Medii de Programare

Note de Curs

Conf. dr. ing. Sorin Borza

INTRODUCERE

Modelul relaţional al bazelor de date a fost formulat şi publicat de către cercetătorul american dr. Edgar Cood la începutul anilor ‘70. Model criticat la început datorită unor imperfecţiuni ale sale, s-a impus treptat pe piaţa bazelor de date, astfel că în prezent ocupă cel mai important segment din aceasta. Datele şi relaţiile sunt reprezentate printr-o structură logică numită relaţie. O relaţie poate fi reprezentată sub forma unei tabele bidimensionale sau a unui fişier cu înregistrări. Modelul relaţional a fost definit riguros din punct de vedere matematic, el constituind un mediu permanent de studiu al proprietăţilor logice ale unui sistem de baze de date. Edgar F. Codd a publicat un set de 13 reguli care determină dacă un **Sistem de Gestiune a Bazelor de Date** (**SGBD**) este relaţional. Ulterior până în anii ’90, Codd apublicat un număr de aproximativ 100 de reguli. În general nici un SGBD pus în vânzare nu respectă în totalitate aceste reguli, ci doar un număr mai mic sau mai mare al lor, dar acest lucru nu împiedică ataşarea proprietăţii relaţional, produsului respectiv.

Visual FoxPro face parte din acea categorie de aplicaţii cunoscute sub numele de **Sisteme de Gestiune a Bazelor de Date Relaţional Obiectuale** (**SGBDRO**). Acesta este ultimul tip de baze de date în care suprtul de memorare al datelor este relaţional iar limbajul de programare este obiectual. Visual FoxPro pune la dispoziţia utilizatorilor utilitare complexe care ajută la crearea relaţiilor (tabelelor), a indecşilor, la introducerea datelor şi la obţinerea rapoartelor pe baza lor. FoxPro oferă un mecanism rapid şi flexibil de manipulare a fişierelor de date (fişierele cu extensia DBF) sau a bazelor de date compuse din mai multe fişiere cu extensia DBF.

Vechiul FoxPro sub DOS sau sub Windows este un dialect a ceea ce este cunoscut ca limbajul **Xbase**. Limbajul procedural din Foxpro, care este şi elementul său central, este puternic şi flexibil având de asemenea un înalt grad compatibilitate cu aplicaţiile Xbase.

Visual FoxPro nu pune la dispoziţie numai un compilator şi un mecanism de manipulare a bazelor de date, ci furnizează şi un set bogat de utilitare puternice de proiectare, încorporate într-un mediu omogen, atât de puternic şi de comod încât majoritatea proiectanţilor de aplicaţii FoxPro preferă să-şi realizeze aplicaţiile în interiorul său.

Limbajele de programare Xbase, din care face parte şi FoxPro, reprezintă o familie de limbaje de programare descendente ale limbajului dBase II produs de firma Ashton-Tate. Limbajul este caracterizat printr-o sintaxă uşor de însuşit, apropiată de engleza curentă şi printr-un bogat set de comenzi, orientate pe înregistrări, pentru manipularea la nivel înalt al datelor. Originea limbajului Xbase este legată de un limbaj brevetat de manipulare a datelor pe calculatoare, creat pentru Laboratorul de

Propulsie cu Reacție (JPL – Jet Propulsion Laboratory) din cadrul NASA la sfârşitul anilor ’70. Acest limbaj a fost cunoscut sub numele JPLDIS, fiind ulterior portat sub numele de Vulcan, pe sistemul de operare CP/M. Sesizând posibilitățile comerciale ale produsului, Wayne Ratliff a părăsit JPL şi a trecut la firma Ashton-Tate unde a elaborat programul dBase II care a fost o realizare deosebită ajutând la dovedirea utilității microcalculatoarelor. În momentul în care a apărut calculatorul IBM PC, dBase II a fost portat pe această platformă de către echipa lui Ratliff, apărând astfel produsul dBase III, o versiune cu mult mai performantă decât ceea ce exista pe piața bazelor de date în acel moment. În jurul anilor ’80 produsul firmei Aston-Tate domina aproximativ 75% din piața bazelor de date pe microcalculatoare. Pe măsură ce aplicațiile Xbase au devenit mai complexe, a crescut, ca o cerință, şi viteza de lucru a produsului. Firma Ashton - Tate a fost la început indiferentă la acest aspect, permițând altor companii care au început să imite limbajul dBase III să se dezvolte pe această direcție.

Una dintre aceste companii a fost Fox Software din Perrysburg Ohio. Primul lor produs **FoxBase**, a fost o imitație dBase II. Fox şi-a creat rapid o reputație bazată pe viteză şi pe maxima compatibilitate cu limbajul dBase. Un alt producător, Nantucket, a produs compilatorul Clipper care a pus mai mult accentul pe crearea programelor executabile pure, urmând propria lor cale, cu extensii nestandard ale limbajului.

În anul 1988, Ashton-Tate începea să piardă piața nu numai față de produsele imitație, dar şi față de numărul din ce în ce mai mare de sisteme de baze de date non-dBase. În acest timp Fox Software lucra pe cont propriu la un nou produs, cunoscut sub numele de **FoxPro**. La târgul Fall Comdex din 1989, Ashton-Tate a anunțat lansarea oficială a produsului dBase IV, care a fost aproape imediat caracterizat de critici ca având deficiențe iremediabile. Tot la acest târg demonstrațiile care sau făcut au scos în evidență faptul că produsul care urma să fie lansat, **FoxPro 1.0**, era mult mai rapid şi avea o serie de avantaje majore asupra proaspătului lansat dBase IV.

La câteva zile după aceste evenimente firma Ashton-Tate a intentat proces împotriva produsului Fox, în sensul că produsul încălca dreptul de copyright al firmei Ashto-Tate asupra produsului dBase III. Acest proces a avut ca efect distrugerea firmei Ashton-Tate şi o mare publicitate pentru firma Fox Software. Firma Borland Internațional a achiziționat Ashton-Tate şi produsul său principal dBase IV, trecând la modificarea şi îmbunătățirea produsului.

FoxPro şi-a dobândit propria sa personalitate o dată cu versiunea 2.0, apărută în iulie 1991. Cu versiunea 2.0 Fox şi-a consolidat poziția sa de lider în inovația tehnologică din domeniul Xbase. În vara anului 1992, compania Fox Software a fost achiziționată de către concernul Microsoft. Produsul FoxPro a continuat să fie dezvoltat sub mediul Dos, apărând versiunile 2.5 şi 2.6. Sub patronajul Microsoft produsul FoxPro a migrat în primul rând spre mediul Windows prin versiunile **FoxPro 2.6 for Windows** şi ulterior **Visual FoxPro** versiunile 3.0, 5.0, 7.0, 8.0 şi în curând 9.0.

O schimbare majoră în **Visual Fox**, față de **Fox Pro**, constă în limbajul de programare orientat obiect, față de limbajul structurat clasic corespunzător versiunii sub DOS. Din motive de compatibilitate în Visual Fox poate fi folosită programarea structurată, instrucțiunile din FoxPro 2.6 fiind compatibile 100% cu mediul Visual Fox.

O altă schimbare majoră constă în trecerea la elemente de programare vizuală. Deşi în versiunile 2.6 pentru DOS şi Windows existau elemente de programare visuală proprii Generatoarelor, în Visual Fox aceste elemente sunt adaptate modelului orientării pe obiecte şi modelului programării conduse de evenimente.

Alte modificări majore se referă la modul de concepere a entităților pentru memorarea datelor. Pe lângă clasicul tabel sau relație în Visual Fox apare **baza de date de tip container**, ca element important de memorare a datelor. Aceasta oferă facilități în plus pentru validarea datelor, atât la nivel de câmp cât şi la nivel de înregistrare.

Visual FoxPro oferă suport pentru tehnica **OLE** de tragere şi plasare (drag-and-drop), instrument care permite transferul de date între VFP, Visual Basic, Windows Explorer, Microsft Word şi Excel etc. VFP permite ca datele construite şi administrate în acest mediu, să fie folosite în alte medii de programare care nu au ca limbaj de programare VisualFox sau cu alte cu alte cuvinte VFP este furnizor OLE DB.

Începând cu versiunea 7.0 s-a introdus tehnologia de asistare a programatorului în construirea instrucțiunilor denumită **IntelliSense.**

Visual Fox permite folosirea serviciilor Web, care reprezintă clase disponibile pe Internet, create de alții şi posibil să fie folosite în propriile aplicații.

Visual Fox permite folosirea unor facilități legate de serverele **COM**, permite conversia datelor în format **XML** dintr-o tabelă şi invers, oferind astfel utilizatorului Visual FoxPro posibilitatea publicării datelor pe **Web**.

Majoritatea limbajelor actuale referitoare la domeniul bazelor de date, fie că sunt obiectuale sau procedurale au incorporate în ele limbajul **Structured Query Language** (**S.Q.L.**).

S.Q.L a fost conceput ca un limbaj standard de descriere a datelor şi acces la informațiile din bazele de date, ulterior dezvoltându-se ca o adevărată tehnologie dedicată arhitecturilor client/server. Se reaminteşte că o bază de date este formată dintr-o colecție de date divers şi un software care să controleze accesul la acestea cunoscut sub numele de **Sistem de Gestiune al Bazelor de Date** (**SGBD**). Orice SGBD modern furnizează un **limbaj de interogare al datelor** de obicei de tip SQL (**Structural Data Language**) sau cel puțin câteva instrucțiuni SQL ce permit utilizatorilor să obțină mult mai rapid şi mai simplu informațiile dorite, decât dacă ar utiliza un program scris într-un limbaj procedural. Limbajele de interogare se pot împărți în două categorii:

a) **limbaje algebrice**, în care interogările asupra relațiilor sunt exprimate prin intermediul unor operatori aplicați asupra lor. În această categorie se încadrează limbajul SQL;

b) **limbaje bazate pe calcul relațional**, în care interogările se bazează pe diverse condiții puse asupra tuplelor existente în relații.

SQL este un limbaj standard de descriere a datelor şi acces la informațiile din bazele dedate, ulterior dezvoltându-se ca o adevărată tehnologie dedicată arhitecturilor client/server.

Utilizat inițial de către firma I.B.M pentru produsul DB2, S.Q.L a devenit la mijlocul deceniului trecut un standard de facto în domeniul bazelor de date. În această perioadă au fost realizate şapte versiuni ale standardului S.Q.L, trei dintre acestea fiind concepute de către Institutul National American de Standarde (ANSI), iar celelalte de către firmele de software IBM, Microsoft Borland. În prezent, ANSI lucrează la standardul S.Q.L 3 care va introduce bazele de date orientate pe obiect. Primul standard SQL a fost creat în 1989 de către ANSI fiind revizuit în 1992. Proliferarea diverselor implementări ale limbajului SQL a determinat formarea consorțiului SAG (The SQL Access Group), care are drept scop conceperea de standarde SQL, având la bază standardul ANSI-SQL'89.

În timp, diverşi membrii ai SAG şi-au dezvoltat propriile standarde SQL, astfel:

- Microsoft a dezvoltat produsul ODBC (**Open Database Connectivity**);
- Borland a dezvoltat standardul IDAPI (**Integrated Database Application Programming Interface**)

În capitolele următoare se va face referire la limbajul VisualFoxPro şi la comenzile **SQL** incorporate în acest produs. La prezentarea comenzilor şi funcțiilor Visual FoxPro se vor folosi o serie de convenții, preluate de la firma producătoare şi care sunt prezentate în continuare:

- **cuvintele cheie** ale limbajului sunt prezentate cu majuscule şi îngroşat. Cuvintele cheie sau cuvintele rezervate sunt construcții sintactice proprii mediului Visual FoxPro;
- **parantezele unghiulare** încadrând text scris cu litere mici indică o porțiune ce trebuie furnizată de programator, aceasta fiind simbolizată prin textul dintre paranteze;
- **parantezele rotunde, virgula, egalul**, vor fi trecute în instrucțiune, comandă sau funcție, exact în poziția în care apar în definiție;
- **parantezele pătrate** încadrează o porțiune opțională;
- **bara verticală** indică alegerea unei opțiuni şi numai a uneia, din mai multe posibile;
- **trei puncte de suspensie**, indică faptul că funcția, comanda sau instrucțiunea se continuă în mod asemănător.

Exemplu sintaxă

```
DIMENSION<numetablou1>(<expN1>[,<expN2>])
     [,<numetablou2>(<expN3> [,<expN4>)]…
     FOR <var>=<expN> TO <expN2> [STEP <expN3>]
```

```
        [instrucțiuni]
        [EXIT]
        [LOOP]
    ENDFOR|NEXT
```

Exemplu program

```
USE SALARIAT
DISPLAY
DISPLAY ALL OFF
GO RECORD 3
DISPLAY FIELDS NUME,PRENUME,SALAR
DISPLAY FIELDS SALAR REST
GO TOP
DISPLAY ALL FOR STARE_CIVL=.F.
DISPLAY ALL TO FILE SALAR.TXT NOCONSOLE
CLOSE ALL
```

Capitolul

1

INTRODUCERE ÎN VISUAL FOXPRO

1.1 Folosirea mediului propriu Visual FoxPro

Mediul Visual FoxPro versiunea 7 se prezintă în figura 1.1. Visual FoxPro defineşte ecranul ca o clasă specială denumită **Screen.** Visual Fox este un mediu de programare orientat obiect. Începând cu versiunea 6.0 există posibilitatea de a declara baza de date, tabela, vederea sau legătura între tabele ca şi o clasă de obiecte, putând astfel gestiona datele ca şi obiecte. Din acest punct de vedere, Visual Fox se comportă ca o bază de date relaţîonal obiectuală.

Visual FoxPro este un mediu orientat pe evenimente, adică se poate trece de la o activitate la alta, aproape în orice moment. De exemplu dacă suntem într-o fereastră de editare a unui fişier de date, fără să închidem această fereastră se poate deschide o fereastră de editare program. Se poate sări oricând înapoi la fereastra de editare a fişierului de date fără a închide fereastra de editare program. Principalele elemente ale mediului Visual FoxPro (VFP) sunt :

- Meniul mediului VFP sau meniul sistem;
- Bara cu butoane (Toolbars)
- Fereastra de comandă (Command).

1.1.1 Bara de meniuri

Bara de meniuri se prezintă în figura 1.1. Fiecare cuvânt care apare în bara de meniuri reprezintă o opţiune bară sau simplu opţiune. Meniul se mai numeşte şi

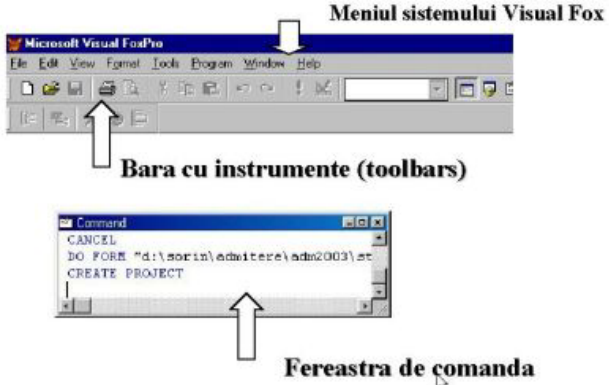


Fig 1.1 Mediul integrat VFP

meniu principal sau meniul sistem.

Părțile componente ale sistemului de meniuri sunt prezentate în figura 1.2.

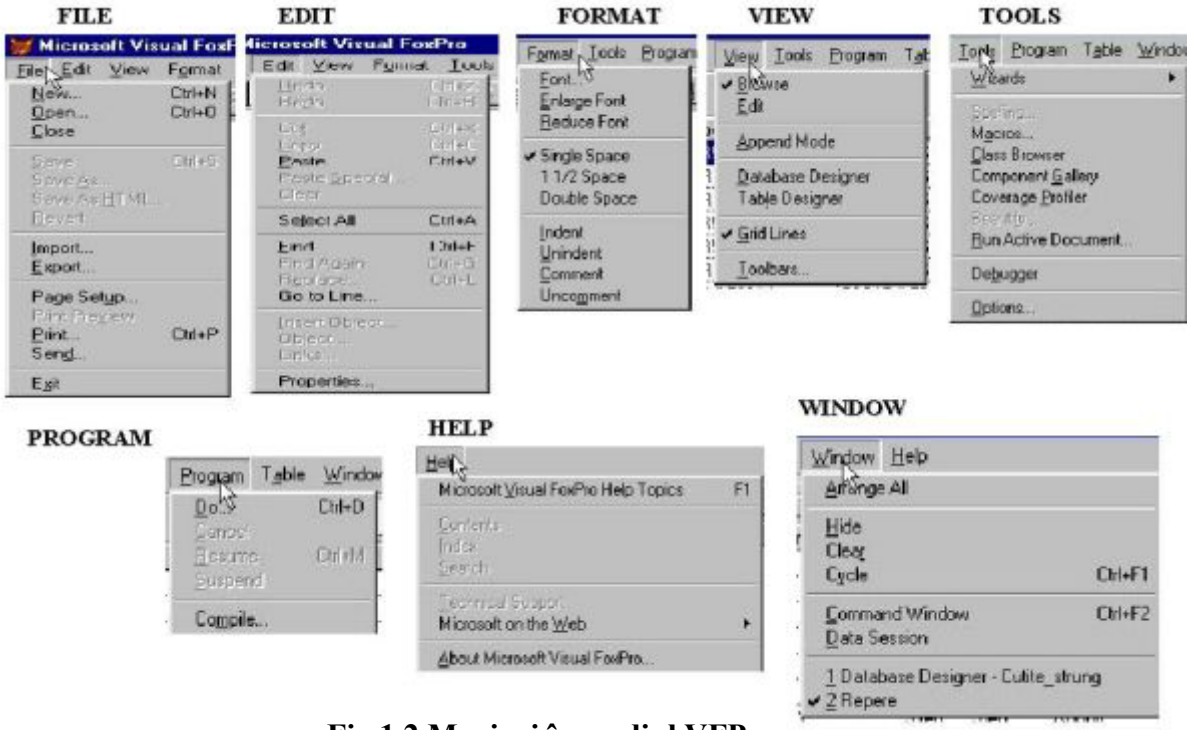


Fig 1.2 Meniuri în mediul VFP

Pentru a selecta o anumită opțiune se apasă tasta Alt în combinație cu tasta subliniată a numelui opțiunii respective. Pentru apelarea opțiunii File se apasă concomitent tastele Alt+F. Submeniul, este lista care apare atunci când selectați o opțiune bară. Pentru a selecta o opțiune a unui submeniu se deplasează cursorul cu ajutorul tastelor poziționale pe opțiunea dorită şi se tastează Enter (↵). Se poate de asemenea tasta litera subliniată (sau supraluminată) asociată unei opțiuni a unui submeniu.

Unele selecții din submeniuri au o cale directă de apelare de la tastatură, afișată în dreapta opțiunii. De exemplu apelarea opțiunii **Help** se poate face prin apăsarea tastei funcționale F1.

Unele opțiuni sunt urmate de trei puncte (…). Acestea reprezintă o indicație vizuală a faptului că, după selectarea lor urmează o fereastră de dialog.

Opțiunile meniului sistem vor fi explicate pe parcursul acestei cărți în funcție de capitolul în care vor fi tratate.

1.1.2 Configurarea mediului Visual Fox

Visual Fox permite utilizatorului individual să realizeze configurarea particularizată a mediului de lucru funcție de dorința acestuia. Acest lucru se poate realiza folosind meniul **Tools** şi opțiunea **Options** a acestuia. Fereastra care apare (fig. 1.3) permite definirea în totalitate a caracteristicilor mediului de lucru în VFP.

Ea conține 13 pagini distincte care împart setul de comenzi în următoarele grupuri:

- **Controls** stabileşte legăturile cu bibliotecile de clasele vizuale şi cu controalele OLE;
- **Data** defineşte accesul regăsirea şi afişarea datelor;
- **Editor** defineşte caracteristicile editorului de text VFP;
- **File Locations** defineşte căile de localizare a diferitelor fişiere existente în VFP;
- **Forms** determină configurarea proiectantului de formulare;
- **General** permite dfinirea anumitor opțiuni ale mediului VFP care nu se regăsesc în celelalte pagini ale ferestrei *Options*;
- **Regional** defineşte modul de formatare al datei sau al numerelor funcție de zona geografică;
- **Projects** defineşte opțiunile managerului de proiect;
- **Remote data** stabileşte modul de legătură al VFP cu datele înderpătate, în cazul lucrului în rețea;
- **View** defineşte trăsăturile barei de stare şi posibilitatea de a folosi proiectele recent folosite;
- **Field Maping** permite definirea în cazul folosirii proiectantului de formulare (Form Designer), care obiect clasă de bază este asociat cu un anumit tip de dată;
- **IDE** permite definirea caracteristicilor de sintaxă precum şi a ferestrelor sau meniurilor;
- **Debug** defineşte caracteristicile depanatorului de programe din mediul VFP.

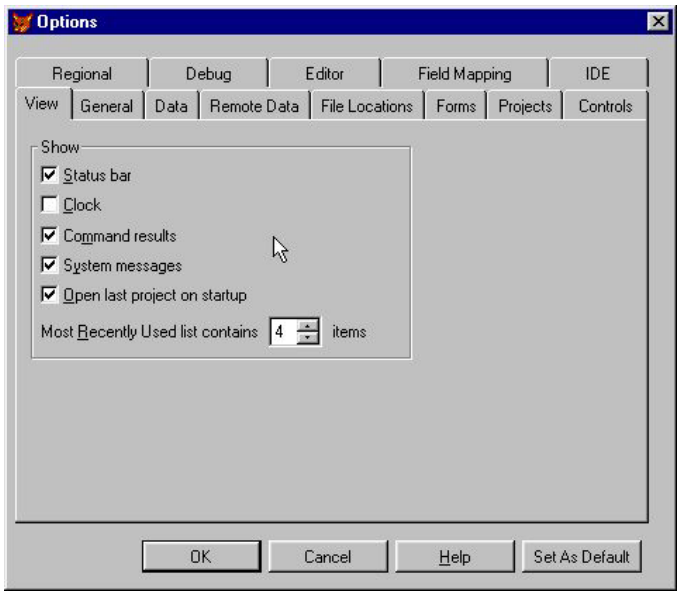


Fig 1.3 Fereastra "Options" a meniului Tools

În continuare vom prezenta câteva dintre cele mai uzuale pagini folosite la configurarea mediului.

Pagina **Data** a ferestrei **Options** din meniul **Tools** este prezentată în figura 1.4. Ea permite stabilirea unor caracteristici cu caracter global referitoare la datele existente în entitățile de memorare VFP.

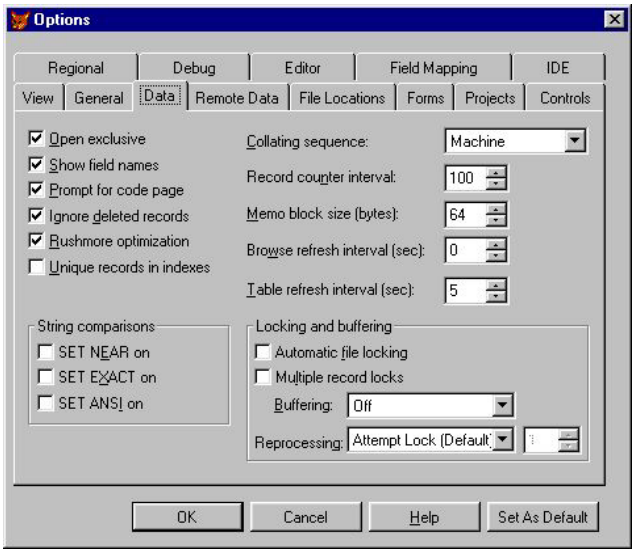


Fig 1.4 Pagina Data a ferestrei Options

Butonul:

- **Open exclusive** permite stabilirea modului de deschidere a unui fişier într-un mediu partajat. În mod exclusiv controlul este cedat în totalitate utilizatorului care a deschis fişierul;
- **Ignore deleted records** permite ignorarea înregistrărilor marcate pentru ştergere în toate comenzile Visual Fox care fac referire la ele;
- **Rushmore optimisation** permite folosirea tehnicii de căutare rapide Rushmore. Uneori se poate renunța la această tehnică. În general comenzile VFP care o folosesc au şi opțiuni separate de dezactivare;
- **SET NEAR** stabileşte ce face VFP atunci când o căutare eşuează. Dacă opțiunea nu este selectată pointerul de înregistrări este poziționat pe sfârşit de fişier. Dacă opțiunea este selectată atunci pointerul de înregistrări se poziționează pe următoarea înregistrare în ordine alfabetică de la poziția unde ar fi trebuit să existe valoarea căutată;
- **SET EXACT** se referă la modul în care sunt evaluate criteriile de căutare. Dacă această opțiune este setată căutarea se realizează caracter cu caracter pe toată lungimea şirului de caractere. Dacă opțîunea nu este selectată evaluarea se face caracter cu caracter pe lungimea termenului din partea dreaptă a expresiei de căutare;
- **SET ANSI** defineşte modul în care se realizează în SQL comparațiile între şiruri. Când opțiunea este selectată, se complectează şirul mai scurt cu blancuri pentru a egaliza cele două şiruri. Apoi se compară cele două şiruri caracter cu caracter pentru a vedea dacă ele coincid. Dacă opțiunea nu este selectată comparația se face caracter cu caracter, pe lungimea şirului mai scurt;

- Grupul de comenzi **Locking and buffering** are ca scop lucrul cu buferele într-un mediu partajat, adică gospodărirea datelor într-un mediu multiutilizator.

Pagina File Locations (fig 1.5), permite localizarea unor anumite grupe de fişiere:

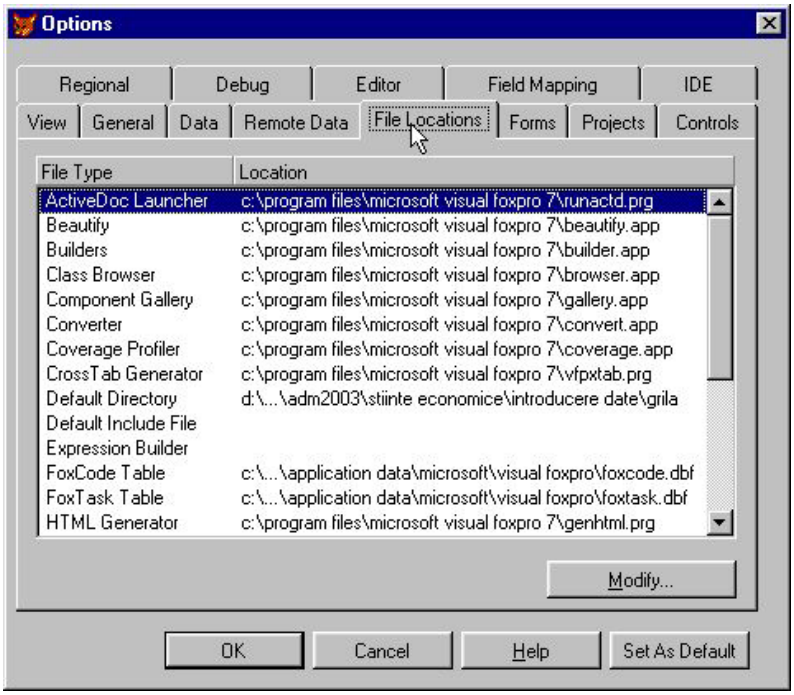


Fig 1.5 Pagina File Locations

- **Default Directory** permite definirea folderului de lucru curent;
- Din această fereastră pot fi definite locațiile pentru alte categorii de fişiere ca de exemplu: fişierul Help, fişierul de resurse, care păstrează informații despre modul în care lucrați, generatorul HTML, corectorul ortografic etc

1.1.3 Fereastra de comandă

În figura 1.1 se pune în evidență un element al mediului Visual FoxPro denumit fereastra de comenzi (**Command**). Aceasta interacționează direct cu sistemul, în ea se poate scrie orice fel de comandă specifică produsului Visual FoxPro. O instucțiune introdusă în fereastra **Command** rămâne acolo până la părăsirea mediului, astfel încât fereastra funcționează şi ca un istoric al comenzilor introduse. Prin deplasarea cu ajutorul tastelor direcționale se poate alege comanda anterioară tastată. Cu ajutorul opțiunilor din meniul **Edit:**

- copy;
- cut;
- paste;

se pot copia, sau muta blocuri de informații, în diverse ferestre Fox active.

1.2 Editorul de texte din Visual FoxPro

Programele scrise de un programator sunt denumite fişiere program, ele sunt fişiere text care conţin instrucţiuni ale limbajului Visual FoxPro. Aceste fişiere au extensia PRG şi deseori sunt denumite fişiere PRG sau fişiere sursă. În mediul Visual FoxPro există două tipuri de fişiere program generate de sistem. Acestea sunt prezentate în tabelul 1.2.

Tabelul 1.2

Generat de	Extensia
Constructorul de meniuri	.MPR
Interogări relaţionale (RQBE)	.QPR

Fişierele program, precum şi fişierele text generate de sistem, pot fi scrise şi modificate cu ajutorul editorului propriu al mediului Visual FoxPro.

Editorul mediului Visual FoxPro se poate porni în mai multe moduri. Din submeniul **F**ile, opţiunea **New** sau **Open,** apoi se specifică **Program** (pentru a edita un fişier program) sau **File** (pentru editarea unui fişier text obişnuit). Visual FoxPro va solicita numele fişierului.

Dacă se porneşte editorul Visual FoxPro dintr-un program sau din fereastra de comenzi, atunci se vor folosi instrucţiunile:

- **MODIFY FILE <nume fişier>** pentru editarea oricărui tip de fişier text;
- **MODIFY COMMAND <nume program>** pentru editarea unui fişier program.

Instrucţiunea (comanda) **MODIFY COMMAND** se deosebeşte de instrucţiunea **MODIFY FILE** sub două aspecte. În primul rând dacă nu se specifică o anumită extensie, comanda **MODIFY COMMAND** crează implicit pentru fişierul program extensia **PRG**, pe când **MODIFY FILE** crează implicit un fişier cu extensia **TXT.** În al doilea rând, Visual FoxPro ţine evidenţa modificărilor făcute cu **MODIFY COMMAND,** executând întodeauna versiunea cea mai nouă, modificată.

Dacă se tastează **MODIFY COMMAND <nume_fişier>** se va deschide fereastra editorului de texte. Textul se introduce caracter cu caracter de la tastatură, la sfârşitul fiecărei linii se tastează Enter. Pentru a muta cursorul în cadrul ferestrei se folosesc tastele sau combinaţiile de taste conform tabelului 1.1.

Tabelul 1.3

Combinaţie Taste	Mută Cursorul
Săgeată dreapta →	Un caracter la dreapta
Săgeată stânga ←	Un caracter la stânga
Săgeată sus ↑	O linie în sus
Săgeată jos ↓	O linie în jos
Page Up PgUp	O fereastră(pagină) de text în sus
Page Down PgDn	O fereastră(pagină) de text în jos
Home Home	La începutul liniei curente
End End	La sfârşitul liniei curente
Ctrl+→	Un cuvânt la dreapta

Ctrl + ←	Un cuvânt la stânga
Ctrl + Home	La începutul textului
Ctrl + End	La sfârşitul textului

O facilitate importantă a editorului Visual FoxPro este posibilitatea de a selecta un număr variabil de caractere. Dacă se ține tasta Shift apăsată şi se deplasează cursorul în cadrul textului cu ajutorul tastelor sau combinațiilor de taste (tabelul 1.3) se selectează numărul de caractere dorit. Acelaşi efect se obține dacă se ține apăsat butonul stâng al mouse-ului şi se deplasează cursorul acestuia peste caracterele care se doresc a fi selectate. Dacă se doreşte selectarea unei linii de text, atunci se poziționează cursorul la începutul rândului, se apasă tasta Shift şi una dintre tastele ↑ sau ↓. Acelaşi efect se poate obține cu ajutorul mouse-ului. Se poziționează cursorul acestuia la începutul unei linii sau pe un caracter al unei linii care se doreşte a fi selectate şi se face clic de trei ori.

Ştergerea unui caracter sau a unei porțiuni de text selectat selectat se face cu ajutorul tastelor Del sau ← Bksp :

- Delete şterge caracterul pe care este poziționat cursorul;
- Backspace şterge caracterul aflat imediat în stânga cursorului.

Dacă există un text selectat la apăsarea uneia dintre aceste taste textul se şterge. Combinația Ctrl + ← Bksp şterge cuvântul deasupra căruia se află cursorul.

Textul selectat poate fi mutat sau copiat în diferite locuri ale fişierului cu ajutorul opțiunilor meniului **E**dit:

- **Cut** Încarcă textul selectat în clipboard(zonă de memorie folosită la transferurile de date), eliminându-l din textul sursă;
- **Copy** Încarcă textul selectat în clipboard fără a-l elimina din textul sursă;
- **Paste** Introduce textul încărcat anterior în clipboard, în textul sursă, la poziția cursorului.

Selectarea caracteristicilor editorului de texte se face prin opțiunea **Proper**ties a meniului **E**dit (fig 1.6).

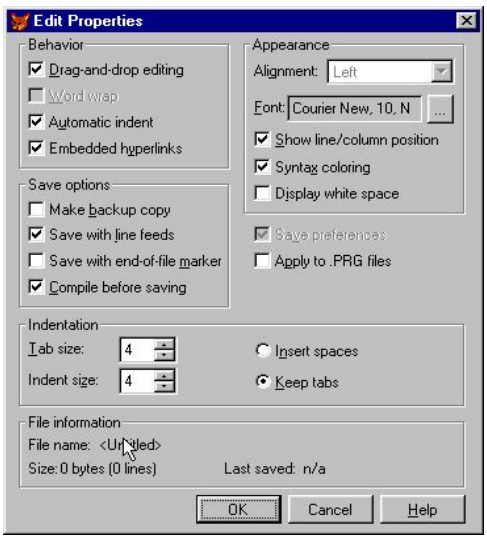


Fig 1.6 Opțiunile ferestrei Edit Properties

Opțiunile acestei ferestre sunt prezentate în tabelul 1.4

Tabelul 1.4

Opțiune	Explicații
Drag and drop editing	Editare drag and drop
Word wrap	Determină saltul automat la linie nouă când se întâlneşte marginea dreaptă a ferestrei de editare.
Automatic ident	Saltul la linia următoare se face sub primul caracter al liniei precedente.
Make backup copy	Păstreză o copie a versiunii anterioare a fişierului de editat, actualizarea acesteia făcându-se la fiecare salvare a fişierului.
Save with line feeds	Fişierul este salvat folosind "carriage return" (întoarcere la începutul liniei) şi "line feed" (salt la linie nouă) la sfârşitul fiecărei linii.
Show line/column position	Permite vizualizarea numărului de coloană sau rând în cadrul editorului
Save with and of file marker	Adaugă caracterul Ctrl+Z la sfârşitul fişierului indicând terminarea logică a acestuia.
Alignment	Specifică modul de aliniere al textului în fereastră: la stânga, la dreapta, în centrul ferestrei.
Tab size	Indică marimea pe ecran a caracterului Tab. Valoarea implicită este 4, el putând lua valori între 0-50.
Indent size	Mărimea spațiului de identare
Compile befor saving	La salvarea ferestrei de editare se face şi compilare automată
Use these preferences as default for .PRG files	Setările efectuate se folosesc implicit pentru fişierele program
Font	Alegerea fontului dorit
Syntax coloring	Foloseşte diverse culori pentru textul care însoțește editarea
Save preference	Salvează setările stabilite
Display white space	Vizualizarea spațiului care nu conține caractere
Applay to .PRG files	Se aplică fisierelor text cu extensia .PRG
Ok	Butonul de validare al ferestrei Preference
Cancel	Butonul de invalidare a setărilor efectuate în fereastra Preference
Help	Apelarea programului de ajutoe în explicarea opțiunilor ferestrei

Celelalte opțiuni ale meniului **Edit** (fig 1.7) sunt prezentate în tabelul 1.5. Aceste opțiuni sunt folosite pentru a acționa asupra textului tastat într-o fereastră de

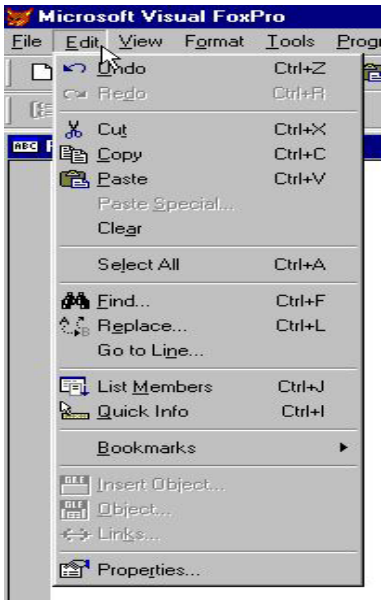


Fig 1.7 Meniul Edit

editare.

Tabelul 1.5

Opțiune	Explicații
Undo	Anulează ultima acțiune asupra unui text dintr-o fereastră de editare, care pot fi ştergerile cu tastele Del sau ←Bksp, înlocuiri de blocuri prin selectare.
Redo	Această opțiune este opusă lui **Undo** şi ea poate fi aleasă repetat pentru anularea unor comenzi **Undo** repetate.
Clear	Determină ştergerea textului selectat, fără încărcarea sa în clipboard.
Select All	Selectează tot textul existent în fereastra de editare.
Go to Line	Are ca efect activarea ferestrei din figura 1.8, prin care se poziționează cursorul pe linia dorită.
Find	Se utilizează pentru căutarea unui şir de caractere în fereastra de editare şi eventual înlocuirea acestuia cu un alt şir de caractere. Are ca efect apariția ferestrei din figura 1.9.
Cut, Copy, Paste	Comenzille clasice Windows de copiere şi mutare folosind **Clipboard**.
Paste Special	Se foloseşte pentru inserarea obiectelor **OLE** din alte aplicații într-un câmp de tip "**Genera**l".
Replace	Înlocuieşte şirul găsit cu şirul specificat prin opțiunea Find, după care caută o nouă apariție a şirului căutat. Se foloseşte frecvent pentru înlocuirea unui şir cu altul, cele două şiruri fiind definite anterior în fereastra de dialog a opțiunii **Find**
Quick Info	Informații despre diverse entități folosite în program
Bookmarks	Jaloane care pot fi puse în diverse faze ale editării unui fişier text
Insert Object	Opțiunea este asemănătoare cu **Paste Special**, doar că nu se face presupunerea că obiectul există deja şi este memorat în **Clipboard**. Opțiunea permite înglobarea obiectelor într-un câmp de tip "**General**". Aceasta este o metodă de a introduce obiecte multimedia într-un fişier de date
Object	Pemite editarea unui obieect **OLE** selectat.
Links	Permite editarea tuturor obiectelor legate.
List Members	Afişează componentele obiectului selectat.
Properties	Activează fereasta cu acelaşi nume a meniului care a fost prezentată în tabelul 1.4

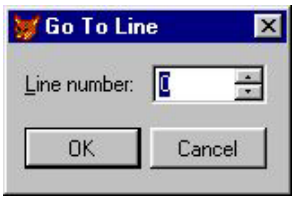


Fig 1.8 Fereastra Go To Line

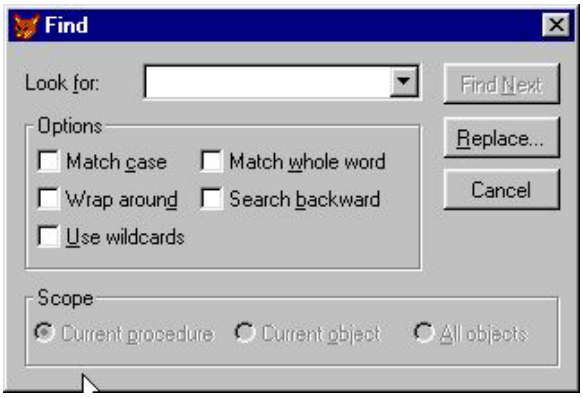


Fig 1.9 Fereastra de căutare Find

Opțiunile ferestrei de dialog Find sunt prezentate în tabelul 1.6.

Tabelul 1.6

Opțiune	Explicații
Look For	Câmp de editare text în care se introduce şirul de caractere căutat.
Replace	Buton care activează câmpul de editare, folosit pentru a specifica şirul de caractere cu care va fi înlocuit şirul de caractere căutat.
Search backward	Buton radio care determină direcția de căutare a şirului de caractere dorit, de la cursor spre începutul fişierului.
Find Next	Buton care determină căutarea înainte a şirului de caractere dorit
Math wholee word	Comutator care dacă este activ, consideră cuvântul (şirul de caractere) găsit numai atunci când acesta este de sine stătător şi dacă este inactiv atunci cuvântul (şirul de caractere) căutat poate să facă parte dintr-un alt cuvânt.
Match case	Căutarea şirului de caractere se face indiferent de formatul caracterelor (mici sau mari)
Use wildcards	Buton radio care permite folosirea caracterelor wildcard adică a caracterelor care permit definirea unor anumite actiuni, de exemplu: "<" cautarea şirului de caractere la începutul cuvântului "*" sau "?" folosite în cazul căutărilor în care sunt omise caractere
Wrap around	Căutarea şirului de caractere se face în întreg fişierul, cu depăşirea limitelor de început şi sfârşit de fişier până la poziția curentă a cursorului.
Zona Scope	Permite definirea căutărilor în: • procedura curentă (Current procedure); • obiectul curent (Current object); • în toate obiectele (All objects).

Cu ajutorul editorului de texte Visual FoxPro se poate edita şi conținutul unui câmp memo. Despre câmpurile memo se va discuta în capitolul 4.

