lantul LIBER

Restul procedurilor din meniuri referitoare la fisierele COMENZI si CAT_P, sint simple si nu s-au prezentat existind exemple de acest tip discutate anterior.

Pentru o mai buna intelegere se dau meniurile, structurile si continuturile fisierelor utilizate, rezultatele obtinute din diferite proceduri pe ecran.

```
**************************************************
* Program evidenta comenzi inlantuind produsele de pe comanda
* FISIERE :  Comenzi,  Prod_c (produse comandate)
*  I. Jian   Cat_p (catalog produse)
**************************************************
```

```
clear
set talk off
* definire meniu principal
mes1='selectati cu sageti si enter'
defi menu m1 mess mes1
defi pad cre of m1 prompt 'CREARE BD' at 0,1 mess 'Creare structura bd'
defi pad com of m1 prompt 'COMENZI' at 0,15 mess 'Comenzi lansate'
defi pad prod of m1 prompt 'PRODUSE' at 0,30 mess 'Produse contractate'
defi pad cat of m1 prompt 'CATALOG' at 0,45 mess 'Catalog de produse'
defi pad term of m1 prompt 'EXIT' at 0,60  mess 'Terminare program'
on  select pad cre of m1 do pcrea
on select pad com of m1 activ popup pc
on select pad prod of m1 activ popup pp
on select pad cat of m1 activ popup pca
on select pad term of m1 deact menu
*    definire popup comenzi  pc*
mes2 ='selectia cu sageti sau prima litera'
defi popup pc from 5,5 to 15,35 mess mes2
defi bar 1 of pc prompt '      COMENZI' SKIP
defi bar 2 of pc prompt space(5)+ repl('=',10) SKIP
defi bar 3 of pc prompt space(10) SKIP
defi bar 4 of pc prompt 'adaugare comenzi'
defi bar 5 of pc prompt 'modificare comanda'
defi bar 6 of pc prompt 'afisare comanda'
defi bar 7 of pc prompt 'terminat'
ON SELECT POPUP pc do cselect
*         definire popup produse  pp  *
defi popup pp from 5,5 to 15,35 mess mes2
defi bar 1 of pp prompt 'PRODUSE COMANDATE' SKIP
defi bar 2 of pp prompt  repl('=',20) SKIP
defi bar 3 of pp prompt space(10) SKIP
defi bar 4 of pp prompt 'adaugare produse pe c-da'
defi bar 5 of pp prompt 'afisare produse de pe c-da'
defi bar 6 of pp prompt 'stergere produse din c-da'
defi bar 7 of pp prompt 'terminat'
ON SELECT POPUP pp do pselect
*         definire popup catalog produse pca     *
mes2 ='selectia cu sageti sau prima litera'
defi popup pca from 5,5 to 15,35 mess mes2
defi bar 1 of pca prompt '      CATALOG' SKIP
defi bar 2 of pca prompt space(5)+ repl('=',10) SKIP
defi bar 3 of pca prompt space(10) SKIP
defi bar 4 of pca prompt 'adaugare produse'
defi bar 5 of pca prompt 'modificare inregistrari'
defi bar 6 of pca prompt 'afisare produse dupa cod' defi bar 7 of pca prompt 'terminat'
ON SELECT POPUP pca do catsel
ACTIVATE MENU M1
CLEAR ALL
RETURN
*  CSELECT   *  Procedura selectie comenzi   *
```

```
PROCED cselect
do case
        case bar()=4
        do pcad
        case bar()=5
        do pcmod
        case bar()=6
        do pcafis
        case bar()=7
        deact popup
        return
endcase
RETURN                  && sfirsit program principal
```
*** PCSELECT ** Procedura selectie produse   ***

```
PROCED pselect
do case
        case bar()=4
        do comand
        case bar()=5
        do comand
        case bar()=6
        do comand
        case bar()=7
        deact popup
        return
endcase
RETURN
```
*** CATSEL ** Procedura selectie catalog   ***

```
PROCED catsel
do case
        case bar()=4
        do padc
        case bar()=5
        do pmodc
        case bar()=6
        do pafisc
        case bar()=7
        deact popup
        return
endcase
RETURN
```
****