1.3 Comenzi şi funcții în Visual FoxPro

Comenzile sau instrucțiunile limbajului vor fi prezentate în această carte cu litere majuscule. Funcțiile specifice limbajului Visual FoxPro sunt reprezentate prin mnemonica instrucțiunii urmată de caracterele “(“ şi “)”. La cele mai importante comenzi sau funcții va fi prezentată detaliat sintaxa. În cadrul sintaxei parametrii opționali se vor scrie încadrați de caracterele “[“,”]”. Expresiile numerice, cele şir de caractere şi cele logice vor fi scrise expN, expC şi expL, încadrate de caracterele “<”,”>”. Posibilitatea folosirii unui parametru în mai multe variante este marcată prin caracterul “|”.

Concluzii

Meniul sistemului Visual, este un instrument important în apelarea sau proiectarea rapidă a diferitelor entități ale mediului. Multitudinea de subopțiuni ale

acestuia permit lucrul rapid, atât în activitatea de programare, cât şi în cea de întreținere de date.

Configurarea mediului Visual Fox, este o activitate importantă, care poate fi efectuată aşa cum s-a arătat din meniul **Tools** subopțiunea **Options**. Aceasta este prima activitate care se desfăşoară atunci când se începe lucrul cu oricare mediu de programare.

Capitolul 2

MEMORAREA DATELOR ÎN VISUAL FOXPRO

2.1 Noțiuni teoretice privind bazele de date şi fişierele de date în Visual FoxPro

Visual FoxPro este un produs Microsoft. În VFP tabela este entitatea fundamentală în care se memorează datele. Spre deosebire de clasicul FoxPro, tabela care este un fişier cu extensia .DBF, poate exista ca entitate independentă (free table) sau ca entitate conținută într-o bază de date, care este un fişier cu extensia .DBC. Într-o bază de date sunt incorporate mai multe tabele relaționale, mai multe interogări, mai multe vederi şi cod program, care toate la un loc alcătuiesc un tot unitar numit **bază de date**. În continuare în această carte vom folosi termenul de fişier de date atunci când ne vom referi la o tabelă relațională sau un fişier cu extensia DBF şi termenul de bază de date când vom face referință la un fişier cu extensia .DBC.

Un fişier cu extensia DBF este o colecție de date formată din unul sau mai multe articole (înregistrări) legate între ele.

Se poate face o analogie între modul de organizare al unui tabel şi modul de organizare al fişier de date.

În primul rând pentru a defini un fişier de date este necesar să-i definim **STRUCTURA** adică caracteristicile fiecărui articol**.** În cadrul structurii unui fişier de

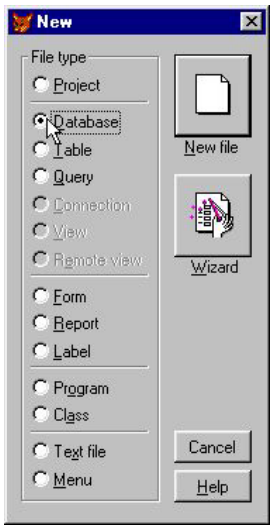


Fig. 4 1 Fereastra de creare a unei entități noi

date definim **CÂMPURILE** care compun structura şi **TIPUL** fiecărui câmp. Toate **articolele** dintr-un fişier, care se mai numesc şi **înregistrări** au aceeaşi structură.

Dacă facem analogie cu o relaţie definită de Codd, atunci fiecare **fişier de date** este o **relaţie**, fiecare **înregistrare** sau **articol** este o **tuplă**, iar fiecare **relaţie** are unul sau mai multe **atribute**, deci unul sau mai multe **câmpuri** care alcătuiesc **structura** fişierului.

Baza de date este o entitate care se crează cu comanda:

CREATE DATABASE <nume_bază de date>

Baza de date se poate crea din menoul **File**, opţiunea **New** (fig 2.1)

Baza de date poate fi creată automat într-un mod predefinit propriu VFP (Wizard) sau prin acţiunea manuală a utilizatorului (**New File**). În această vom prezenta realizarea interactivă, manuală a entităţilor, modul Wizard fiind prezentat numai în cazurile în care autorul consideră acest lucru util.

În fereastra proprie bazei de date, pot fi realizate mai multe acţiuni: creare tabelă, adăugare de tabelă, creare vedere la distanţă sau local, etc.. Acestea se pot activa din meniul contextual care apare, acţionând asupra butonului drept al mouse-

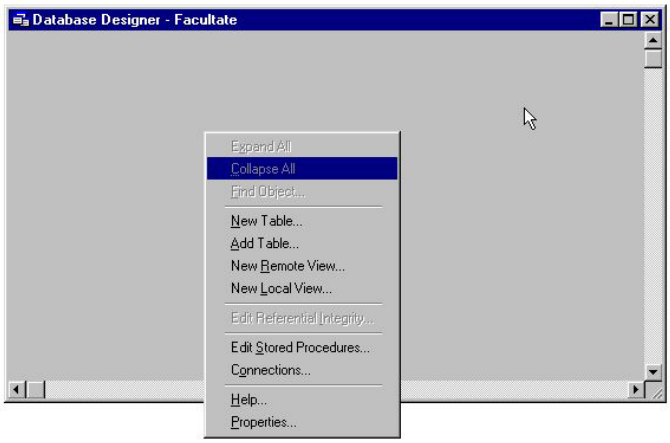


Fig. 2 2 Meniul contextual al ferestrei Database Designer

ului (fig 2.2).

Asupra tabelelor create în baza de date se pot întreprinde o serie de acţiuni, activând opţiunea **Table** a meniului principal (fig. 2.3).

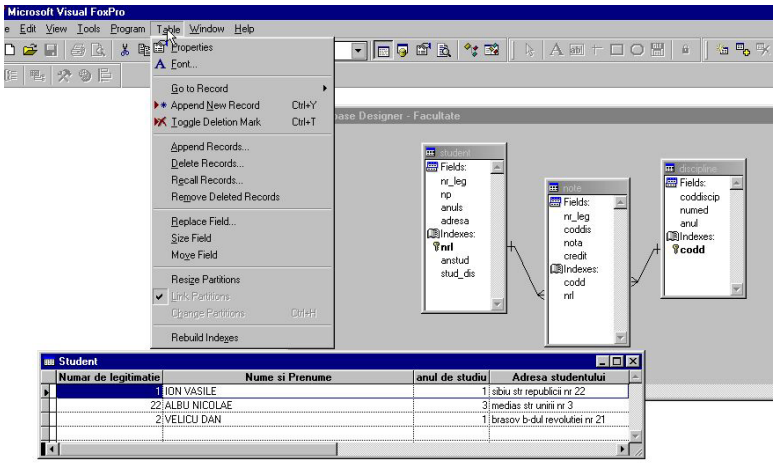


Fig. 2 3 Meniul Table prin care pot fi întreprinse diverse acţiuni asupra tabelelor din baza de date

Opțiunile meniului **Table** au denumiri identice cu comenzi Visual Fox, de acțiune asupra înregistrărilor dintr-o tabelă, care vor fi explicate în acest capitol.

Aşa cum s-a menționat anterior, o tabelă poate exista în mediul VFP în două moduri:

- Independent, fără să aparțină unei baze de date;
- Aparține unei baze de date;

Indiferent de modul de creare a unei tabel, tipurile de date care pot exista în înregistrările tabelei sunt aceleaşi (fig 2.4).

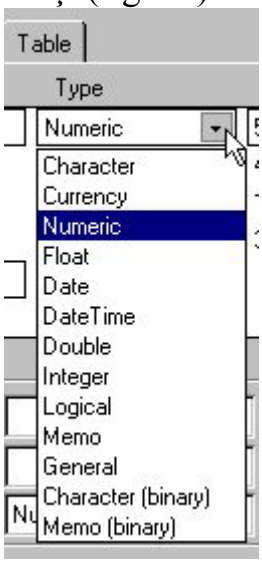


Fig. 2 4 Tipuri de date în tabele Visual Fox

2.1.2 Tipuri de date ale câmpurilor FoxPro

Fiecare câmp (atribut) dintr-un fişier de date este destinat memorării unui element individual de date. Tipurile de câmpuri acceptate de Visual FoxPro sunt:

- caracter (character);
- unități monetare (currency);
- dată calendaristică (Date, Date Time);
- numeric (Numeric, Float, Double, Integer);
- logic (Logical);
- memo;
- general ;
- character (binary);
- memo (binary).

Câmpurile **character** sunt cele mai flexibile şi cele mai utilizate. Ele conțin caractere de tip ASCII, limita de lungime impusă pentru acest tip de câmp este 254 caractere.

Tipul câmpului	Descriere	Mărime	Rang
Character	Orice caracter admis de către calculator	1 până la 254 de caractere	Orice caracter

Câmpurile unităţi monetare sau **Currency** sunt folosite pentru a reprezenta câmpuri numerice însoţite de simbolul monedei în care se lucrează, leu, dolar, euro, etc.

Câmpurile **dată calendaristică** memorează doar datele de acest tip. El are lungime fixă de 8 caractere. Asupra acestui tip de câmp pot fi executate o serie de operaţii de transformare a datei calendaristice în şir de caractere sau tip de dată numeric. Visual FoxPro nu acceptă date calendaristice inexistente, el execută o permanentă verificare a câmpurilor din care este formată data calendaristică. Acest câmp se poate seta după dorinţă în dată calendaristică de un anumit tip (american, englez, german. Francez etc.). Există două tipuri de câmp dată calendaristică: Date şi Date Time, care pot prezenta data calendaristică sub forma lună, zi, an sau sub forma, lună, zi, an, oră, minut, secundă. Pentru ţine cont de problema anului 2000 este necesară este necesară luarea în considerare a opţînii **Tools->Options**. În **zona Year 2000 compliance** se va seta caseta **Strict data lavel** la valoarea dorită. Opţiunile acestei casete sunt strâns legate de comenzile **SET CENTURY** şi **SET STRICTDATE.**

Tipul câmpului	Descriere	Mărime	Rang
Date	Date formată implicit din lună, zi, an	8 bytes	Când se foloseşte formatul strict {^0001-01-01}, 1 ianuarie anul 1 la {^9999-12-31}, 31 decembrie anul 9999
Date Time	Data formată din lună, zi, an şi timp: oră, minut, secundă	8 bytes	Când se foloseşte formatul strict, de la, {^0001-01-01}, 1 ianuarie anul 1 {^9999-12-31}, la 31 decembrie anul 9999, între orele 00:00:00 a.m. la 11:59:59 p.m.

Câmpurile **numerice** memorează numere cu punct zecimal fix, care pot fi pozitive sau negative. Imaginea unui câmp numeric este limitată la 20 de caractere. Visual FoxPro are o precizie de calcul de 16 cifre.

Câmpurile numerice în **virgulă mobilă** sau în notaţia ştiinţifică sunt un caz particular al tipului numeric. Tipul numeric virgulă mobilă diferă de tipul numeric prin faptul că numerele în virgulă mobilă sunt destinate în special depozitării numerelor foarte mari sau foarte mici, care vor fi convertite dacă este necesar, în notaţia ştiinţifică şi depozitate cu maximum de precizie posibil. Un număr în notaţia ştiinţifică se caracterzează prin două câmpuri:

- mantisă;
- exponent;

despărţite printr-o literă.

De exemplu considerăm numărul 0.345000000E+05, în care 0.345000000 reprezintă mantisa, iar 05 reprezintă exponentul. Valoarea acestui număr se interpretează $0.345*10^5$.

Tipul câmpului	Descriere	Mărime	Rang
Double	Precizie dublă În virgulă mobilă	8 bytes	+/-2.94065645841247E-324 la +/-8.9884656743115E307
Float	La fel ca şi câmpul numeric numai că reprezentarea este în virgulă mobilă	8 bytes în memorie;	- .9999999999E+19 la .9999999999E+20
Integer	Valoare întreagă	4 bytes	-2147483647 la 2147483647

Câmpul de tip **logic** are întodeauna lungimea de un caracter, el putând memora doar două valori adevărat sau fals. Litera T mare în câmp reprezintă valoarea logică adevărat, iar litera F mare este interpretată ca valoarea fals.

Câmpurile **memo** sunt câmpurile din Visual FoxPro în care se pot memora informaţii de diverse tipuri: text, informaţii binare cum ar fi fişiere executabile DOS, imagini, biţi de sunet sau orice altceva care nu se potriveşte cu un alt tip de câmp. Aceste câmpuri sunt practic nelimitate, singura limitare referindu-se la spaţiul disponibil pe memoria auxiliară. Infomaţiile dintr-un câmp memo nu sunt memorate în fişierul de date DBF. Ele sunt memorate într-un fişier separat, fişier care are extensia FPT şi acelaşi nume cu al fişierului DBF. Câmpurile memo sunt cele mai vulnerabile câmpuri ale sistemului FoxPro. Un câmp memo se poate distruge prin simpla întrerupere de curent şi face imposibilă folosirea fişierului cu extensie FPT. De aceea în practică, dacă se lucrează cu astfel de câmpuri se impune lucrul cu mai multe copii de siguranţă.

Câmpurile de tip **General** sunt folosite pentru a introduce obiecte de tipul OLE. Acestea pot fi câmpuri multimedia sau entităţi Windows ca de exemplu foaie de calcul Excel sau document Word.

Câmpurile Character (Binary) şi Memo(Binary) sunt folosite în cazul în care se lucrează în reţea.

Tipul câmpului	Descriere	Mărime	Rang
Character (Binary)	La fel cu tipul **Character,** dar valorile nu sunt translatate atunci când se schimbă codul de pagină, de exemplu parolele unor utilizatori stocate într-o tabelă şi folosite în diverse ţări	1 la 254 caractere	Orice caracter
Memo (Binary)	La fel cu tipul **Memo,** dar valorile nu sunt translatate atunci când se schimbă codul de pagină, de exemplu scriptul de logare al unui utilizator (în diverse ţări)		

Tipul de dată Variant, nu apare la crearea fişierelor de date, dar ele există în mediul VFP.

Tipul câmpului	Descriere	Mărime	Rang
Variant	Acest tip de dată poate fi de orice tip recunoscut de VFP sau poate avea valoarea NULL. Acest tip de variabilă este definit în sintaxa limbajului prin caracterul "e" situat înaintea numelui datei.	Funcție de tipul predefinit de dată.	Funcție de tipul predefinit de dată.

2.2 Implementarea fişierelor de date în limbajul FoxPro.

Principalele operații care se execută în vederea implementării unui fişier de date sunt:

1) **crearea structurii** fişierului;
2) **deschiderea / închiderea** fişierului de date, într-o zonă de lucru (WORK AREA) în vederea prelucrării;
3) **introducerea** datelor conform cu structura fişierului;

2.2.1 Crearea fişierului de date independent (**free table**) din fereastra de comenzi sau cu ajutorul sistemului de meniuri

Pentru a crea o un fişier de date se poate folosi instrucțiunea:

CREATE [< nume fişier > / ?]

din fereastra de comenzi. Dacă nu se tastează numele fişierului sau se tastează caracterul ? atunci apare o fereastră de dialog (figura 2.5), în care se introduce numele fişierului de date care se va crea.

Pentru exemplul prezentat s-a creat fişierul de date cu numele tbl_freetable. După

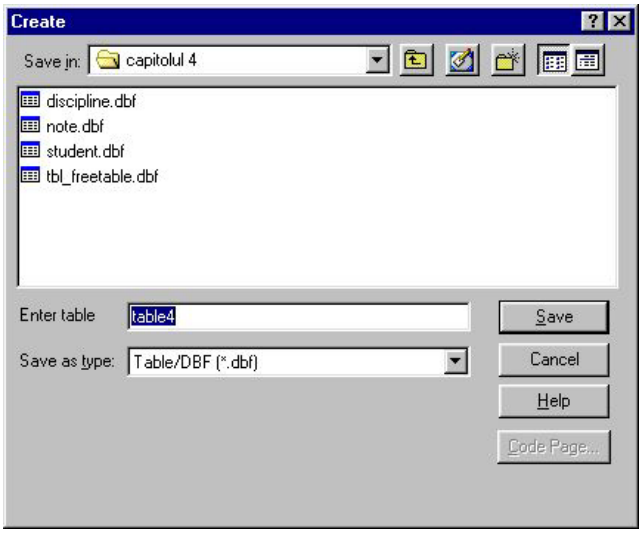


Fig. 2.5 Fereastra de definire a numelui fişierului de date

tastarea numelui fişierului, se va acționa butonul **Save** pentru crearea fişierului.

Se poate folosi în acelaşi scop sistemul de meniuri, selectând meniul **F**ile, subopțiunea **N**ew. Pe ecran va apare fereastra de dialog pentru selectarea tipului de fişier (figura 2.6).

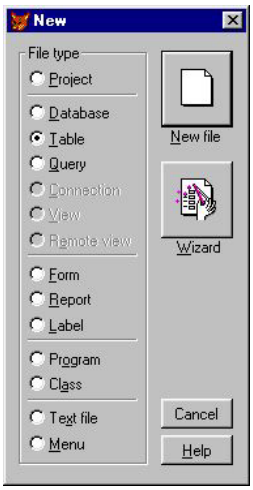


Fig. 2. 6 Ferastra NEW

Se va alege butonul **Table** din lista entităților afişate.

Oricare din cele două metode va deschide fereastra de dialog din figura 2.7.

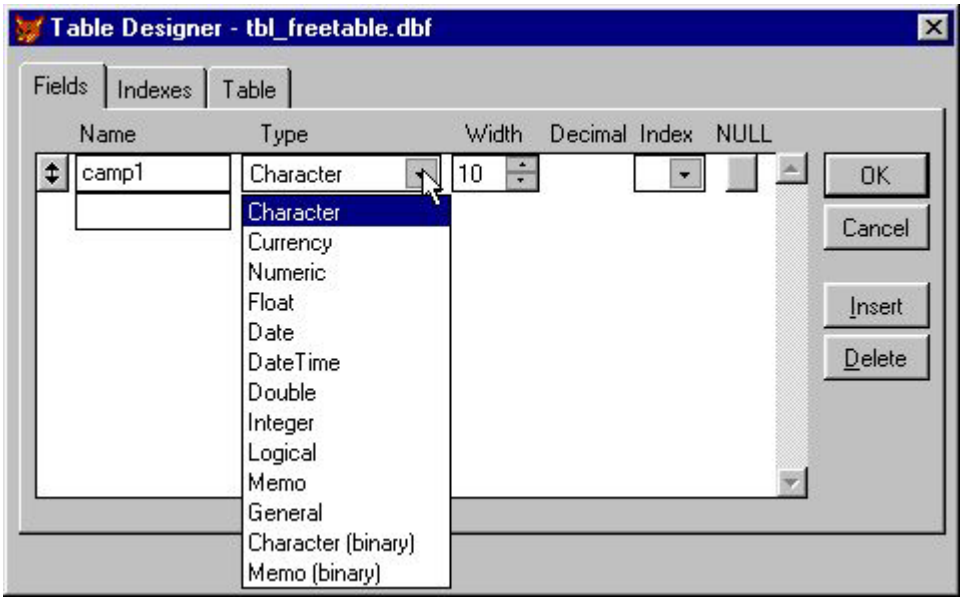


Fig. 2. 7 Fereastra Table Designer pentru definirea structurii unui fişier de date

Aceasta este o fereastră de natură modală, adică pentru a selecta o altă operație această fereastră trebuie închisă, prin butonul **OK** de terminare a operației de creare tabelă sau prin butonul **Cancel** de anulare a operației. În pagina **Fields** sunt definite câmpurile fişierului de date.

Modul de lucru este simplu. Se introduce numele câmpului (atributului) dorit, maxim 10 caractere, în caseta Name, se alege tipul câmpului şi mărimea acestuia. Tipurile câmpurilor sunt cele enunțate anterior.

Prin fereastra de dialog se poate naviga folosind tastatura sau mouse-ul. Dacă se foloseşte tastatura atunci cu ajutorul tastei **TAB** se poate selecta obiectul dorit într-un sens sau combinația **Shift+Tab** pentru selectarea obiectului cu deplasare în sens invers. Dacă se foloseşte mouse-ul, se face clic pe obiectul care se doreşte a fi selectat.

După definirea tuturor câmpurilor, am definit structura fişierului DBF. Structura unui fişier de date poate fi oricând modificată. Ştergerea unui câmp din structură are ca efect pierderea irecuperabilă a datelor din acel câmp. Dacă se modifică lungimea unui câmp atunci datele sale vor fi modificate astfel:

- dacă lungimea câmpului se măreşte atunci informațiile memorate sunt completate la dreapta cu spații;
- dacă lungimea câmpului este micşorată, atunci se va face o trunchiere a datelor originale şi se vor elimina caracterele cele mai din dreapta.

În pagina Indexes sun definiți indecşii ataşați fişierului (fig 2.8). În acest caz pot fi definiți doar indexi de tip **Regular** şi **Candidate.** Asupra noțiuni "index" se va reveni pe parcursul acestui capitol.

Pagina **Table** afişează în cazul fişierelor de date independente date referitoare la fişierul creat (fig. 2.9).

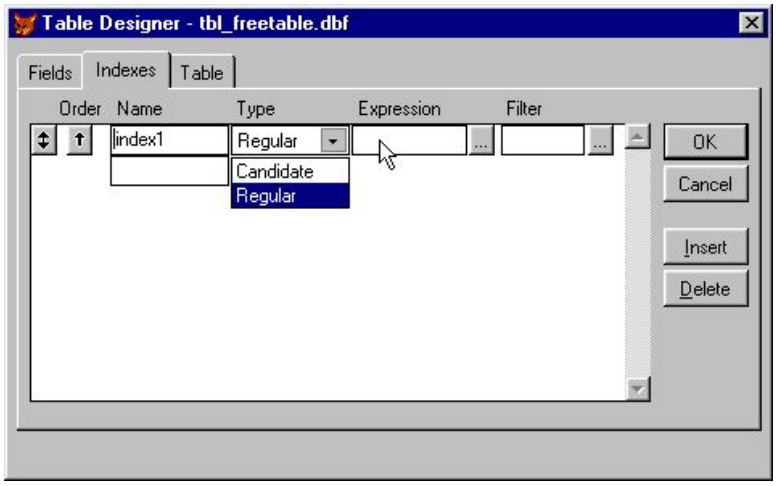


Fig. 2. 8 Crearea indexilor in fişierele de date independente

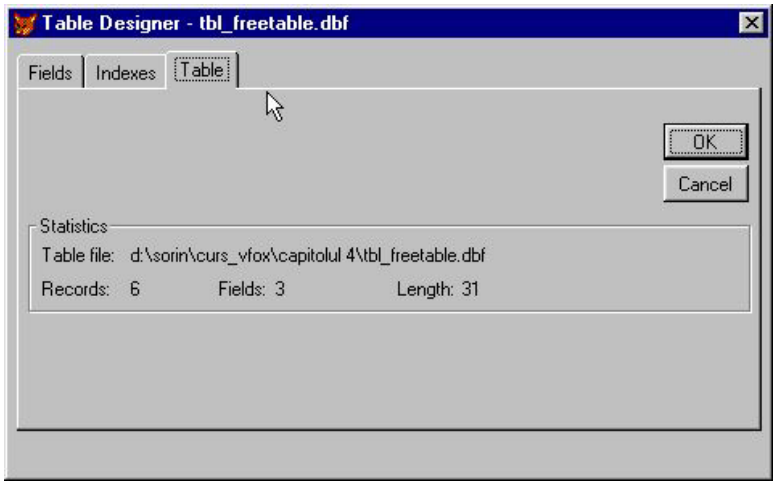


Fig. 2. 9 Pagina Table

2.2.2 Crearea fişierelor de date din program sursă PRG

Crearea unor fişiere de date din program sursă Visual FoxPro se realizează cu comanda:

CREATE TABLE | DBF < nume > <nume lung de fişier> [FREE] **(**<nume câmp1>/<tip>[<lungime>[<zecimale>] **)** ,[<nume câmp2>...] [NULL |NOT NULL] [CHECK lExpresie1 [ERROR TextEroare1]] [DEFAULT eExpresie1] [PRIMARY KEY | UNIQUE] [REFERENCES NumeTabel2 [TAG NumeEticheta1]] [NOCPTRANS] [, NumeCâmp2 ...] [, PRIMARY KEY eExpresie2 TAG NumeEticheta2 |, UNIQUE eExpresie3 TAG NumeEticheta3] [, FOREIGN KEY eExpresie4 TAG NumeEticheta4 [NODUP] REFERENCES NumeTabela3 [TAG NumeEticheta5]] [, CHECK lExpresie2 [ERROR TextEroare2]])| FROM ARRAY NumeMatrice

Aceasta este de fapt o comandă a limbajului SQL implementat în mediul VFP. Semnificaţia parametrilor comenzii este prezentată în tabelul 2.1.

Tabelul 2.1

Parametru Comandă	Semnificaţie
Table sau DBF	Parametrii sinonimi ai comenzii
Nume	Numele fişierului de date care se crează.
Nume lung	Numele lung al fişierului poate conţine 128 de caractere şi poate fi specificat doar în cazul în care fişierul de date aparţine unei baze de date
FREE	Specifică faptul că tabela este independentă în cazul în care o bază de date este activă. În cazul în care nici o bază de date nu este deschisă acest parametru poate lipsi, dacă se doreşte crearea unei tabele indepandente.
Nume cimp1	Numele primului câmp din structura fişierului.
Tip	Tipul primului câmp definit.
Lungime	Lungimea câmpului definit.
Zecimale	Numărul de zecimale al câmpului, în cazul în care acesta este câmp numeric
Nume cimp2	Numele celorlalte câmpuri care se introduc
NULL	Permite introducerea valorii null în câmp. Dacă unul dintre câmpuri conţine valoarea null atunci numărul de câmpuri care poate fi creat într-un fişier de date se reduce de la 255 la 252.
NOT NULL	Nu permite existenţa valorilor null în cadrul câmpului. Aceasta este valoarea implicită la crearea fişierului de date
CHECK	Permite specificarea unor reguli de validare pentru câmp
ERROR	Afişează un mesaj de eroare în cazul în care regula de validare pentru câmp nu este îndeplinită
DEFAULT	Specifică valoarea iniţială a câmpului
PRIMARY KEY	Crează un index primar pentru câmpul specificat
UNIQUE	Crează un index candidat pentru câmp
REFERENCES	Stabileşte numele tabelei părinte cu care se realizează o

	legătură persisentă
NOCPTRANS	Previne translatarea câmpurilor memo şi caracter cu cod de pagină diferit
FOREIGN KEY	Crearea unui index extern
NODUP	Crearea unui index candidat extern
FROM ARRAY	Reprezintă numele unei matrici existente care conține numele, tipul, lungimea şi numărul de zecimale pentru fiecare câmp .

Această comandă este deosebit de utilă pentru a crea într-o aplicație fişiere de date temporare adică acele fişiere care după o anumită fază a aplicației sau după terminarea execuției aplicației pot să fie şterse.
În cazul acestei comenzi tipurile câmpului sunt date de către următoarele litere:

- C şir de caractere de o anumită lungime;
- N numeric, de o anumită lungime şi cu un anumit număr de zecimale;
- F virgulă mobilă de o anumită lungime şi cu un anumit număr de zecimale;
- I integer;
- B double;
- Y currency
- L logic;
- M memo;
- D dată calendaristică;
- T date time;
- G general

Exemplul 2.1
Să se creeze fişierul de date SALARIAT.dbf cu următoarea structură:

- Matr N(6) numărul matricol al salariatului;
- Nume C(12) numele salariatului;
- Prenume C(20) prenumele salariatului;
- Data_nast D data naşterii salariatului;
- Sex L dacă salariatul este de sex masculin valoarea câmpului este T (true);
- Stare _civilă L dacă salariatul este căsătorit valoarea câmpului este T (true);
- vârsta N(3) vârsta salariatului.

Se deschide fereastra de editare în care se va scrie următoarea comandă:

CREATE TABLE salariat(nume C(12), prenume C(20), data_nast D, sex L, stare_civilă L, vârsta N(3))

2.2.3 Crearea unui fişier de date într-o bază de date

În mediul Visual Fox crerea unui fişier de date într-o bază de date, permite pe

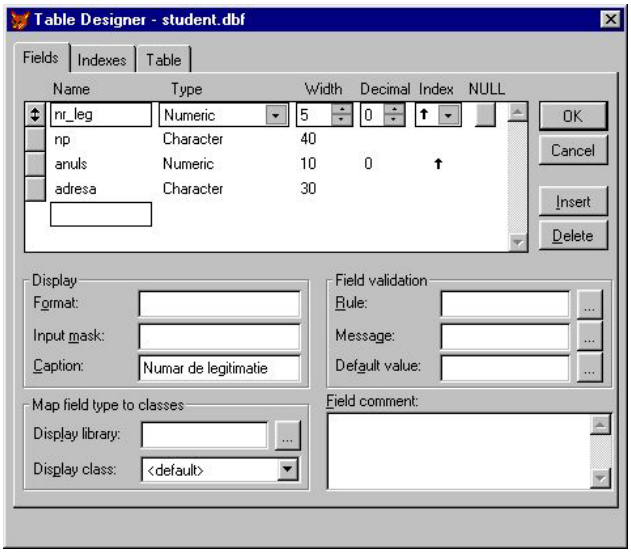


Fig. 2. 10 Crearea structurii unui fişier de date într-o bază de date

lângă crearea câmpurilor şi a indexilor, impunerea unor restricţii la nivel de câmp sau la nivel de fisier de date. Un fişier de date, poate fi creat iniţîal într-o bază de date sau un fişier de date independent poate fi ataşat la o tabelă. Fereastra pentru crearea structurii tabelei este prezentată în fig. 2.10.

Acest mod de creare a tabelei prezintă o serie de avantaje:

- Permite nume lungi de câmpuri, până la 128 de caractere. *În definirea numelui unui câmp nu se foloseşte caracterul "spaţiu",* el poate fi înlocuit cu caracterul "_" (liniuţa de subliniere);
- Prezintă trei zone distincte**: Display, Map field type to classes, field validation** pentru a impune diverse restricţii asupra câmpurilor şi o zonă de comentarii asupra câmpului realizat, **Field comment**;
- În caseta **Table** pot fi introduse restricţii asupra fişierului de date în ansamblu;
- Permite definirea indexului de tip cheie primară;
- Permite realizarea în cadrul bazei de date a relaţiilor permanente între tabele de tipul one to one, one to many sau many to many.

Zona **Display** permite introducerea unor restricţii asupra câmpului definit. Restricţia poate fi introdusă sub forma unui format particularizat al câmpului (în caseta **Format**) sau sub forma unei măşti de intrare (în caseta **Input mask**).

Principalele caracterele folosite pentru proprietatea **Format** sunt prezentate în tabelul 2.2

Tabelul 2.2

Caracter folosit	Desctriere
!	Converteşte literele mici în litere mari.
$	Vizualizează simbolul monezii pentru câmpul de tip currency

^	Vizualzează datele numerice folosind notația științifică.
A	Permite folosirea doar a caracterelor alfabetice.
D	Foloseşte setarea curentă pentru SET DATE.
E	Editează data calendaristică în formatul British.

Pentru masca de intrare principalele caractere folosite sunt prezentate în tabelul 2.3

Tabelul 2.3

Caracter folosit	Descriere
!	Converteşte literele mici în litere mari.
#	Permite folosirea caracterelor numerice a spațiului şi a semnului.
$	Permite folosirea simbolului monetar într-o poziție fixă.
$$	Permite folosirea simbolului monetar într-o poziție aleatoare după valoarea numerică.
9	Permite folosirea valorilor numerice şi a semnului "-".
A	Permite folosirea numai a caracterelor alfabetice.
L	Permite folosirea numai a valorilor logice .t. (true), .f. (False).
N	Permite folosirea numai a caracterelor alfanumerice.
X	Permite folosirea oricărui caracter.
Y	Permite folosirea caracterelor Y, y, N şi n pentru valorile logice true respectiv false

În caseta **Caption** pot fi introduse caractere care vor înlocui denumirea câmpului într-o comandă VFP.

Exemplul 2.2

Se consideră baza de date Facultate (fig 2.11) în care se crează tabela Student

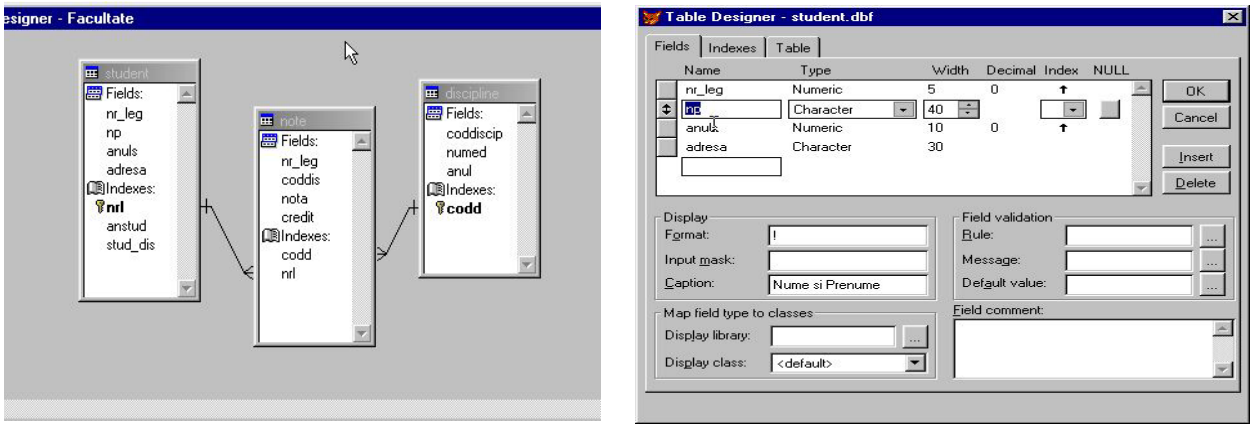


Fig. 2 11 Baza de date "Facultate" şi tabela "Student"

a) Câmpul "np" apare în comenzile VFP cu denumirea Nume si Prenume (caseta Caption);
b) În caseta Format s-a folosit caracterul "!" pentru ca informația din câmp să fie scrisă cu litere mari indiferent de caracterul introdus în câmp.

În zona **Field Validation**, pot fi introduse restricții asupra câmpului. Acestea se introduc sub forma unor expresii, în caseta **Rule,** unde se apelează **Expression Builder** (fig. 2.12)

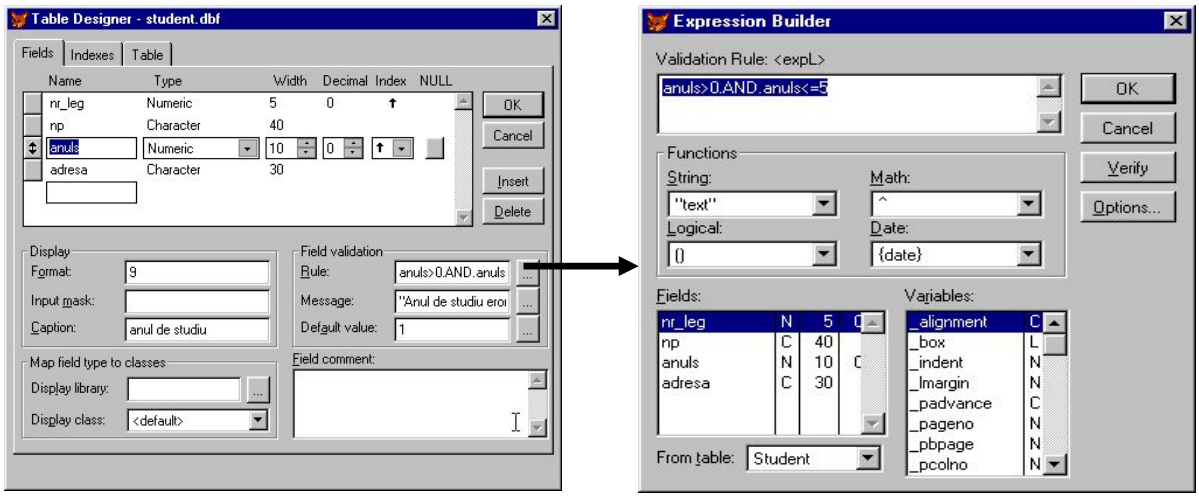


Fig. 2 12 Introducerea restricţiilor în caseta Rule şi Expression Builder

Expression Builder este apelat de la butonul "…" situat în dreapta casetei Rule. El are rolul de a forma o expresie. O expresie, este o entitate care conţine operanzi şi operatori. Asupra acestui aspect, se va reveni în această carte când vom discuta despre limbajul de programare VFP. **Expression Builder** permite introducerea într-o expresie a câmpurilor existente în tabelă, a unei funcţii predefinite VFP sau a unei variabile de memorie VFP. Din figură, se observă că a fost introdusă restricţia: valoarea câmpului anuls să fie mai mare decât 0 şi mai mică sau egală cu 5. În cazul în care regula de validare impusă asupra câmpului, nu este îndeplinită, apare un mesaj predefinit sau un mesaj scris de către utilizator, în caseta **Message**. Valoarea iniţială a câmpului poate fi definită în caseta **Default Value.** Pentru exemplul prezentat valoarea iniţială este 1.

Un comentariu de câmp este o notă care clarifică semnificaţia sau scopul câmpului. VFP stochează acest element ca un câmp Memo, de aceea poate fi oricât de lung. Toate acestea pot fi introduse în caseta **Field comment**.

Zona **Map field to classes** permite alegerea unei clase sau a unei biblioteci de clase pentru un câmp în cazul în care acesta este introdus într-un formular.

În caseta Table (fig 2.13), este permisă introducerea unor restricţii la nivel de

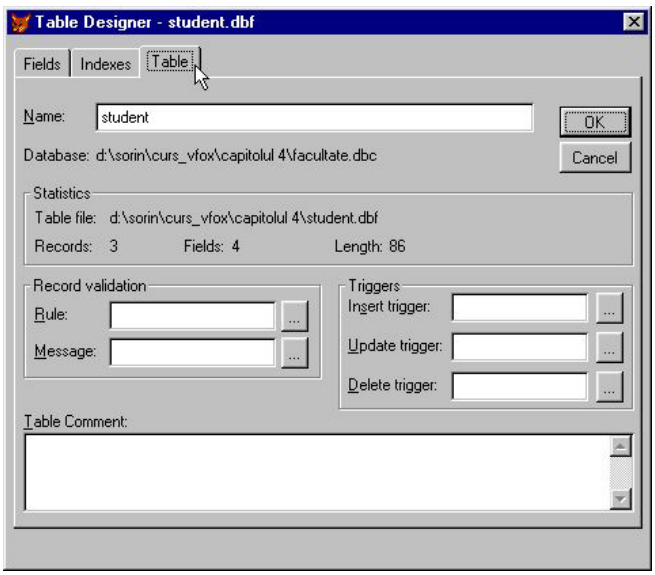


Fig. 2 13 Pagina Table pentru fişierul de date Student

înregistrare într-un fişier de date.

Validarea la nivel de înregistrare, înseamnă, modificarea unor valori din câmpurile înregistrării şi mutarea într-o nouă înregistrare. Codul de validare nu poate modifica un câmp din înregistrarea curentă, nu poate modifica contorul de înregistrări sau pointerul de înregistrare dar poate compara valoarea unui câmp cu a altuia sau poate executa o validare prin căutare într-un alt fişier. Orice expresie de validare trebuie să returneze un rezultat de tip logic. Pentru aceasta se foloseşte o funcţie utilizator scrisă în caseta **Rule**, din zona **Record validation**. Caseta **Message** conţine un mesaj care însoţeşte validarea. Pentru a realiza o funcţie de validare se foloseşte funcţia VFP, GETFLDSTATE, care stabileşte dacă un câmp dintr-un tabel s-a schimbat în timpul comenzii curente sau dacă starea de ştergere s-a modificat, prin valorile pe care le poate returna (tabelul 2.3).

Tabelul 2.3

Valoare returnată	Descriere
1	Câmpul şi indicatorul de ştergere nu şi-au modificat valoarea
2	S-a modificat valoarea câmpului şi a indicatorului de ştergere
3	Câmpul şi indicatorul de ştergere nu şi-au modificat valoarea într-o operaţie de adăugare
4	S-a modificat valoarea câmpului şi a indicatorului de ştergere într-o operaţie de adăugare

Zona **Triggers** defineşte declanşatorii (trigerii) care se execută atunci când se adaugă (**Insert Trigger**), se modifică (**Update Trigger**) sau se şterge (**Delete Trigger**) o înregistrare din tabel. Ca şi în cazul validării la nivel de câmp şi la nivel de înregistrare, codul sursă al triggeru-lui se stochează în baza de date. Întodeauna când se execută o operaţie de adăugare, modificare sau ştergere într-o tabelă, VFP verifică dacă există un declanşator aferent tabelei şi în cazul în care acesta există este apelat codul program sursă, al respectivului trigger.

2.2.3 Legături între tabelele bazei de date

Tabelele existente înr-o bază de date pot fi legate între ele prin diverse câmpuri. Un index primar a unui câmp sau o cheie primară dintr-o tabelă, are o cheie de căutare sau o cheie externă , care poate fi un index regular al unui câmp, într-o altă tabelă. Pentru exemplul prezentat în figura 2.3 legătura dintre tabela **Student** şi tabela **Note** se realizează între cheia primară **Nrl** care este un undex primar în tabela

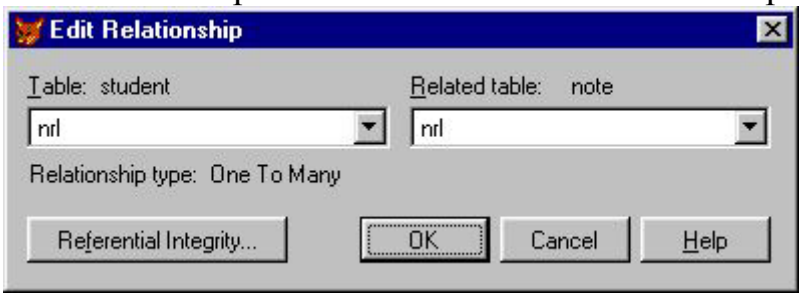


Fig. 2 14 Legătura între tabele (Relationship)

Student şi indexul regular **Nrl,** din tabela **Note** (fig 2.14). Aceasta este o relație de tip *one to many*. În Visual Fox pot exista relații de tip *one to one, one to many şi many to one*. Relațiile de tip *many to many* se obțin din precedentele.

Fereastra **Edit Relationship** se poate activa poziționând cursorul pe legătura stabilită între tabelele din baza de date şi executând clik pe butonul din dreapta al mouse-ului. Din meniul contextual care apare se apelează opțiunea cu acelaşi nume. Editarea legăturii se poate realiza şi din meniul Database, care se activează la vizualizarea bazei de date (MODIFY DATABASE).

Într-o tabelă se pot stabilii şi *relații de autoreferire,* prin care un câmp dintr-o tabelă este legat cu un câmp din aceeaşi tabelă.

În capitolul 2 s-au prezentat noțiuni teoretice despre integritatea referențială. În general aceasta stabileşte ce operații sunt permise între tabelele legate. Integritatea referențială tratează ca invalide, înregistrările care nu îndeplinesc anumite criterii. Ea se aplică doar în cazul în care se modifică datele din cheile stabilite în relație. Aceste modificări se referă la:

- Adăugarea unei înregistrări noi;
- Modificarea unei înregistrări existente;
- Ştergerea unei înregistrări existente.

În cazul în care se doreşte adăugarea unei înregisrări noi în *tabela copil* (**Note**) legată la *tabela părinte* (**Student**), apar două cazuri:

1. Permite adăugare de înregistrare nouă în tabela copil indiferent dacă există corespondent în tabela părinte (**Ignore**);
2. Se generează o eroare atunci când se introduce o înregistrare în tabela copil care nu are corespondent în tabela părinte (**Restrict**)

În cazul în care se doreşte modificarea unei înregisrări în *tabela copil* (**Note**) legată la *tabela părinte* (**Student**), apar trei cazuri:

1. Permite modificarea în înregistrarea din tabela copil chiar dacă aceasta nu are corespondent în tabela părinte (**Ignore**);
2. Nu permite modificarea unei înregistrări din tabela copil, care nu are corespondent în tabela părinte, acest lucru generând eroare (**Restrict**);

3. Sunt modificate automat toate înregistrările din tabela copil care au avut aceeaşi valoare a cheii părinte vechi cu noua valoare a cheii. De exemplu un student îşi modifică numărul de legitimaţie, atunci în tabela copil **Note**, câmpul **nr_leg** se modifică automat cu noua valoare (**Cascade**).

În cazul în care se doreşte ştergerea unei înregisrări în *tabela părinte* (**Student**)

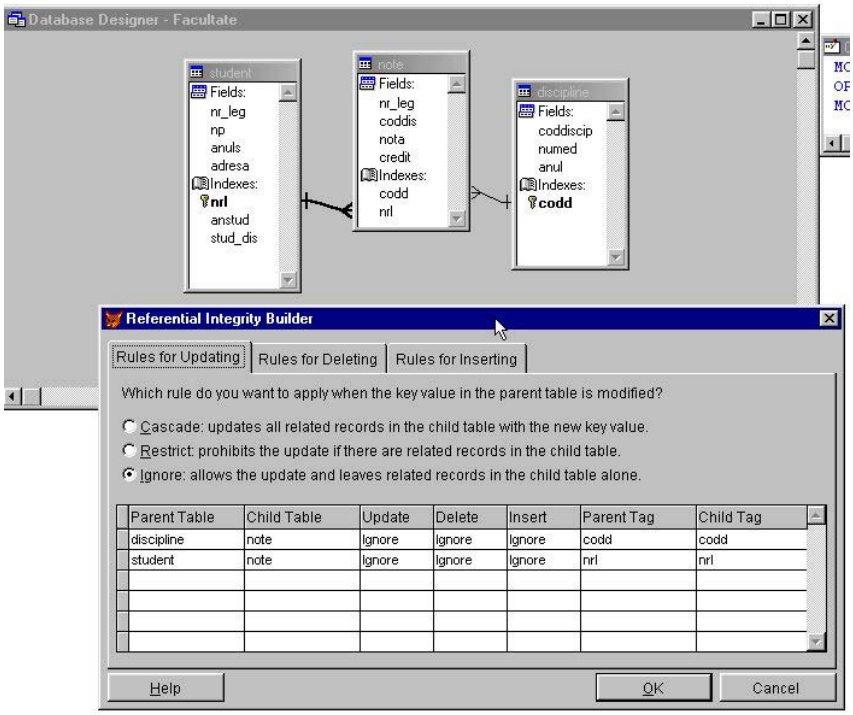


Fig. 2 15 Constructorul de integritate referenţială

legată la *tabela copil* (**Note**), apar trei cazuri:

1. Permite ştergerea în înregistrarea din tabela părinte indiferent dacă în tabela copil există sau nu înregistrări legate cu înregistrarea din tabela părinte (**Ignore**);
2. Apare un mesaj de eroare atunci când ştergerea se face pe o înregistrare în tabela părinte pentru care există înregistrări corespunzătoare în tabela copil (**Restrict**);
3. Se şterg automat, toate înregistrările din tabela copil, legate de înregistrarea din tabela părinte ştearsă (**Cascade**).

În VFP, integritatea referenţială poate fi definită automat folosind Constructorul de integritate referenţială (**Referential Integrity Builder**). Acesta poate fi apelat fie de la butonul **Referenţial Integrity,** din fereastra de editare a legăturii sau din meniul **Database** al ferestrei **Database Designer**. Fereastra care apare permite definirea restricţiilor definite anterior (fig 2.15).

2.3 Activarea fişierelor de date

2.3.1 Deschiderea / închiderea unui fişier de date Visual FoxPro într-o zonă de lucru

Deschiderea fişierului de date presupune rezervarea în memoriea internă a calculatorului a unei zone în care Visual FoxPro va memora toate informaţiile necesare utilizării acestui fişier. Aceste zone se numesc zone de lucru ele fiind numerotate de la 1-32767 în versiunea Visual FoxPro 7.0 standard. Numărul de zone de lucru care pot fi deschise în acelaşi timp, diferă funcţie de versiunea VFP cu care se lucrează. Primele 10 zone de lucru pot fi marcate cu literele de la A-J.

Manipularea fişierelor de date este la îndemâna programatorului pe baza comenzilor specializate în acest scop. *În această carte se prezintă modul de lucru pe un calculator nelegat în reţea, de aceea o serie de parametrii ai comenzilor,care se referă la modul de lucru în reţea sunt omişi.* După deschiderea fişierului de date se trece la lucrul efectiv cu aceasta, adică:

- prelucrarea datelor din fişierul de date (citire);
- modificarea datelor (scriere de date noi peste cele existente);
- adăugarea datelor noi la cele existente (scriere de date noi);
- ştergerea informaţiilor existente în fişierul de date.

După terminarea lucrului cu fişierul de date urmează **închiderea** acestuia, operaţie ce constă în :

- desfacerea legăturii dintre zona de lucru şi fişierul de date ;
- eliberarea memoriei alocate la deschidere (eliberarea zonei de lucru);
- închiderea fişierului de date memorând modificările efectuate de la deschidere, până în acest moment.

Pentru a deschide un fişier de date a cărui structură a fost creată şi în care au fost sau nu introduse date se foloseşte comanda **USE** cu sintaxa:

USE [<nume_fis> | ?]
[**IN** <expN1>]
[**AGAIN**]
[**INDEX** <lista fis index> | ?
[**ORDER** [<expN2>
| <fisiere index idx>
| [**TAG**] <nume tag>
[**OF** <fisiere cdx>]
[**ASCENDING** | **DESCENDING**]]]]
[ALIAS <alias>]
[**EXCLUSIVE**]
[**SHARED**]
[**NOUPDATE**]

USE nume _fis

este cea mai uzuală formă a comenzii pentru deschiderea fişierului de date în zona de lucru activă sau curentă. Implicit această zonă este zona 1.

Opţiunea **IN** <expN1> defineşte zona de lucru în care fişierul de date este deschis. <expN1>, este o expresie numerică având valoarea cuprinsă între 1-225. În cazul în care în zona de lucru există un fişier deja activ acesta este închis şi devine activ fişierul deschis ultima dată.

Opţiunea **AGAIN** se foloseşte pentru a deschide acelaşi fişier de date în mai multe zone de lucru.

Exemplul 2.3

USE salariat

USE salariat IN 2 AGAIN

se deschide fişierul de date Salariat în zona 1 şi în zona 2 de lucru

Opţiunea **[INDEX <**lista fis_index**> | ?**

[ORDER [<expN2**>**

| <fisiere index idx**>**

| [TAG] <nume tag**>**

[OF <fisiere cdx**>]**

[ASCENDING | DESCENDING]]]] se foloseşte pentru a deschide fişierele indexate. Se va explica opţiunea în subcapitolul referitor la indexarea fişierelor de date.

Opţiunea **ALIAS** se referă la aliasul atribuit de utilizator unui fişier de date. Aliasul este un nume oarecare atribuit la deschiderea fişierului. În unele cazuri aliasul poate fi definit de către mediul FoxPro.

Exemplul 2.4

USE salariat IN 3 ALIAS s în zona de lucru 3 fişierul salariat are aliasul “s”.

Opţiunile **EXCLUSIVE** şi **SHARED** sunt specifice modului de lucru în reţea, caz în care mai mulţi utilizatori au acces la acelaşi fişier în acelaşi timp. În cazul lucrului în reţea dacă la un fişier de date au acces mai mulţi utilizatori acesta trebuie dechis NOEXCLUSIVE în caz contrar un singur utilizator având acces la fişierul de date.

Opţiunea **NOUPDATE** se referă la faptul că fişierul este deschis numai în citire, deci datele din fişier nu pot fi modificate.

Exemplul 2.5

USE pers **NOUPDATE**

Închiderea unui fişier activ într-o zonă de lucru se poate face folosind comanda USE fără nici o opţiune sau folosind alte comenzi specifice ca de exemplu CLOSE ALL. Spre deosebire de comanda USE care închide doar fişierul din zona de lucru activă comanda CLOSE ALL închide toate fişierele active la un moment dat în mai multe zone de lucru.

2.3.2 Selectarea unei zone de lucru în FoxPro

La pornirea Visual FoxPro zona de lucru curentă este 1 sau zona A . În cazul în care lucrăm cu mai multe fişiere de date atunci zona de lucru se poate schimba folosind instrucțiunea:

SELECT <exp N**>** |<expC>

<expN> reprezintă numărul zonei de lucru ce va fi activată;

<expC> reprezintă numele zonei de lucru care va fi activată, acesta poate fi o literă A-J sau un alias definit de utilizator.

Exemlul 2.6

Presupunem că dorim să lucrăm cu trei fişiere Cu denumirile Fisa, Fisb, Fisc, pe care vrem să le deschidem în trei zone de lucru diferite.

SELECT 1
USE Fisa
SELECT 2
USE Fisb
SELECT 3
USE Fisc

În continuare apelul la unul dintre fişiere se face folosind doar instrucțiunea **SELECT**. Astfel, pentru a lucra cu Fisc este suficient a selecta numai zona de lucru 3: **SELECT** 3

Implicit aliasul, adică numele asociat fişierului este chiar numele acestuia. Selectarea fişierului Fisc ca fişier de lucru se poate face prin comanda:

SELECT Fisc

Exemplul 2.7

Presupunem că dorim să deschidem fişierul Salariati în zona de lucru 2 atribuindu-i aliasul ‘salar’.

USE salriati **IN** b **ALIAS** salar

SELECT salar se selectează zona de lucru 2 prin aliasul salar al fişierului din această zonă de lucru.

În Visual FoxPro pentru majoritatea comenzilor se pot folosi doar primele 4 caractere din comandă .

Pentru a obține numărul zonei de lucru curentă se foloseşte funcția:

SELECT ()

Aceasta returnează un număr ce reprezintă numărul zonei de lucru curente sau numărul ultimei zone de lucru nefolosite.

- **SELECT()** sau **SELECT (0)** returnează numărul zonei de lucru curente;
- **SELECT(1)** returnează numărul ultimei zone de lucru nefolosite

2.2. Accesul la fişierele de date

2.2.1 Accesul la structura unui fişier de date

Pentru a modifica structura unui fişier de date se foloseşte comanda

MODIFY STRUCTURE <nume_fişier**>**

<nume_fişier> este numele fişierului căruia i se modifică structura. Dacă numele fişierului nu este specificat apare o fereastră de dialog care cere să se specifice numele fişierului dorit.

Fereastra de dialog care apare este prezentată în figura 2.7 sau 2.10. În această fereastră se pot modifica numele câmpurilor, dimensiunile acestora, se pot adăuga câmpuri noi sau se pot şterge câmpuri existente. Ştergerea şi inserarea de câmpuri se realizează prin poziționarea cursorului pe indicatorul de înregistrări şi acționarea tastelor Insert sau Delete.

Vizualizarea structurii unui fişier de date se realizează cu comanda:

DISPLAY STRUCTURE
[**IN** <expN> | <expC>]
[**TO PRINTER [PROMPT]**
| **TO FILE** <fişier>]
[**NOCONSOLE]**

<expN> specifică zona de lucru în care se găseşte fişierul, iar <expC> specifică numele fişierului. Implicit afişarea se face pe ecran. Acest lucru poate fi inhibat prin folosirea opțiunii **NOCONSOLE**. Structura fişierului poate fi copiată într-un fişier (opțiunea **TO FILE**) sau la imprimantă (opțiunea **TO PRINTER**). Opțiunea **PROMPT** este specifică mediului Window.

Comanda **LIST STRUCTURE** este identică cu precedenta comandă, singura deosebire constând în faptul că instrucțiunea **DISPLAY STRUCTURE** face pauză după umplerea unui ecran cu informații.

Informația afişată are următorul conținut:

- pe prima linie se afişează fişierul de date la care se referă comanda;
- pe a doua linie se afişează numărul de înregistrări din fişierul DBF;
- pe a treia linie se afişează data calanderistică când s-a scris ultima informație în baza de date;
- pe următoarele linii se afişează câmpurile ce alcătuiesc structura fişierului de date cu caracteristicile acestora (nr.câmp, nume, tip, lățime).

Un alt mod de a crea un fişier de date este acela de a copia structura unui fişier existent în unul nou, care va avea inițial structura identică cu a fişierului sursă. Comanda de copiere a structurii este:

COPY STRUCTURE TO <fişier> **FIELDS** <listă câmpuri>

<fişier> este numele noului fişîer în care s-a copiat structura. Dacă nu se doreşte copierea tuturor câmpurilor, atunci câmpurile care se doresc a fi copiate se pot specifica cu opțiunea **FIELDS**.

Exemplul 2.8

Se copiază din structura fişierului Pers.dbf câmpurile Nume şi Prenume în noul fişier Pers_1.dbf.

```
USE PERS
COPY STRU TO PERS_1 .DBF;
    FIELDS NUME, PRENUME
USE PERS_1.DBF
LIST STRU
CLOSE ALL.
```

Se mai poate copia structura unui fişier de date în înregistrările unui nou fişier de date, folosind comanda:

COPY TO <fişier>**STRUCTURE EXTENDED**

Aceast fişier de date are o structură fixă formată din 4 câmpuri iar câmpurile fişîerului de date activ devin înregistrări. Câmpurile fixe au următoarea semnificație:

- Field_name (nume câmp) de tip şir de caractere, în care se depozitează numele câmpurilor structurii;
- Field_type (tip câmp) de tip şir de caractere, de lungime 1, în care se depozitează un caracter ce corespunde tipului câmpului structurii(C-şir de caractere,N-numeric, D-dată calendaristică, F-virgulă mobilă, L-logic, M-memo);
- Field_len(lungime câmp), de tip numeric, memorează lungimea fiecărui câmp al structurii;
- Field_dec (număr de zecimale din câmp), de tip numeric, în care se memorează numărul de zecimale pentru un câmp numeric al structurii.

Exemplul 2.9

Se execută comanda **COPY STRUCTURE** asupra fişierului SALARIAT.DBF.

```
USE SALARIAT.DBF
COPY STRU EXTENDED TO TEMP
```

Fişierul TEMP va avea structura:

NR	FIELD_NAME	FIELD_TYPE	FIELD_LEN	FIELD_DEC
1	MATR	N	12	---
2	NUME	C	12	---
3	PRENUME	C	20	---
4	DATA_NAS	D		
5	SEX	L		
6	ST_CIVILĂ	L	3	
7	VÂRSTA	N		

Structura fişierului SALARIAT.DBF devine înregistrare pentru TEMP.DBF. Ulterior se poate crea un nou fişier de date folosind înregistrările fişierului Temp.dbf cu comanda:

CREATE PERS1 **FROM** TEMP se creează fişierul de date PERS1.DBF cu noua structură.

2.2.2 Accesul la câmpurile unui fişier de date

Accesul la câmpurile unui fişier de date este controlat de comanda **SET FIELDS** cu sintaxa

SET FIELDS TO [<câmp1>][<câmp2>]/ALL

SET FIELDS ON /OFF

Dacă **SET FIELDS** este ON atunci **FIELDS** reprezintă lista câmpurilor ce pot fi accesate. Accesul la toate câmpurile unui fişier de date se poate face prin comanda

SET FIELDS TO ALL

Numărul câmpurilor dintr-o bază de date este dat de funcţia **FCOUNT().**

Exemplul 2.10

Explicaţii

A) Se determină numărul de înregistrări din fişierul specificat;
B) Se determină mărimea câmpului Nume din fişierul pers;
C) Se determină numele unui câmp din fişierul de date identificat printr-un număr.

Semnul "?" se foloseşte pentru listarea pe ecran a rezultatului unei funcţii. Se va revenii pe larg în capitolul

2.2.3 Accesul la înregistrările unui fişier de date

Accesul la conţinutul unui fişier de date se poate face în două moduri:

1. **secvenţial** când pentru obţinerea unei informaţii se parcurg înregistrările una după cealaltă în ordinea firească a lor;
2. **direct** când putem stabili unde anume pe suport este înregistrarea vizată şi o utilizăm fără să ne preocupăm de înregistrările anterioare.

Fiecare înregistrare (articol) din fişier are ataşat un număr, care poartă denumirea de indicator de înregistrare. Fişierul cu extensia Dbf are două caractere speciale care marchează, începutul şi sfârşitul unui fişier, **bof**, **eof**. Corespunzător acestor două caractere speciale, există funcţiile:

A
USE PERS
?**FCOUNT()**
6

B
?FSIZE('NUME')
12

C
?FIELD(1)
NUME

- **BOF()** care returnează rezultatul adevărat (“T”) dacă indicatorul de inregistrare este poziționat la începutul fişierului;
- **EOF()** care returnează rezultatul adevărat (“T”) dacă indicatorul de înregistrare este poziționat la sfârşitul fişieruluide date .

Pentru a căuta o anumită înregistrare în fişierul de date se foloseşte


```
USE SALARIATI
GO TOP
?RECNO()
GO RECORD RECNO()+1
GO RECNO()+4
GO 10
CLOSE ALL
```

comanda **GO** sau **GOTO** care poziționează indicatorul de înregistrare al fişierului pe o anumită înregistrare. Sintaxa acestor comenzi este identică:

GO [RECORD] <expN1>
[**IN** <expN2> | **IN** <expC>]
GO TOP | BOTTOM
[**IN** <expN2> | **IN** <expC>]
GOTO [RECORD] <expN1>
[**IN** <expN2> | **IN** <expC>]
GOTO TOP | BOTTOM
[IN <expN2> | IN <expC>]

Cuvântul rezervat **RECORD** este folosit opțional, expN1 reprezintă numărul înregistrării în fişier. Localizarea se poate face şi într-o altă zonă de lucru precizată prin expresie numerică <expN2> sau prin aliasul fişierului dedate <expC>.

GO TOP - poziționare pe început de fişier
GO BOTTOM - poziționare pe sfârşit de fişier

Numărul înregistrării curente (valoarea numerică a indicatorului de înregistrări la un moment dat) dintr-un fişier din zona de lucru <expN> sau identificat prin aliasul <expC> este dat de funcția **RECNO()** cu sintaxa:

RECNO([<expN> | <expC>])

Dacă ambele expresii <expN> şi <expC> lipsesc atunci funcția returnează numărul înregistrării curente din fişierul de date activ.

Exemplul 2.11

Explicații

A) deschiderea fişierului de date;
B) poziționare la începutul fişierului;
C) vizualizare număr înregistrare curentă;
D) poziționarea pe înregistrarea următoare;
E) se poziționează pe al patrulea articol de la înregistrarea curentă;
F) poziționează pe articolul cu indicatorul de înregistrări zece;

```
        USE SALARIATI
A       GO RECO 3
B       SKIP
        ?RECNO()
C       SKIP 10
        ?RECNO()
D       SKIP -5
        ?RECNO()
E       ?RECOUNT()
F       ?RECSIZE()
        CLOSE ALL
```

G) închiderea fişierului de date.

Un alt tip de deplasare, cu indicatorul de înregistrări, de-a lungul fişierului de date se realizează cu comanda **SKIP** care are sintaxa:

SKIP [<expN1> | <expN2> | <expC>]

- <expN1> reprezintă numărul de înregistrări peste care se sare. Expresia poate să fie atât pozitiv cât şi negativă, saltul poate avea loc înainte şi înapoi în fişierul de date;
- <expN2> şi <expC> specifică fişierul la care se referă funcţia, prin numărul zonei de lucru în care este deschisă (<expN2>) sau prin aliasul corespunzător (<expC>). Dacă cele două expresii nu se specifică comanda se referă la fişierul de date activ.

Funcţia **RECCOUNT()** returnează numărul de înregistrări dintr-un fişier de date. Funcţia are sintaxa:

RECCOUNT([<expN2> | <expC>])

Seminificaţia celor două expresii este identică cu cea de la comanda SKIP.

Funcţia **RECSIZE()** returnează mărimea unei înregistrări dintr-un fişier DBF Sintaxa acestei funcţii este:

RECSIZE([<expN2> | <expC>])

Seminificaţia celor două expresii este identică cu cea de la comanda SKIP.

Exemplul 2.12

Explicaţii

A) poziţionare pe înregistrarea trei din fişier;
B) incrementatrea indicatorului de înregistrări;
C) valoarea indicatorului de înregistrări se măreşte cu 10;
D) valoarea indicatorului de înregistrări se micşorează cu 5;
E) se vizualizează numărul de înregistrări dein fişier;
F) se vizualizează mărimea înregistrării din fişier.

2.2.4 Domeniul înregistrărilor

În limbajul Visual FoxPro există comenzi care acționează asupra mai multor înregistrări dintr-un fişier de date. Alegerea înregistrărilor se face folosind în comandă condiția de selectare, din mulțimea înregistrărilor unui fişier se aleg doar acelea care îndeplinesc condiția de selecție. În Visual FoxPro condițiile de selecție se realizează prin cuvintele rezervate **FOR** şi **WHILE** care sunt clauze a mai multor comenzi ale limbajului.

Clauza **FOR** (" Pentru") se foloseşte pentru selectarea înregistrărilor în funcție de o condiție logică. Se selectează acele înregistrări pentru care expresia logică este adevărată.

Expresia logică este acea expresie care întoarce un singur rezultat: adevărat (true) sau fals (false).

Clauza **WHILE** ("cât timp") realizează selecția asemănător cu clauza **FOR,** aceasta făcându-se în funcție de valoarea adevărată a expresiei logice. Spre deosebire de clauza **FOR** care după găsirea unei înregistrări care nu respectă condiția logică, continuă testarea celorlalte înregistrări, clauza **WHILE** întrerupe căutarea înregistrărilor când o înregistrare nu respectă condiția logică.

Datorită faptului că viteza de prelucrare este mult mai mare când se foloseşte clauza **FOR** este recomandabil să se folosească această clauză ori de câte ori este posibil.

Numărul înregistrărilor care se testează printr-o condiție de selecție este variabil şi el constitue domeniul de selecție. Acesta constitue o clauză distinctă pentru o mulțime de comenzi FoxPro.

Clauza domeniu inclusă într-o astfel de comandă va avea următoarele forme:

- **ALL** selectează toate înregistrările din fişierul de date;
- **NEXT**<expN> se referă la următoarele <expN> înregistrări, începând de la înregistrarea curentă;
- **RECORD** <expN> acțiunează numai asupra înregistrării cu numărul <expN>;
- **REST** selectează toate înregistrările de la cea curentă şi până la sfârşitul fişierul de date.

2.5 Operații asupra înregistrărilor

Asupra înregistrărilor dintr-un fişier de date se pot face următoarele operații principale:

- adăugare;
- modificare;
- ştergere.

2.5.1 Adăugarea de înregistrări într-un fişier de date

În paragrafele anterioare am arătat modul în care se creează un fişier de date, şi modul în care poate fi manipulată structura fişierului. Adăugarea de înregistrări se poate face în două moduri :

1. adăugarea de noi înregistrări la sfârşitul fişierului de date
2. introducerea de înregistrări noi în interiorul fişierului de date

Cea mai uzuală comandă folosită pentru introducerea de date este comanda **APPEND** cu sintaxa:

APPEND [BLANK]

Această comandă adaugă înregistrări la sfârşitul fişierului de date activ. Dacă se tastează comanda cu acelaşi nume în fereastra **Command Window,** se deschide fereastra "Append" prin care datele se introduc interactiv în fişierul de date (fig.2.16)

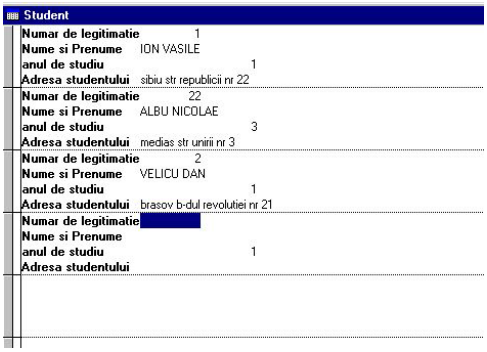


Fig. 2 16 Fereastra Append

A
B

USE SALARIAT
SET CARRY ON
SET CARRY TO SEX
APPEND
SET CARRY OFF
USE

Saltul la câmpul următor se realizează fie la apăsarea tastei Enter, fie atunci când câmpul este ocupat în totalitate cu date. Pentru a edita un câmp de tip memo, când cursorul se află în câmpul respectiv se tastează combinaţia de taste Ctrl+PgDn. Ieşirea cu salvare din fereastra de editare a câmpului memo se face cu Ctrl+W.

Datele introduse în fereastra Append sunt influenţate de comanda **SET CARRY** cu sintaxa:

SET CARRY TO <listă câmpuri>[ADITIVE]

Această comandă poate determina ca toate câmpurile sau numai anumite câmpuri introduse într-o nouă înregistrare să fie identice cu câmpurile din înregistrarea precedentă

Exemplul 2.13

Dorim să introducem persoanele de sex masculin în fişierul SALARIAT.

Explicaţii

A) Se setează câmpul SEX, pentru a fi identic cu înregistrarea precedentă. Se va scrie automat caracterul T în toate înregistrările care vor fi scrise după înregistrarea curentă.

B) Când se termină introducerea datelor se tastează **SET CARRY OFF** prin care se revine la starea inițială în care fiecare câmp dintr-o înregistrare nouă poate avea valori diferite.

O anumită listă de câmpuri poate fi extinsă prin folosirea clauzei ADDITIVE, care permite ca la o anumită listă de câmpuri existente în comanda **SET CARRY** să se adauge una nouă.

Comand **APPEND BLANK** adaugă o înregistrare "blank", adică o înregistrare în care toate câmpurile sunt vide, la sfârşitul fişierului de date urmând ca mai târziu să se încarce informația utilă prin alte comenzi FoxPro.

Comanda **APPEND FROM**<fişier>
[**FIELDS**<listă câmpuri>]
[**FOR** <expL>]
[**TYPE]**

adaugă înregistrări dintr-un alt fişier de date în fişierul de date curent. Dacă se doreşte ca în fişierul de date să fie introduse numai anumite câmpuri atunci se foloseşte opțiunea **FIELDS** în care se vor introduce câmpurile care se vor copia, separate prin virgulă. Pentru a prelua doar anumite înregistrări se va folosi opțiunea FOR. Clauza **TYPE** se foloseşte pentru a specifica tipul fişierului din care vor fi preluate datele. Acesta este diferit de tipul DBF şi aparține altor medii de programare.

Adăugarea unei înregistrări noi după înregistrarea curentă, se realizează cu comanda **INSERT** care are sintaxa:

INSERT [BEFORE][BLANK].

Dacă se foloseşte clauza opțională **BEFORE** atunci se adaugă înaintea înregistrări curente, o înregistrarea nouă. Clauza opțională **BLANK** este analoagă cu aceeaşi clauză de la comanda **APPEND**. Fereastra care apare în urma acestei comanzi este identică cu aceea de la comanda **APPEND**. În general este o comandă mai puțin folosită deoarece introducerea de înregistrări noi printre cele vechi se foloseşte mai rar cu această comandă, existând alte tehnici prin care se realizează acest lucru.

2.5.2 Modificarea înregistrărilor unui fişier de date

Modificarea informațiilor stocate într-un fişier de date se poate face cu una din comenzile :

CHANGE
EDIT
BROWSE
REPLACE

Primele trei comenzi nu modifică propriu-zis conținutul unui fişier de date, ci deschid o fereastră de editare în care utilizatorul va modifica informațiile dorite.
Comenzile **CHANGE** şi **EDIT** sunt identice , ele fiind diferite de comanda **BROWSE** numai prin modul în care sunt aranjate câmpurile în fereastra de editare. În timp ce la comenzile **EDIT** şi **CHANGE** câmpurile sunt aşezate pe verticală, la comanda **BROWSE** câmpurile sunt aşezate pe orizontală.

2.5.2.1 Comanda BROWSE

Sintaxa comenzii este:

BROWSE

[**FIELDS** <listă câmpuri>]
[**FONT** <expC1> [, <expN1>]]
[**STYLE** <expC2>]
[**FOR** <expL1>]
[**FORMAT]**
[**FREEZE** <câmpuri>]
[**KEY** <expr1> [, <expr2>]]
[**LAST | NOINIT]**
[LEDIT]
[REDIT]
[LOCK <expN2>]
[LPARTITION]
[**NOAPPEND]**
[**NOCLEAR]**
[NODELETE]
[**NOEDIT | NOMODIFY]**
[**NOLGRID] [NORGRID]**
[**NOLINK]**
[**NOMENU]**
[**NOOPTIMIZE]**
[**NOREFRESH]**
[**NORMAL]**
[**NOWAIT]**
[**PARTITION** <expN3>]
[**PREFERENCE** <expC3>]
[**REST]**
[SAVE]
[**TIMEOUT** <expN4>]
[TITLE <expC4>]
[VALID [:F] <expL2>
[ERROR <expC5>]]
[**WHEN** <expL3>]
[WIDTH <expN5>]
[[**WINDOW** <nume fereastră1>]
[**IN [WINDOW]** <nume fereastră2>
| IN SCREEN]]

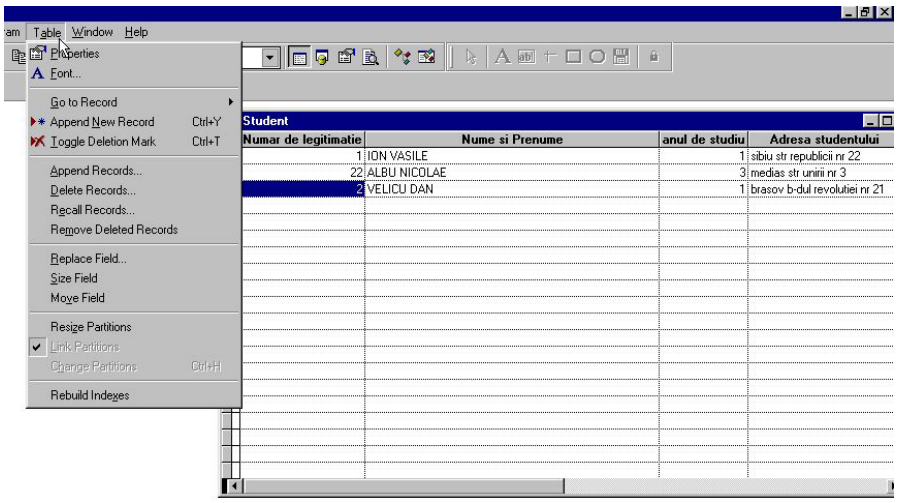


Fig. 2 17 Fereastra Browse şi meniul Table aferent acesteia

[**COLOR SCHEME** <expN6> | **COLOR** <listă perechi de culori>]

Comanda **BROWSE** fără nici o opțiune, afişează toate câmpurile din fişierul de date oferind utilizatorului, accesul cu toate drepturile la informațiile existente (fig 2.5).

La tastarea comenzii **BROWSE**, în meniul principal apare meniul **Table** care permite configurarea ferestrei de editare a fişierelor.

Opțiunea **Font** permite schimbarea fontului de editare.

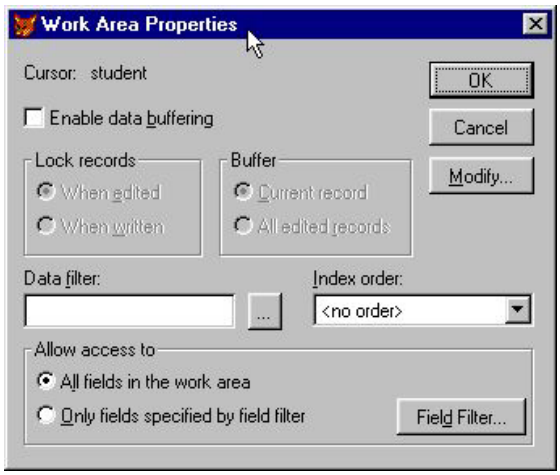


Fig. 2 18 Fereastra Browse - Properties

Opțiunea Properties deschide fereastra cu acelaşi nume (fig 2.18).

Fereastra afişează numele fişierului activ în fereasta **Browse**, permite blocarea articolelor în cazul lucrului în rețea (**Lock records**), permite definirea unui filtru asupra datelor în caseta **Data filter**, permite deschiderea fişierului indexat prin alegerea indexului dorit din caseta **Index order**, permite accesul la toate articolele din fişier sau numai la cele specificate de către filtru, prin alegerea butonului dorit din zona **Allow access to**. Butonul Field Filter permite activarea anumitor câmpuri din cadrul tabelei.

Opțiunea **Go to Record** permite poziționarea pe anumite articole din cadrul fişierului de date.

Opțiunea **Append New Record** permite adăugarea unui articol nou din fereastra **Browse.**

Opțiunea **Toggle Deletion Mark** permite marcarea pentru ştergere articolului sau a grupului de articole dorit

Opțiunea **Resize Partitions** este similară cu opțiunea PARTITIONS a comenzii **BROWSE.** Aceasta activează butonul de împărțire al ferestrei de editare în două zone distincte. Cu ajutorul tastelor direcționale sau al mouse-ului se definesc cele două regiuni denumite partiții.

Opțiunea **Size Field** permite redimensionarea câmpului activ (câmpul în care se află cursorul) cu ajutorul tastelor săgeți sau al mouse-lui.

Opțiunea **Move Field**, permite schimbarea ordinii de afişare a câmpurilor, în fereastra de afişare.

Opțiunea **Remove Deleted Records** permite refacerea articolelor şterse.

Opțiunea **Recall Records** permite demarcarea articolelor marcate pentru ştergere

Opțiunea **Delete Records**, permite marcarea pentru ştergere a unor înregistrări din fişierul de date.

Opțiunea **Append Record**, permite adăugarea de articole noi în fişierul de date dintr-un alt fişier de date.

Opțiunea **Rebuild Indexes,** permite refacerea fişierelor de index ataşate fişierului de date sau refacerea etichetelor index ataşate fişierului de date.

În continuare vom prezenta opțiunile comenzii **BROWSE**.

Opțiunea **FIELDS** a comenzii **BROWSE,** oferă utilizatorului posibilitatea de a afişa doar anumite câmpuri ale fişierului de date activ, în fereastra de editare. Numele câmpurilor care se doresc a fi afişate se vor scrie în lista de câmpuri, despărțite de virgulă. Opțiunile de editare ale câmpurilor sunt prezentate în tabelul 2.2

Tabelul 2.2

Opțiuni de editare a câmpurilor	Explicații
[:P = <expC2>]	Se poate specifica un cod PICTURE specific comenzilor de intrare ieşire (Capitolul ??????)
<câmp1> [:R] [:mărime câmp]	Câmpul se poate vizualiza pe un număr de caractere.
[:V = <expr1> [:F] [:E = <expC1>]]	Permite validarea câmpului introdus. După ce câmpul sa modificat şi se iese din editarea acestuia se evaluează <expr1> şi dacă valoarea ei este .T. (adevărat) atunci data introdusă este corectă. [:F] se foloseşte pentru a forța validarea [:E] se foloseşte pentru a schimba mesajul de eroare în cazul în care, valoarea expresiei <expr1> este falsă.
[:B = <expr2>, <expr3> [:F]]	Reprezintă intervalul în care trebuie să se găsească câmpul după editare. <expr2> reprezintă limita inferioară, iar <expr3> reprezintă limita superioară.
[:H = <expC3>]	Schimbă numele câmpului care se afişază în fereastra de editare.
[:W = <expL1>]	Se permite editarea câmpului doar dacă expresia logică <expL1> are valoarea adevărat.

Exemplu 2.14

```
USE SALARIAT
BROWSE FIELDS NUME:R:10, MATR:H='MATRICOL'
BROW FIELDS VIRSTA:V=VIRSTA>15 AND VIRSTA<50: VIRSTA;
            INVALIDA'
BROW FIELDS SALAR:W=SALAR>200000
```

A

B

C

Explicaţii

```
          USE SALARIAT
A         BROW VALID SALAR > 200000 ERROR 'DATE ERONATE'
B         BROW WHEN SEX=.F.

C         BROW FOR SALAR>1000000

          CLOSE ALL
```

A) Comandă BROWSE în care se afişază câmpul Nume pe lungime de 10 caractere şi se schimbă numele câmpului Matr în Matricol.
B) Comandă BROWSE în care asupra cămpului Virsta se pune condiţia de validare ca acesta să fie cuprins între valorile numerice 15 şi 50. În cazul în care se tastează valori diferite acestui interval apare mesajul de eroare VIRSTA INVALIDA.
C) Comandă **BROWSE** care inhibă editarea câmpului SALAR dacă valoarea salarului este mai mică decât 200000.

Opţiunea **FOR** permite apariţiei în fereastra de editare doar a acelor înregistrări care îndeplinesc condiţia logică.

Opţiunea **VALID** acţionează la nivel de înregistrare. Oţiunea este evaluată în momentul în care se trece de la o înregisrare la alta şi verifică dacă înregistrarea respectivă îndeplineşte o anumită valoare. Clauza ERROR oferă utilizatorului posibilitatea de a introduce un mesaj, diferit de mesajul sistem “Invalid input”.

Opţiunea **WHEN** se foloseşte pentru a specifica dacă o înregistrare poate fi sau nu modificată. Dacă expresia logică este .T., atunci este validat accesul la înregistrare pentru a face modificări.

Exemplul 2.15

Explicaţii

A) în cazul în care câmpul Salar din înregistrarea în care sau făcut modificările nu îndeplineşte condiţia logică Salar>200000, trecerea la o altă înregistrare, generează mesajul ‘DATE ERONATE’.
B) comanda permite numai modificarea înregistrărilor în care câmpul Sex are valoarea .F..
C) comanda vizualizează în fereastra de editare doar acele înregistrări care îndeplinesc condiţia logică.

Opţiunile **NOEDIT** şi **NOMODIFY** ale comenzii BROWSE sunt identice, ele impiedică modificarea datelor din tabele, dar permit ştergerea şi adăugarea de înregistrări.

Opţiunile **NODELETE** şi **NOAPPEND** inhibă posibilităţile de ştergere şi de adăugare de noi înregistrări în fişierul de date.

Opţiunea **NOMENU** inhibă apariţia meniului Table (Fig 2.17) în meniul principal.

Alte opţiuni ale comenzii **BROWSE** sunt prezentate în tabelul 2.2.

Tabelul 2.4

Opţiune	Semnificaţie
FONT şi STYLE	Opţiuni pentru alegerea tipului fontului de editare specifice mediului Window.
FORMAT	Informează comanda Browse că trebuie să folosească variabile din fişierul format activ. În această carte nu se vor folosi fişierele format
FREEZE	Limitează editarea doar la un singur câmp din fereastra de editare.
LAST	Preia ultima configuraţie de fereastră de editare **Browse** salvată.
LEDIT şi REDIT	Comută partiţiile stângă sau dreaptă în modul **Change.**
LOCK	Specifică numărul de coloane care rămân fixate în poziţia lor iniţială în fereastra de editare.
LPARTITION	Plasează iniţial cursorul în fereastra din stânga în cazul în care fereastra de editare este partiţionată.
NOLINK	Opţiunea nu corelează partiţiile. Deplasarea cursorului într-o partiţie nu implică deplasarea cursorului şi în cealaltă partiţie
NOOPTIMIZE	Dezactivează tehnica **Rushmore** de căutare a înregistrărilor într-un fişier de date. Aceasta este cea mai rapidă tehnică de căutare existentă în mediul VFP.
NOREFRESH	Comandă folosită doar în mediu multiutilizator.
NORMAL	Fereastra de editare este setată la atributele normale.
NOWAIT	Se poate folosi în cazul în care din program se apelează comanda Browse şi are rolul ca după afişarea ferestrei de editare programul utilizator se continuă.
PARTITION	Împarte fereastra de editare în două partiţii.
PREFERENCE	Salvează atributele ferestrei pentru o utilizare ulterioară
REST	Se utilizează împreună cu clauza **FOR** şi impune comenzii să caute înregistrările de la înregistrarea curentă până la sfârşitul fişierului.
SAVE	Se foloseşte într-un program şi forţează păstrarea pe ecran a ferestrei şi a câmpurilor memo asociate.
TIMEOUT	Specifică cât timp aşteaptă comanda **Browse** ca utilizatorul să introducă datele, înainte de a închide automat fereastra de editare.
TITLE	Oferă posibilitatea de a da un titlu ferestrei de editare.
WIDTH	Defineşte lăţimea maximă a unei coloane din fereastra de editare.
WINDOW	Afişază fereastra de editare în interiorul unei ferestre definită anterior
COLOR	Permite să se indice o configuraţie de culori prestabilite sau o listă de atribute pentru culoare.