*** COMAND   **Procedura cautare comanda dupa NRC -nr comanda**

****

```
PROCEDURE COMAND
clear
use comenzi in 1 index inrc alias cd use prod_c in 2 alias pc
use cat_p in 3 index icodp alias cp
defi window fc from 0,0 to 3,79
defi window fp from 4,0 to 19,79
```

```
activ window fc
r1='d'
DO while r1$'DdYy'
clear
vnrc=space(5)
@ 0,1 say 'Nr.comanda: ' get vnrc mess 'Dati nr.comanda'
read
seek vnrc
if eof()
@ 1,0 say 'Nr.C-DA inexistenta!! '
wait 'Continuati ? d/n ' to r1
loop
endif
@ 1,1 say 'Furnizor: ' +codf
@ 1,25 say 'Data:  '+dtoc(data)
activ window fp          && fereastra produse
 do case
 case bar()=4            && adaugare produse pe comanda
  do adaugp
 case  bar()=5           && afisare produse de pe comanda
  do afisp
 case bar()=6
  do stergp
other
  @ 0,3 say 'functie inexistenta'
  wait 'Continuati?d/n ' to r1
  loop
endcase
***** Revenire din proceduri produse
select 1                 && zona fis comenzi
deactiv  window fp       && sterge fereastra produse
clear
@ 0,3 say 'Continuati  cu alta comanda ' get r1
read
ENDDO
 deactiv wind fc         && sterge fereastra comanda
 close all               && inchide toate fisierele
 RETURN
***sfirsit procedura comand
*********************************************************
*PADAUG * procedura adaugare produse pe comanda selectata
*        se aloca dinamic peste inregistrarile sterse
*********************************************************
proced adaugp
sele 2
        go 1             && pointer libere
        liber=ur
r2='d'
        DO WHILE r2 $ 'dDYy'
clear
```

```
vcodp=space(5) @ 1,1 say 'Cod produs: ' get vcodp
read
da=seek(vcodp,3)
  if .not. da
? 'Nu exista codul produs  '+ vcodp styl 'i'
? 'Actualizati catalogul de produse ' styl 'i'
wait ' Continuati ? d/n ' to r2
loop
  endif
@ 1,30 say cp->den
        if liber #-1            && exista libere
        go liber                && alocare dinamica
        liber=ur                && corectie libere
        repl cant with 0, pret with 0
        else                    && nu exista libere
APPEND BLANK                    && adaugare inregistrare
        endif
repl codp with vcodp,datal with date()
@ 2,1 say 'Cantitate comandata:' get cant
@ 3,1 say 'Pret unitar: ' get pret
@ 4,1 say 'Data livrarii: ' get datal
read
* *actualizare pointeri
repl ur with cd->pt
repl cd->pt with recno()
wait 'Mai adaugati produse? d/n ' to r2
enddo
        go 1            && memorare pointer libere
        repl ur with liber
return
**sfirsit procedura ADAUGP
************************************************
* AFISP ** procedura afisare produse de pe comanda
************************************************
PROCEDURE AFISP
sele 2
set relation to codp into cp     && spre catalog
n=cd->pt
  if n=-1
? 'Nu exista produse comandate ' styl 'i'
wait
return
  endif
clear
nl=2                    && contor linii afisare
@ 0,0 say 'CODP    DENUMIRE            CANTITATE    PRET    VALOARE  '
@ 1,0 say repl('=',79)
t=0
        DO while n # -1
go n                    && pozitionare pe inregistrare
```

```
* S-a pozitonat si pe produsul din catalog cu codul CODP
  if nl<15
@ nl,0 say 'I '+codp+' I '+cp->den +' I '+ str(cant,7,2)+ ' I '+str(pret,7,2)+;
          ' I '+str(pret*cant,10,2) +' I'
nl=nl+1 n= ur           && adresa urmatorului produs de pe comanda
  else
wait
nl=2                    && ecran nou
@ 0,2 clear             && sterge ecran fara cap tabel
  endif
t=t+pret*cant           && total comanda
          ENDDO
@ nl,0 say repl('=',78)
@ nl+1,29 say 'TOTAL VALOARE COMANDA '+str(t,12,2)
wait
return
** sfirsit procedura AFISP
************************************************
```