2.5.2.2 Comanda REPLACE

Modificarea câmpurilor unui fişier de date poate fi efectuată şi cu comanda **REPLACE** care spre deosebire de comenzile prezentate anterior nu deschide o fereastră de editare. Sintaxa acestei comenzi este:

REPLACE <câmp1> **WITH** <expr1> [**ADDITIVE]**
[, <câmp22> **WITH** <expr2>
[**ADDITIVE**]] ...

USE SALARIAT

A **GO RECORD** 4
REPLACE NUME **WITH** ‘IONESCU’, PRENUME **WITH** ‘ION’, MATR; **WITH** 1012

B **APPE BLANK**
REPLACE NUME **WITH** ‘POPESCU’, PRENUME **WITH** ‘ION’, MATR; **WITH** 1011
REPLACE SEX **WITH** .T., STARE_CIVL WITH .F.

C **GO TOP**
REPLACE SALAR **WITH** 1000000 **NEXT** 3

D **REPLACE** SALAR **WITH** 150000 **FOR** NUME=’IONESCU AND; ‘PRENUME=’ION

CLOSE ALL

[<domeniu>]
[**FOR** <expL1>]
[**WHILE** <expL2>]
[NOOPTIMIZE]

Comanda înlocuieşte vechea valoare din câmpul1 cu valoarea rezultată în urma evaluării expresiei expr1, câmpul2 cu expr2 şi aşa mai departe. Pentru câmpurile memo se foloseşte clauza **ADDITIVE** prin care câmpul memo nu este şters, valoarea expresiei fiind ataşată la sfârşitul câmpului.

Comanda **REPLACE** se foloseşte în general cu comanda **APPEND BLANK** pentru a introduce înregistrări noi într-un fişier.

Exemplul 2.16

Explicaţii

A) Pentru înregistrarea cu numărul 4 se înlocuieşte conţinutul câmpurilor Nume, Prenume, Matr;
B) Se adaugă o nouă înregistrare la sfârşitul fişierului, în care se vor complecta câmpurile: Nume, Prenume, Natr, Sex, Stare_civl;
C) Se modifică câmpul Salar pentru trei înregistrări începând de la înregistrarea curentă;
D) Se modifică câmpul Salar pentru înregistrarea care conţine în câmpurile Nume şi Prenume, expresiile IONESCU şi ION.

2.5.3 Ştergerea înregistrărilor dintr-un fişier de date.

Ştergerea unei înregistrări dintr-un fişier de date se poate realiza la două nivele şi anume:

1. **la nivel logic**, când înregistrarea nu este propriu zis ştearsă din fişierul de date ci ea este "marcată pentru ştergere", indicând astfel această stare a înregistrării. Înregistrările marcate pentru ştergere pot fi determinate de funcţia **DELETED**().
2. **la nivel fizic**, când înregistrarea este, efectiv ştearsă din fişierul de date, ulterior ea nu mai poate fi recuperată.

Marcarea pentru ştergere se face prin comanda **DELETE** cu sintaxa:

A
```
USE SALARIAT
SET DELETED ON
GO TOP
DELETE NEXT 4
```

B
```
DELETE NEXT 1 FOR SEX=F
```

C
```
DELETE REST FOR STARE_CIVL = T
```

D
```
DELETE NEXT ALL
CLOSE ALL
```

DELETE [<domeniu>] [FOR<expL1>] [WHILE<expL2>][NOOPTIMIZE]

Dacă nu se specifică nimic în comandă atunci se marchează pentru ştergere înregistrarea curentă. <domeniu> este domeniul înregistrărilor. Clauzele FOR şi WHILE sunt clauzele generale de căutare a înregistrărilor, în funcţie de o expresie logică. Clauza **NOOPTIMIZE** se referă la dezactivarea metodei Rushmore.

Controlul la înregistrările marcate pentru ştergere este dat de către comanda:

SET DELETED ON|OFF

Dacă se alege opţiunea ON atunci înregistrările marcate pentru ştergere nu vor fi accesibile celorlalte comenzi Visual FoxPro care vor folosi aceste înregistrări. Opţiunea OFF permite accesul celorlalte comenzi Visual FoxPro la înregistrările marcate pentru ştergere.

Exemplul 2.16

Explicaţii

A) Şterge primele 4 înregistrări de la începutul fişierului.
B) Şterge primul articol care are îndeplinită condiţia logică câmpul Sex=F.
C) Şterge toate articolele din fişier care au în câmpul Stare_civl valoarea T.
D) Şterge toate articolele rămase în fişier de la înregistrarea curentă până la sfârşitul fişierului.

Înregistrare marcată pentru ştergere nu este ştearsă fizic. Informaţia marcată pentru ştergere poate fi recuperată cu ajutorul comenzii **RECALL** cu sintaxa:

RECALL [<domeniu.] **[FOR**<expL1>] [**WHILE**<expL2>].

Clauzele acestei comenzi sunt identice cu clauzele comenzi DELETE. Dacă nu se specifică nici o clauză se recuperează înregistrarea curentă.

Ştergerea efectivă a înregistrărilor dintr-un fişier de date se face cu comanda **PACK**. După aplicarea acestei comenzi înregistrările nu mai pot fi refăcute.

Toate înregistrările dintr-un fişier pot fi şterse global prin comanda ZAP. Această comandă activează o fereastră de dialog care validează sau nu comanda. Informaţiile şterse din fişier sunt distruse definitiv, fără posibilităţi de refacere. Comanda ZAP este echivalentă cu comenzile: DELETE ALL şi PACK.

În limbajul Visual Fox, există o comandă prin care, fără ştergere sau marcare pentru ştergere, se poate înhiba accesul la unele înregistrări. Această comandă este:

SET FILTER TO [<expL>]

Ca efect al acestei comenzi în fişierul de date vor apare doar înregistrările care îndeplinesc condiţia expL.

Exemplul 2.18

A
```
USE SALARIAT
SET FILTER TO STARE_CIVL=.T.
BROW
```
B
```
?FILTER()
CLOSE ALL
```

Explicaţii

A) Se aleg din fişierul de date numai acele înregistrări care au în âmpul Stare_civl valoarea adevărat.

B) Se vizualizează condiţia de filtrare cu ajutorul funcţiei FILTER().

2.6 Vizualizarea conţinutului unei fişier de date

Vizualizarea conţinutului unui fişier de date se poate face folosind una dintre comenzile pentru modificarea înregistrărilor fişierului de date, prezentate în paragraful precedent. În acest caz, pentru a avea o vizualizare strictă a fişierului de date este necesară inhibarea posibilităţilor de scriere în câmpurile înregistrărilor şi ştergerea acestora.

Comenzile specifice pentru vizualizare înregistrărilor sunt comenzile **DISPLAY** şi **LIST**.

Comanda **DISPLAY** are sintaxa:

DISPLAY
[[FIELDS] <listă câmpuri>
| **FIELDS LIKE** <mască>
| **FIELDS EXCEPT** <mască>]
[<domeniu>]
[FOR <expL1>]
[WHILE <expL2>]
[**OFF]**

[**TO PRINTER** [PROMPT]
| **TO FILE** <nume fişier>]
[**NOCONSOLE]**
[**NOOPTIMIZE**]

Afişarea informaţiilor se face astfel :

- pe prima linie se reprezintă lista câmpurilor;
- pe următoarele linii se reprezintă înregistrări din fişierul de date. Pe prima poziţie se afişează numărul de ordine al înregistrării din fişierul de date.

Opţiunea **FIELDS** specifică lista de câmpuri pentru afişare. Dacă lipseşte se afişează toate câmpurile. Opţiunile **FIELDS LIKE** şi **FIELDS EXCEPT** afişează sau nu mai multe câmpuri după o anumită mască de identificare.

USE SALARIAT
A **DISPLAY**

B **DISPLAY ALL OFF**
GO RECORD 3
C **DISPLAY FIELDS** NUME,PRENUME,SALAR

D **DISPLAY FIELDS** SALAR **REST**
GO TOP
E **DISPLAY ALL FOR** STARE_CIVL=.F.

F **DISPLAY ALL TO** FILE SALAR.TXT **NOCONSOLE**
CLOSE ALL

Opţiunile <domeniu> **FOR** şi **WHILE** sunt cele specifice domeniului înregistrărilor.

Opţiunea **OFF**, inhibă apariţia în formatul de afişare, a informaţiei de pe coloana 0 reprezentând numărul de ordine al înregistrărilor.

Opţiunile **TO PRINTER** şi **TO FILE**, permite transmiterea rezultatelor vizualizării la imprimantă sau într-un fişier ASCII.

Opţiunea **NOCONSOLE** inhibă vizualizarea conţinutului fişierului pe ecranul videoterminalului calculatorului

Implicit dacă în [domeniu] nu se specifică nimic atunci cu **DISPLAY** se vizualizează înregistratrea curentă.

Exemplul 2.19

Explicaţii

A) Vizualizarea primei înregistrări din fişierul Salariat;
B) Vizualizarea tuturor înregistrărilor din fişier, fără afişarea pe coloana 0 a indicatorului de înregistrări;

C) Vizualizarea câmpurilor Nume, Prenume şi Salar de la treia înregistrare din fişierul de date;
D) Vizualizarea tuturor înregistrărilor de la înregistrarea cu numărul 3 până la sfârşitul fişierului de date;
E) Vizualizarea tuturor înregistrărilor care conţin valoarea fals pentru câmpul Stare_civl;
F) Vizualizarea tuturor înregistrărilor în fişierul text, cu numele Salar.txt, fără afişarea rezultatelor vizualizării pe ecran.

Comanda **LIST** este asemănătoare cu comanda DISPLAY, având aceeaşi sintaxă, dar cu următoarele deosebiri:

- domeniul implicit al comenzii vizualizează toate articolele fişierului de date;
- în timp ce comanda **DISPLAY** afişează ecran cu ecran, trecerea de la un ecran la altul făcându-se prin apăsarea unei taste, comanda **LIST** afişează fără pauză între ecrane, toate înregistrările;
- Comanda **LIST** nu afişează înregistrările marcate pentru ştergere, în cazul în care avem activă comanda **SET DELETED ON**, pe când comanda **DISPLAY** afişează şi aceste înregistrări.

Pentru ca prima linie de vizualizare care conţine câmpurile dintr-un fişier să nu apară ca urmare a comenzilor **DISPLAY** sau **LIST** se foloseşte comanda **SET HEADING OFF.**

2.7 Fereastra Data Session

Pentru afişarea zonelor de lucru, a fişierelor active într-o aplicaţie precum şi a legăturilor dintre acestea se foloseşte fereastra **DATA SESSION** (fig 2.19), care se

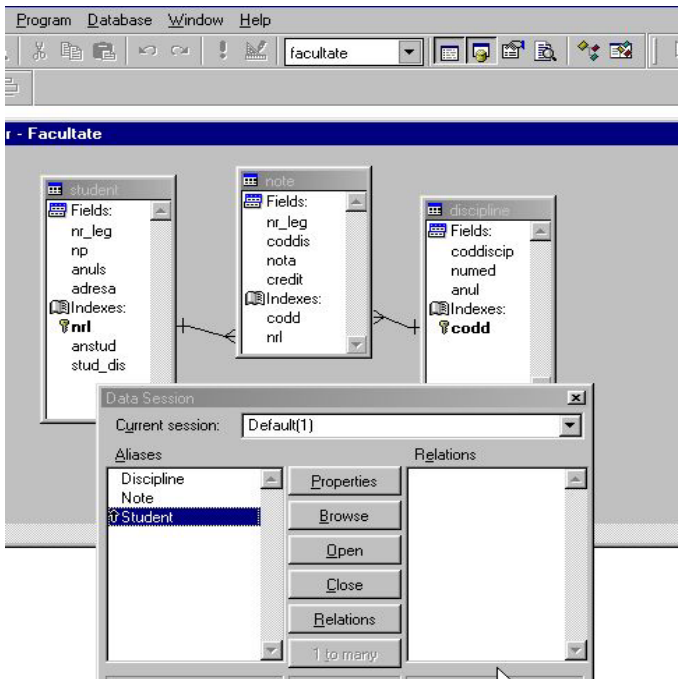


Fig 2. 19 Fereastra Data Session

apelează din meniul Window.

Această fereastră afişază o listă derulabilă de zone de lucru, aliasurile tabelelor deschise şi o schemă a tuturor relaţiilor stabilite între fişierele de date.

În tabelul 2.5 sunt descrise funcţiile butoanelor din partea centrală a ferestrei.

Tabelul 2.5

PROPERTIES	Activează fereastra **Work Area Properties** (fig 2.18)
RELATIONS	Activează fereastra de setare a legăturilor dintre fişierele de date.
BROWSE	Activează fereastra de editare cu acelaşi nume corespunzătoare unui fişier de date.
OPEN	Activează un fişier de date.
CLOSE	Închide un fişier de date.

Înregistrările dintr-un fişier pot corespunde uneia sau mai multor înregistrări din alt fişier, caz în care se spune că între cele două fişiere există o legătură sau în terminologia relaţională a bazelor de date, o **relaţie**. Aşa cum s-a arătat anterior în acest capitol, fişierul de unde se stabileşte relaţia se numeşte fişier "părinte", iar fişierul "copil" este fişierul în care s-a stabilit relaţia. Pentru stabilirea unei relaţii între două fişiere de date, trebuie ca ambele să fie deschise în câte o zonă de lucru. Se poate stabili o relaţie, numai între câmpuri care memorează date comune.

Pentru a stabili interactiv o relaţie între două fişiere, acestea se deschid în două zone de lucru diferite. Zona curentă de lucru este zona în care este deschis fişierul părinte. Se acţionează apoi butonul **Relation**, pe ecran va apare fişierul părinte cu o săgeată îndreptată în jos şi spre dreapta. Se stabileşte fişierul copil din zonele de lucru şi relaţia de legătură cu ajutorul constructorului de expresii.

2.8 Ordonarea unui fişier de date

Ordonarea înregistrărilor în Visual FoxPro se poate realiza în două moduri ;

- prin ordonarea propiu-zisă a fişierului de date în ordinea dorită a înregistrărilor. Se creează practic un nou fişier de date care conţine aceleaşi înregistrări ca şi primul, dar de data aceasta înregistrările sunt ordonate după unul sau mai multe câmpuri, numite cheie de ordonare.
- prin indexarea fişierului de date, aceasta presupunând crearea unui fişier care conţine informaţii cu privire la ordinea înregistrărilor în fişierul de date. Aceasta ***NU CONŢINE*** înregistrările fişierului de date, fişierul index ţine evidenţa articolelor din fişierul de date într-o anumită ordine.

2.8.1 Sortarea fişierelor de date

Sortarea fişierelor de date, adică aranjarea înregistrărilor într-o anumită ordine, funcție de unul sau mai multe câmpuri ale fişierului de date, se realizează cu comanda **SORT** a cărei sintaxă este:

SORT TO <nume fişier>
ON <câmp1>[/A][/D][/C],
[Câmp 2**][/A][/D][/C]** ...
[**ASCENDING][DESCENDING**]
[<domeniu>][**FOR** <expL1>][**WHILE** <expL2>]
[**FIELDS**<listă câmpuri>]
[**NOOPTIMIZE]**

Comanda creează un nou fişier de date, al cărui nume este dat de către clauza **TO**. Câmpul sau câmpurile după care se face sortarea sunt denumite chei de sortare, numele acestora fiind dat de către clauza **ON.** Parametrii A şi D se folosesc pentru a realiza ordonarea crescătoare sau descrescătoare a înregistrărilor din fişier şi sunt similari cu clauzele **ASCENDING** şi **DESCENTING.** Parametrul C se foloseşte pentru a compara şiruri de caractere şi pentru a nu face deosebire între literele mari sau mici. Clauza ASCENDING este implicită. Fişierul de date rezultat în urma operației de sortare poate avea toate câmpurile fişierului de date inițial sau poate avea numai anumite câmpuri carepot fi specificate în clauza **FIELDS**.
Dacă nu se doreşte ordonarea tuturor înregistrărilor dintr-o bază de date atunci se pot folosi clauzele :

- **<**domeniu**>** prin care se poate selecta un anumit număr de înregistrări din fişierul de date;
- FOR sau WHILE prin care, înregistrările se selectează în funcție de o anumită condiție logică

Exemplu 2.20

Se consideră fişierul de date Salariat.dbf.

```
CLEA
CLOSE ALL
DELE FILE MATRICOL.DBF
DELETE FILE ALFAB.DBF
DELETE FILE SALAR.DBF
DELETE FILE SALNUM.DBF
USE SALARIAT
```

Explicaţii

A) aranjarea înregistrărilor din fişierul de date după câmpul Matr în fişierul Matricol în ordine ascendentă;
B) aranjarea înregistrărilor din fişierul de date după câmpurile Nume şi Prenume în fişierul Nume în ordine ascendentă;
C) aranjarea înregistrărilor din fişierul de date care îndeplinesc condiţia logică salar>500000, în ordine descendentă, după cheia Matr, în fişierul Salar;
D) aranjarea înregistrărilor în ordie ascendentă după câmpurile Salar şi Nume, în fişierul Salnum. Noul fişier de date are doar câmpurile Nume şi Salar.

2.8.2 Indexarea fişierelor de date

Indexii sunt structuri de date care au drept unic scop, creşterea vitezei de accesare a articolelor. La crearea structurii fişierelor de date am prezentat tipurile de indecsi care pot fi create. În afară de metoda prezentată, fişierele index pot fi create prin comenzi tastate în fereastra de comandă. Fişierele de index sunt fişiere auxiliare pe disc, ele sunt diferite de fişierele de date, căutarea se face după valorile anumitor câmpuri care poartă denumirea de cheie de indexare. Indexul este definit asupra unui câmp unic din fişier, el conţine valorile câmpului de indexare, ordonate astfel încât să se poată face o căutare liniară asupra lor. Indexul mai conţine un câmp în care se găsesc pointeri sau adrese la blocuri sau înregistrări din fişierul de date. Dimensiunile fişierului index sunt mai mici decât dimensiunile fişierului de date, de aceea căutarea se face mult mai rapid. Indexarea se poate face pe un câmp care are valori distincte în cadrul fişierului de date, caz în care indexul se numeşte index primar. Indexarea se

A
```
SORT TO MATRICOL ON MATR
SELE 2
USE MATRICOL
BROW
SELE 1
```

B
```
SORT TO ALFAB ON NUME,PRENUME
SELE 3
USE ALFAB
BROW
SELE 1
```

C
```
SORT TO SALAR ON MATR FOR SALAR >500000 DESC
SELE 4
USE SALAR
BROW
SELE 1
```

D
```
SORT TO SALNUM ON SALAR,NUME FIELDS NUME,SALAR
SELE 5
USE SALNUM
BROW
CLOSE ALL
```

poate face după o cheie de indexare (unul sau mai multe câmpuri) care nu are valori distincte în tot fişierul, caz în care fişierul se numeşte index prin clusterizare. Un fişier mai poate avea mai multe fişiere de index secundare, care pot fi specificate pe un câmp oarecare din fişier, în afară de câmpurile principale de indexare.

În limbajul Visual FoxPro există două tipuri de fişiere index:

- fişierele cu extensia **IDX** care conţin un singur index;
- fişierele cu extensia **CDX**, care sunt fişiere index compuse.

Numărul de indecşi dintr-un fişier index compus este limitat doar de capacitatea memoriei disponibile; fiecare index poartă denumirea de etichetă (tag). Un fişier index compus care are acelaşi nume cu fişierul de date DBF şi care este legat de acesta poartă denumirea de fişier index compus structurat. Fişierele CDX au avantaje faţă de fişierele IDX prin faptul că ele conţin mai mulţi indecşi înglobaţi. Deschiderea unui fişier de date care are ataşaţi mai mulţi indecşi simpli, poate dura foarte mult. De asemenea fişierele indexate compuse asigură o integritate sporită fişierelor index, prin legarea strânsă a fişierelor de date de indecşii care îi sunt asociaţi.

Fişierele index atât cele simple cât şi cele compuse se prezintă sub forma unor arbori B^+. Fişierele indexate compuse au câte un arbore B^+ pentru fiecare etichetă de indexare.

Fişierele index trebuie să fie deschise ori de câte ori se deschide şi fişierul de date DBF şi se realizează modificări ale articolelor din el. Altfel apar anomalii în funcţionarea fişierelor index. În cazul fişierelor compuse structurale acest lucru poate fi neglijat deoarece aceste fişiere de index se deschid automat la deschiderea fişierului de date **DBF**.

Fişierele index simple se recomandă a se crea atunci când dorim să creeăm un index temporar care există un timp limitat de timp, după care se şterge cu o comandă **DELETE FILE** <nume_fişier_index>. În cazul în care se creează fişiere index care vor fi folosite în mod frecvent, un timp mai îndelungat, soluţia cea mai bună este de a folosi fişiere index compuse, adică fişierele index **CDX**, deoarece etichetele de index într-un astfel de fişier sunt actualizate în mod permanent.

Indecşii pot fi creaţi cu comanda **INDEX**, iar cu comanda **REINDEX**, poate fi refăcut un index alterat. Comanda **INDEX** de creare a unui fişier index simplu sau compus are sintaxa:

INDEX ON <exp_cheie> TO <fişier_idx>
| TAG <nume_etichetă>
[**OF** <fişier_cdx>]
[**FOR** <expL>]
[**COMPACT]**
[**ASCENDING | DESCENDING]**
[**UNIQUE]**
[**ADDITIVE]**

Cheia de indexare este descrisă de către expresia aflată în clauza **ON**. Această expresie poate fi de tip: şir de caractere, numeric, logic, dată calendaristică, funcţie

definită de utilizator. Această ultimă noțiune va fi explicată pe larg în capitolul "Proceduri şi funcții". Cheie de indexare nu poate fi un câmp de tip memo. Lungimea unei expresii a unei chei de indexare poate fi de 100 de caractere pentru fişierele IDX şi 240 de caractere pentru fişierele CDX.

Dacă se specifică clauza TO se crează un fişier index de tip IDX, iar dacă se specifică clauza TAG se crează un fişier compus. Fişierele compuse sunt de două categorii:

- structurale, caz în care au acelaşi nume cu fişierul de date, fără a mai fi specificat un alt nume de fişier indexat în clauza **OF** a comenzii **INDEX**;
- nestructurale, adică ele poartă alt nume decât cel al fişierului de date, nume care este specificat în clauza OF a comenzii **INDEX**

Clauza **COMPACT** se referă doar la fişierele index simple şi oferă posibilitatea de a realiza fişiere index compactate, adică fişiere index cu dimensiuni mai reduse decât un fişier index obişnuit. Totodată accesul la înregistrări în fişierele compactate este mai rapid. Fişierele index compuse sunt totdeauna compactate.

În cazul în care se doreşte indexarea numai a anumitor înregistrări din fişierul de date se foloseşte clauza **FOR**. Această clauză poate fi specificată numai atunci când este creată o etichetă a unui fişîer index compus. Expresia logică din clauza **FOR** poate conține variabile de memorie, funcții definite de utilizator sau câmpuri ale unor alte fişiere DBF, deschise în alte zone de memorie.

Dacă este specificată clauza **UNIQUE** la crearea unui fişier index, atunci doar prima înregistrare pentru care valoarea cheii de indexare este aceeaşi, este introdusă în fişierul index.

Clauza **ASCENDING** este implicită şi semnifică ordonarea înregistrărilor în ordine crescătoare. Clauza **DESCENDING** se referă doar la fişierele indexate compuse.

Deschiderea unui fişier index se realizează cu comanda **USE** a cărei sintaxă este:

USE [<fişier> | ?]
[**IN** <expN1>]
[**AGAIN**]
[**INDEX** <lista_fişiere_indexate> | ?
[**ORDER** [<expN2>
| <fişiere_indexate_idx>
| [**TAG**] <nume_etichetă>
[**OF** <fişiere_cdx>]
[**ASCENDING** | **DESCENDING**]]]]
[**ALIAS** <alias>]
[**EXCLUSIVE**]
[**SHARED**]
[**NOUPDATE**]

CLOSE ALL
SET TALK OFF
DELE FILE inume.idx
DELE FILE matricol.idx
USE salariat
INDEX ON nume to inume

A

INDEX ON matr to matricol

B

INDEX ON nume **TAG** nume

C

INDEX ON matr **TAG** matricol

D

INDEX ON salar **TAG** salar **FOR** salar>500000 **DESCENDING**

E

USE salariat **INDEX** inume
BROW

F

USE salariat **INDEX** matricol
BROW

G

USE salariat **ORDER TAG** nume
BROW

H

USE salariat **ORDER TAG** matricol
BROW

I

USE salariat **ORDER TAG** salar
BROW

J

CLOSE ALL

Comanda **USE** care se foloseşte pentru deschiderea unui fişier de date poate fi folosită şi pentru deschiderea unui fişier indexat de tip IDX sau CDX.

Clauza **INDEX** prezintă o listă a fişierelor **IDX**. Dacă se tastează caracterul ? se deschide o fereastră de dialog din care se va alege fişierul indexat dorit. Fişierele din listă vor fi separate prin virgulă.

Clauza **ORDER** determină fişierul din lista de fişiere **IDX** care va devenii activ sau eticheta (TAG) care va deveni activă. Apelul acestora se poate face şi prin expresia numerică <expN>, care reprezintă numărul de ordine al fişierului în lista de fişiere sau numărul de ordine al etichetei în lista de etichete.

Clauza **TAG** individualizează o etichetă cu un anumit nume.

Clauza **OF** face referinţă la un fişier **CDX** nestructural, deci care are nume diferit de cel al fişierului de date activ.

Exemplul 2.21

Se consideră fişierul de date Salariat.dbf, care este indexat şi deschis în mai multe moduri.

Explicaţii

A) crearea unui fişier indexat simplu cu numele inume.idx;
B) crearea unui fişier indexat simplu cu numele matricol.idx;
C) crearea unui fişier indexat compus cu acelaşi nume cu numele fişierului de date, Salariat.cdx, cu cheia de indexare nume şi eticheta nume;
D) adăugarea etichetei matricol, corespunzătoare cheii de indexare matr, în fişierul Salariat.cdx;
E) adăugarea etichetei Salar, corespunzătoare cheii de indexare Salar în fişierul Salariat.cdx, pentru înregistrările care îndeplinesc condiţia logică "salar>500000 ", aranjarea înregistrărilor făcând-se descrescător;
F) deschiderea fişierului indexat simplu inume.idx;
G) deschiderea fişîerului indexat simplu matricol.idx
H) deschiderea fişierului indexat compus Salariat.cdx, cu selectarea etichetei Nume;
I) se selectează eticheta Matricol din fişierul indexat compus Salariat.cdx;
J) se selectează eticheta Salar din fişierul indexat compus Salariat.cdx.

Asupra fişierelor indexate pot acţiona o serie de funcţii care returnează informaţii despre fişierele indexate:

- **NDX**() returnează numele fşierelor indexate simple deschise într-o anumită zonă de lucru;
- **CDX()** returnează numele fşierelor indexate compuse deschise într-o anumită zonă de lucru;
- **TAG(), SYS(21)** returnează numele fişierului simplu sau al etichetei unui fişier indexat compus, dintr-o listă existentă;
- **KEY(), SYS(14)** returnează numele cheii de indexare specificată la crearea fişierului index simplu sau al etichetei unui fişier indexat compus;
- **ORDER()** returnează numele fişierului index simplu activ sau al etichetei active.

În cazul în care cheia de indexare este formată din mai multe câmpuri este necesară formarea unei expresii de indexare de tip şir de caractere. Dacă câmpurile sunt de tipuri diferite, este obligatorie folosirea unor funcţii specifice mediului Visual Fox pentru conversia datelor dintr-un tip de dată în alt tip de dată. În figura 2.20 este prezentată realizarea unui index folosind o cheie de indexare formată din două câmpuri de tipuri diferite. Câmpul de tip numeric este convertit în tip şir de caractere folosind funcţia STR. Formarea cheii de indexare din cele două câmpuri se realizează folosind operatorul de concatenare "+".

Acelaşi index se poate realiza folosind comanda de indexare în fereastra de comandă:

INDEX ON STR(anuls)+np **TAG** STUD_DIS

Deschiderea fişierului se realizează cu comanda:

USE STUDENT ORDER TAG STUD_DIS

În general în mediul Visual Fox folosirea fişierelor de index de tip Idx, nu este interzisă dar se folosesc mai rar. Efectul prin care se crează un fişier de index atât din

fereastra **Table Designer** sau prin comanda **INDEX**, este identic, un fişier cu extensia **CDX**, care are eticheta de index STUD_DIS

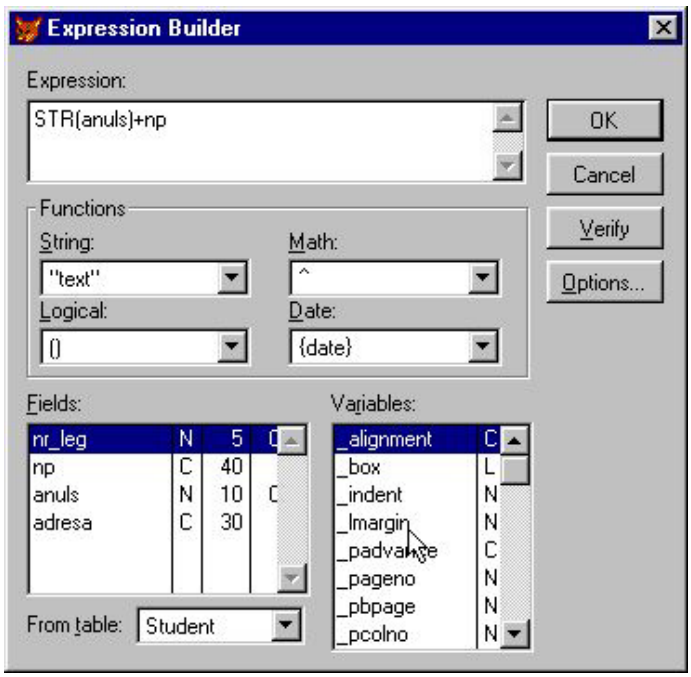


Fig 2. 20 Index care are cheia de indexare formată din două câmpuri de tip diferit

Concluzii

În mediul Visual Fox *suportul de memorare* al datelor este *relaţia* sau *tabela* sau *fişierul de date*, care poate exista independent sau într-o bază de date. Existenţa tabelei într-o bază de date VFP, este soluţia recomandată de către autor datorită facilităţilor pe care le oferă.

Din punct de vedere obiectual, o bază de date poate fi considerată o clasă de tip container cu numele **Data Environment**, clasă care conţine obiecte de tip **Cursor,** care sunt tabelele bazei de date, vederile acesteia sau legăturile dintre fişierele de date, ale bazei de date. Despre toate acestea vom discuta în capitolul referitor la programarea orientată obiect în mediul Visual Fox.

Capitolul

3

VARIABILE, CONSTANTE, EXPRESII, FUNCȚII SPECIFICE TIPURILOR DE DATE ÎN LIMBAJUL VISUAL FOX

3.1 Obiecte de date

Variabilele reprezintă informația pe care o poate manipula un limbaj de programare. Limbajul Visual Fox oferă două tipuri de **variabile scalare**:

- **câmpurile**, care constitue elementele constructive ale fişierelor de date DBF şi au un caracter permanent;
- **variabile de memorie**, care există temporar, pe durata execuției unui program. Ele pot fi salvate într-un fişier şi aduse în memoria calculatorului, în funcție de dorința utilizatorului.

Variabilele scalare se mai numesc în mediul Visual Fox **obiecte de date.** Spre deosebire de alte limbaje în Visual Fox, variabilele nu sunt definite în program într-o zonă anume şi păstrează tipul cu care au fost definite pe toată durata programului. Lungimea şi tipul unei variabile de memorie sunt ajustabile pe durata execuției unui program.

Un program Visual Fox este o înşiruire de instrucțiuni care au trei părți distinctive şi anume:

- **semantica** sau semnificația instrucțiunii;
- **sintaxa** sau modul de scriere al instrucțiunilor;
- **lexica** sau regulile pe baza cărora se formează elementele sintaxei.

Totalitatea instrucțiunilor unui program formează "**programul sursă"**. Pentru ca acesta să poată fi înțeles de către mediul Visual Fox, este necesar să fie compilat adică să fie transformat în instrucțiuni maşină, instrucțiuni care sunt interpretate de către calculator. Programul sursă se tipăreşte într-o fereastră specială denumită "fereastra de program" şi care poate fi apelată prin comanda **MODIFY COMMAND** nume_program din fereastra de comandă, sau din meniul principal opțiunea **File,** subopțiunea **New** şi opțiunea **Program** din lista care apare în fereastra de dialog New. În general la începutul unui program se inițializează variabilele globale, constantele şi variabilele care vor fi folosite în program, fară ca acest lucru să fie

obligatoriu, alocarea tipului variabilei se realizează dinamic prin program. Alocarea dinamică înseamnă că tipul este atribuit pe parcursul executării programului în momentul creării variabilei, putând fi schimbat ori de câte ori doreşte programatorul.

Într-un program sursă Visual Fox, se pot face diverse comentarii. Acestea fac referiri la diverse puncte ale programului pentru ca acesta să fie mai uşor de înțeles. Comentariile se introduc pe un rând nou, cu ajutorul caracterului "*****", tastat pe prima poziție a unui rând, sau cu ajutorul combinației de caractere "**&&**", introduse pe un rând în locul în care se doreşte să se facă un comentariu.

Exemplul 3.1

```
**************************************************
*Programul  calculează aria şi volumul unei sfere, functie de
*o anumită rază. Rezultatele se memoreaza in fisierul sfera.dbf
*cu structura formata din cimpurile aria, raza si sfera
**************************************************
close all
@ 1,1 say 'Raza = ' get raza default 0 picture '9999.99'
read
aria= 4*pi()*raza*raza  && Calculeaza aria sferei
volum=aria*raza/3  && Calculează volumul sferei
@2,1 say 'Aria       = ' get aria && Afişază aria sferei
@3,1 say 'Volumul = ' get volum && Afişază volumul sferei
return
```

Comentariile nu sunt interpretate de către compilator, ele sunt folosite doar pentru explicitarea programelor sursă.

3.1.2 Atributele unui obiect de date

Oricare obiect de date, câmp sau variabilă de memorie, are următoarele atribute:

- identificator;
- tip;
- valoare;
- lungime.

Un şir de caractere care defineşte un nume de variabilă,constantă, procedură sau funcție se numeşte **identificator,** care rămine nemodificat pe durata rulării unui program.

În limbajul Visual Fox **identificatorii** respectă următoarele reguli de bază şi anume:

- pot să fie formați din oricâte caractere, dar numai primele 10 caractere sunt luate în considerare;
- trebuie să înceapă cu un caracter alfabetic sau liniuța de subliniere;

- pot să cuprindă caractere alfabetice, numerice sau liniuţa de subliniere “_”.

Modul de scriere cu majuscule sau litere mici nu are importanţă. Compilatorul interpretează la fel literele mari şi literele mici. Limbajul Visual Fox nu are o listă de cuvinte rezervate, dar nu este indicat să folosiţi cuvinte cheie Visual Fox ca şi identificatori. Cuvintele cheie pot fi trunchiate până la 4 caractere, compilatorul Visual Fox luându-le în considerare. Identificatorii sunt de două tipuri:

- definiţi de către utilizatori, în cadrul unui program;
- standard, care sunt proprii mediului Visual Fox şi care definesc funcţii standard, precum şi **variabile de sistem**.

Variabilele de sistem, conţin informaţii globale referitoare la sistemul Visual Fox. De obicei variabilele de sistem au pe prima poziţie o liniuţă de subliniere “_”. Variabilele de memorie admit un singur tip de date la un moment dat, cu toate că tipul unui obiect de date se poate modifica pe parcursul unui program. În general variabilele de memorie sunt locaţii de memorie care au un nume (identificator) şi conţin informaţii de o anumită natură.

3.1.3 Constante, expresii, funcţii

Constantele sunt denumite uneori reprezentări literale, deoarece valorile lor sunt valori concrete, ele nu fac referire la alte valori ca şi variabilele sau câmpurile din fişierele de date. Constantele sunt de următoarele tipuri:

- **Caracter;**
- **Logic;**
- **Dată calendaristică;**
- **Numeric**

Constantele **caracter,** care se mai numesc şi şiruri pot fi formate din maximum 64K de caractere, fiind definite prin intermediul ghilimelelor. Ghilimelele pot fi formate din caracterele: “, ‘, [,]. De exemplu şirul Universitate poate fi scris:

‘Universitate’
“Universitate”
[Universitate]

Constantele caracter conţin în general orice caracter permis de către sistem. Caracterele sunt reprezentate în codul ASCII. Pentru a introduce caractere speciale care nu se tipăresc într-un şir se recomandă folosirea funcţiei CHR() care returnează caracterul al cărui cod ASCII a fost introdus între paranteze. De exemplu caracterele de întoarecere a carului (carriage return) şi de salt la linie nouă (line feed) se obţin cu ajutorul funcţiei CHR() astfel:

CHR(13) - carriage return;
CHR(10) - line feed.

Constantele logice sau constantele booleene pot conţine doar două valori: TRUE (adevărat) şi FALSE (fals). Visual Fox permite utilizarea caracterelor YES

pentru TRUE şi NO pentru FALSE. Datorită acestui fapt, mulțimea constantelor logice este .T., .F., .Y., .N.. În cazul constantelor logice nu se face diferență între litere mari şi litere mici, adică .T. este identic cu .t..

Constantele dată calendaristică se folosesc pentru a memora informații de tip zi, lună, an. Ele sunt definite folosind acoladele astfel:

identificator_constantă = {<cifre1> <delimitator> <cifre2> <delimitator> <cifre3>}

Forma de prezentare a datei adică a grupului de cifre <cifre1>, <cifre2>, <cifre3> este în funcție de modul în care se utilizează comanda:

SET DATE [TO] AMERICAN | ANSI | BRITISH | FRENCH | GERMAN | ITALIAN | JAPAN | USA | MDY | DMY | YMD.

Dacă se consideră cifre1=luna=ll, cifre2=ziua=zz, cifre3=anul=aa, atunci în tabelul 3.1 sunt sintetizate acțiunile comenzii SET DATE.

Tabelul 3.1

Tipul datei	Formatul
AMERICAN	ll/zz/aa
ANSI	aa.ll.zz
BRITISH	zz/ll/aa
FRENCH	zz/ll/aa
GERMAN	zz.ll.a
ITALIAN	zz-ll-aa
JAPAN	aa/ll/zz
USA	ll-zz-aa
MDY	ll/zz/aa
DMY	zz/ll/aa
YMD	aa/ll/zz

Formatul implicit este formatul AMERICAN.

Constantele numerice sunt întodeauna de tipul N, niciodată F. reprezentarea internă depinde de valorile acestor constante. Constantele numerice pot apare şi în reprezentarea ştiințifică:

Valelect = 1.6E-12;

Nravog = 2.24E+1

Variabilele şi constantele reprezintă obiectele de bază manipulate într-un program.

Expresiile sunt construcții care combină între ele variabilele şi constantele pentru obținerea unor valori noi, prin aplicarea operatorilor. Expresiile constau din următoarele:

- **Operanzi simpli** (constante, variabile sau funcții);
- **Operanzi combinați cu operatori** şi cu alți operanzi.

Funcțiile sunt formate din următoarele elemente:

- **Un nume**, care poate fi propriu sistemului Visual Fox sau poate fi definit de utilizator;
- **Zero unul sau mai mulți parametrii** cuprinşi între două paranteze "(" şi ")".

Operanzii sunt constante, variabile, funcţii. Operandul are mai multe atribute, fiecare operand având un anumit tip de date. Constantele, expresiile şi funcţiile sunt efemere, spre deosebire de variabile. Valorile lor există în memorie doar pentru o scurtă perioadă de timp (până la evaluarea expresiei următoare), iar atributele lor de lungime nu sunt fixe, spre deosebire de câmpurile fişierelor de date DBF, dar toţi operanzii au un singur tip de dată.

O expresie este formată din mai mulţi operanzi legaţi de operatori. Funcţie de tipul operanzilor care există într-o expresie sunt permişi anumiţi operatori. De exemplu operatorul "+" poate avea semnificaţia de adunare pentru varibilele numerice şi de concatenare a şirurilor pentru variabile şir de caractere.

Operatorii caracter sunt operatorii "+" şi "-" care concatenează două sau mai multe şiruri de caractere. Efectul celor doi operatori este asemănător, deosebirea dintre ei constă în faptul că operatorul "-" mută toate spaţiile libere de la sfârşitul operanzilor la sfârşitul rezultatului.

Exemplul 3.2

Se consideră următoarele comenzi tastate în fereastra de comandă "Command":

```
A='Universitatea    '
B='Lucian Blaga     '
C='din Sibiu'
?A+B
Universitatea     Lucian Blaga
?A+B+C &&Vizualizare rezultat concatenare variabile folosind operatorul "+"
Universitatea     Lucian Blaga     din Sibiu
?A-B
UniversitateaLucian Blaga
?A-B-C &&Vizualizare rezultat concatenare variabile folosind operatorul "-"
UniversitateaLucian Blagadin Sibiu
```

Explicaţii

A) Concatenarea şirurilor A şi B folosind operatorul "+";
B) Vizualizare şi comentariu aferent operaţiei de concatenare a trei şiruri folosind operatorul "+";
C) Concatenarea şirurilor A şi B folosind operatorul de concatenare "-";
D) Vizualizare şi comentariu aferent operaţiei de concatenare a trei şiruri folosind operatorul "-";

Operatorii dată calendaristică, sunt identici cu operatorii pentru şirurile de caractere; există doar patru metode permise de combinare a operanzilor şi a acestor operatori:

- <data> + <număr> rezultatul este de tip dată calendarisică;
- <număr> + <data> rezultatul este de tip dată calendarisică;
- <data> - <număr> rezultatul este de tip dată calendarisică;
- <data> - <data> rezultatul este de tip numeric;

Operatorii relaţionali sunt folosiţi în diverse expresii logice şi sunt prezentaţi în tabelul 3.2.

Tabelul 3.2

Operatorul	Descriere
=	Egal
==	Egalitatea exactă(identitate); numai pentru tipul caracter.
>	Mai mare decât
<	Mai mic decât
>= **sau** =>	Mai mare sau egal cu
<= **sau** =<	Mai mic sau egal cu
<>	Diferit de
#	Diferit de (identic cu <>)
!=	Diferit de (identic cu <>)
$	Este cuprin în (numai pentru tipurile caracter şi memo)

Aceşti operatori se aplică doar la operanzi de acelaşi tip. Operatorii relaţionali se pot aplica doar operanzilor şiruri de caractere, dată calendaristică şi numerici. Operanzii dată calendaristică nu pot fi comparaţi cu operanzii numerici cu toate că şi operanzii dată calendaristică sunt reprezentaţi intern sub forma unor numere. Operatorul "==" este folosit pentru a determina dacă două şiruri sunt identice, adică au aceeaşi lungime şi aceleaşi caractere. Operatorul "=" compară două şiruri de caractere şi le consideră egale dacă fiecare caracter din şirul din dreapta este egal cu fiecare caracter din şirul din stânga. Pot apare cazurile:

```
A='alfa'
B='alfabet'
? A=B
.F.
? B=A
.T.
```

Comparaţia se face pe lungimea şirului mai scurt. Pentru a înlătura acest lucru când se foloseşte operatorul relaţional "=", este necesară folosirea comenzii SET EXACT ON, în acest caz caracterele trebuie să corespundă poziţie cu poziţie, în timp ce poziţiile libere de la sfârşitul fiecărui şir sunt ignorate.

Operatorul $ se aplică doar şirurilor de caractere şi returnează rezultatul .T. dacă şirul din stânga este conţinut în şirul din dreapta.

Exemplul 3.3

```
A= 'Universitatea Lucian Blaga'
A ? 'Lucian' $ A
  .T.
B ?'universitatea' $ A
  .F.
```

Explicaţii

A) Se caută şirul Lucian în şirul A, şirul este găsit rezultatul este adevărat;

B) Ultima comparaţie este falsă deoarece la compararea şirurilor de caractere se ţine cont de tipul literelor mari sau mici.

Operatorii logici care se folosesc în limbajul Visual Fox sunt AND, OR şi NOT. Operatorii relaţionali şi operatorii logici AND şi OR sunt operatori binari pentru că sunt folosiţi între doi operanzi. Operatorul NOT este un operator unar deoarece are nevoie de un singur operand.

Tabelul de adevăr al operatorului AND este următorul:

AND	**.T.**	**.F.**
.T.	**.T.**	**.F.**
.F.	**.F.**	**.F.**

Tabelul de adevăr al operatorului OR este următorul:

OR	**.T.**	**.F.**
.T.	**.T.**	**.T.**
.F.	**.T.**	**.F.**

Tabelul de adevăr al operatorului NOT este următorul:

NOT	
.T.	**.F.**
.F.	**.T.**

Pentru operatorul NOT se poate folosi un sinonim şi anume caracterul “!”. De exemplu !A este echivalent cu .NOT. A

Operatorii numerici sau operatorii aritmetici ai limbajului Visual Fox sunt prezentaţi în tabelul 3.3.

Tabelul 3.3

Operatorul	**Descrierea**
- unar	Generează negativul unui număr
+ unar	Pozitivul unui număr
+	Adunare
-	Scădere
*****	Înmulţire
/	Împărţire
**** sau ^**	Ridicare la putere
%	Modulo întoarce restul care rezultă la împărţirea a două expresii numerice

Exemplul 3.4

```
set talk off
```

În tabelul 3.4 sunt prezentate regulile de precedență ale operatorilor:

```
A=134
?-A
-134
?+A
-134
B=200
?A+B
66
?43.3 % 22
1.3
```

Tabelul 3.4

Nivelul de precedență	Operatorul
1	Parantezele
2	Operatorii de semn + şi -
3	Ridicarea la putere
4	Modulo
5	Înmulțirea şi împărțirea
6	Adunarea scăderea şi concatenarea şirurilor
7	Operatorii relaționali
8	Operatorul .NOT.
9	Operatorii logici binare .AND. şi .OR.
Cel mai scăzut	

3.1.4 Operații cu variabile de memorie

Operația de **creare** a unei variabile de memorie se poate face cu instrucțiunea :

STORE <expresie> **To <**listă de identificatori**>**

Expresia poate fi de un anumit tip permis de către limbaj, iar lista de identificatori, poate cuprinde unul sau mai multe nume de variabile, despărțite prin virgulă. A doua metodă de creare a unei variabile constă din folosirea semnului de atribuire între identificator şi expresie:

[<identificator>] = <expresie>

Exemplul 3.5

```
STORE 5 to Alfa, Beta
STORE “Universitate” to Var
Var=”Universitate”
Alfa=5
Beta=5
```

Membrul <identificator> din cadrul operaţiei de atribuire este opţional. Există cazuri în care se doreşte apelarea unei funcţii şi nu interesează salvarea valorii întoarsă de către aceasta.

Mai există şi alte comenzi Visual Fox care crează variabile de memorie:

- ACCEPT şi INPUT permit introducerea valorilor unor variabile de la tastatură;
- PUBLIC şi PRIVATE declară tipul variabilelor permise într-un program;
- DECLARE şi DIMENSION definesc variabile de tip matrice;
- PARAMETERS defineşte variabilele dintr-o procedură;
- SUM, AVERAGE, COUNT, TOTAL definesc o anumită variabilă ca rezultat al unor operaţii matematice.

Ştergerea variabilelor din memorie se realizeaă cu comanda:

RELEASE ALL [LIKE|EXCEPT <nume generic>]| <listă identificatori>

Această comandă asigură înlăturarea fie a unor anumite variabile sau a tuturor variabilelor din memorie. Clauza ALL elimină toate variabilele de memorie, iar clauzele LIKE şi EXCEPT determină anumite variabile de memorie care vor fi eliminate.

Nume generic este un nume de identificator care poate conţine caracterele de înlocuire:

- "?" corespunde unui singur caracter;
- "*" corespunde oricărei secvenţe de caractere.

Astfel de exempl A? corspunde identificatorilor formaţi din două caractere, la care primul caracter este litera "A", iar A*, corespunde tuturor şirurilor de caractere care au prima literă litera "A". Variabilele de memorie pot fi înlăturate din memorie şi cu comenzile CLEAR MEMORY şi CLOSE ALL.

Vizualizarea variabilelor de memorie este realizată cu comenzile:

- DISPLAY MEMORY;
- LIST MEMORY.

Vizualizarea conţinutului unei variabile se poate face folosind comenzile de intrare/ieşire @..SAY,GET, precum şi comenzile speciale de editare "?","\" sau"\\"

În continuare se vor prezenta variabilele, constantele şi funcţiile specifice diferitelor tipuri de date ale limbajului de programare Visual Fox.

3.2 Tipuri de date în limbajul Visual Fox

Tipurile de date admise de către limbajul Visual Fox sunt următoarele:

- **caracter (character);**
- **unităţi monetare (currency);**
- **dată calendaristică (Date, Date Time);**
- **numeric (Numeric, Float, Double, Integer);**
- **logic (Logical);**

- **memo;**
- **general;**
- **variant.**

Se observă că variabilele de memorie au aceleaşi tipuri de date ca şi câmpurile din fişierele de date DBF.

3.2.1 Tipul şir de caractere

Un şir de caractere reprezintă o mulțime ordonată de caractere , în care fiecare termen al mulțimii are o poziție bine determinată , putându-se asocia un număr care reprezintă poziția caracterului în cadrul şirului. Numărul caracterelor dintr-un şir reprezintă lungimea şirului. O porțiune dintr-un şir se numeşte subşir. O constantă şir de caractere denumită şi şir de caractere, se reprezintă prin mulțimea caracterelor care o compun încadrate între apostroafe simple ' ' sau apostroafe duble " ". De asemenea o constantă de tip şir de caractere poate fi delimitată de caracterele “[“ şi “]”. Nu sunt permise folosirea apostroafelor de tipuri diferite pentru acelaşi şir de caractere. De exemplu:

- 'studenti ' "studenți" corect
- 'studenți" "studenți' incorect

Dacă se doreşte înscrierea unui apostrof într-un şir de caractere atunci şirul de caractere va fi delimitat de apostroafe de celălalt tip: "10' (minute)". Şirul nul se specifică prin două apostroafe alăturate "" sau". Caracterele desemnate prin 0 binar şi 26(Ctrl+Z) nu trebuie utilizare, deoarece în anumite cazuri pot să determine pierderea datelor. Constantele caracter pot fi formate, ca şi variabilele de tip caracter din până la 64K de caractere admise de sistemul Visual Fox.

Operanzii care intră în componența şirurilor de caractr sunt :

- câmpuri a unor fişiere de date de tip caracter sau şir de caractere;
- funcții ce returnează şiruri de caractere;
- variabile de tip şiruri de caracterere;
- constante de tip şir de caractere.

Operatori care se aplică asupra şirurilor de caractere sunt:

- de concatenare simplu “+” sau special “-“;
- de comparare sau relaționali

Operatorul simplu de concatenare face ca din două şiruri de caractere să se obțină un al treilea prin alipirea celui de al doilea şir la coada primului şir. De exemplu:

'UNIVERSITATEA '+'LUCIAN BLAGA' ⇒ 'UNIVERSITATEA LUCIAN BLAGA

Operatorul special '-' este asemănător cu cel simplu cu deosebirea că spațiile de la sfârşitul primului şir sunt trecute la sfârşitul şirului al doilea.

'FAC '-' DE ING' ⇒ 'FAC DE ING '

Evaluarea şirurilor de caractere se realizează cu ajutorul operatorilor relaționali. Dacă este îndeplinită condiția logică atunci rezultatul returnat este adevărat, în caz contrar rezultatul este fals.

În cazul operatorilor relaţionali algoritmul de comparare a două şiruri de caractere este următorul: se testează primele caractere din fiecare şir care se compară. În funcţie de codul ASCII pe care îl au aceste două caractere rezultatul comparării poate fi adevărat (.T.) sau fals (.F.). Dacă codul ASCII a primului caracter din primul şir este mai mic decât codul ASCII al primului caracter din al doilea şir şi dacă între $expC_1$ şi $expC_2$ se găseşte operatorul relaţional "<", atunci rezultatul comparării este .T.. Dacă între cele două şiruri se află operatorul relaţional ">" atunci rezultatul comparării este .F. Dacă codurile ASCII ale acestor prime caractere din şir este identic atunci se trece la compararea celorlalte caractere din şiruri. Dacă lungimile celor două şiruri diferă atunci şirul de lungime mai mică este completat cu caracterul al cărui cod ASCII este egal cu 0 până la egalitatea lungimii celor două şiruri. Aşa cum s-a amintit la paragraful precedent compararea şirurilor de caractere cu operatoru "=" este controlată de către comanda SET EXACT ON|OFF.

3.2.1.1 Funcţii referitoare la codificarea caracterelor din şiruri

Funcţiile referitoare la codul ascii a caracterelor sunt prezentate în tabelul 3.6.

Tabelul 3.6

Funcţie	Semnificaţie	Exemplu program
CHR(<expN>)	funcţia returnează caracterul ASCII corespunzător codului numeric (<expN>) transmis	? **CHR** (65) A *rezultatul funcţiei* ? **CHR** (49) 1 *rezultatul funcţiei*
ASC(<expC>)	funcţia returnează codul ASCII al caracterului dorit (<expC>)	?ASC('A') 65 *rezultatul funcţiei* ? ASC(' AMIC') 65 *rezultatul arată că în cazul în care şirul conţine mai multe caractere atunci funcţia returnează codul numeric al primului caracter*

3.2.1.2 Funcţii referitoare la subşirurile de caractere

Limbajul Visual Fox are implementate mai multe funcţii care faciliteză lucrul cu părţile componente ale unui şir de numite subşiruri. În tabelul 3.7 sunt prezentate principale funcţii care acţionează asupra subşirurilor de caractere precum şi o serie de exemple program de folosire a funcţiilor. Vizualizarea conţinutului unei variabile sau al rezultatului unei funcţii se poate face cu ajutorul caracterului "?", folosit în fereastra de comandă.

Tabelul 3.7

Funcție	Semnificație	Exemplu program
SUBSTR(<expC>, <expN$_1$>, [<expN$_2$>])	Extrage un subşir de caractere dintr-un şir dat. <expN$_1$ > reprezintă poziția de unde începe subşirul de extras <expN$_2$ > este opțională şi reprezintă numărul de caractere care se extrag. Dacă lipseşte atunci şubşirul se consideră de la poziția expN$_1$ până la sfârşit.	?**SUBSTR** ('FACULTATE', 1,3) FAC *rezultatul funcției* ? **SUBSTR** (' UNIV LUCIAN BLAGA' ,7) LUCIAN BLAGA *rezultatul funcției*
LEFT(<expC>, <expN>)	Extrage din şirul <expC>, un subşir de caractere începând de la poziția <expN>, poziționat la stânga.	?**LEFT** (‘Programare’,4) Prog *rezultatul funcției*
RIGHT(<expC>, expN>)	Extrage din şirul <expC>, un subşir de caractere începând de la poziția <expN>, poziționat la dreapta.	?**RIGHT**(‘Programare’,4) mare *rezultatul funcției*
REPLICATE(<expC>, <expN>)	Returnează un şir de caractere obținut prin repetarea şirului <expC> de <expN> ori.	? **REPLICATE** (' FAC’,3) FACFACFAC *rezultatul funcției*
SPACE(<expN>	Returnează un număr de spații egal cu <expN>	?**REPLICATE** (‘ ‘,10) == SPACE(10) .T. *cele două funcții în această formă sunt identice*
ALLTRIM (<expC>)	Elimină spațiile de la începutul şi sfârşitul şirului de caractere <expC>	?**ALLTRIM** (' SIBIU ') SIBIU *rezultatul funcției*
LTRIM (<expC>)	Elimină spațiile de la începutul şirului	? ‘Anul I ‘+ **LTRIM** (' IE ') Anul I IE *rezultatul funcției*
RTRIM (<expC>)	Elimină blancurile de la sfârşitul şirului de caractere	?**RTRIM** (' SIBIU ')+’EST’ SIBIUEST *rezultatul funcției*
PADC (<expR>, <expN>, [expC])	Adaugă la stânga şi la dreapta expresiei <expR>, care poate fi o variabilă de tip şir de caractere, numeric sau dată calanderistică, şirul de caractere expC sau în lipsa acestuia blank, până se obține un şir de lungime <expN>	?**PADC** (' ING1', 10,'-') ---ING1--- *(în total 10 caractere)*
PADL(<expR>, <expN>, [<expC>]	Adaugă la stânga expresiei <expR>, care poate fi o variabilă de tip şir de caractere, numeric sau dată calanderistică, şirul de caractere expC sau în lipsa acestuia blank, până se obține un şir de lungime <expN>	?**PADL**(‘Anul I’,15,’*’) *********Anul I *(în total 15 caractere)*
PADR(<expR>, <expN>, [<expC>]	Adaugă la dreapta expresiei <expR>, care poate fi o variabilă de tip şir de caractere, numeric sau dată calanderistică, şirul de caractere	?**PADR**(‘Anul I’,15,’*’) Anul I********* *(în total 15 caractere)*

	expC sau în lipsa acestuia blank, până se obţine un şir de lungime <expN>	
STRTRAN (<expC$_1$>, <expC$_2$>,[<expC$_3$>],[<expN$_1$>],[<expN$_2$>])	Înlocuieşte un subşir al unui şir dat, cu un alt şir de caractere. expC$_1$-şirul de bază în care se fac înlocuirile expC$_2$-subşir al şirului de bază, care va fi înlocuit cu şirul <exp C$_3$>. Dacă şirul <exp C$_3$> lipseşte se consideră şirul nul, obţinându-se astfel efectul de ştergere a unui subşir dintr-un şir dat. expN$_1$ arată la a câta apariţie în şirul expC$_1$ a lui expC$_2$ se face înlocuirea expN$_2$ -arată numărul de înlocuiri care se fac.Dacă lipseşte se înlocuiesc toate subşirurile găsite până la sfârşit.	?**STRTRAN** ('BLABLABLA', 'BLA', BOOM', 2, 1) BLABOOMBLA *rezultatul funcţiei* ?STRTRAN'AAABBBCCCC','B',' ', 2,2) AAAB CCCC *rezultatul funcţiei*
STUFF(<expC$_1$>, <expN$_1$>, <expN$_2$>, <expC$_2$>)	Înlocuieşte într-un şir de caractere <expC$_1$>, un subşir <expC$_2$>, al acestuia, de la poziţia<expN$_1$>, un număr de <expN$_2$> caractere din şirul de bază.	?**STUFF**('aaaaaaaaa',3,0,'bb') aabbaaaaaaa *rezultatul funcţiei* ?**STUFF**('aaaaaaaaa',3,6,'bb') aabba *rezultatul funcţiei*
CHRTRAN(<expC$_1$>, <expC$_2$>, <expC$_3$>)	Transformă caracterele din <expC$_1$>, folosind şirurile din <expC$_2$> şi <expC$_3$ ca tabel de transformare. Caracterele din <expC$_1$> care se potrivesc cu câte un caracter din <expC$_2$> sunt înlocuite cu caracterele corespunzătoare din <expC$_3$>	?**CHRTRAN** ('bomba','ba,'pe') pompe *rezultatul funcţiei* ?**CHRTRAN** ('CAPAT', 'CT', ' ') APA *rezultatul funcţiei* ?**CHRTRAN** ('COMPUSE', 'MPSE', 'NFZ') CONFUZ *rezultatul funcţiei*

3.2.1.3 Funcţii care returnează informaţii despre şir

De multe ori în programe este necesar să se cunoască informaţii despre şirurile de caractere cu care se lucrează. Funcţiile care returnează informaţii despre şiruri în limbajul Visual Fox sunt prezentate în tabelul 3.8.

Tabelul 3.8

Funcţia	Semnificaţie	Exemplu program
LEN(<expC>)	Returnează lungimea unui şir de caractere <expC>	?**LEN**('aaaaaaaaa') 9 *rezultatul funcţiei*
ISALFA(<expC>)	Returnează adevărat dacă şirul de caractere <expC> începe cu un caracter alfabetic	?**ISALFA**('Sibiu') .T. *rezultatul funcţiei*
ISDIGIT(<expC>)	Returnează adevărat dacă şirul de caractere <expC> începe cu o cifră	?**ISDIGIT**('1234567') .T. *rezultatul funcţiei*
ISLOWER(<expC>)	Returnează adevărat dacă şirul de caractere <expC> începe cu o literă mică	?**ISLOWER**('alfabet') .T. *rezultatul funcţiei*
ISUPPER(<expC>)	Returnează adevărat dacă şirul de caractere	?**ISUPPER**('Alfa')

	<expC> începe cu o majusculă	.T. *rezultatul funcției*
AT(<expC$_1$>, <expC$_2$>, [<expN])	Caută de la începutul şirului <expC$_2$> subşirul <expC$_1$> şi returnează poziția subşirului în cadrul şirului. Funcția AT() este sensibilă la tipul caracterelor, litere mici sau majuscule. <expN> este opțional şi specifică de câte ori apare subşirul în şirul căutat.	?**AT**('SIBIU', 'Universitatea "Lucian Blaga" din SIBIU') 34 *rezultatul funcției*
ATC(<expC$_1$>, <expC$_2$>, [<expN])	Caută de la începutul şirului <expC$_2$> subşirul <expC$_1$> şi returnează poziția subşirului în cadrul şirului. Funcția ATC() este insensibilă la tipul caracterelor, litere mici sau majuscule. <expN> este opțional şi specifică de câte ori apare subşirul în şirul căutat.	?**AT**('sibiu', 'Universitatea "Lucian Blaga" din SIBIU') 34 *rezultatul funcției*
RAT(<expC$_1$>, <expC$_2$>, [<expN])	Caută de la sfârşitul şirului <expC$_2$> spre începutul şirului, subşirul <expC$_1$> şi returnează poziția subşirului în cadrul şirului. Funcția RAT() este sensibilă la tipul caracterelor, litere mici sau majuscule. <expN> este opțional şi specifică de câte ori apare subşirul în şirul căutat.	?**RAT**('SIBIU','Universitatea"Lucian Blaga" din SIBIU') 34 *rezultatul funcției*
OCCURS(<expC$_1$>, <expC$_2$>)	Caută şirul <expC$_1$> în şirul <expC$_2$>, returnând numărul de apariții al şirului.	?**OCCURS**('a','abracadabra') 5 *rezultatul funcției*

Un şir de caractere se poate întinde pe mai multe linii, trecerea la linie nouă fiind realizată de prezența în şir a combinației de caractere CHR(13)CHR(10)

3.2.1.4 Funcții care transformă şirurile de caractere

Diferențierea între caracterele alfabetice mici şi mari a dus la necesitatea transformări caracterelor mici în caractere mari şi invers. Funcțiile care realizează acest lucru sunt prezentate în tabelul 3.9

Tabelul 3.9

Funcție	Semnificație	Exemplu program
LOWER(<expC>)	Transformă toate majusculele în litere mici, restul caracterelor rămânând neschimbate	?**LOWER**(AAAbbCCC') aaabbccc
UPPER(<expC>)	Transformă toate caracterele mici în majusculele corespunzătoare, restul caracterelor din şir rămânând neschimbate.	?**UPPER**('universitate' UNIVERSITATE
PROPER(<expC>)	Transformă primul caracter dintr-un şir în majusculă dacă este alfabetic, restul fiind caractere mici	?**PROPER**('universitate') Universitate

3.2.1.5 Funcţii speciale pentru şirurile de caractere

Tabelul 3.10 prezintă aceste funcţii care se referă la şirurile de caractere şi care au un caracter mai special.

Tabelul 3.10

Funcţia	Semnificaţia	Exemplu program
LIKE(<nume generic>, <expC>	Compară şirul de caractere <expC>, cu numele generic şi întoarce .T. dacă există potrivire.	?**LIKE**('calc*','calculator') .T.
CHRSAW([expN])	Întoarce rezultatul adevărat(.T.) în cazul în care un caracter este prezent în bufferul de tastatură. Expresia numerică reprezintă timpul de răspuns, în secunde.	?**CHRSAW**() .F.
SOUNDEX(<expC>)	Returnează o reprezentare fonetică(cod Soundex) a unui şir de caractere	?**SOUNDEX**('FFFF') F000

3.2.2 Tipul dată calendaristică

Expresiile de tip dată calanderistică pot conţine :

- câmpuri de acest tip ale unui fişier de date;
- funcţii ce returnează valori tip dată calanderistică;
- variabile tip dată calanderistică;
- constante de tip dată calendaristică.

Ordinea de specificare azilei, lunii şi anului, modul de scriere al anului (cu două sau patru cifre) cât şi separatorul dintre cele trei omponente ale datei sunt determinate de comenzi şi funcţii specifice limbajului Visual Fox.

Memorarea datei calendaristice într-o variabilă de memorie se face cu comanda

STORE {11/18/95}TO DATA_C

Principalele funcţii care se folosesc pentru datele calendaristice sunt prezentate în tabelul 3.11

Tabelul 3.11

Funcţia	Semnificaţia	Exemplu program
DATE ()	Returnează data curentă a sistemului	?**DATE** ( ) 12/07/97
SET MARK TO <expC>	Determină schimbarea delimitatorului pentru grupul de caractere zz/ll/aa. ExpC trebuie să reprezinte un singur caracter	**SET MARK TO '-'**
DOW (<expD>)	Returnează numărul zilei din săptămâna. De exemplu la ziua de duminică îi corespunde cifra 1.	?**DOW** ({10-22-95}) 1

CDOW (<expD>)	Returnează numele zilei din săptămână în limba engleză	?**CDOW** ({10-22-95}) Sunday
MONTH (<expD>)	Returnează numărul lunii din an.	?**MONTH** ({10-22-95}) 10
CMONTH (<expD>)	Returnează denumirea lunii calendaristice din an în limba engleză.	?**CMONTH** ({10-22-95}) october
DAY (<expD>)	Returnează valoarea numerică a zilei în cadrul lunii.	?**DAY**({12/08/97}) 8
YEAR <expD>)	Extrage anul dintr-o constantă de tip dată calendaristică	?**YEAR** (date()) 1997
GOMONTH (<expD>,± <expN>)	Întoarce data calendaristică care este cu <expN> luni înainte sau după o anumită dată <expD>.	?**GOMONTH**(DATE(), 3) 03/08/98 - data cu 3 luni înainte de la data curentă ?**GOMONTH** {11/18/95},5) 04/18/96 ?**GOMONTH**(DATE(),- 15) - data cu 15 luni în urmă, de la data curentă
DMY(<expD>)	Transformă data în forma zi Luna anul.	?**DMY**({12/08/97}) 08 December 97
DTOC <expD>)	Transformă data calanderistică într-un şir de caractere.	? **DTOC** (DATE()) 11/18/1995 - şir de caractere
DTOS(<expD>	Transformă data într-un şir de caractere în forma SSAALLZZ (secol, an, lună, zi)	?**DTOS({**10/04/97}) 19971004
SECONDS()	Returnează numărul de secunde trecute de la miezul nopţii.	?**SECONDS**() 33567.750
TIME()	Returnează timpul curent în forma hh:mm:ss	?**TIME**() 09:24:34
CTOD(<expC>)	Converteşte un şir de caractere într-o dată calendaristică.	?**CTOD**('01/01/98') 01/01/98

3.2.3 Tipul memo

Tipul de dată **memo** este specific fişierelor de date, în cazul în care se doreşte memorarea unui număr mare de caractere. Câmpurile memo sunt câmpuri care permit memorarea unui volum variabil de informaţie sau variabile de memorie care preiau valoarea câmpurilor memo din fişierele de date. Un fişier de date care are cel puţin un câmp memo are asociat un fişier suplimentar în care sunt depuse informaţiile conţinute în acest câmp. Între fiecare înregistrare a unui câmp memo (în cazul nostru câmpul Adresa) şi înregistrările fişierului asociat, fişier care are acelaşi nume ca şi fişierul de date dar are extensia **FPT**, se stabileşte o relaţie univocă (Fig. 3.1). În câmpul memo din fişierul de date, se păstrează date referitoare la adresa corespunzătoare din fişierul FPT. Acest mecanism este invizibil pentru utilizator. Accesul la un câmp memo se face astfel:

- se selectează înregistrarea dorită din fişierul de date;
- implicit şi independent de utilizator se face legătura cu fişierul a cărei extensie este FPT, adică cu fişierul memo;
- se citeşte din câmpul fişierul memo de la poziţia determinată anterior, informaţia corespunzătoare câmpului memo.

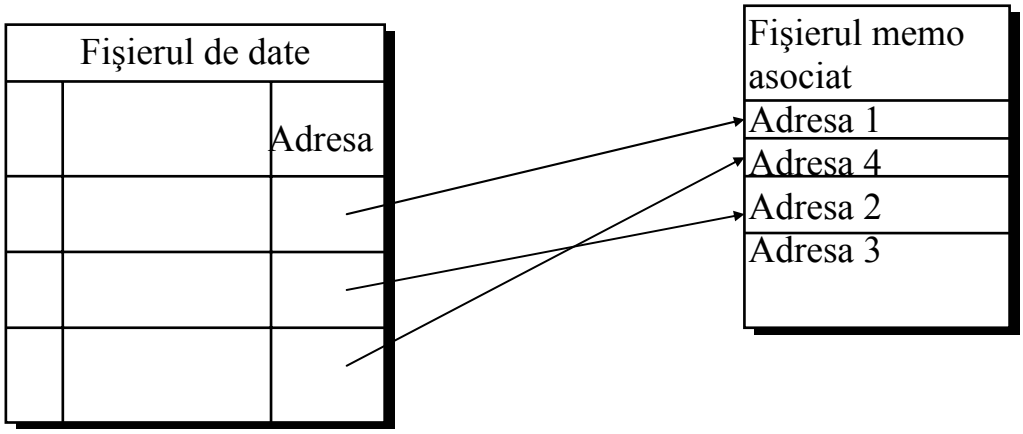


Fig 3.1 Legătura dintre fişierul de date şi fişierul memo ataşat (FPT)

Editarea conţinutului unui câmp memo se poate face din fereastra de editare a unei comenzi pentru modificarea conţinutului unui fişier de date (Browse, Change, Edit, Append, etc). În acest caz ne poziţionăm pe câmpul memo dorit şi-l deschidem prin tastarea combinaţiei de taste Ctrl+PgDn sau dublu click de la mouse. Se introduce conţinutul câmpului în fereastra de editare care apare. După terminare se apasă combinaţia de taste Ctrl+W dacă se doreşte salvarea conţinutului introdus, sau tasta Esc, dacă nu se doreşte salvarea conţinutului introdus.

Considerăm spre exemplu fişierul de date Discipline.dbf în care avem un câmp memo corespunzător bibliografiei unei discipline. Comanda Browse din figura 3.2 prezintă câmpurile fişierului de date, Discipline.dbf. Dacă se doreşte modificarea conţinutului câmpului memo Bibliografie, se poziţionează spotul pe respectivul câmp şi se acţionează combinaţia de taste Ctrl+PgDn. Ca rezultat va apare fereastra de editare a câmpului, fig 3.2.

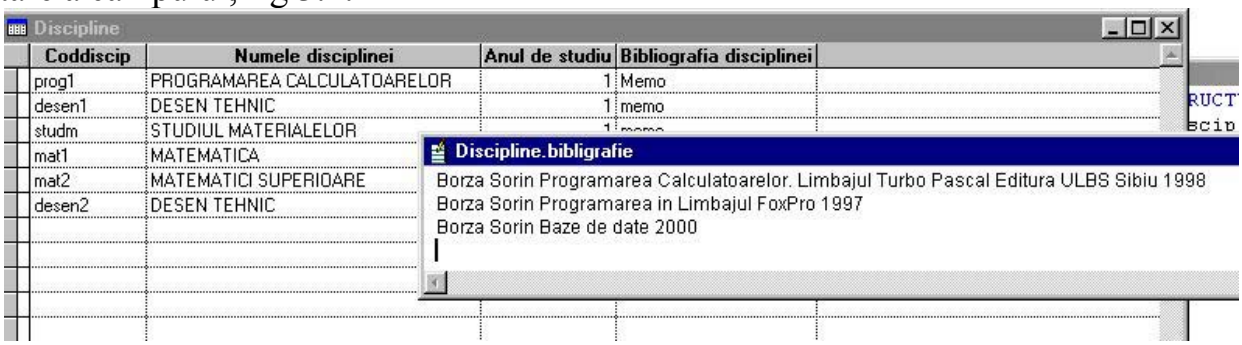


Fig. 3 2 Editarea câmpului Memo

Dacă se doreşte direct modificarea unui câmp memo atunci se poate executa comanda:

MODIFY MEMO <câmp memo 1 >[, **<**câmp memo 2**>…]**
[NOEDIT]
[NOWAIT]
[RANGE <$expN_1$>,$expN_2$**>]**
[WINDOW] <nume fereastră1**>**
[IN [WINDOW]<nume fereastră2**> | SCREEN]]**
[SAVE].

Opțiunile comenzii sunt prezentate în tabelul 3.11.

Tabelul 3.11

Opțiune	Semnificație
<câmp memo 1>, <câmp memo 2>…	Reprezintă câmpurile memo pentru care se vor deschide ferestrele de editare
NOEDIT	Nu permite modificarea câmpului memo.
NOWAIT	Se foloseşte numai în interiorul unui program, acesta continuîndu-şi execuția după deschiderea ferestrei de editare fără a mai aştepta ca utilizatorul să modifice câmpul memo respectiv.
RANGE <$expN_1$>, <$expN_2$>	Se foloseşte pentru a edita doar porțiuni dintr-un câmp memo, începând de la caracterul <$expN_1$> până la caracterul <$expN_2$>
IN SCREEN	Specifică faptul că fereastra de editare se deschide în ecranul Visual Fox.
WINDOW şi IN WINDOW	Definesc faptul că fereastra de editare va fi deschisă într-o fereastră definită de utilizator.
SAVE	Se foloseşte din interiorul unui program, pentru a păstra fereastra pe ecran şi după ce se iese din editare.

Exemplul 3.6
Explicații

USE DISCIPLINE
A **MODIFY** MEMO bibliografia **RANGE** 1,20
B **MODIFY MEMO** bibliografia **NOEDIT**

A) Modificarea câmpului memo bibliografia, al primei înregistrări, numai primele 20 de caractere;
B) Afişarea câmpului memo bibliografia al primei înregistrări fără modificarea sa.

Într-un câmp memo poate fi încărcat conținutul unui fişier. În acest caz se foloseşte comanda:

APPEND MEMO <câmp memo> **FROM** <fişier> **OWERWRITE**

Semnificația opțiunilor comenzii este prezentată în tabelul 3.12.

Tabelul 3.12

Opțiunea	Semnificația
<câmp memo>	Numele câmpului în care se va scrie conținutul fişierului.
<fişier>	Numele fişierului al cărui conținut este copiat în câmpul memo <câmp memo>
OVERWRITE	Conținutul fişierului se scrie peste conținutul câmpului memo

Pentru a copia conținutul unui câmp memo într-un fişier se foloseşte comanda:

COPY MEMO <câmp memo> **TO** <fişier > **[ADDITIVE]**

Parametrii comenzii sunt prezentaţi în tabelul 3.13

Tabelul 3.13

Opţiunea	Semnificaţia
<câmp memo>	Numele câmpului care se va copia în fişier.
<fişier>	Numele fişierului în care va fi copiat câmpul. În cazul în care extensia fişierului nu este specificată, Visual Fox va atribui automat extensia .TXT
ADDITIVE	Conţinutul câmpului se va scrie la sfârşitul fişierului.

Un câmp memo dintr-o anumită înregistrare se poate înlocui cu un alt câmp memo..

Exemplul 3.7

Să se scrie un program prin care să se schimbe conţinutul câmpului memo Capitol1 cu conţinutul câmpului memo Capitol2 pentru aceeaşi înregistrare. Cele două câmpuri memo există într-o tabelă cu numele Carte

```
SET TALK OFF
A  SELECT 1
   USE CARTE
B  Var=capitol1
   SELECT 2
C  USE carte AGAIN
   REPLACE capitol1 WITH b.capitol3
D  REPLACE capitol3 WITH var
```

A) Fişierul de date se deschide în zona 1 de lucru;
B) Variabil denumită Var devine variabilă de memorie de tip memo. În versiunea standard FOX ea poate ocupa maximum 64 K de memorie iar în versiunea extinsă poate ocupa toată memoria disponibilă .Ea este tratată ca un şir de caractere asupra ei putându-se aplica toate fiuncţiile specifice tipului de şir de caractere;
C) Fişierul de date Carte.dbf se deschide şi în zona 2 de lucru;
D) Se schimbă câmpurile între ele cu ajutorul variabilei Var.

3.2.3.1 Funcţiile tipului memo

Asupra câmpurilor sau variabilelor de tip memo se pot aplica mai multe funcţii care sunt prezentate în tabelul 3.14. Exemplu se referă la câmpurile memo ale fişierului de date Carte.dbf (fig 6.1 şi fig 6.2).

Tabelul 3.14

Funcţia Memo	Semnificaţia funcţiei	Exemplu program
SET MEMOWIDTH TO <expN>	Modifică lăţimea liniei de afişare a unui câmp memo. Lăţimea prestabilită este 50, valoarea minimă fiind 8, iar valoarea maximă 254. Valoarea lui <expN> poate depăşi 256, dar lăţimea de afişare a câmpului memo pe ecran	**Set memowidth** to 80

	nu va depăşi această valoare.	
MEMLINES (<câmp memo>	Returnează numărul de rânduri ale unui câmp memo	?**memlines**(capitol1) 4
MLINE (<câmp memo>, <expN1>[,<expN2>	Funcţia extrage linia <expN$_1$> dintr-un câmp memo. Dacă nu se doreşte extragerea liniei în totalitate atunci poate fi considerat cămpul memo ca începând de la al (<expN$_2$>+1) lea caracter	?**mline**(capitol1,1) Capitolul 1 ?**mline**(capitol1,1,10) 1
AT (<expC>, <câmp memo>, [<expN>])	Returnează poziţia unui şir de caractere într-un câmp memo. Dacă <expN> este precizat funcţia returnează poziţia celei de a <expN> apariţii a şirului <expC> în câmpul memo	?**at** (‘1’,capitol1) 11 ?**at**(‘ca’,capitol1,1) 1 ?**at**(‘ca’,capitol1,2) 22
ATC (<expC>, <câmp memo> [,<expN>])	Funcţia este identică cu funcţia AT,ignorând deosebirile dintre literele mari şi mici	?**atc**(‘CA’,capitol1,3) 61
ATCLINE (<expC>, <câmp memo>	Returnează numărul de linie al şirului <expC> în câmpul memo <câmp memo>, ignorând deosebirile dintre literele mari şi mici	?**atcline**(‘cAMp’,capitol1) 3
ATLINE (<expC>, <câmp memo>	Returnează numărul de linie al şirului <expC> în câmpul memo <câmp memo>.	?**atcline**(‘camp’,capitol1) 3
LEFT (<câmp memo>, <expN>)	Returnează numărul de caractere specificat prin <expN>, din partea stângă a câmpului memo	?**left**(capitol1,11) Capitolul 1
LEN (<câmp memo>	Returnează lungimea câmpului memo.	?**len**(capitol1) 93
LTRIM (câmp memo)	Returnează caracterele câmpului memo cu spaţiile libere de la început eliminate.	?**ltrim**(capitol1) Capitolul 1 Aceasta carte va prezenta modul de utilizare a campurilor MEMO intr-un fisier ‣
RIGHT (<câmp memo>, <expN>)	Returnează numărul de caractere specificate de la dreapta câmpului memo.	?**right** (capitol1,15) tr-un fisier ‣
RTRIM (<câmp memo> şi **TRIM**	Returnează câmpul memo cu spaţiile de la sfârşit eliminate.	?**rtrim**(capitol1) Capitolul 1 Aceasta carte va prezenta modul de utilizare a campurilor MEMO intr-un fisier ‣
SUBSTR (<câmp memo> ,<expN$_1$>, [expN$_2$])	Extrage un subşir dintr-un câmp memo, de la o anumită poziţie, un anumit număr de caractere.	?**substr**(capitol1,12,10) Aceasta ?**substr**(capitol1,85) fisier ‣

3.2.4 Tipul numeric

Cu toate că este un limbaj orientat spre fişiere de date, şi nu spre calcule matematice ca de exemplu limbajele Pascal sau Fortran, tipul numeric este implementat astfel încât să realizeze majoritatea operaţiilor matematice întâlnite în practică. De asemenea sunt prevăzute o serie de funcţii matematice care calculează funcţiile matematice elementare cum ar fi exponenţiala, logaritmul, funcţiile trigonometrice, etc.

Operanzii numerici care intervin în expresii pot fi :

- câmpuri numerice ale unui fişier de date;
- funcţii care returnează valori numerice;
- variabile de tip numeric;
- constante numerice.

Numărul de cifre zecimale cu care este afişat implicit un număr este 2. Numărul de zecimale care se afişază poate fi modificat de către utilizator cu comanda:

SET DECIMAL TO <expN>

<expN> reprezintă numărul de zecimale care se afişază. Această comandă controlează doar aspectul extern al numărului nu şi modul în care lucrează intern Visual Fox.

În limbajul Visual Fox sunt prevăzute mai multe tipuri de date numerice: numeric (N), Currency, Integer, double aşa cum a fost prezentat în detaliu în capitolul 4. Documentaţia Visual Fox sugerează memorarea datelor cu specific comercial în câmpuri de tip numeric (N) sau Currency şi a celor ştiinţifice în câmpuri de tip Double, Float (F), etc. Conversia din tipul numeric în tipul flotant se face cu funcţia FLOAT, iar din tipul flotant în tipul numeric prin funcţia FIXED.

3.2.4.1 Funcţii numerice şi matematice

Principalele funcţii metematice sunt prezentate în tabelul 3.13.