*** STERGP * Stergere produse de pe comanda ***

```
 *   inregistrarile sterse se ataseaza listei de libere (UR din record 1)
************************************************
PROCEDURE STERGP
SELECT 2
go 1
liber=ur          && cap liste inregistrari libere
n=cd->pt
  if n=-1
? 'Nu exista produse pe comanda'
wait
return
  endif
r2='d'
          DO WHILE r2 $ 'DdYy'
c=1                     && indicator prima din lant
gasit =0                &&indicator  produs negasit
n=cd->pt                && inceput lant
clear
vcodp=space(5)
@ 1,1 say 'Cod produs de sters' get vcodp
read
  DO WHILE n # -1
go n                    && urmatorul articol
  IF .not. codp=vcodp
ultim=n                 && nr. articol anterior in lant
n=ur
c=c+1                   && nu e primul produs
loop
 ELSE
gasit=1
disp
```

```
? 'Inregistrarea s-a sters'
m=ur                    && salvare pointer
repl ur with liber      && se adauga la libere
liber=recno()
gasit =1
   exit
  ENDIF
   ENDDO
  IF gasit =0
? 'Nu exista produsul pe comanda'
wait ' Continuati stergerile? d/n ' to r2
loop
  ENDIF
  IF C=1                && produsul s-a gasit
repl cd->pt with m      && articolul sters a fost primul
  ELSE
go ultim                && refacere lant
repl ur with m
  ENDIF
wait 'Continuati stergeri? d/n' to r2
          ENDDO
go  1                   && memorare cap lista libere
repl ur with  liber
return
** sfirsit STERGP
********************
```

13.IMPLEMENTAREA STRUCTURILOR DE TIP RETEA SIMPLA

In structurile arborescente, fiecare nod parinte poate avea mai multi fii dar, iar fiecare nod fiu are un singur parinte. Din acest motiv modelele de BD ierarhice nu permit legaturi orizontale intre entitatile de pe acelasi nivel ierarhic (fig.1), sau referinte de la doua entitati spre aceeasi entitate (fig.2).

Fig.1 Fig.2

Consideram gestiunea unei biblioteci universitare,care contine un fisier de CARTI si unul de STUDENTI (cititori) (fig.3)

Fig.3

Un student poate imprumuta de la biblioteca mai multe carti. Fiecare titlu de carte poate exista in mai multe exemplare, cu numar de inventar distinct(NRI), care pot fi imprumutate la studenti diferiti.Aceste legaturi sint de tipul retea simpla (N:M). Pentru implementarea acestei BD vom folosi structura din Fig.4, care transforma legaturile N:M in 2 legaturi mai simple 1:N, care se pot implementa prin referinte si liste inlantuite.

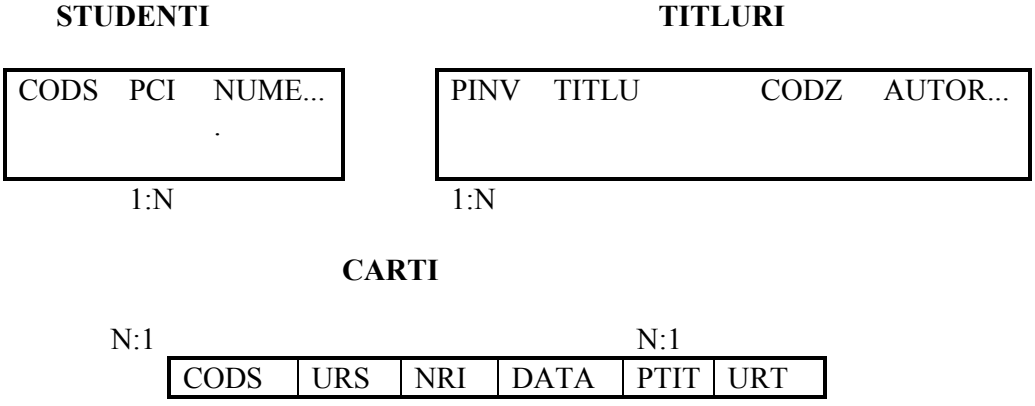


Fig.4

Fisierul STUDENTI este indexat dupa CODS si NUME. El refera prin pointerul PCI (pointer carti imprumutate) prima carte din lantul de carti imprumutate de un student, avind pointerul de inlantuire URS. Cartile se identifica prin numarul de inventar NRI, dupa care fisierul este indexat. Cartile neimprumutate, au cimpurile CODS si DATA completate cu spatii, iar cele imprumutate contin codul studentului si data cind s-a imprumutat.