Tabelul 3.15

Funcţia	Semnificaţia	Exemplu program
ABS(<expN>)	Returnează o valoare numerică egală cu valoarea absolută a argumentului	?**ABS**(-34) 34
SIGN(<expN>)	Returnează semnul argumentului	?**SIGN**(23) +1 ?**SIGN**(-23) -1 ?**SIGN**(0) 0
INT(<expN>)	Returnează partea întreagă a unui număr zecimal. Partea zecimală a unui număr se obţine scăzând din număr	?**INT**(1.234) 1

	partea întreagă	?**INT**(-43.678) -45 ?3.45-**INT**(3.45) 0.45
CEILING (<expN>)	Returnează cel mai mic întreg mai mare sau egal cu argumentul <expN>.	?**CEILING**(9.56) 10 ?**CEILING**(-9.56) -9
FLOOR (<expN>)	Returnează cel mai mare întreg mai mic sau egal cu argumentul <expN>	?**FLOOR**(6.78) 6 ?FL00R(-9.45) -10
ROUND(<$expN_1$>, <$expN_2$>)	Rotunjeşte primul argument <$expN_1$> , funcție de al doilea argument <$expN_2$>, care defineşte numărul de zecimale care se vor păstra în valoarea returnată.	?**ROUND**(9.567,3) 9.567 ?**ROUND**(9.567,2) 9.57 ?ROUND(9.567,0) 10
EXP(<expN>	Returnează funcția exponențială cu baza E	?**EXP**(3.4) 29.96
LOG(<expN>)	Returnează logaritmul natural al argumentului	?**LOG**(3.6) 1.72
LOG10(<expN>)	Returnează logaritmul zecimal al argumentului	?**LOG10**(3.6) 0.75
SQRT(<expN>)	Returnează rădăcina pătrată a argumentului	?**SQRT**(34) 3.83
MAX(<$expN_1$>, $expN_2$>)	Returnează maximum dintre <$expN_1$> şi <$expN_2$>	?**MAX**(43.67,78.9) 78.9
MIN(<$expN_1$>, $expN_2$>)	Returnează minimum dintre <$expN_1$> şi <$expN_2$>	?**MIN**(4.56,2.45) 2.45
RAND [(<expN>)]	Returnează un număr pseudoaleator. Dacă <expN> lipseşte, funcția întoarce un număr pseudoaleator cuprins între o şi 1.0. Inițializarea de la ceasul sistemului se realizează cu argumentul -1 (rand(-1))	?**RAND()** 0.23 ?**RAND(-1)** 0.10
FIXED(<expN>)	Transformă un număr din format IEEE număr binar lung (tipul flotant F), într-un număr în format BCD (tipul numeric N), returnând acelaşi argument <expN>	
FLOAT(<expN>)	Transformă un număr din format BCD(binary coded decimal) (tipul numeric N), într-un număr binar lung corespunzător standardului IEEE (tipul flotant F), returnând acelaşi argument <expN>	

3.2.4.2 Funcții trigonometrice

Funcţiile trigonometrice ale limbajului Visual Fox sunt reprezentate în tabelul 3.16.

Tabelul 3.16

Funcţia	Semnificaţia	Exemplu program
ACOS(<expN>)	Arccosinus. ExpN este cuprinsă între -1 şi +1, iar valoarea returnată de funcţie este în radiani fiind cuprinsă în intervalul 0 şi PI	?**ACOS**(0.3) 1.27
ASIN(<expN>)	Arcsinus. ExpN este cuprinsă între -1 şi +1, iar valoarea returnată de funcţie este în radiani, fiind cuprinsă în intervalul -PI/2 şi PI/2	?**ASIN**(0.3) 0.30
ATAN(<expN>)	Arctangentă. ExpN poate lua orice valoare, valoarea returnată de funcţie este dată în radiani fiind cuprinsă în intervalul -PI/2 şi PI/2.	?**ATAN**(0.5) 0.46
ATN2(<$expN_1$>, <$expN_2$>)	Funcţia primeşte două argumente numerice ca argumente reprezentând coordonatele punctului în plan a cărui arctangentă se calculează ($expN_1$ reprezintă abscisa iar <$expN_2$> reprezintă ordonata) şi returnează unghiul corespunzător în radiani. Raportul <$expN_1$>/<$expN_2$> trebuie să se găsească în intervalul -PI şi PI	?**ATN**2(5,4) 0.90
SIN(<expN>)	Sinus. Argumentul <expN> este dat în radiani	?**SIN**(PI()/4) 0.71
COS(<expN>)	Cosinus. Argumentul <expN> este dat în radiani	?**COS**(PI()/4) 0.71
TAN(<expN>)	Tangentă. Argumentul <expN> este dat în radiani	?**TAN**(PI()/4) 1
PI()	Returnează valoarea lui π (3.141592653589…)	**?ROUND**(PI(),10) 3.1415926536
DTOR(<expN>)	Transformă gradele în radiani rezultatul fiind dat în radiani.	?**DTOR**(90) 1.57
RTOD(<expN>)	Transformă radianii în grade	?**RTOD**(1.57) 89.95

3.2.5 Tipul logic

O expresie de tip logic este o combinaţie de operatori şi operanzi, realizată după anumite reguli sintactice, pentru a forma o construcţie corectă, a cărei evaluare va avea ca rezultat o valoare logică.

Operanzii ce intră în componenţa expresiilor logice sunt de următoarele tipuri:

- câmpuri de tip logic a unui fişier de date;
- funcţii ce returnează valori logice;
- variabile de tip logic;
- alte expresii de tip logic.

Operatorii logici în ordinea priorităţilor sun prezentaţi în tabelul 3.17.

Tabelul 3.17

Operator	Semnificație
()	grupează expresiile logice
!, NOT	negația logică
AND	şi logic
OR	sau logic

În general rezultatul unei expresii care conține operatori relaționali este de tip logic.

3.2.6 Funcții pentru calcule financiare

În limbajul Visual Fox există câteva funcții care rezolvă relativ uşor unele probleme financiare. Presupunem următoarele probleme:

Exemplul 3.8

Să se calculeze suma care trebuie restituită lunar către o bancă considerând că s-a efectuat un împrumut de 10,000,000 lei, dobânda lunară practicată de bancă este de 3,6% iar împrumutul a fost acordat pe un an. Pentru rezolvare se foloseşte funcția financiară **PAYMENT** cu sintaxa:

PAYMENT(<expN1>,<expN2>,<expN3>)

unde expresiile <$expN_{1..3}$> reprezintă:

- $expN_1$ valoarea împrumutului (10,000,000);
- $expN_2$ rata dobânzii (0.036);
- $expN_3$ numărul de rate pentru achitarea împrumutului (12) .

Rezolvare:

```
?PAYMENT(10000000,0.036,12)
1040939.03
```

Exemplul 3.9

Presupunem că am deschis un cont într-o bancă care inițial este gol. Să se calculeze suma existentă în cont după un an, dacă depunem lunar 500000, dobânda lunară practicată de bancă este 3,3%. Pentru rezolvare se foloseşte funcția Visual Fox, **FV() (Future value)** cu următoarea sintaxă:

FV(<expN1>,<expN2>,<expN3>)

unde expresiile <$expN_{1..3}$> reprezintă:

- $expN_1$ suma care se depune lunar (500,000);
- $expN_2$ rata lunară a dobânzi (0.033);
- $expN_3$ numărul de depuneri după care se calculează suma din cont (12) .

Rezolvare:

```
?FV(500000,0.033,12)
```

7218172.63

Exemplul 3.10

Presupunem că avem un cont într-o bancă. Ce valoare trebuie să aibă acest cont pentru a putea scoate lunar 1,000,000 lei timp de 1 an astfel ca la sfârşitul intervalului contul să fie nul. Dobânda practicată de bancă este de 4% lunar. Pentru rezolvare se foloseşte funcţia Visual Fox, **PV() (Present Value)** cu următoarea sintaxă:

PV(<expN1>,<expN2>,<expN3>)

unde expresiile <expN$_{1..3}$> reprezintă:

- expN$_1$ suma care se scoate lunar din cont (1,000,000);
- expN$_2$ rata lunară a dobânzi (0.04);
- expN$_3$ numărul de luni în care se scot bani din contul respectiv(12) .

Rezolvare:

?PV(1000000,0.04,12)
9385073.76

3.2.7 Funcţii referitoare la toate tipurile de date

În limbajul Visual Fox există câteva funcţii care se aplică global asupra expresiilor fără a ţine cont de tipul acesteia. Aceste funcţii sunt prezentate în tabelul 3.18.

Tabelul 3.18

Funcţia	Semnificaţie	Exemplu program
TYPE(<expC>)	Testează tipul unei expresii. Rezultatul returnat de funcţie este de tip caracter cu următoarea semnificaţie: • **C** şir de caractere; • **N** numeric; • **D** dată calendaristică; • **L** logic; • **M** memo; • **U** nedefinit.	?**TYPE**(date()) D ?**TYPE**(‘1=5’) L ?**TYPE**(‘356’) N ?**TYPE**(‘”universitate”’) C ?**TYPE**(‘universitate’) U
EVALUATE (<expC>)	Evaluează rezultatul expresiei transmisă ca şir de caractere. Acesta poate fi: numeric,logic, dată calendaristică, şir de caractere, memo.	?**EVALUATE**(‘6*3’) 18
EMPTY(<expr>)	Evaluează dacă o expresie este vidă sau nu. Semnificaţia de expresie vidă are diferite înţelesuri funcţie de tipul expresiei astfel: • şir de caractere expresia conţine numai spaţii, nuluri (chr(0)), taburi(chr(10)), sfârşit de linie (chr(13), chr(10)); • numeric expresia este 0; • data calendaristică expresia este {/ /} sau {}; • logic expresia este falsă (.F.);	?**EMPTY**(‘aa’) .F. ?**EMPTY**(‘ ‘) .T. ?**EMPTY**(5-5) .T. ?**EMPTY**({}) .T.

	• memo fără conținut.	
INLIST(<expr$_1$>, <expr$_2$>, [<expr$_3$>]	Funcția testează dacă expr$_1$ se găseşte printre expresiile expr$_2$, expr$_3$,…	A='joi' ?**INLIST**(a,'miercuri', 'joi', 'vineri', 'sâmbătă') .T.
BETWEEN (<expr$_1$>,<expr$_2$>, <expr$_3$>	Funcția testează dacă o expresie se încadrează între două expresii de acelaşi tip. Expresia poate fi de orice tip permis de limbajul Visual Fox	?**BETWEEN**(date(),{01/01/97},{12/31/97} .T. ?**BETWEEN**(3,1,5) .T. ?**BETWEEN**('h','a','z') .T.
MIN(<expr$_1$>, <expr$_2$>, [<expr$_3$>…]	Returnează valoarea minimă a două sau mai multor expresii de acelaşi tip.	?**MIN**(3*4,9*3,8*2) 12 ?**MIN**('a','b','c','d') a ?**MIN(date**(),{010/01/97}) 01/01/97
MAX(<expr$_1$>, <expr$_2$>, [<expr$_3$>…]	Returnează valoarea maximă a două sau mai multor expresii de acelaşi tip.	?**MAX**(3*4,9*3,8*2) 27 ?**MAX**('a','b','c','d') d ?**MAX**(date(),{010/01/97}) 12/09/97

Concluzii

Expresiile sunt construcții sintactice fundamentale în orice limbaj de programare. Ele sunt formate din operanzi şi operatori. De multe ori într-o expresie apar funcții proprii mediului de programre. Acestea sunt de obiecei legate de un anumit tip de dată. Pe lângă funcțiile proprii mediului de programare, pot exista funcții scrise de către un anumit utilizator şi care de asemenea pot face parte dintr-o expresie. Despre acest tip de funcții se va discuta în capitolul 9 al acestei cărți.

Capitolul

4

INSTRUCȚIUNI DE SELECȚIE

Instrucțiunile de selecție sunt acele instrucțiuni care în funcție de evaluarea unei **expresii logice** execută o anumită secvență de program.

Expresia logică este aceea expresie care are ca rezultat, cuvintele logice adevărat sau fals, respectiv **TRUE** (.T.) sau **FALSE** (.F.). În cadrul unei expresii logice se folosesc operatorii relaționali şi operatorii logici.

În limbajul FoxPro există două tipuri de instrucțiuni de selecție:

- instrucțiuni de selecție cu două căi de tip:
  1) **IF**…**ELSE**…**ENDIF**
  2) **IIF**()
- instrucțiuni de selecție cu mai multe că de tip:
  DO CASE ... **ENDCASE**

4.1 Instrucțiuni de selecție cu două căi

Instrucțiunile de selecție cu două căi evaluează o expresie logică. Funcție de valoarea de adevăr a acesteia se execută o ramură sau alta de instrucțiuni. În limbajul FoxPro, instrucțiunea de selecție cu două căi este instrucțiunea **IF**…**ELSE**…**ENDIF** şi funcția **IIF**(). Din punct de vedere logic, o structură de acest tip este reprezentată în figura 4.1.

Forma generală a instrucțiunii este:

IF <expL> <instr1>
ELSE <instr2>
ENDIF

Etapele de evaluare ale instrucțiunii sunt următoarele:

1. Se evaluează expresia logică;

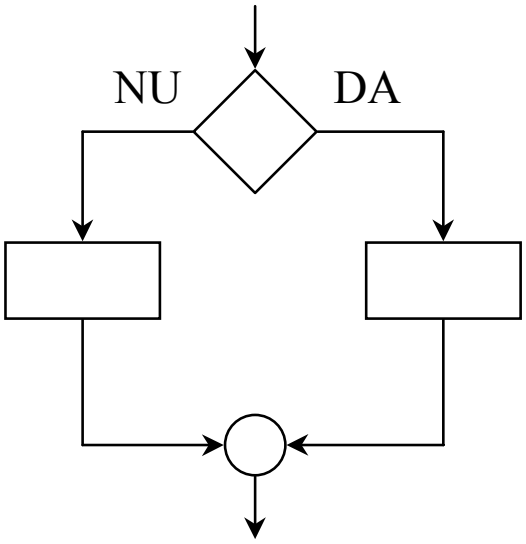


Fig. 4.1. Structura IF…THEN…ELSE

2. Dacă aceasta este adevărată se execută instrucțiunile de pe ramura DA sau .T., adică instrucțiunile <instr1>;
3. Dacă expresia logică este falsă atunci apar două cazuri:
   a) există clauza **ELSE** în comandă şi se vor executa instrucțiunile <instr2>;
   b) nu există clauza **ELSE** în comandă şi se trece la prima instrucțiune din program care urmează după instrucțiunea **IF**…**ENDIF**.

Exemplul 4.1

Să se scrie un program în limbajul FoxPro care citeşte două numere a şi b şi le compară afişând relația care există între numere.

```
SET TALK OFF
CLOSE ALL
CLEAR
@ 1,1 SAY 'CITESTE NUMARUL a'GET a PICT '999'
        DEFAULT 0
@ 1,1 SAY 'CITESTE NUMARUL b'GET b PICT '999'
        DEFAULT 0
IF a>b
    \a mai mare decât b
ELSE
    \a mai mic sau egal cu b
ENDIF
CLOSE ALL
```

A

Explicații

A) Expresia logică care se evaluează este a>b. Dacă este adevărată se execută instrucțiunea "`\a mai mare decit b`", altfel se execută instrucțiunea "`\a mai mic sau egal cu b`"

Două instrucțiuni de selecție se numesc imbricate dacă una din instrucțiuni se găseşte în clauza IF sau în clauza ELSE a celeilalte instrucțiuni

Mai multe comenzi **IF**...**ENDIF** pot fi **imbricate** obținându-se condiționarea unui grup de instrucțiuni prin mai multe expresii logice sau obținându-se selecții diverse între mai multe grupuri de instrucțiuni.

Schemele generale de imbricare ale unei instrucțiuni de tip **IF**…**ELSE**…**ENDIF** sunt:

```
IF <expL1>
     IF <expL2> <instr1>
     ELSE <instr2>
     ENDIF
ELSE <instr3>
ENDIF
```

şi

```
IF <expL1> <instr1>
ELSE
     IF <expL2> <instr2>
     ELSE <instr3>
     ENDIF
ENDIF
```

Clauza **ELSE** este opțională. Cuvântul rezervat **ENDIF** este necesar să apară întodeauna, lipsa lui într-o instrucțiune selectivă generând o eroare de sintaxă.

Exemplul 4.2

Să se scrie un program FoxPro care calculează următoarea funcție:

$$F(x) = \begin{cases} e^x \text{ pentru } x < -1 \\ 0 \text{ pentru } x \in [-1,1] \\ \ln(x) \text{ pentru } X > 1 \end{cases}$$

```
SET TALK OFF
CLEAR
CLOSE ALL
F=0
@1,1 SAY 'Se citeste numarul x' GET x;
     PICT '99.99' DEFAULT 0
```

```
READ
IF x < -1
     F=EXP(x+1)
ELSE
     IF x <= 1
          F=0
     ELSE
          F=LOG(x)
     ENDIF
ENDIF
@4,1 SAY 'Valoarea functiei F=' GET F;
     PICT '99.99999'
CLOSE ALL
```

A

B

Explicaţii

A) Se testează prima ramură a funcției şi se calculează valoarea funcției, luând în considerare valoarea de adevăr a expresiei logice x <-1;

B) Se testează cea de a doua ramură a funcției F(x) pornind din clauza **ELSE** a primei instrucțiuni **IF**.

O altă posibilitate de a ramifica pe două căi un program FoxPro este folosirea unei funcții **IIF**() cu următoarea sintaxă:

IIF(<expL>,<expr1>, <expr2>)

Etapele de lucru ale acestei funcții sunt:

1. Se evaluează expresia logică <expL>;
2. Dacă aceasta este adevărată se execută grupul de instrucțiuni <expr1>;
3. Dacă expresia logică este falsă se execută grupul de instrucțiuni <expr2>.

Cele două expresii <expr1> şi <expr2> nu trebuie să fie neapărat de acelaşi tip, ele putând fi de tip numeric, şir de caractere, dată calendaristică sau logic. La fel ca şi instrucțiunile de tip **IF**…**ELSE**…**ENDIF**, două sau mai multe funcții **IIF**() pot fi imbricate.

Exemplul 4.3

Să se scrie un program în limbajul FoxPro care citeşte două numere. Dacă primul număr este mai mare decât al doilea se tipăreşte logaritmul primului număr. În caz contrar tipăreşte suma celor două numere.

```
SET TALK OFF
CLEA
CLOSE ALL
a=0
b=0
@1,1 SAY 'se citeste a' GET a PICT '9999.99'
READ
```

A

```
@2,1 SAY 'se citeste b' GET b PICT '9999.99'
READ
SET DECIMALS TO 3
```
B
```
var=IIF(a>b,'Logaritmul natural al numarului;
    a = '+STR(log(a),10,3),;
    'Suma numerelor= '+STR((a+b),6,1))
@5,5 GET VAR
CLOSE ALL
SET DECIMALS TO
```

Explicații

A) Se citesc cele două numere a şi b cu două zecimale;

B) Se foloseşte funcția **IIF**() pentru a testa cele două numere şi pentru a afişa rezultatul.

Exemplul 4.4

Să se scrie un program FoxPro care rezolvă ecuația de gradul al doilea:

$$Ax^2 + Bx + C = 0$$

cu coeficienți întregi, folosind funcția IIF(). Algoritmul de rezolvare este următorul:

1. Dacă A = 0, B = 0, C = 0 ecuația este nedeterminată;
2. Dacă A = 0, B = 0, C # 0 ecuația este imposibilă;
3. Dacă A= 0 şi B # 0 ecuația de gradul I cu soluția X = -C/B;
4. Dacă A # 0 se calculează expresia delta = B*B - 4*A*C. Dacă delta > 0 rădăcinile ecuației sunt:

$$X1 = (-B + \sqrt{B*B - 4*A*C}) / (2*A)$$
$$X2 = (-B + \sqrt{B*B - 4*A*C}) / (2*A)$$

```
SET TALK OFF
CLOSE ALL
CLEAR
```
A
```
ACCEPT 'Citeste coeficientul A: ' TO a
INPUT  'Citeste coeficientul B: ' TO b
@3,3 SAY 'Citeste coeficientul C:' GET c PICT '99';
    DEFAULT 0
READ
```
B
```
a=VAL(a) <instr2>
```
C
```
varrez=IIF(A=0,IIF(B=0,IIF(C=0,'Ecuatie ;
    nedeterminata','Ecuatie imposibila'),;
    'Ecuatie de gradul I, X='+STR(-C/B,8,2)),;
    'Delta = '+STR((B*B-4*A*C),8,2))
```
D
```
IF a<>0 AND VAL(SUBSTR(varrez,9,8))>=0
    x1=-b+SQRT(b*b-4*a*c)/(2*a)
    x2=-b-SQRT(b*b-4*a*c)/(2*a)
    @ 4,4 SAY 'Ec de gradul 2 are radacinile:;
```

```
            '+STR(x1,6,2)
        @4,50 GET x2
    ELSE
        ?varrez
    ENDIF
    CLOSE ALL
    RETURN
```

Explicații

A) Se citesc variabilele de intrare, coeficienții A,B,C, folosind trei tipuri de instrucțiuni de intrare. Recomandabil a se folosi doar instrucțiunile @..**SAY**…**GET**;

B) Se transformă variabila şir de caractere a în variabilă numerică;

C) Varibilei varrez i se atribuie o valoare şir de caractere în funcție de evaluarea expresiilor din funcțiile **IIF** imbricate;

D) Se evaluează pasul numărul patru al algoritmului în cazul în care coeficientul A este diferit de 0. Din variabila varrez se extrage doar subşirul începând de la al noulea caracter, care se transformă în valoare numerică.

4.2 Instrucțiuni de selecție cu mai multe căi

Instrucțiunile de selecție pe mai multe căi evaluează o expresie funcție de care se execută o anumită ramură de program. Aceste instrucțiuni acceptă în general comenzile în cascadă ca şi în cazul mai multor instrucțiuni **IF**…**ELSE**…**ENDIF** sau funcții **IIF**().

În limbajul FoxPro instrucțiunea de selecție cu mai multe căi este **DO CASE**…**ENDCASE** cu următoarea sintaxă:

```
DO CASE
    CASE <expL1>
        <instructiuni1>
    [CASE <expL2>
        <instrucțiuni2>
    . . . . . .
    CASE <expLN>
        <instructiuniN>]
OTHERWISE
    <instructiuni>
ENDCASE
```

Comanda determină execuția grupului de instrucțiuni pentru care expresia logică este adevărată. Se evaluează expresia logică <expL1>. Dacă este adevărată, se execută instrucțiunile <instructiuni1> după care comanda se încheie. Dacă expresia logică <expL1> este falsă, atunci se trece la evaluarea următoarelor expresii logice şi unde se găseşte prima dată expresia logică adevărată se execută grupul de instrucțiuni

corespunzător. Dacă nici una dintre expresiile logice nu are valoarea adevărat (.T.) pot apare două cazuri:

- când nu există clauza **OTHERWISE** şi se trece la execuția primei instrucțiuni care urmează după cuvântul rezervat **ENDCASE**;
- în prezența clauzei **OTHERWISE** se va executa grupul de instrucțiuni care urmează cuvântului cheie, după care se trece la prima comandă care urmează după cuvântul rezervat **ENDCASE**.

În general instrucțiunea **DO CASE** înlocuieşte mai multe instrucțiun IF imbricate.

Exemplul 4.5

Să se citescă două numere şi se afişeze relația în care se găsesc acestea. În primul caz se vor folosi instrucțiuni **IF**, iar în al doilea caz instrucțiuni **DO CASE**.

Primul caz:

```
SET TALK OFF
CLEAR
CLOSE ALL
@ 1,1 SAY 'Citeste numarul A :' GET a DEFAULT 0
@ 2,1 SAY 'Citeste numarul B :' GET b DEFAULT 0
READ
IF a>b
     ?'Numarul A mai mare decit numarul B'
ELSE
     IF a=b
          ? 'Numarul B egal cu numarul A'
     ELSE
          ? 'Numarul A mai mic decit numarul B'
     ENDIF
ENDIF
```

Al doilea caz:

```
SET TALK OFF
CLEA
a=0
b=0
@1,1 SAY 'se citeste a' GET a
READ
@2,1 SAY 'se citeste b' GET b
READ
var1='a mai mare ca b'
var2= 'a mai mic ca b'
var3 = 'a egal cu b'
DO CASE
     CASE a>b
```

```
        ?var1
        CASE a=b
        ?var3
        CASE a<b
        ?var2
    ENDCASE
```

Instrucțiunile **DO CASE** se folosesc în cazul în care o expresie ia mai multe valori şi în funcție de fiecare valoare se execută un anumit set de instrucțiuni. Foarte frecvent se folosesc aceste instrucțiuni în cazul meniurilor.

Exemplul 4.6

Se consideră un fişier de date cu numele Salariat.dbf cu următoarele câmpuri:

- Nume C(10) numele salariatului
- Adresa C(30) adresa salariatului
- Vârsta N(2) vârsta salariatului
- Secția C(15) secția în care lucrează salariatul
- Casatorit L are valoara logica .T. dacă salariatul este căsătorit

Să se scrie un program FoxPro care simulează un meniu cu următoarele cinci opțiuni:

- Adăugare
- Modificare
- Căutare
- Sortare
- Exit

```
SET TALK OFF
USE salariati
CLEA
@ 8,16 SAY 'MENIU PRINCIPAL PENTRU FISIERUL PERSONAL'
@ 9,16 SAY '======================================='
@ 10,20,16,60 BOX REPLICATE(CHR(177),14)
a=0
IF a=0
    @11,22 SAY 'Adaugare' GET a DEFAULT 0 PICT '9';
        MESSAGE 'Pentru actualizare tasteaza 1'
    READ VALID a=1 or a=0
ENDIF
IF a=0
    @12,22 SAY 'Modificare' GET a DEFAULT 0 PICT '9';
        MESSAGE 'Pentru modificare tasteaza 2'
    READ VALID a=2 or a=0
ENDIF
IF a=0
```

```
        @13,22 SAY 'Cautare' GET a DEFAULT 0 PICT '9';
             MESSAGE 'Pentru cautare tasteaza 3'
        READ VALID a=3 or a=0
    ENDIF
    IF a=0
        @14,22 SAY 'Sortare dupa nume' GET a DEFAULT 0;
             PICT '9' MESSAGE 'Pentru sortare tasteaza 4'
        READ VALID a=4 or a=0
    ENDIF
    IF a=0
        @15,22 SAY 'Exit' GET a DEFAULT 0 PICT '9';
             MESSAGE 'Pentru iesire tasteaza 5'
        READ VALID a=5
    ENDIF
B   DO CASE
        CASE a=1
             APPEND
        CASE a=2
             BROWSE
C       CASE a=3
             CLEAR
             @ 1,1 SAY 'Tasteaza numele cautat :';
                  GET mnume DEFAULT REPL(' ',10)
             READ
             GO TOP
             LOCATE FOR mnume=nume WHILE NOT EOF()
             IF FOUND()
                  BROWSE FIELDS Nume,Virsta,Adresa;
                       FOR nume=mnume NOAPPEND NODELETE;
                       NOEDIT
             ELSE
                  WAIT WIND 'Nume negasit'
             ENDIF
             CONTINUE
        CASE a=4
             SORT ON NUME TO snume
             USE snume
             BROWSE
        CASE a=5
             CLOSE ALL
             CLEAR
             RETURN
        ENDCASE
```

Explicaţii

A) Se scriu instrucțiunile pentru realizarea meniului cu opțiunile prevăzute în textul problemei;
B) Se scrie instrucțiunea **DO CASE** pentru valorile pe care le poate lua variabila A;
C) Pentru procedura de căutare se foloseşte instrucțiunea **LOCATE** împreună cu funcția **IF FOUND**(). Dacă este îndeplinită condiția logică de căutare din instrucțiunea **LOCATE** înregistrarea este găsită şi se execută instrucțiunile de pe ramura adevărat a instrucțiunii **IF FOUND**().

Concluzii

Instrucțiunile de selecție, deservesc structura selectivă din orice limbaj de programare. Acest set de instrucțiuni, există în toate limbajele de programare, înțelegerea lor fiind obligatorie în activitatea de programare

Capitolul

5

STRUCTURI DE CONTROL. INSTRUCȚIUNI REPETITIVE

Structura repetitivă, este una dintre structurile fundamentale ale programării structurate, ea permite reluarea unui număr de instrucțiuni ale unui program de un număr determinat sau nedeterminat de ori

Se numesc instrucțiuni repetitive acele instrucțiuni care determină un grup de instrucțiuni să se execute de un anumit număr de ori într-un program.

Setul de instrucțiuni care se repetă se numeşte **corpul ciclului.** Instrucțiunile repetitive sunt de două categorii:

1. Ciclul nedeterminat sau ciclu condiționat;
2. Ciclul controlat prin contori, sau ciclu care se repetă de un număr determinat de ori, în funcție de o variabilă care ia valori determinate.

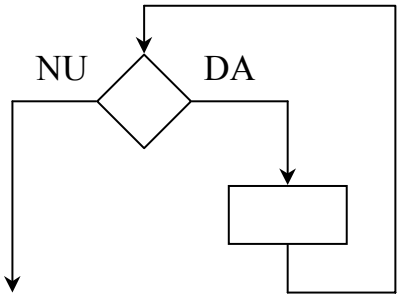
5.1. Cicluri nederminate sau condiționate

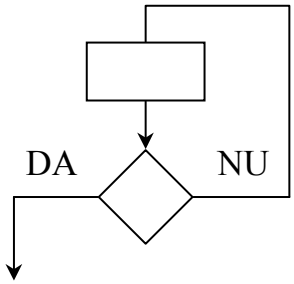
Un anumit număr de instrucțiuni care se repetă de un număr nedeterminat de ori, funcție de o anumită condiție logică, formează un ciclu repetitiv nedeterminat sau condiționat.

În cadrul limbajelor de programare există două tipuri de cicluri repetitive condiționate şi anume cicluri repetitive cu test inițial şi cicluri repetitive cu test final.

La **ciclurile repetitive cu test inițial** condiția logică se găseşte înainte de grupul de instrucțiuni care formează corpul ciclului. Etapele de execuție pentru un astfel de ciclu sunt:

1. Se evaluează expresia logică;
2. Dacă aceasta este **adevărată,** se execută instrucțiunile care formează corpul ciclului;
3. Dacă expresia logică este **falsă**, se execută prima instrucțiune din program





Ciclu repetitiv cu test initial Ciclu repetitiv cu test final

Fig. 5.1. Cicluri repetitive condiționate

care urmează după corpul ciclului.

Scheme logice ale ciclului repetitiv cu test inițial şi ale ciclului repetitiv cu test final sunt reprezentate în figura 5.1.

La **ciclurile repetitive cu test final** condiția logică este pusă după instrucțiunile care formează corpul ciclului. Etapele de execuție ale unui ciclu repetitiv cu test final sunt:

1. Se execută instrucțiunile care formează corpul ciclului;
2. Se evaluează expresia logică;
3. Dacă aceasta este falsă, se execută instrucțiunile din cadrul corpului ciclului;
4. Dacă expresia logică este adevărată se execută prima instrucțiune din program care urmează după instrucțiunile ce formează corpul ciclului.

Deosebirea dintre cele două tipuri de cicluri constă în faptul că la ciclul condițional cu test inițial, instrucțiunile care formează corpul ciclului pot să nu se execute nici o dată, pe când la ciclul condiționat cu test final instrucțiunile din corpul ciclului se execută cel puțin o singură dată.

Cele două tipuri de cicluri se pot înlocui unul cu celălat, adică în program instrucțiunile repetitive pot fi executate fie cu test final, fie cu test inițial.

5.2. Ciclu condițional cu test inițial. Comanda DO WHILE … ENDDO

În limbajul Visual FoxPro există numai ciclul repetitiv condițional cu test inițial. Instrucțiunea pentru ciclul condițional cu test inițial este: **DO WHILE** ....**ENDDO** cu sintaxa:

```
DO WHILE <expL>
     <instrucțiuni>
     [LOOP]
     [EXIT]
ENDDO
```

Instrucțiunea **DO WHILE** se termină în mod obligatoriu cu instrucțiunea **ENDDO**. În cazul în care se doreşte ieşirea forțată dintr-o buclă **DO WHILE** se foloseşte opțiunea **EXIT**. Pentru reevaluarea forțată a condiției de execuție a ciclului se foloseşte instrucțiunea **LOOP**. Utilizarea ei determină numai testarea condiției de execuție a ciclului, iar în continuare controlul programului este preluat de structura repetitivă în conformitate cu rezultatul evaluării condiției.

În cazul în care expresia <expL> este falsă la prima testare a condiției, instrucțiunile care formează ciclul repetitiv **DO WHILE**…**ENDDO** nu se execută nici o dată. De asemenea, dacă expresia logică este întotdeauna adevărată, programul nu mai iese niciodată din ciclu şi trebuie terminat forțat prin apăsarea tastelor (Ctrl+F2).

Prelucrarea repetitivă a articolelor dintr-un fişier de date este o acțiune frecventă. Condiția din structura repetitivă este de obicei parcurgerea tuturor articolelor din fişierul de date până la întâlnirea marcajului de sfârşit de fişier **EOF**. Funcția **EOF**() returnează rezultatul adevărat dacă s-a detectat sfârşitul de fişier şi fals dacă nu avem sfârşit de fişier.

Exemplul 5.1

Să se scrie un program care calculează media a zece numere.

```
SET TALK OFF
CLEAR
NR = 0
MEDIA = 0
TOTAL = 0
CONTOR =0
DO WHILE CONTOR <=10
     @1,1 SAY 'CITESTE NUMARUL' GET NR PICT'999 '
     READ
```

A

B

```
        TOTAL = TOTAL + NR
        CONTOR = CONTOR+1
        @ 2,2 SAY 'S-a citit numarul' GET CONTOR;
            DEFAULT 0 PICTURE '99' DISABLE
        NR=0
    ENDDO
C   MEDIA = TOTAL / CONTOR
    SET DECIMALS TO 3
    @5,5 SAY 'MEDIA= ' + STR(MEDIA,8,3)
    SET DECIMALS TO
```

Explicații

A) Se inițializează variabilele care vor fi folosite în program;

B) Se foloseşte instrucțiunea **DO WHILE** pentru citirea celor zece numere. Instrucțiunile cuprinse între **WHILE** şi **ENDDO** se execută atâta timp cât expresia logică CONTOR<=10 rămâne adevărată;

C) Se calculează media numerelor şi se afişează.

Exemplul 5.2

La un examen de admitere nu se prezintă un anumit număr de candidați înscrişi. Să se scrie un program care listează candidații absenți la una dintre cele două probe ale examenului. Se consideră absent candidatul pentru care în câmpul NOTA1 sau NOTA2 există valoarea 0. Evidența candidaților este ținută în fişierul de date Candidați.DBF cu următoarele câmpuri:

Nota1	N	2	nota la prima probă
Nota2	N	2	nota la cea de a doua probă
Nume	C	20	numele candidatului
Prenume	C	20	prenumele candidatului

```
    USE CANDIDATI
    SET TALK OFF
    CLOSE ALL
    CLEAR
    FLAG=.T.
    USE CANDIDAT
    lin = 4
    col = 6
    @ 1,5 SAY 'CANDIDATI ABSENTI LA CONCURS'
    \=====================================
```

A

```
DO WHILE .NOT. EOF()
     IF NOTA1<=10 OR NOTA2 <=10
          IF NOTA1 < 1
               @ LIN,COL SAY NUME+' '+PRENUME+;
                    + ' ABSENT ; LA PRIMA PROBA'
               LIN=LIN+1
          ELSE
               IF NOTA2 < 1
                    @ LIN,COL SAY NUME+' '+PRENUME;
                         +'ABSENT LA PROBA NR 2'
                    LIN=LIN+1
               ENDIF
          ENDIF
     ELSE
B         WAIT WIND 'EROARE LA INTRODUCEREA;
               NOTELOR'
          FLAG=.F.
          MNUME=NUME
          EXIT
     ENDIF
SKIP
ENDDO
C    IF FLAG=.F.
     BROWSE FOR NUME=MNUME
ENDIF
CLOSE ALL
RETURN
```

Explicaţii

A) Instrucţiunea **DO WHILE** .**NOT**. **EOF**() este un exemplul clasic de căutare într-un fişier de date. În cazul în care în câmpul Nota1 sau Nota2 avem valoarea 0 înseamnă că respectivul candidat nu s-a prezentat şi programul îl va afişa;

B) În cazul în care Nota1 sau Nota2 depăşeşte valoarea 10 se va semnala eroare, se va seta o variabilă numită flag pe valoarea .F. (false) şi se va părăsi forţat bucla **WHILE** cu ajutorul instrucţiunii **EXIT**;

C) Se corectează câmpul eronat cu ajutorul comenzii **BROWSE**.

Exemplul 5.3

Să se scrie un program în limbajul FoxPro care calculează suma numerelor de la 1 la 10 cu excepția valorilor 3 şi 5.

```
SET TALK OFF
CLEAR
I = 0
total = 0
DO WHILE I <= 10
     IF  I = 3 OR I = 5
          I = I + 1
          LOOP
     ENDIF
     total = total + I
     I = I + 1
ENDDO
SET TEXTMERGE ON
\Total = <<total>>
SET TEXTMERGE OFF
CLOSE ALL
RETURN
```

A

Explicații

A) În cazul în care variabila I are valoarea 3 sau 5 se va face incrementarea variabilei urmată de saltul la începutul ciclului folosind instrucțiunea **LOOP** fără a mai fi executate restul instrucțiunilor din corpul ciclului.

5.3. Cicluri condiționate specifice fişierelor de date. Comanda SCAN…ENDSCAN

Un tip special de instrucțiune repetitivă folosită în limbajul FoxPro doar pentru fişierele de date este instrucțiunea **SCAN** ...**ENDSCAN** cu sintaxa:

```
SCAN [NOOPTIMIZE]
     [domeniu][FOR<expL1>][WHILE <expL2>]
     <instrucțiuni>
     [LOOP]
     [EXIT]
     ENDSCAN
```

Această comandă realizează parcurgerea fişierului de date curent şi executarea grupului de instrucţiuni <instrucţiuni> pentru fiecare înregistrare specificată prin: domeniu, **FOR** sau **WHILE**. Comanda este deosebit de utilă în cazul fişierelor de date mari, când pentru a găsi anumite înregistrări se parcurge fişierul de date o singură dată. Domeniul înregistrărilor a fost specificat în capitolele anterioare şi se referă la:

- toate înregistrările fişierului de date (**ALL**);
- următoarele înregistrări de la înregistrarea curentă până la sfârşitul fişierului (**REST**);
- următoarele înregistrări dorite de la înregistrarea curentă (**NEXT**).

Expresiile logice din clauza **FOR** sau **WHILE** determină condiţia de căutare. Clauza **NOOPTIMIZE** inhibă optimizarea **RUSHMORE**. Clauzele **LOOP** şi **EXIT** au aceeaşi semnificaţie cu cele de la comanda **DO WHILE**…**ENDDO**.

Exemplul 5.4

Să se scrie un program care consultă fişierul Clienţi.DBF şi măreşte costurile tuturor clienţilor din judeţul “SB” care încep cu litera “B”. Se vor afişa câmpurile Companie, Oraş, Cost.

```
SET TALK OFF
CLOSE ALL
CLEAR
USE CLIENTI
SCAN ALL FOR judet ='SB'
     IF SUBSTR(companie,1,1)='B'
          mcost=cost*1.1
          REPL COST WITH mcost
          DISP FIELDS companie,oras,cost
     ENDIF
ENDSCAN
CLOSE ALL
RETURN
```

5.4. Cicluri controlate prin contori

Ciclurile repetitive sau iterative controlate prin contori se execută de un număr determinat de ori în funcţie de valorile pe care le poate lua o variabilă numită variabila de control. În general aceasta ia valori între o valoare iniţială şi o valoare

finală. În limbajul FoxPro acest tip de ciclu repetitiv se realizează cu instrucțiunea **FOR**…**ENDFOR** cu sintaxa:

```
FOR variabila_de_control=valoare_inițială
    TO valoare_finală [STEP <expN>] <instrucțiuni>
    [EXIT]
    [LOOP]
    ENDFOR/NEXT
```

Comanda determină execuția repetată a corpului ciclului (<instrucțiuni>), contorizarea numărului de iterații făcându-se prin variabila de control. Valoarea inițială a acestei variabile este dată de o expresie numerică, iar valoarea finală de o altă expresie numerică. Modul în care se ajunge de la valoarea inițială la valoarea finală, este controlat de către clauza **STEP** prin expresia numerică <expN>, care intră în componența ei. Dacă această clauză lipseşte atunci se consideră pasul egal cu unu, adică variabila de control este incrementată la fiecare iterație.

Deci, la prima iterație a grupului de instrucțiuni variabila de control va avea valoarea egală cu valoarea inițială iar la a doua iterație va avea valoarea:

```
variabila de control = valoarea inițială ± <expN>
```

unde <expN> este o valoare numerică care poate fi pozitivă sau negativă.

Etapele de lucru ale acestei instrucțiuni sunt:

1. se evaluează variabila de control cu valoarea inițială;
2. se compară valoarea inițială cu valoarea finală. Dacă este diferită se execută corpul ciclului;
3. se incrementează variabila de control dacă opțiunea **STEP** lipseşte. Dacă nu, se măreşte sau se micşorează variabila de control cu valoarea <expN> din clauza **STEP**;
4. se repetă paşii 2 şi 3 de atâtea ori până când valoarea variabilei de control devine mai mare sau mai mică decât valoarea finală, moment în care se trece la prima instrucțiune din afara corpului ciclului, adică prima instrucțiune care urmează după cuvântul rezervat **ENDFOR**.

Valoarea inițială, valoarea finală şi valoarea numerică cu care se face incrementarea se evaluează la intrarea în buclă, modificarea lor ulterioară ne influențând numărul de iterații. Nu este indicat să se modifice de către utilizator valoarea variabilei de control în timpul unui ciclu iterativ, deoarece acest lucru poate conduce la rezultate imprevizibile.

Clauzele **EXIT** şi **LOOP** sunt identice cu cele de la instrucțiunea **DO WHILE**…**ENDDO**.

Cuvântul rezervat **ENDFOR** este obligatoriu să fie prezent la sfârşitul unui ciclu iterativ controlat prin contori. El poate fi înlocuit cu cuvântul **NEXT** având aceeaşi semnificație.

Exemplul 5.5

Folosind un ciclu iterativ controlat prin contori, să se scrie un program care calculează media unui număr aleator de numere.

```
SET TALK OFF
CLOSE ALL
CLEAR
Media = 0
Total = 0
Numnr = 0
@ 1,1 SAY 'Pentru care numere se calculeaza;
    media :' GET numnr PICT '9999'
READ
FOR I = 1 TO numnr
@ 2,1 SAY 'Citeste numerele = ' GET NR;
    PICT '9999.99' DEFAULT 0
READ
Total=total + nr
ENDFOR
MEDIA =TOTAL/numnr
@ 6,6 SAY 'Media numerelor = '+ STR (MEDIA)
CLOSE ALL
RETURN
```

A

Explicaţii

A) În instrucţiunea **FOR**...**ENDFOR** lipseşte clauza **STEP** deci variabila de control I va fi incrementată la fiecare iteraţie de la valoarea 1 la valoarea atribuită varibilei numnr. Instrucţiunea **FOR**…**ENDFOR** se poate scrie în alt mod, efectul instrucţiunii fiind acelaşi:

```
FOR I = numnr TO 1 STEP -1
    @ 2,1 SAY 'Citeste numerele= ' GET NR;
        PICT '9999.99' DEFAULT 0
    READ
    Total=total + nr
ENDFOR
```

Instrucţiunea **FOR**…**ENDFOR** este mai rapidă decât instrucţiunea **DO WHILE** … **ENDDO** din această cauză este recomandabil folosirea instrucţiunii **FOR** în loc de **DO WHILE** ori de câte ori este cazul.

Exemplul 5.6

Să se scrie un program în limbajul FoxPro care ilustrează faptul că instrucțiunea **FOR** … **ENDFOR** este mai rapidă decât instrucțiunea **DO WHILE** … **ENDDO**. Se va utiliza funcția **SECONDS**() care returnează numărul de secunde trecute de la miezul nopții (ora zero).

```
SET TALK OFF
CLOSE ALL
CLEAR
incepdo = SECONDS()
I = 0
DO WHILE I < 1000000
    I = I + 1
ENDDO
sfdo = SECONDS()
FOR I = 1 TO 1000000
ENDFOR
sffor = SECONDS()
timpdo = sfdo - incepdo
timpfor = sffor - sfdo
@1,1 SAY 'Nr secunde pentru Do While= '  GET timpdo;
    PICT '99.99999'
@2,1 SAY 'Nr secunde pentru For= ' GET timpfor;
    PICT '99.999999'
CLOSE ALL
RETURN
```

Concluzii

Instrucțiunile repetitive, deservesc structura repetitivă din orice limbaj de programare. Acest set de instrucțiuni, există în toate limbajele de programare, înțelegerea lor fiind obligatorie în activitatea de programare

Capitolul

6

Formulare în mediul Visual Fox

De-a lungul anilor, a fost abandonat modelul de programare conform căruia programul controla ceea ce făcea utilizatorul. Acum utilizatorii au pretenția să poată controla aplicația. Cu alte cuvinte s-a trecut de la programarea structurată la programarea orientată pe obiecte. În mediile de programare vizuale (Visual Fox, Delphi, Visual Basic, Acces, etc) orientate spre baze de date, entitatea fundamentală care apare în aplicații este **formularul**. Acesta se aseamănă cu o fereastră dar se deosebeşte fundamental, prin faptul că este format din obiecte.

6.1 Paradigma obiectelor de tip formular

Programele orientate spre obiecte acceptă două tipuri fundamantale de obiecte vizuale: **containerele şi obiectele**. *Un container este orice obiect în care puteți plasa alte obiecte*. Prin urmare, un formular este un container, deoarece el conține obiecte, cum ar fi casetele de text, casetele de validare şi multe alte tipuri de obiecte. O caracteristică interesantă a formularelor este aceea că, întrucât sunt obiecte, puteți crea noi formulare pe baza celor existente. Datorită proprietății de moştenire a obiectelor, puteți crea definiția unui formular generic şi apoi să o folosiți pentru a crea restul formularelor unei aplicații. De exemplu, puteți defini aspectul global, culoarea, fonturile şi alte proprietăți. Apoi, salvați pur şi simplu acest formular sub forma unei clase vizuale şi folosiți-l drept şablon pentru toate noile formulare.

Există mai multe metode de a crea un nou formular:

- Selectați **File, New**, ***Form*** din meniul de sistem
- Introduceți comanda **CREATE FORM** în fereastra de comenzi
- Introduceți comenzile:
- **NewForm = CREATEOBJECT("FORMULAR")**
  NewForm.Open
- Folosiți vrăjitorul **Form Wizard.**

Deşi vrăjitorii vă permit să creați un obiect elementar, cum ar fi un formular, veți constata deseori că trebuie să modificați şi să îmbunătățiți ceea ce creează aceştia.
Figura 6.1 prezintă un formular în care nu s-a introdus nici un obiect. De asemenea, această figură prezintă zona de lucru a instrumentului **Form Designer**, în care puteți construi o reprezentare vizuală a formularului.

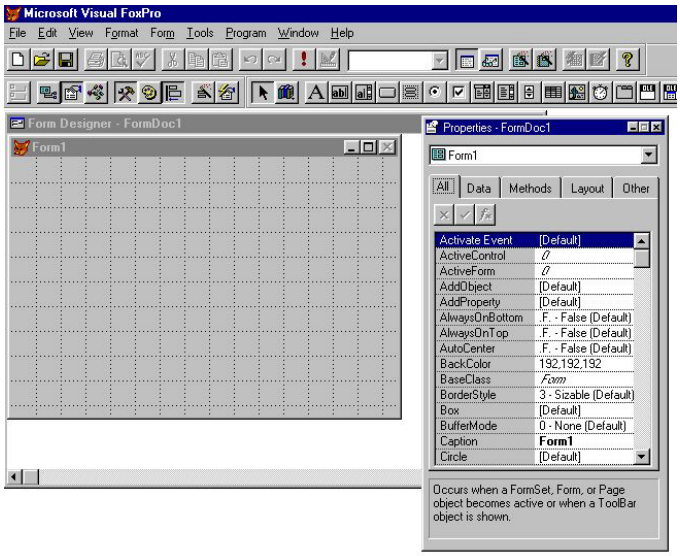


Fig 6.1 Formularul nou

Pentru a adăuga obiecte în formular, Visual FoxPro utilizează bara cu instrumente Controls.

6.2. Bara cu instrumente *Controls*

Aceasta defineşte pictogramele barei cu instrumente Controls, care sunt următoarele:

Semnificatia fiecărui buton este următoarea:

Control / Nume	**Picto-gramă**	**Descriere**
Selectare obiecte / *Select Pointer*		Redimensionează şi mută obiectele
Vizualizare clase / View Classes		Vă permite să afişaţi bibliotecile de clase înregistrate sau să deschideţi noi biblioteci de clase
Etichetă / Label		Utilizat pentru afişarea unui text fix.
Casetă de text		Utilizat pentru afişarea unei singure linii de text
Casetă de editare		Utilizat pentru afişarea mai multor linii de text
Buton de comandă		Utilizat pentru a crea un buton pe care utilizatorul îl poate alege pentru a executa o comandă.
Grup de butoane		Utilizat pentru a crea un grup de butoane pe care utilizatorul le poate alege pentru a executa diverse comenzi
Grup de butoane de opţiune		Utilizat pentru a afişa mai multe opţiuni din care utilizatorul poate selecta una singură
Casetă de validare		Utilizat pentru a indica dacă ceva este selectat sau nu, adevărat sau fals sau pentru a afisa opţiuni multiple din care utilizatorul poate

		selecta mai multe opțiuni.
Casetă combinată		Utilizat pentru crearea unei liste derulante care să permită utilizatorului să selecteze unul dintre articolele listei sau pentru crearea unei casete combinate care să permită utilizatorului să selecteze dintr-o listă sau să introducă o nouă valoare.
Casetă cu listă		Utilizată pentru afişarea unei liste derulante de articole
Casetă de modificare valorică		Utilizat pentru incrementarea sau decrementarea unei valori întregi în interiorul unui domeniu.
Grid		Utilizat pentru crearea unui control de tip browser
Control cu imagine		Utilizat pentru afişarea unei imagini grafice
Timer		Utilizat pentru planificarea evenimentelor şi pentru stabilirea intervalelor dintre ele
Cadru de pagină		Utilizat pentru reprezentarea mai multor pagini ale unui control într-un singur formular
OLE 2.0		Utilizat pentru a permite legarea şi înglobarea obiectelor (OLE) dintr-un server OLE
Legătură OLE		Utilizat pentru a permite legarea şi înglobarea obiectelor (OLE) în câmpul general al unui tabel
Linie		Utilizat pentru desenarea diverselor stiluri de linii pe formularul dumneavoastră
Shape		Utilizat pentru a desena dreptunghiuri, pătrate, cercuri şi elipse
Separator		Utilizat la crearea unei bare cu instrumente personalizate în vederea separării instrumentelor în grupuri
Blocare generator		Utilizat pentru deschiderea generatoarelor corespunzătoare obiectelor atunci când acestea sunt selectate
Blocare buton *Button Lock*		Utilizat pentru adăugarea mai multor obiecte de acelaşi tip, fără a fi necesar să reexecutați clic pe butonul controlului respectiv din bara cu instrumente
Container		Permite introducerea mai multor entități Vfox
Control WEB		Permite legătura la Internet

6.2.1 Controlul (butonul) Select Pointer

Butonul **Select Pointer** reactivează indicatorul mouse-ului. În mod normal, atunci când plasați un obiect pe ecran, mouse-ul revine la modul de indicare după poziționarea controlului. În schimb, dacă folosiți instrumentul *Button Lock* pentru a plasa mai multe obiecte de acelaşi tip, trebuie să executați clic pe acest buton pentru a reveni la modul de indicare.

6.2.2 Controlul (butonul) View Classes

Butonul **View Classes** vă permite să vizualizaţi alte clase de obiecte salvate ca fişere VCX. Înainte de a utiliza acest buton, puteţi să înregistraţi şi să selectaţi propriile dumneavoastră biblioteci folosind **Tools, Options, Controls**. Puteţi, de asemenea, să înregistraţi o bibliotecă de clase personalizată apăsând butonul **View Classes** din bara cu instrumente şi selectând opţiunea **Add**. Este afişată caseta de dialog **Open** care vă permite să selectaţi biblioteca din orice director. Selectarea bibliotecii duce la deschiderea şi, simultan, la înregistrarea ei.

După înregistrarea lor, numele acestor biblioteci de clase personalizate apar atunci când selectaţi butonul **View Classes**. Atunci când selectaţi o bibliotecă personalizată, butoanele din secţiunea mediană a acestor bare cu instrumente sunt înlocuite cu butoanele corespunzătoare noilor obiecte.

Puteţi reveni oricând la obiectele prestabilite selectând **Standard** din lista View Classes.

6.2.3 Controlul (butonul) Builder Lock

Instrumentul **Builder Lock** comută în modul generare, construcţie obiect. Generatoarele vă ajută să definiţi proprietăţile unui obiect prin afişarea unor ecrane de introducere care vă solicită să specificaţi cele mai importante proprietăţi ale controlului. Generatoarele reprezintă o metodă eficientă de învăţare a celor mai importante proprietăţi ale obiectelor. Totodată ele permit o generare mai uşoară a obiectului dorit. Nu toate obiectele din mediul Visual Fox permit generarea lor cu ajutorul constructorului (builderului). Instrumentul **Button Lock** vă permite să adăugaţi instanţe multiple ale aceluiaşi obiect de bază, fără să fie necesar să reveniţi în bara cu instrumente **Controls** şi să-l reselectaţi. Executaţi clic o dată pe acest buton pentru a-l apăsa, după care selectaţi controlul pe care vreţi să-l folosiţi. Adăugaţi numărul dorit de instanţe ale controlului selectat, fără să reveniţi la bara cu instrumente. Puteţi chiar să schimbaţi controlul selectat, executând clic pe un alt control din bara cu instrumente. Puteţi, de asemenea, să activaţi instrumentul **Button Lock,** executând dublu clic pe butonul oricărui control.

6.2.4 Obiectul Label sau Etichetă

Etichetele sau obiectele **Label**, sunt simple şiruri de text adăugate unui formular pentru identificarea câmpurilor sau pentru afişarea unor informaţii fixe de tip caracter, destinate utilizatorului. O etichetă poate avea o singură linie sau, în cazul în care valoarea proprietăţii **WordWrap** este .T., se poate întinde pe mai multe linii. Cele mai uzuale proprietăţi ale etichetelor sunt cele care permit schimbarea fontului, a dimensiunii şi a stilului fontului. Puteţi, de asemenea, să definiţi o culoare de fundal

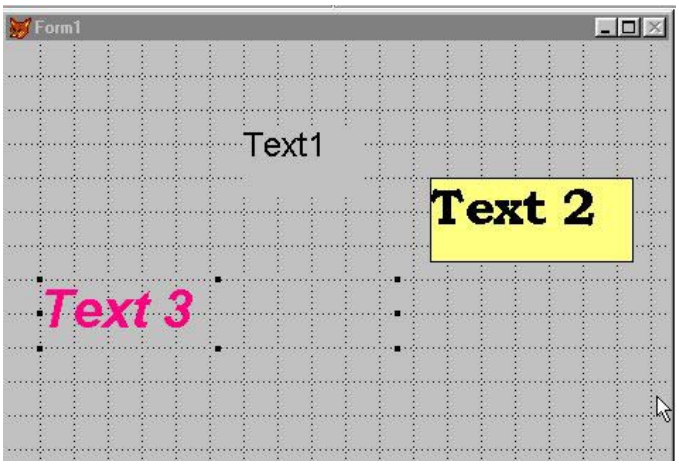


Fig 6.1 Exemple de obiecte etichetă sau text

sau să stabiliți un fundal transparent, astfel încât să fie vizibile toate obiectele de dedesubt. Textul se poate înconjura cu un chenar cu linie simplă, continuă. Figura 6.2 prezintă aceste posibilități.

În fereastra **Properties** sunt definite proprietățile obiectului text fig 6.3.

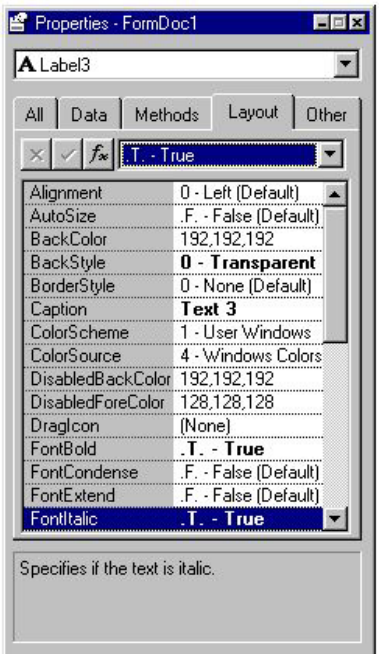


Fig 6.2 Fereastra cu proprietățile obiectului

Obiectul text nu are o sursă de date. Introduceți textul care doriți să apară pe ecran, direct în proprietatea **Caption** a obiectului.

Majoritatea obiectelor au o proprietate **Visible**. Dacă îi atribuiți valoarea .F. atunci când definiți formularul, obiectul nu va apărea la rularea formularului. El există totuşi, însă nu este vizibil. Această proprietate permite personalizarea formularelor în vederea afişării numai a obiectelor necesare în contextul specific.
O utilizare interesantă a acestei proprietăți o reprezintă definirea mai multor obiecte suprapuse, dintre care numai unele sunt vizibile la un moment dat.

Grupul de proprietăți **Font** permite definirea fontului cu care se va tipării textul din cadrul obiectul. Acestea fac referință la mărime, culoare, fond, subliniere etc.

Proprietățile **Left** şi **Top** definesc poziția obiectului în formular, iar proprietățile **Height** şi **Width** definesc mărimea obiectului pe verticală respectiv orizontală.

Proprietatea **ToolTipBox** permite afişarea unui text, dacă mouse-ul este poziționat pe obiect, în cazul în care proprietatea **ShowTips** a formularului are valoarea **True**.

Deşi acest aspect nu a fost discutat referitor la forme, aceste obiecte au ca şi etichetele, evenimente asociate lor. De exemplu, ele pot să răspundă unui simplu clic, unui dublu clic sau unui clic cu butonul drept at mouse-ului. Toate evenimentele au metode prestabilite.

În general metoda **Click Event** execută o acțiune în momentul în care se execută click pe butonul stâng al mouse-ului, **Dblclick Event** execută o acțiune în momentul în care se execută dublu click pe butonul stâng al mouse-ului, **RightClick Event** execută o acțiune în momentul în care se execută click pe butonul drept al mouse-ului. **GotFocus Event** şi **LostFocus Event** execută acțiuni, în momentul în care controlul se transmite unui obiect sau în momentul în care se părăseşte un obiect. Metodele **Init Event** şi **Load Event** execută acțiuni de inițializare a obiectului, fie la activarea sa, fie la încărcarea sa în memorie. În obiectul **Form** aceste metode pot fi folosite pentru deschiderea bazelor de date şi a tabelelor în diverse zone de lucru controlate de către utilizator.

6.2.5 Obiecte Text Box

Deosebirea fundamentală între o etichetă şi un obiect **TextBox** sau casetă de text constă în sursa de date a celor două. În cazul unei etichete textul este invariabil şi este dat de valoarea proprietății **Caption** a obiectului. Pe de altă parte, sursa unei casete de text este de obicei un câmp al unui tabel, dar poate fi la fel de bine o

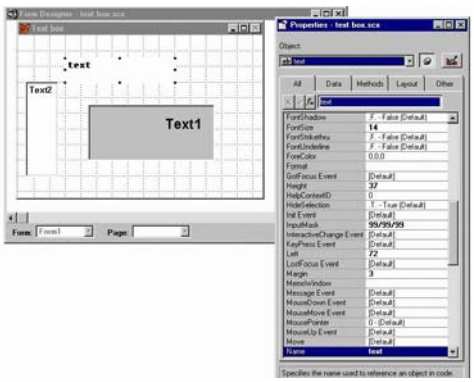


Fig.6. 4 Exemple de obiecte Text box

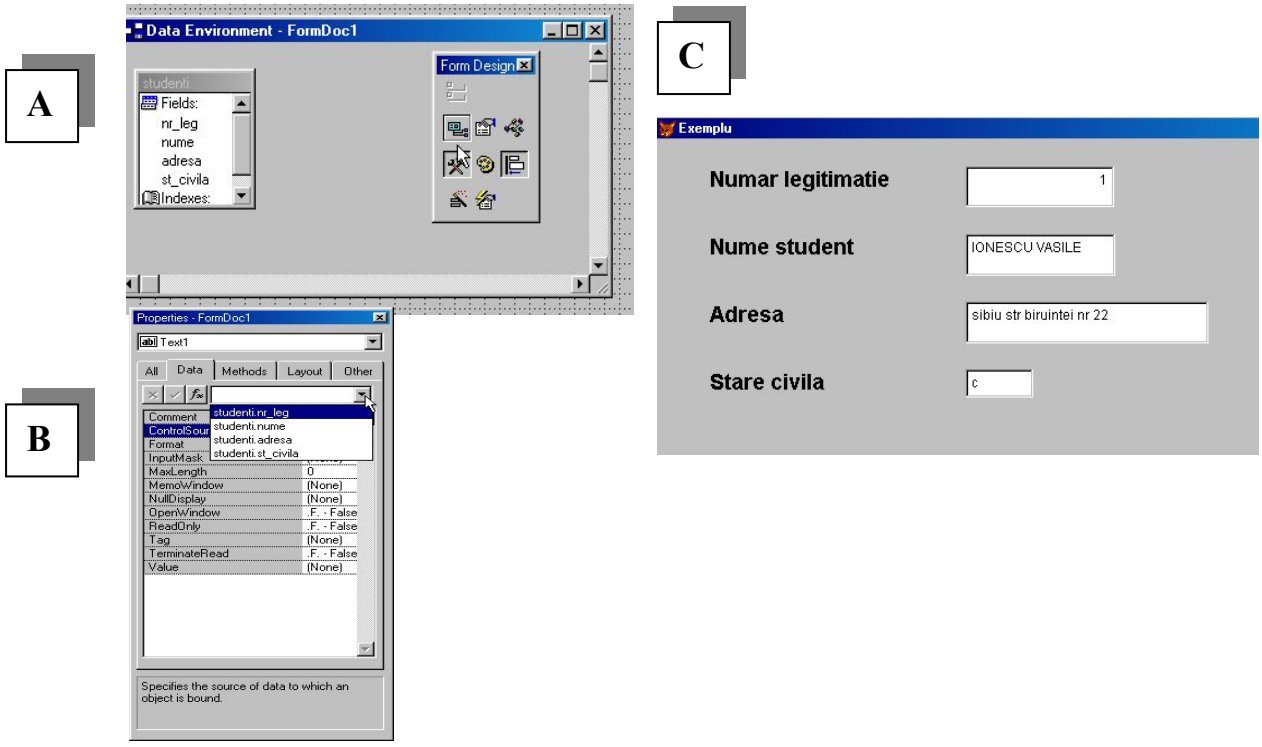
variabilă de memorie. Printre cele mai comune proprietăți ale casetei de text pe care le veți modifica se numără cele referitoare la culoare, font, dimensiunea şi poziția casetei de text şi, evident, sursa de date. Figura 6.4 prezintă un exemplu de casetă de text etichetată Text1.

Pe lângă proprietatea **Visible**, discutată în legătură cu etichetele, casetele de text (ca şi alte obiecte cu surse de date) au o proprietate **Enable**. Atunci când valoarea proprietăţii **Enable** este .T., utilizatorul poate accesa obiectul. Evident, codul asociat unei metode **When** poate forţa părăsirea câmpului. Însă în cazul în care valoarea proprietăţii **Enable este .T**., metoda **When** nu va fi niciodată executată. De altfel, un câmp dezactivat poate fi afişat diferit de un câmp activat folosind culori diferite pentru prim-plan şi pentru fundal. (Vezi proprietăţile **DisableBackColor** şi **DisableForeColor**) Utilizarea acestora în locul proprietăţii **Visible** semnalează utilizatorului existenţa câmpului sau a opţiunii, precum şi faptul că nu poate fi selectată în contextul respectiv.

O altă opţiune referitoare la câmpurile care posedă surse de date este să le protejaţi la scriere (**ReadOnly**). Folosiţi opţiunea **ReadOnly** pentru a permite utilizatorului să vizualizeze textul, fără să-l poată modifica. Dacă înălţimea casetei de text este suficient de mare, textul continuă automat pe linia următoare. În schimb, atunci când spaţiul se epuizează, caseta de text trebuie să întrerupă afişarea textului. Pentru date de tip caracter de lungime mai mare, folosiţi o casetă de editare. Deşi puteţi defini o fereastră memo distinctă pentru afişarea variabilelor de tip memo legate la un câmp de text, este recomandabil să folosiţi obiecte de tip casetă de editare, **Edit Box**, pentru a face acest lucru.

Exemplul 6.1

Să se realizeze un formular pentru vizualizarea datelor existente în tabela Studenti din baza de date Facultate (vezi Capitolul 4).



Explicaţii

A) După crearea unui nou formular şi activarea **Form Designer** se acţionează asupra butonului **Data Environment,** pentru activarea ferestrei cu acelaşi nume pentru ataşarea bazei de date şi a tabelei dorite la formular;
B) Se construiesc obiectele **Label** şi **Text Box**. Proprietatea **Control Source** din caseta **Data** permite legarea obiectului **Text** de câmpul dorit din tabelă. Proprietăţile **Format** şi **InputMask** permit definirea unor formate personalizate şi a unei măşti de intrare pentru obiectul **Text** (vezi capitolul 4);
C) Formularul realizat vizualizează datele din prima înregistrare a tabelei Studenti

6.2.6 Obiecte Edit Box

Obiectele de editare Edit Box îmbunătăţesc caracteristicile casetelor de text, permiţând utilizatorului să modifice textul afişat. În primul rând, ele posedă o bară de derulare verticală, care permite utilizatorului să parcurgă textele lungi mai rapid decât prin derulare linie cu linie. Având în vedere faptul că o casetă de editare

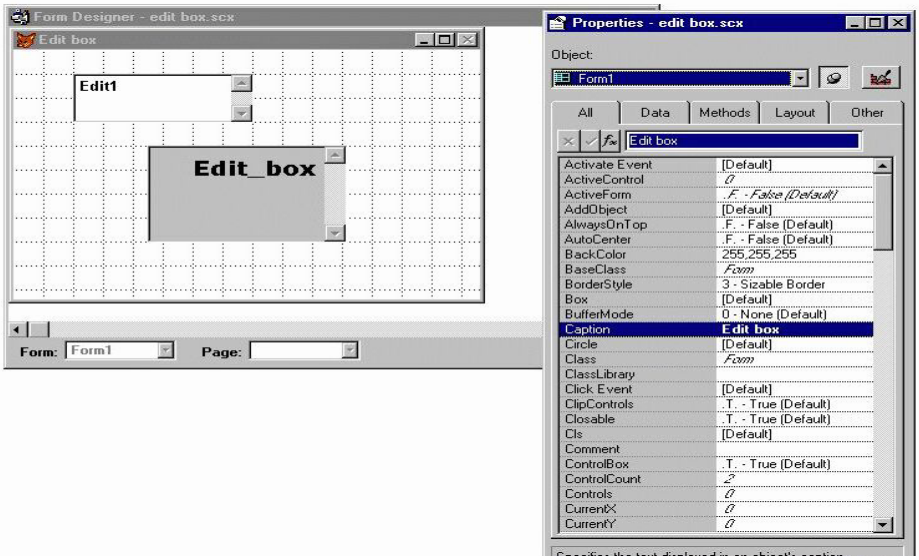
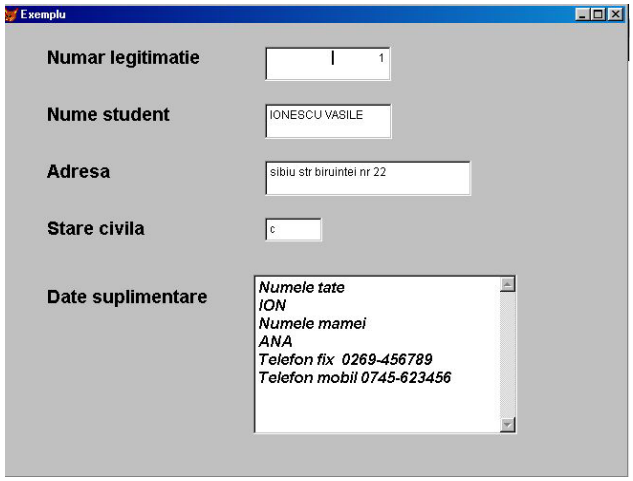


Fig.6. 5 Controlul Edit Box

acceptă până la 2.147.483.647 de caractere, barele de derulare se pot dovedi vitale. Figura 6.5 ilustrează o casetă de editare situată în jumătatea inferioară a ecranului. În mare, o casetă de editare prezintă aceleaşi caracteristici ca şi o casetă de text.
O proprietate specială ests **Allow Tabs**, care specifică dacă utilizatorul poate utiliza tabulatoare în textul casetei de edttare. Dacă activaţi această opţiune, utilizatorul va trebui să apese Ctrl+Tab pentru a trece la controlul următor (în locul tastei T). Prin urmare, dacă permiteţi utilizarea tabulatoarelor; nu uitaţi să indicaţi pe formular că utilizatorul trebuie să apese Ctrl+Tab pentru a părăsi câmpul.
Exemplul 6.2

Se adaugă la formularul prezentat la exemplul 6.1 obiectul **Edit Box** prin care sunt vizualizate date suplimentare



6.2.7 Obiecte Command Button

Un buton de comandă are aspectul unui dreptunghi tridimensional cu un text în interior. Formularele utilizează deseori butoane de comandă pentru a permite selectarea unei opțiuni dintr-un set de opțiuni. Una dintre cele mai uzuale utilizări ale butoanelor de comandă este pentru parcurgerea unui tabel. De obicei se utilizează patru butoane pentru deplasarea indicatorului de înregistrări la începutul tabelului, la înregistrarea anterioară, la cea care urmează, respectiv la sfârşitul fişierului.
Printre celelalte utilizări ale butoanelor de comandă se numără închiderea unui formular, părăsirea aplicației sau tipărirea unui raport. În toate cazurile, evenimentul principal asociat unui buton de comandă este un clic. Atunci când utilizatorul

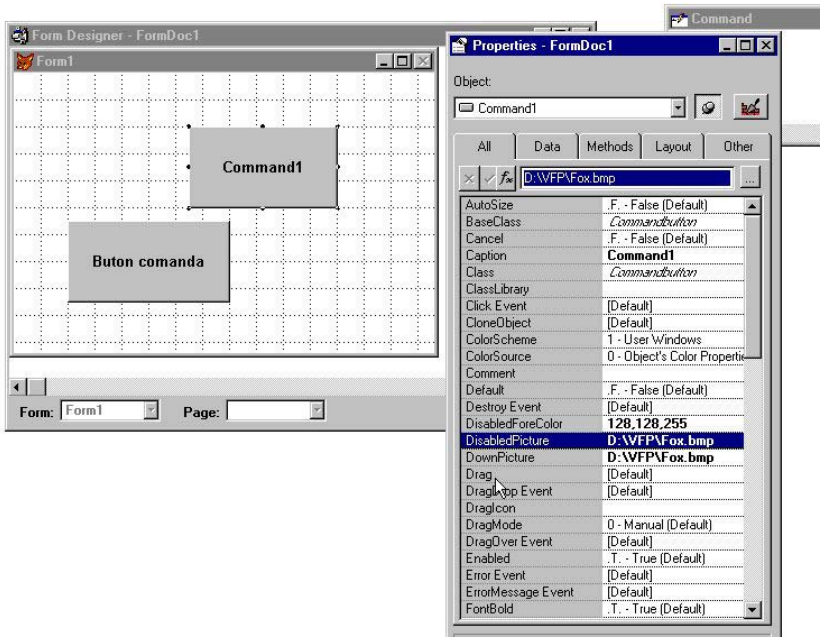


Fig.6. 6 Buton de comandă

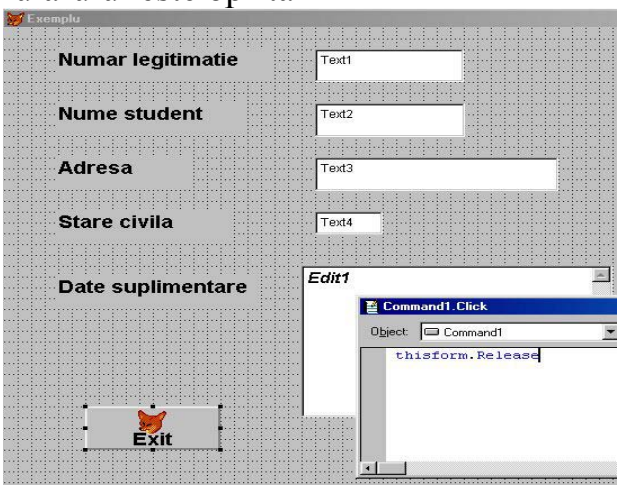
execută clic pe un buton, el **se aşteaptă să efectueze o acțiune, cum ar fi deplasarea la**

o altă înregistrare sau la un formular existent. Totuşi butoanele de comandă pot executa multe alte acţiuni. Puteţi chiar să asociaţi valoarea unui buton cu o sursă de date. Spre deosebire de multe obiecte care vă permit să controlaţi atât culoarea de prim-plan, cât şi culoarea de fundal, butoanele de comandă nu vă permit să stabiliţi decât prim-planul (textul). Setul de culori Windows defineşte culoarea de fundal. În schimb, puteţi folosi culori de prim-plan diferite pentru starea activată, respectiv dezactivată.

O altă caracteristică interesantă a butoanelor de comandă este capacitatea acestora de a afişa o imagine bitmap folosind proprietăţile **Picture, Disabled Picture şi DownPicture**. Toate aceste trei proprietăţi vă permit să selectaţi un fişier BMP sau DIB pentru a-l afişa pe buton. Proprietatea **Picture** defineşte imaginea ce urmează a fi folosită atunci când butonul este neapăsat. Proprietatea **DownPicture** defineşte imaginea afişată atunci când utilizatorul execută clic pe buton. În sfârşit, proprietatea **DisabledPicture** defineşte imaginea folosită atunci când butonul este dezactivat. O imagine clasică pentru butonul neapăsat este cea a cercului roşu cu o bară deasupra.

Exemplul 6.3

Să se insereze în formularul de la exemplele precedente un buton de comandă prin care execuţia formularului este oprită



Explicaţii

Butonul are o imagine ataşată şi un text (Exit). În procedura Click Event ataşată butonului s-a apelat metoda **Release**, prin care obiectul formular este eliminat din memorie. Numele butonului este Command1.

6.2.8 Obiecte Command Group

Puteţi definii butoanele de comandă fie independent unul câte unul, fie în grup. Principalul avantaj al utilizării unui grup este că vă permite să plasaţi codul comun într-o singură metodă aparţinând grupului.

Cele trei butoane din figura 6.7, efectuează acţiuni asemănătoare. Fiecare emite un mesaj cu o altă destinaţie. În loc să introduceţi în mod repetat codul

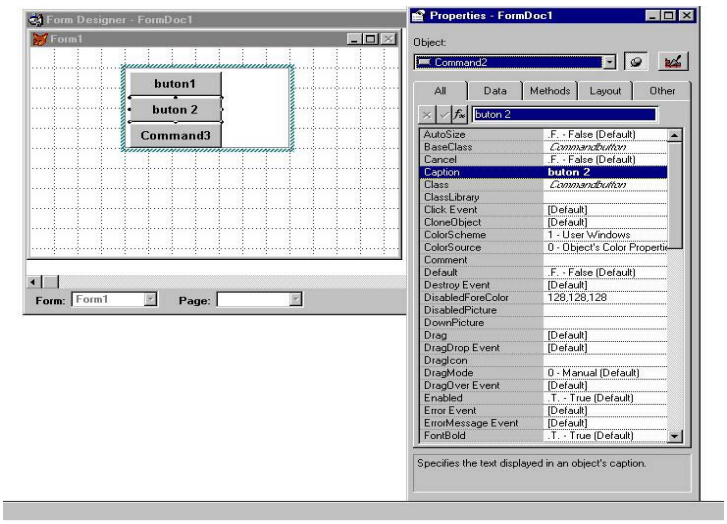


Fig.6. 7 Butoane de comandă sub forma unui grup

generării unui mesaj în fiecare buton, este preferabil să le combinaţi într-un singur obiect. Evenimentul **Click Event** al grupului de butoane de comandă, utilizează proprietatea **Value** a grupului pentru a stabili butonul apăsat de utilizator.
Codul program pentru metoda amintită este:

```
DO CASE
    CASE This.Value = 1
      = MESSAGEBOX (‘Mesaj trimis de BUTON 1’)
    CASE This.Value = 2
      = MESSAGEBOX (‘Mesaj trimis de BUTON 2’)
    CASE This.Value = 3
       = MESSAGEBOX (‘Mesaj trimis de COMMAND 3’)
ENDCASE
```

Proprietatea **Value** conţine numărul butonului pe care s-a executat clic. VFP numerotează butoanele secvenţial, în ordinea în care le adăugaţi în grupul de comandă. Proprietăţii ButtonCount trebuie să-i atribuiţi o valoare egală cu numărul de butoane din grup.

Pentru a declara unul dintre butoanele de comandă ca fiind cel prestabilit, atunci când utilizatorul apasă [↵], atribuiţi valoarea .T. proprietăţii **Default** a butonului respectiv. Aveţi în vedere faptul că numai un singur buton al unui formular poate fi declarat drept prestabilit. Nu puteţi specifica în acelaşi timp un buton dintr-un grup de comandă şi un buton independent drept butoane prestabilite sau butoane din două sau mai multe grupuri de comandă.

6.2.9 Obiecte Option Group (butoane radio)

Un alt tip de butoane sunt butoanele radio care nu apar niciodată singure. De altfel, un singur buton radio nu prea are sens - este ca şi cum radioul dumneavoastră

ar avea un singur buton. Scopul unui set de butoane radio este să ofere o serie de opțiuni din care utilizatorul trebuie să selecteze una. Spre deosebire de un grup de comandă pe care utilizatorul poate să-l ignore, un grup de butoane radio are întotdeauna un buton selectat.

Prin urmare, cea mai importantă proprietate a unui grup de butoane radio (pe care Visual FoxPro îl numeşte grup de opțiune) este **ButtonCount**. Trebuie să definiți această proprietate înainte de a începe să etichetați butoanele individuale. Tot înaintea configurării butoanelor individuale, va trebui să definiți butonul prestabilit. În mod prestabilit, primul buton devine cel prestabilit, însă puteți schimba butonul prestabilit modificând proprietatea **Value** a grupului de opțiune. Pentru a declara al doilea buton drept butonul prestabilit, atribuiți proprietății **Value** valoarea 2. Concomitent, este stabilită şi proprietatea **Value** a fiecărui buton. Valorile pot fi 0 (deselectat) sau 1 (selectat).

După stabilirea numărului de butoane din grupul de opțiune, le puteți selecta pe fiecare în parte, pentru a le defini titlurile. Deschideți lista derulantă situată la baza casetei de dialog Properties. Butoanele radio individuale apar imediat sub grupul de opțiune. Figura 6.8 ilustrează această listă.

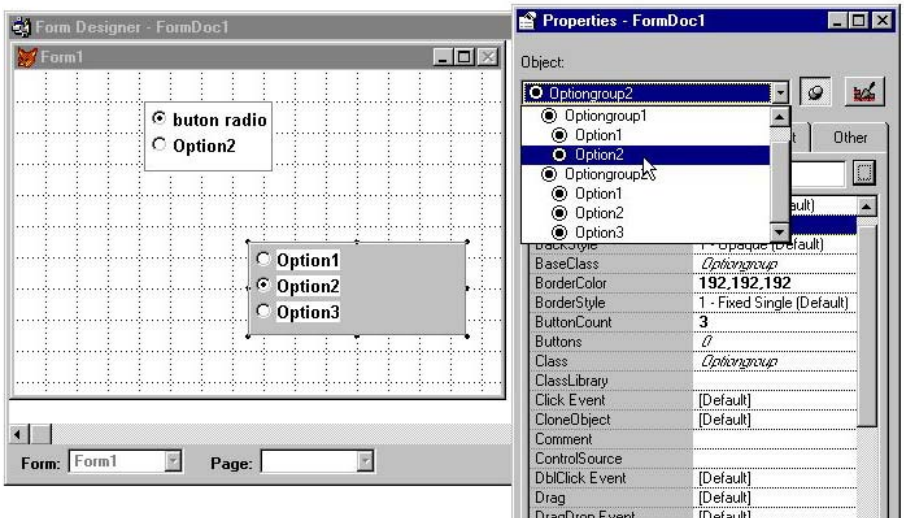


Fig.6. 8 Butoane radio

Ca şi în cazul butoanelor de comandă, puteți folosi o imagine pentru un buton de opțiune, însă numai dacă atribuiți în prealabil valoarea **Graphical** proprietății **Style**. Desigur, în acest fel butonul de opțiune va semăna cu un buton de comandă. Rețineți faptul că deosebirea fundamentală este că un grup de butoane de opțiune are întotdeauna o valoare selectată. Butonul selectat este apăsat atunci când este acceptată selecția.

Atât grupurile de comandă, cât şi cele de opțiune, pot avea un chenar. Figura 6.8 ilustrează grupul de comandă cu chenarul afişat; chenarul grupului de opțiune este dezactivat. În timp ce proprietatea **Border Style** controlează existența unui chenar cu linie simplă continuă, proprietatea **BackStyle** controlează afişarea acestuia. Atunci

când proprietatea **BackStyle** are valoarea **Transparent**, VFP nu afişează chenarul şi culoarea de fundal.

6.2.10 Obiecte Check Box sau casete de validare

În general, casetele de validare (fig 6.9), reprezintă câmpuri sau variabile logice individuale. Prin convenţie, o casetă de validare goală semnifică faptul că opţiunea respectivă nu este selectată. Atunci când utilizatorul o selectează, în casetă apare un X. Spre deosebire de grupurile de comandă sau de opţiune, casetele de validare funcţionează independent unele de altele. În consecinţă, puteţi avea orice număr de casete de validare într-un formular. Mai mult, utilizatorul poate să nu selecteze nici una, să le selecteze pe toate sau numai o parte dintre ele.