Fisierul TITLURI contine datele comune pentru toate cartile cu acelasi titlu (titlu,autor,nr.pagini,cod limba,..).Pointerul PINV (pointer nr.inventar) refera prima carte din lantul cartilor cu acel titlu, care se inlantuie intre ele prin pointerul URT. Pointerul PTIT face

o referinta de la o carte, identificata prin NRI, spre inregistrarea care contine titlul cartii. Toti pointerii folosesc numarul inregistrarii din fisier a inregistrarii referite, si utilizeaza -1 pentru a indica sfirsitul de lant.

Identificarea titlurilor dupa domeniu se face utilizind codul de zecimal CODZ de clasificare, dupa care fisierul titluri se indexeaza.

In fig.5 se prezinta structura BD si modul de inlantire:

STUDENTI

CODS	PCI	NUME....
123	10	Pop Gh
519	-1	Dumitru T

TITLURI

	PINV	TITLU	CODZ	AUTOR...
15	10	Memorii		
29|	17	Timpul		

CARTI

	CODS	URS	NRI	DATA	PTIT	URT
10	123	17	4563	021293	15	20
17	123	25	4598	241293	29	25
20			4634		15	-1
25	123	-1	5678	100194	29	134

Fig.5

Prezentam mai jos programul principal si procedurile:

```
*********************************
* PBIBL * Program gestiune biblioteca
* I.Jian dec.1993
*********************************
set exact on
set talk off
clear
mes2='Selectati cu sageti sau prima litera '
defi popup b1 from 5,5 to 15,40 mess mes2
defi bar 1 of b1 prompt 'OPERATII CU CARTI' skip
defi bar 3 of b1 prompt 'Adaugare titluri '
defi bar 4 of b1 prompt 'Imprumutare carti '
defi bar 5 of b1 prompt 'Carti imprumutate '
defi bar 6 of b1 prompt 'Restituire carti '
defi bar 7 of b1 prompt 'Terminat '
on select  popup b1 do pselect
activ popup b1
return                          && **sfirsit program principal**
********
PROCEDURE PSELECT
```

```
DO CASE
        case bar()=3
   do atitl
        case bar()=4
   do cauts                    && Imprumut carti unui student
        case bar()=5
   do cauts                    && Afisare carti imprumutate
        case bar()=6
   do cauts                    && Restituire carti
        case bar()=7
   close all
   DEACTIV POPUP
   return
ENDCASE
RETURN
**********************************
* CAUTS * Cautare un student dupa cod
**********************************
PROCEDURE CAUTS
SELECT 1
use stud index icods alias st
use carti index inri in 2 alias ca
use titluri in 3 alias tl
 SELECT 2
set relation to ptit into tl
  SELECT 1
r1='d'
        DO WHILE r1 $ 'DdYy'
clear
vcod =space(5)
@ 1,1 say 'Cod student ' get vcod
read
vcod =trim(vcod)
seek vcod
if eof()
  ?'Studentul cod: '+ vcod+' nu exista '
  wait
  loop
endif
  DO CASE
 case bar()=4
set exact on
do imprumut                    && imprumut carti
 case bar()=5
set exact off
do afis_c                      && afisare carti imprumutate
 case bar()=6
set exact on
do rest_c                      && restituire carti
  ENDCASE
```

```
 SELECT 1
wait 'Continuati pt. alt student ?d/n ' to r1
        ENDDO
clear
close all
set exact off
RETURN                  && * sfirsit CAUT_S
* ***************************
```

*** IMPRUMUT * imprumut carti**

```
***************************
PROCEDURE IMPRUMUT
R2='d'
        DO WHILE r2 $ 'DdYy'
clear
* n=pci                  && pointer carti imprumutate
select 2                && fisier carti (volume)
vnri=0
@ 2,2 say 'Nr.inventar carte: ' get vnri
read
* seek vnri             && cautare carte
*  if eof()
  if vnri >recc() .or.vnri <0
 ? 'Nr.inventar: ', vnri, ' inexistent'
wait
return
  endif
go vnri
  if cods # space(5)
? 'Cartea e imprumutata studentului cods: ' ,cods
d1=seek(cods,1)
? st->nume,'  adresa: ',st->adresa
wait
return
  endif
repl urs with st->pci,cods with st->cods,data with date()
repl st->pci with recno()       && actualizare pointeri
? 'S-a inregistrat ',nri,tl->titlu,tl->autor
?   '  pt.studentul: ',st->nume,'  cods:',st->cods
wait 'Mai sint carti pentru acelasi student? d/n ' to r2
        ENDDO
RETURN                  && * sfirsit imprumut carti
************************************
```

*** AFIS_C * afisare carti imprumutate**

```
************************************
PROCEDURE AFIS_C
        DO WHILE cods =vcod       && se pot afisa pe grupuri
clear
@ 2,0 say 'Studentul '+Nume +'cods: '+cods +'  adr:  '+adresa
  if pci=-1
@ 3,5 say 'Nu are carti imprumutate! '
```

```
wait
skip
loop
  endif
@  3,5 say 'A imprumutat cartile: '
n=pci                          && pointer spre carti
select 2                       && fisierul carti
  DO WHILE n # -1
go n                           && pozitionare pe carte
* afisare titluri carti imprumutate
? nri,'  ',tl->titlu,'  ',tl->autor
n=urs                          && urmatoarea carte   ENDDO
select 1
wait
skip                           && urmatorul student din grup
        ENDDO
RETURN                         && * sfirsit AFIS_C
**********************************
```

*** ATITL * adaugare titluri carti**

```
**********************************
PROCEDURE ATITL
use carti in 2                 && fisier carti
use titluri in 3               && fisier titluri carti
 SELECT 3
r1='d'
        DO WHILE r1 $ 'DdYy'
clear
append blank
@ 1,10 say 'ADAUGARE TITLURI CARTI'
@ 3,2 say 'Titlul cartii: ' get titlu
@ 4,2 say 'Autori: ' get autor
@ 5,2 say 'Cod zecimal: ' get codz
@ 6,2 say 'Cod limba: ' get codl
read
n= recno()                     && Nr.inregistrare titlu adaugat
 SELECT 2
*  Se genereaza nr. de exemplare in fis. carti
go bottom
p=-1                           && indicator urmatoarea inregistrare
ni=nri+1                       && nr.inventar urmator dupa ultima carte
ne=0                           && contor nr.exemplare
@ 8,5 say 'Nr. exemplare: ' get ne valid ne>0
read
  DO WHILE ne>0                && generare numere inventar
append blank
repl nri with ni               && nr inventar urmator
repl ptit with recno(3)        && pointer spre titlul cartii
repl urt with p                && urmatoarea carte cu acelasi titlu
? 'Nr.inventar: ',nri          && nr inventar generat
ni=ni+1
```

```
ne=ne-1                         && contor cicluri -nr exemplare
p=recno()                       && adresa inregistrare (precedenta)
  ENDDO
* repl tl->pinv with p          && pointer cap de lant in fis.titluri
SELECT 3
repl pinv with p
wait 'Mai adaugati titluri? d/n' to r1
          ENDDO
wait ' Reindexati fisierul titluri? d/n ' to r1
  if r1$'DdYy'
index on codz to icodz
  endif
close all
clear
return
* sfirsit * ATITL
**************************************
```

*** REST_C * Restituire carti imprumutate**

```
**************************************
PROCEDURE REST_C
r2='d'
n=pci             && pointer prima carte imprumutata
  SELECT 2        && fisier carti
          DO WHILE r2 $ 'DdYy'
clear
  if st->pci=-1
? 'Studentul ',st->cods,'  nu mai are carti imprumutate '
wait
return
  endif
input 'Nr.inventar carte: ' to vnri
n=st->pci                       && prima carte
n1=1                            && este primul din lant
gasit =.n.                      && indicator articol gasit
  DO WHILE n # -1               && parcurgere lant
go n
  IF nri # vnri
preced=n                        && articol precedent in lant
n1= n1+1                        && nu e primul
n=urs                           && urmatorul articol
loop
 ELSE
gasit =.y.
? ' Cartea ','Nri: ',nri,' RESTITUITA '
? tl->titlu,'  autor: ',tl->autor
m=urs                           && salvare pointer urmator
repl cods with space(5), urs with 0     && initializari cimpuri
exit                            && terminare cautare
  ENDIF
  ENDDO
```

```
 IF .not. gasit
? ' Nr. inventar: ',vnri,' gresit '
wait 'Continuati restituire pt. acelasi student? d/n ' to r2
loop
 ENDIF
* cartea s-a gasit
  IF n1=1                    && articolul sters este primul in lant
repl st->pci with m
  ELSE
go preced
repl urs with m              && refacere lant
  ENDIF
wait 'Continuati restituiri pt. acelasi student?d/n ' to r2
        ENDDO
RETURN
* sfirsit REST_c
*******************************
```

Alaturat se prezinta structura si continutul fisierelor dupa utilizare, meniul si rezultatele obtinute pe ecran.