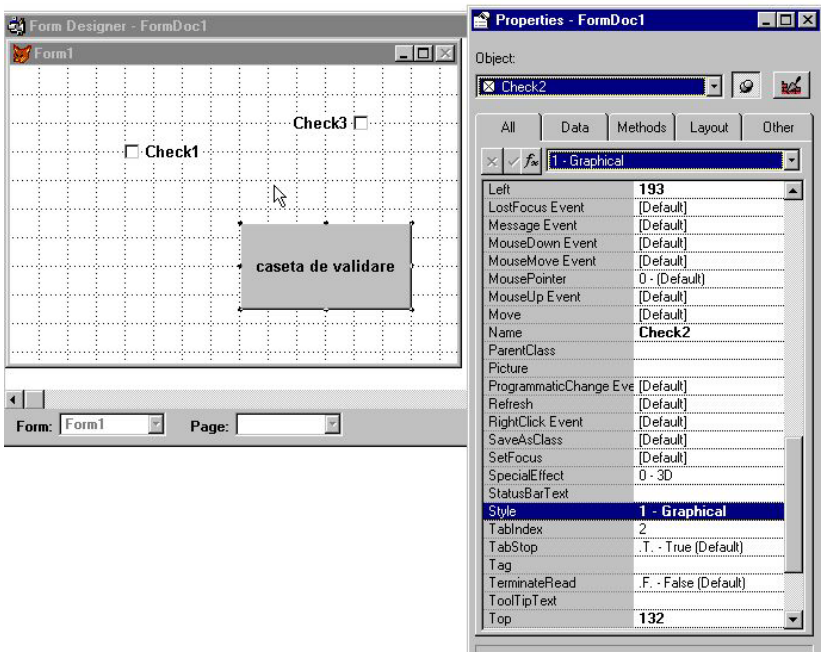


Fig.6. 9 Casete de validare

Ca şi în cazul butoanelor de opţiune, puteţi afişa o casetă de validare sub forma unui buton grafic, folosind proprietatea **Style.** Atunci când este afişată în acest fel, caseta de validare poate conţine o imagine bitmap, precum şi un titlu. Dacă butonul apare neapăsat, opţiunea nu este selectată (are valoarea 0 sau .F.). Dacă butonul apare apăsat, valoarea este egală cu 1 sau .T. VFP stabileşte dacă va fi returnată o valoare numerică sau logică, în funcţie de tipul sursei de date asociate controlului. În schimb, casetele de validare pot avea şi valoarea 2. Această stare reprezintă o valoare **.NULL.**, cu alte cuvinte, nici bifată, nici nebifată. În unele aplicaţii s-ar putea să doriţi să iniţializaţi casetele de validare cu această valoare pentru a stabili dacă utilizatorul a efectuat o selecţie sau a sărit-o pur şi simplu. O casetă de validare având valoarea .**NULL.** apare sub forma unei casete umbrite.

6.2.11 Obiecte Spinner sau caseta de modificare valorică

Un alt control prezentat în figura 6.10 este caseta de modificare valorică, în cazul căreia trebuie să existe o valoare numerică drept sursă a controlului (proprietatea **ControlSource**), valoare reprezentând numărul de exemplare. Săgeţile din dreapta casetei permit utilizatorului să incrementeze sau să decrementeze valoarea afişată. În mod prestabilit, valoarea de incrementare este 1.00. Puteţi însă să înlocuiţi această valoare cu orice altă valoare, inclusiv cu valori fracţionare.

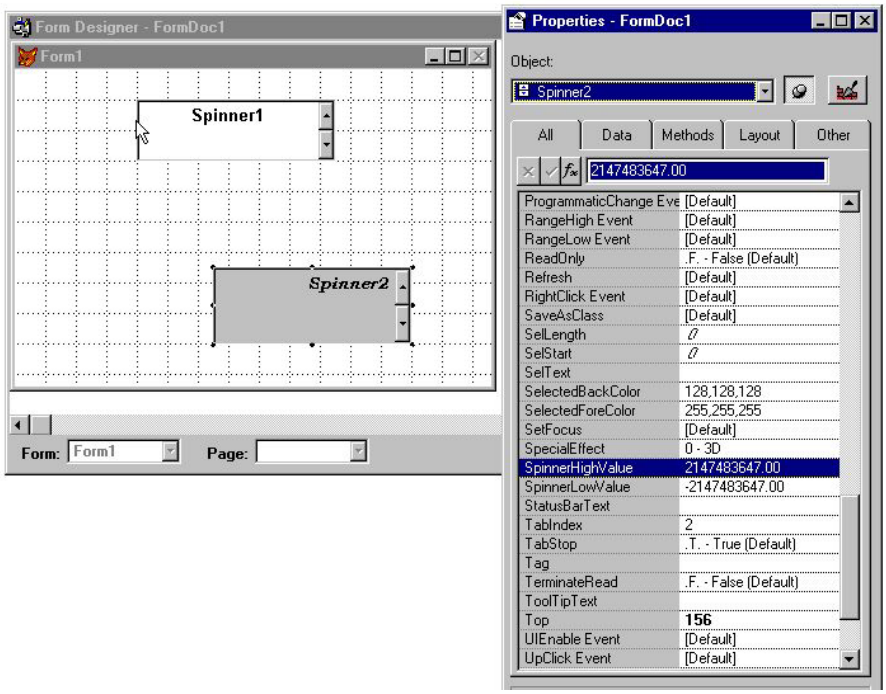


Fig.6. 10 Control spinner

Pe lângă proprietatea **Increment**, casetele de modificare valorică sunt caracterizate de un domeniu limită, definit de proprietăţile **KeyboardHighValue** şi **KeyboardLowValue**, **SpinnerHighValue** şi **SpinnerLowValue.** În mod normal, proprietăţile corespunzătoare tastaturii şi casetei de modificare valorică au aceeaşi valoare. Utilizatorul nu poate introduce de la tastatură o valoare situată în afara aestui domeniu şi nu poate utiliza săgeţile casetei pentru a ieşi în afara domeniului.

6.2.12 Obiecte Image sau obiectul imagine

Obiectul imagine este format din fişiere imagine (bmp, pcx, jpg, gif, etc) salvate sub forma unor fişiere separate. Le puteţi folosi drept embleme în formulare şî rapoarte. Nu le puteţi edita, ci doar să le afişaţi. Figura 6.11 ilustrează un formular cu obiect imagine, în care a fost utilizat fişierul WINLOGO.BMP din directorul \WINDOWS.

În mod prestabilit, atunci când VFP inserează o imagine într-un formular, dimensiunea acesteia va fi cea a imaginii bitmap originale. Desigur, acest lucru ar

putea duce la suprapunerea imaginii peste obiectele existente. Puteți fie să trageți celelalte obiecte în alte poziții pentru a face loc imaginii, fie să modificați modul în care VFP afişează imaginea, folosind proprietatea **Stretch**.

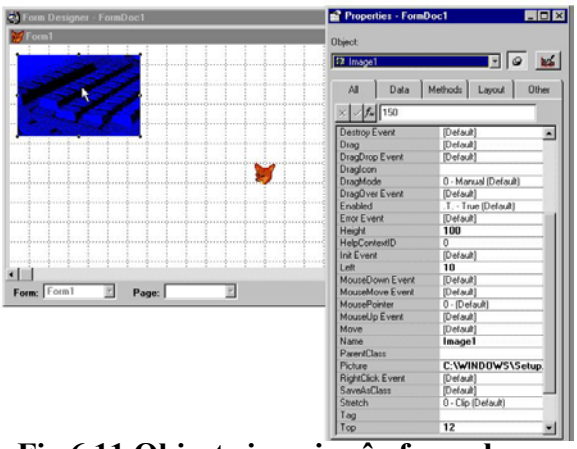


Fig 6.11 Obiecte imagine în formular

Proprietatea **Stretch** a unei imagini identifică cele trei moduri de afişare a imaginii:

0 Clip
1 Stretch
2 Zoom

Dacă selectați **Clip,** Visual FoxPro retează imaginea atunci când îi reduceți dimensiunile în raport cu originalul. Indiferent ce marcaj de selecție folosiți atunci când redimensionați imaginea, Visual FoxPro repoziționează întotdeauna imaginea în funcție de colțul din stânga-sus al acesteia (proprietățile Left şi Top). În consecință, dacă este necesară retezarea imaginii, FoxPro retează laturile din dreapta şi de jos, după caz, chiar dacă ați deplasat marcajul de selecție din colțul din stânga-sus.

A doua opțiune este **Stretch** care menține proporția relativă dintre înălțime şi lățime. În consecință, dacă modificați zona imaginii înjumătățind-o pe lățime, opțiunea **Stretch** înjumătățeşte automat şi înălțimea, păstrând în acest fel proporțiile relative ale obiectului original.

Opțiunea **Zoom** modifică modul de afişare a imaginii, astfel încât aceasta să se încadreze în noile dimensiuni ale obiectului. Această tehnică distorsionează dimensiunile relative ale imaginii, însă nu o retează. Orice dimensiune nemodificată rămâne intactă. În consecință, o imagine redimensionată prin tragerea marcajelor de selecție situate la mijlocul laturii superioare sau inferioare pare comprimată.

6.2.13 Obiecte Line sau obiectul linie

Obiectul linie vă permite să desenați linii drepte între oricare două puncte ale formularului. Pentru a desena o linie, selectați controlul şi poziționați mouse-ul în locul în care vreți să înceapă linia. Apoi executați clic şi trageți mouse-ul până la punctul final. În timp ce trageți, veți vedea o casetă reprezentând un dreptunghi

imaginar care înconjoară linia, de genul celui care apare la redimensionarea unei ferestre. Linia uneşte colţurile opuse.

Atunci când eliberaţi butonul mouse-ului, Visual FoxPro afişează o linie care începe

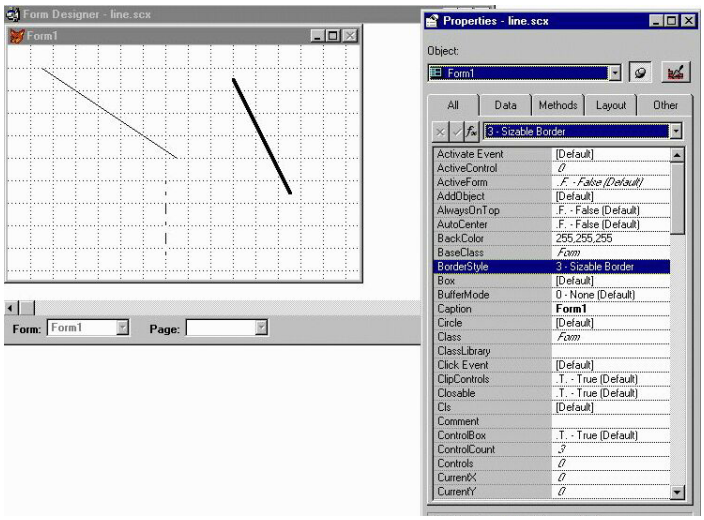


Fig.6. 12 Obiectul Line

în colţul din stânga-sus şi se termină în colţul din dreapta-jos al acestei casete. Dar dacă vreţi ca linia să înceapă în colţul din stânga-jos şi să se termine în colţul din dreapta-sus? Având linia selectată (lucru semnalat de prezenţa micilor pătrate, numite marcaje de selecţie, pe laturile şi în colţurile dreptunghiului imaginar), deschideţi pagina **Layout** a casetei de dialog Properties. În continuare, selectaţi proprietatea **LineSlant** şi schimbaţi-i valoarea din \ în /.

Puteţi, de asemenea, să schimbaţi grosimea liniei prin modificarea proprietăţii **BorderWidth.** Această valoare reprezintă grosimea liniei în pixeli.

O altă proprietate interesantă este **BorderStyle.** Ei îi corespund şapte stiluri de linie diferite, stilul prestabilit fiind cel de linie continuă. De asemenea, îi corespund 16 stiluri diferite ale creionului (moduri de desenare). Unele dintre acestea vă permit să afişaţi linii care trebuie să intersecteze alte obiecte, chiar şi imagini grafice. Figura 6.12 de mai sus ilustrează câteva dintre aceste stiluri de linie.

6.2.14 Obiecte Shape

Obiectul **Shape** permite să creaţi pătrate, dreptunghiuri, ovaluri şi cercuri. În

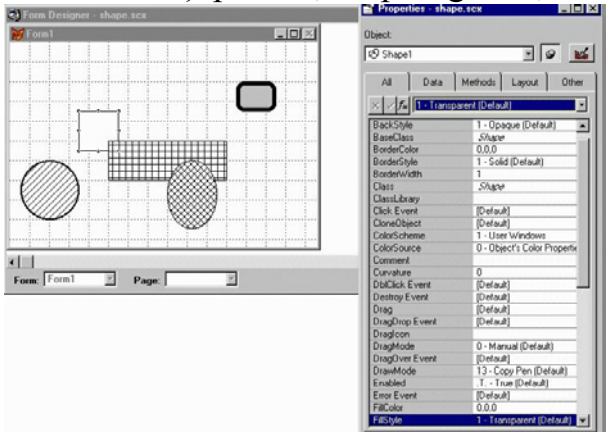


Fig. 6.13 ilustrează diferite forme geometrice

esență, procedura folosită pentru crearea formelor este asemănătoare celei utilizate pentru linii, cu câteva excepții.

Dreptunghiuri şi pătrate

Forma prestabilită este dreptunghiul. Un pătrat este un tip special de dreptunghi cu toate laturile egale. În mod similar, ovalurile şi cercurile derivă şi ele din dreptunghiuri având colțurile din ce în ce mai rotunjite.

Formele posedă şi umbrire, care în mod prestabilit este transparentă. Un obiect transparent poate avea o culoare asociată, însă fiind transparent, aceasta nu va fi afişată. În schimb, veți vedea tot ce se găseşte în spatele lui, fie că este vorba de fundalul ecranului, de un text sau de alte obiecte. Pentru a vedea culoarea asociată unui obiect, schimbați valoarea proprietății **FillStyle** din **Transparent** în unul din celelalte opt modele de umplere.

Un model de umplere opac (**FillStyle = 0**) afişează întreaga suprafață a obiectului cu culoarea (proprietatea **FillColor**) selectată. Pe de altă parte, dacă intenționați să tipăriți formularul, testați în prealabil fidelitatea cu care imprimanta reproduce diversele culori. Unele imprimante tipăresc culorile cu o calitate destul de slabă; prin urmare, va trebui să utilizați diverse modele de umplere (proprietatea FillStyle) în locul culorilor.

O caracteristică a modelelor de umplere este că pot fi făcute opace în raport cu celelalte obiecte din fundal, folosind proprietatea **BackStyle**. În cazul modelelor de umplere, VFP utilizează valoarea proprietății **BackColor** pentru definirea fundalului modelului de umplere şi valoarea proprietății **FillColor** pentru liniile modelului. În schimb, dacă atribuiți proprietății **BackStyle** valoarea **Transparent**, VFP ignoră valoarea proprietății **BackColor** şi afişează orice obiect situat în spatele obiectului în cauză.

O ultimă caracteristică specială a formelor este proprietatea **SpecialEffect**. Atribuiți acestei proprietăți valoarea 0 pentru a da chenarului un efect tridimensional. Din păcate, nu puteți controla în nici un fel direcția sursei de lumină sau lățimea umbririi. În mod prestabilit, este vorba de o formă tridimensională cu sursa de lumină venind din colțul din stânga-sus al ecranului. Pe de altă parte, efectul este foarte slab şi este vizibil numai în cazul dreptunghiurilor şi al pătratelor cu colțuri nerotunjite şi cu o grosime a chenarului de 1.

Cercuri şi ovale

Un cerc sau un oval nu este nimic altceva decât un pătrat sau un dreptunghi cu colțurile rotunjite. Practic, puteți utiliza proprietatea Curvature pentru a controla gradul de rotunjire a colțurilor dreptunghiului. Un dreptunghi perfect are curbura zero. Pe de altă parte, un oval perfect are o valoare a curburii de 99. Valorile intermediare reprezintă diverse grade de curbură a colțurilor.

6.3 Fereastra pentru secvențe de cod. Definirea metodelor

Intr-o astfel de fereastră se pot scrie secvente de program care se vor executa la apariția unor evenimente (clic pe obiect, inițializarea obiectului, etc).

Pentru modificarea unei proprietăți a unui obiect în cadrul unui formular se poate folosi construcția:

Numeformular.Numeobiect.Proprietate = valoare

Unde **numeformular** se referă prin construcție la ThisForm (acest formular)

Execuția unei metode a formularului se realizează cu construcția

Nume formă.metodă

Deschiderea unei ferestre cod se poate face, astfel:

- clic dreapta pe suprafața formularului şi selectarea opțiunii Code;
- dublu clic pe numele metodei din cadrul ferestrei Properties;
- selectarea optiunii Code din meniul View.

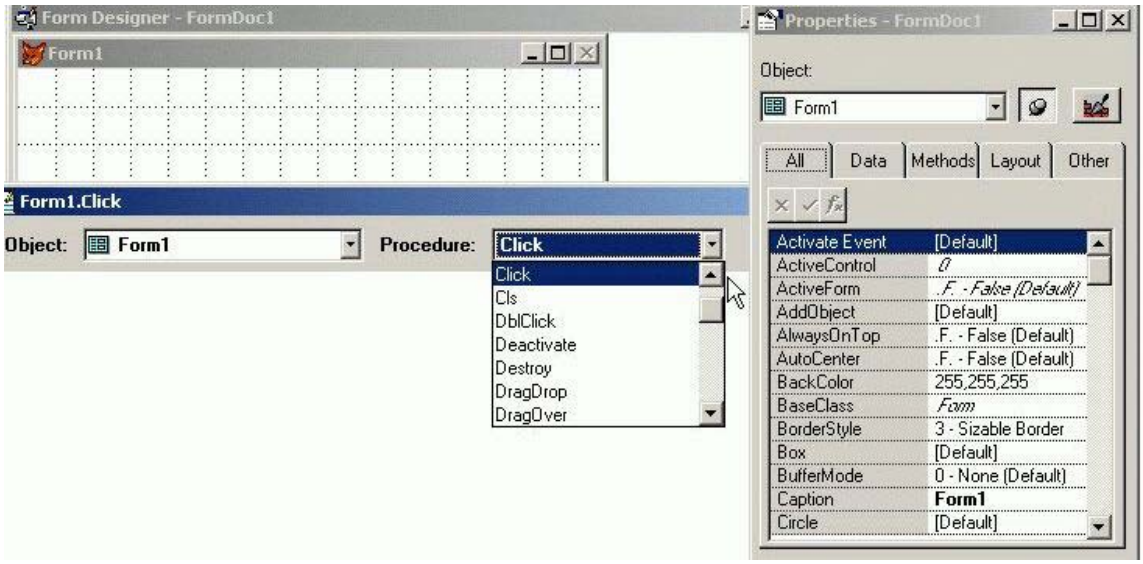


Fig.6. 12 Fereastră de cod

In interiorul acestei ferestre sunt disponibile 2 liste derulante:

- **Object** care permite selectarea unui obiect;
- **Procedure** care permite alegerea unei anumite metode.

6.4 Definirea proprietăților globale ale formularului

Prima etapă a creării unui formular o reprezintă definirea proprietăților globale ale acestuia. Deoarece toate proprietățile au valori prestabilite, este suficient să specificați proprietățile pe care vreți să le modificați. Din fericire, nu trebuie modificate decât câteva proprietăți (din totalul de 60). Printre cele mai evidente sunt:

- **Dimensiunea**
- **Poziția pe ecran**
- **Titlul**

- **Culoarea de fundal şi de prim-plan**

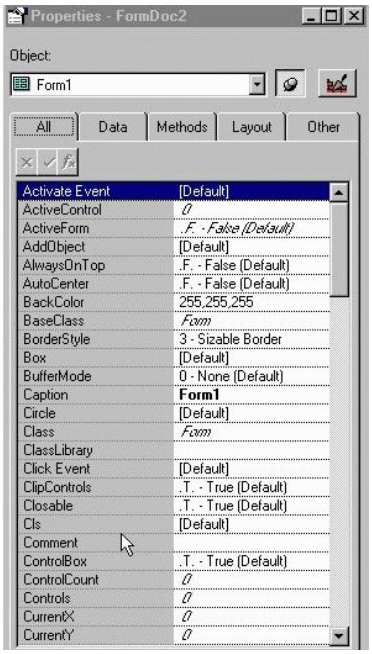


Fig.6. 14 Fereastra de proprietăți

Visual FoxPro împarte proprietățile obiectelor în cinci pagini: **Data, Methods, Layout, Other** şi **All** ce include toate proprietățile. Pentru a modifica aspectul unui formular, afişați pagina **Layout.**

Figura 6.14, ilustrează câteva proprietăți ale paginii **Layout**. Pentru a modifica

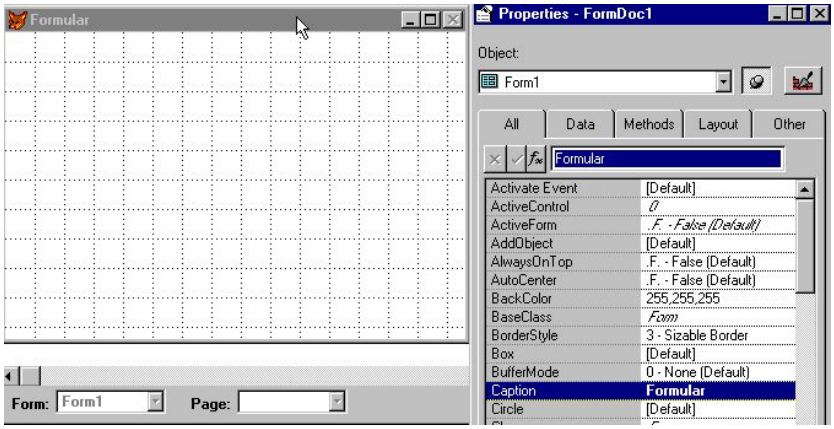


Fig.6. 15 Proprietatea Caption

o proprietate, executați clic pe ea sau puneți-o în evidență cu tastele direcționale. Visual FoxPro afişează valoarea curentă în caseta de editare din partea superioară a paginii. În figura 6.15 este pusă în evidență proprietatea **Caption**, a cărei valoare prestabilită a fost deja schimbată din Form1 în **Formular**. Remarcați
faptul că această modificare este imediat reflectată în bara de titlu a formularului propriu-zis.

În modi implicit în partea stângă sus a ferestrei **Properties** apar trei butoane:

Primul este etichetat cu un X.

El anulează modificările proprietății, cel puțin până în momentul în care apăsați ↵ sau selectați o altă proprietate.

Al doilea buton afişează un semn de validare.

Acesta verifică din punct de vedere sintactic ecuațiile folosite pentru definirea proprietății. Anumite proprietăți au un domeniu de valori limitat constând din valori logice sau valori selectate din liste predefinite (cum ar fi cazul stilului de chenar). Pentru aceste proprietăți, butonul de validare acceptă pur şi simplu noua valoare în mod asemănător apăsării tastei ↵ sau executării unui clic pe altă proprietate.

Al treilea buton deschide caseta de dialog **Expression Builder**. Folosiți acest buton pentru proprietățile care acceptă valori calculate, cum ar fi **Top, Left, Height sau Width.**

În stânga casetei de editare pot apărea o serie de butoane suplimentare, de exemplu: o săgeată cu vârful în jos indicând o listă derulantă. Multe proprietăți, cum ar fi **Border Style, Draw Style, Font**, toate proprietățile logice şi multe altele, afişează o listă derulantă. Executând clic pe acest buton, afişati valorile posibile ale proprietății. Nu puteți introduce valori într-o asemenea listă. De exemplu, proprietatea **Style Border** afişează patru stiluri numerotate de la 0 la 3. Caseta de editare a proprietății nu va accepta nici o altă valoare sau caracter. (Evident, nu ar şti ce să facă cu alte valori.) Prin urmare, spre deosebire de scrierea manuală a codului, caz în care riscați să introduceți o valoare incorectă, caseta de editare a proprietăților permite numai introducerea valorilor valide.

Câteva proprietăți, cum ar fi **ForeColor şi BackColor,** afişează un buton cu trei puncte (puncte de suspensie). Acest buton indică faptul că la apăsarea butonului este afişată o casetă de dialog cu opțiuni pentru proprietatea respectivă. Executând clic pe acest buton, deschideți caseta de dialog **Color**, care afişează o grilă cu 48 de culori predefinite. În plus, vă permite să definiți până la 16 culori personalizate.

Patru proprietăți definesc pozitia şi dimensiunea majorității obiectelor vizuale, inclusiv ale formularelor. **Top şi Left** poziționează colțul din stânga-sus al obiectului. În mod prestabilit, Visual FoxPro plasează un nou formular în colțul din stânga-sus al ecranului (Top = 0 şi Left = 0). Puteți fie să-i schimbați poziția folosind aceste două proprietăți, fie să mutați fizic obiectul în fereastra **Form Designer**. Puteți, de asemenea, să folosiți proprietatea **AutoCenter** pentru a centra automat formularul. În mod similar, proprietățile **Height** şi **Width** definesc dimensiunile formularului.

S-ar putea să doriți să centrați formularul pe o singură direcție, pe orizontală sau pe verticală. Nu există nici un buton şi nici o opțiune pentru a face acest lucru. Atribuiți pur şi simplu proprietății **Left** sau **Top** (după caz) expresia uneia dintre următoarele expresii:

Top: =(_Screen.Height – This.Height) / 2

sau

Left: =(_Screen.Width - This.Width) / 2.

Aceste expresii se bazează pe posibilitatea de a face referire la oricare dintre proprietățile obiectului. Suprafața de lucru FoxPro are întotdeauna numele de obiect SCREEN şi are aceleaşi proprietăți referitoare la poziție şi la dimensiuni ca şi un formular. Expresia utilizează cuvântul rezervat de referire relativă "This" pentru a face referire la proprietățile **Width** şi **Height** ale obiectului.curent. “This” se referă la formularul care este proiectat.

La baza ferestrei Properties, apare în majoritatea cazurilor, o scurtă descriere a proprietății.

6.5 Mediul de date

Majoritatea formularelor trebuie să facă referire la datele din unul sau mai multe tabele. Pentru a include tabelele necesare în definiția formularului deschideți fereastra **View**, **Data Environment**. Această fereastră seamănă cu Table View a instrumentului Database Designer.

Pentru adăugarea de tabele în zona de lucru Data Environment, deschideți meniul derulant **Data Environment**, care apare în meniul de sistem după selectarea proprietății. Inițial, este activat numai butonul **Add**. Selectați-l şi alegeți un tabel din caseta de dialog **Add Table sau View.** Sau executați clic pe butonul drept în fereastra Data Environment şi să selectați Add din meniul derulant.

Pentru a stabili o relație între mai multe tabele, executați clic pe un câmp al tabelului principal şi trageți-l în tabelul de căutare. În fereastra **Data Environment** apare o linie care uneşte câmpul tabelului principal cu indexul corespunzător din tabelul de căutare, după cum se poate vedea în figura 6.16. Este de semnalat faptul că totuşi trebuie să existe dinainte un index corespunzator.

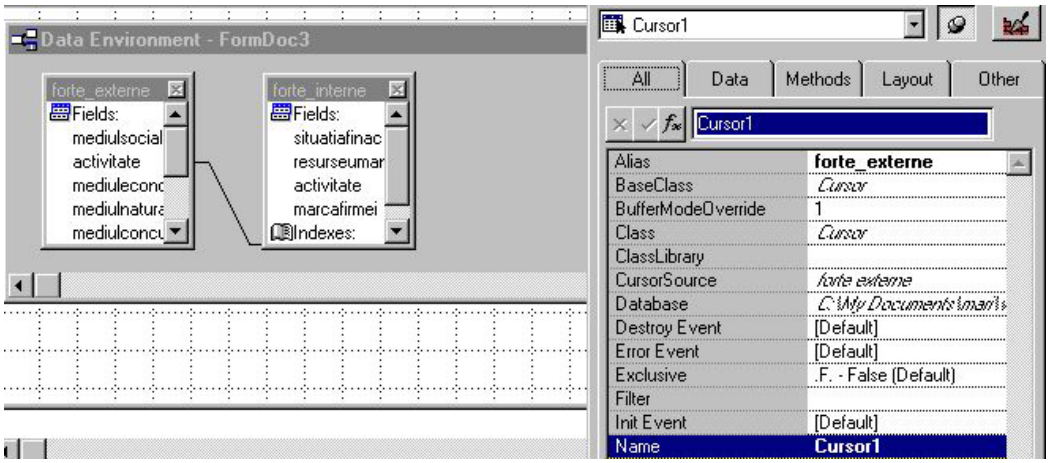


Fig.6. 16 Legătura între două tabele folosind Data Environment

6.6 Adăugarea obiectelor într-un formular

După ce am definit mediul de date putem adăuga obiecte în formularul nostru. Visual FoxPro acceptă mai multe tipuri de obiecte. Unele obiecte nu solicită nici o acţiune din partea utilizatorului odată ce au fost definite. Acestea sunt liniile, formele şi etichetele. Celelalte obiecte trebuie să fie legate la anumite câmpuri din tabelele sau vederile specificate în mediul de date. Există obiecte care leagă Visual FoxPro cu alte fişiere sau aplicaţii.

De exemplu, obiectul **Image** afişează o imagine bitmap creată de altă aplicaţie şi stocată separat sub forma unui fişier BMP. Obiectele OLE vă permit să lucraţi cu documente din alte aplicaţii şi să le includeţi în aplicaţia dumneavoastră.

Pentru a adăuga un obiect într-un formular, trebuie să deschideţi bara cu instrumente **Form Controls**, dacă nu este deja deschisă. Pentru a o deschide, selectaţi opţiunea **Toolbar** din meniul derulant **View** şi alegeţi **Form Control.**

6.7 Adăugarea manuală a obiectelor

Pentru a adăuga orice obiect într-un formular, executaţi clic pe pictograma acestuia după care executaţi clic pe formular. Visual FoxPro afişează obiectul cu dimensiunea prestabilită, definită de clasa de bază. Pentru unele obiecte, cum ar fi casetele de text sau cele de editare; va trebui să definiţi dimensiunea în momentul în care le plasaţi. În asemenea cazuri, după selectarea obiectului, executaţi clic cu mouse-ul în formular şi trageţi pentru a crea un dreptunghi reprezentând dimensiunea obiectului. Atunci când eliberaţi butonul mouse-ului, Visual FoxPro atribuie obiectului dimensiunile acestui dreptunghi.

6.7.1 Utilizarea instrumentului Builder pentru adăugarea manuală a obiectelor

Dacă plasaţi obiectele pe formular în maniera descrisă anterior, va trebui să definiţi manual toate proprietăţile acelui obiect. Pe de altă parte, Visual FoxPro vă pune la dispoziţie un instrument numit **Builder** (generator) care vă ajută să

Fig 6.17 Pictograma Builder Lock

configuraţi proprietăţile importante ale fiecărui obiect.

Pictograma **Builder Lock (**Fig 6.17**)**seamănă cu o baghetă magică. Trebuie doar să executați clic pe ea înainte de a selecta obiectul pe care vreți să-l adăugați. În continuare, instrumentul rămâne selectat până când executați din nou clic. Puteți, de asemenea, să lansați un generator executând clic cu butonul drept pe obiect şi selectând opțiunea **Builder** din meniu.

Exemplu 6.4

Să se folosească generatorul **Builder** pentru a adăuga un grup de butoane de de tip radio.

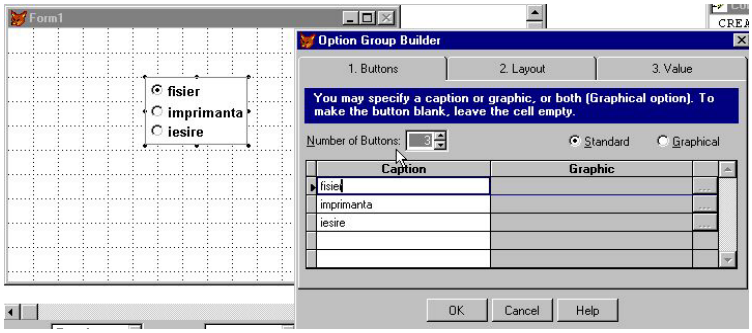


Fig.6. 18 Prima pagina din Group Builder

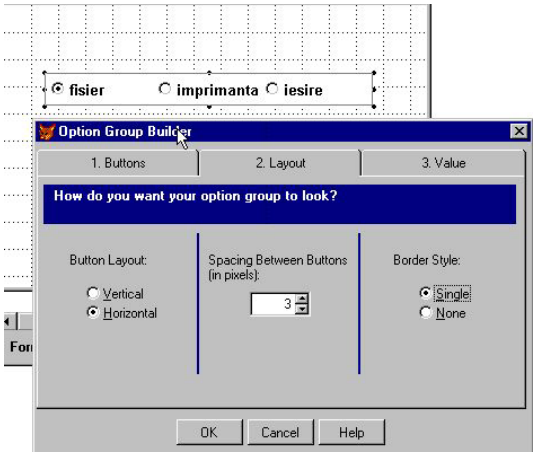


Fig.6. 19 Pagina a doua din Group Builder

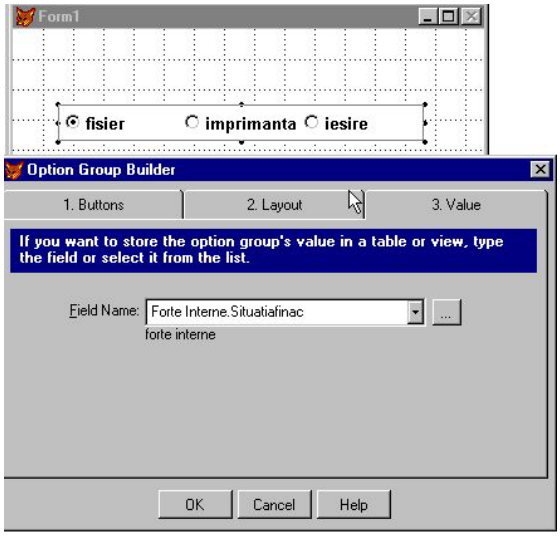


Fig.6. 20 Pagina a treia din Group Builder

Explicaţii

A) Întâi executaţi clic pe butonul **Builder Lock** şi apoi alegeţi opţiunea dorită. În continuare, plasaţi grupul de butoane în formular executând clic o dată şi trăgând pentru a-l dimensiona. Visual FoxPro afişează instrumentul **Option Group Builder**. Acesta are trei pagini, prima dintre ele fiind prezentată în figura 6.18. Această pagină defineşte numărul de butoane, titlurile şi tipul lor. În cazul de faţă, cele trei butoane au fost definite ca butoane standard cu titlurile: Fisier, Imprimanta şi Iesire. Atunci când introduceţi numele titlurilor sau schimbaţi numărul de butoane, puteţi vedea imediat aceste modificări în formularul propriu-zis.

B) În cea de a doua pagină (vezi figura 6.19) definim configuraţia butoanelor cu ajutorul a trei opţiuni. În primul rând, puteţi plasa butoanele pe verticală sau pe orizontală. În afara cazului în care aveţi puţine butoane, opţiunea cea mai uzuală este plasarea pe verticală. În al doilea rând, puteţi schimba spaţierea butoanelor, exprimată în foxeli. Şi, în sfârşit, puteţi defini stilul chenarului din jurul grupului de butoane. Aceasta înseamnă fie afişarea unei casete cu o singură linie, fie renunţarea la chenar.

C) În ultima pagină generatorul vă întreabă dacă doriţi să salvati valoarea care identifică butonul selectat de utilizator. Aşa cum rezultă din figura 6.20 În general, este vorba de un câmp al unui tabel sau al unei vederi, însă poate fi şi o variabilă de memorie. Dacă selectaţi Yes, puteţi selecta atât tabelul sau vederea, cât şi câmpul. Pentru a selecta tabelul sau vederea, executaţi clic pe butonul cu cele trei puncte. Este afişată caseta de dialog Open. După selectarea unui tabel, executaţi clic pe săgeata cu vârful în jos a casetei cu listă derulantă pentru a afişa câmpurile tabelului sau ale vederii. Selectaţi din listă câmpul dorit sau introduceţi direct numele unei variabile de memorie.

Fiecare generator vă ajută să definiţi diverse proprietăţi ale obiectului asociat. Unele generatoare, cum ar fi **ComboBox Builder sau List Builder**, necesită parcurgerea a până la patru cadre de pagină pentru definirea completă a obiectului. Altele necesită parcurgerea a numai două pagini. În ambele cazuri, generatorul vă ajută să definiţi suficiente proprietăţi pentru a crea un obiect funcţional. Ulterior, puteţi oricând reveni în caseta de dialog **Properties** pentru a configura alte atribute ale obiectului, cum ar fi culorile sau fonturile.

Puteţi, de asemenea, să reveniţi la generator pentru obiecte deja existente pe ecran. Pur şi simplu selectaţi obiectul pe care vreţi să-l treceţi în revistă şi executaţi clic pe butonul **Builder** din colţul din dreapta sus al casetei de dialog **Properties**.

6.8 Proprietăţile tipice ale obiectelor

Orice obiect are proprietăţi care îi definesc poziţia şi dimensiunile. Visual FoxPro poziţionează obiectele folosind coordonatele colţului din stânga-sus al acestora. Iată de ce proprietăţile corespunzătoare se numesc **Top şi Left**. Pentru

definirea dimensiunilor, fiecare obiect are o înălțime (**Height**) şi o lățime (**Width**). Rețineți faptul că înălțimea se măsoară de sus în jos.

Poziția şi dimensiunile sunt exprimate în unitățile de măsură definite de proprietatea **ScaleMode**. Există două posibilități: pixeli sau foxeli. **Pixelii** sunt cei mai uşor de înțeles. Fiecare pixel reprezintă un punct colorat pe ecran. Atunci când folosiți pixeli, poziționarea textului nu depinde de dimensiune sau de alte caracteristici ale fontului, ci se reduce la o simplă numărare a pixelilor.

Pe de altă parte, Visual FoxPro defineşte un **foxel** ca fiind dimensiunea medie a unui caracter. În cazul fonturilor neproporționale, cum ar fi Courier, dimensiunea medie a caracterelor este identică cu dimensiunea oricărui caracter. În schimb, atunci când se folosesc seturi de caractere proporționale, dimensiunea medie de un **foxel** nu este riguros egală cu dimensiunea nici unui caracter. Mai mult, dimensionarea şi poziționarea obiectelor se complică deoarece nici un set aleator de cinci caractere nu va încăpea în mod necesar într-o casetă de text definită cu o lățime de cinci foxeli.

În cazul obiectelor plasate într-un formular folosind ca unități de măsură **foxelii,** sunt utilizate caracteristicile fontului prestabilit pentru stabilirea poziției şi a dimensiunilor lor.

Visual FoxPro vă oferă o serie de funcții care vă ajută să stabiliți caracteristicile fontului. Prima funcție este **FONTMETRIC()** care returnează diverse atribute despre orice font din sistem. În tabelul 6.1 sunt enumerate informațiile pe care le puteți obține cu această funcție.

Tabelul 6.1

Atribut	Descriere
1	Înălțimea caracterelor în pixeli
2	Porțiunea ascendentă (numărul de unități cu care depăşeşte linia de bază) în pixeli
3	Porțiunea descendentă (numărul de unități de sub linia de bază) în pixeli
4	Interlinierea (spațiul dintre linii) în pixeli
5	Interlinierea suplimentară în pixeli
6	Lățimea medie a caracterelor în pixeli
7	Lățimea maximă a caracterelor în pixeli
8	Grosimea fontului
9	Italic (0=Nu, diferit de zero=Da)
10	Subliniat (0=Nu, diferit de zero=Da)
11	Tăiat (0=Nu, diferit de zero=Da)
12	Primul caracter definit al fontului
13	Ultimul caracter definit al fontului
14	Caracterul prestabilit (care înlocuieşte caracterele pe care fontul nu le conține)
15	Caracterul de despărțire a cuvintelor
16	Distanța între caractere şi familia
17	Setul de caractere
18	Lățire (lățime suplimentară)
19	Aspectul orizontal pentru fontul de dispozitiv
20	Aspectul vertical pentru fontul de dispozitiv

Pe lângă atributul fontului pe care vreți să-l obțineți, trebuie să specificați numele fontului, dimensiunea în puncte şi codul stilului. Sintaxa funcției FONTMETRIC() este următoarea:

FONTMETRIC (<atributul fontului>, <numele fontului>,<marimea in puncte>, <codul stilului>)

Exemplul 6.5

Să se calculeze lătimea medie a caracterelor unui font Arial, aldin, cu o dimensiune de 14 puncte.

? FONTMETRIC(6, 'ARIAL', 14, 'B')

La scrierea acestei expresii, s-a presupus că se cunoaşte fontul, dimensiunea în puncte şi stilul. În cazul unui program real, nu puteți face această presupunere. În consecință, trebuie să folosiți o altă funcție, **WFONT().**

Funcția **WFONT()** returnează una din trei informații posibile pentru orice fereastră activă, în funcție de valoarea primului parametru. Valorile posibile sunt:

1 numele fontului
2 dimensiunea fontului
3 stilul fontului

Sintaxa generală a comenzii este următoarea:

WFONT(<atribut>, <numele ferestrei>)

Exemplul 6.6

Se presupune că numele formularului deschis este **FORM1**, şi se cer informațiile referitoare la lățimea medie a caracterelor .

? FONTMETRIC (6, WFONT (1, 'FORM1'), WFONT (2, 'FORM1'), WFONT(3, ; 'FORM1'))

Exemplul 6.7

Se cere să creați o casetă de text de 10 caractere, folosind un font neproporțional, cum ar fi Courier. Această casetă de text va apărea într-un formular care utilizează un font Arial de 10 pt, normal.

= 10 * FONTMETRIC (6, 'Courier New', 10, 'N') / FONTMETRIC (6, WFONT(1,; 'FORM1'), WFONT(2, 'FORM1'), WFONT(3, 'FORM1'))

Explicații

A) Se calculează lătimea necesară pentru acest obiect. Această ecuație calculează numărul de pixeli necesari pentru 10 caractere cu font Courier New de 10 pt, normal. În continuare, împarte această valoare la numărul

mediu de pixeli ai unui caracter din fontul formularului. Rezultatul indică numărul de foxeli pe care trebuie să-i aibă lățimea casetei de text.

6.9 Adăugarea noilor proprietăți şi metode

Puteți adăuga noi proprietăți unui formular, dacă acesta este deschis. Selectați întâi formularul, după care deschideți meniul derulant **Form**. Prima opțiune a meniului adaugă o nouă proprietate, iar a doua adaugă o nouă metodă. Selectând opțiunea **New Property**, se va afişa caseta de dialog prezentată în figura 6.21.

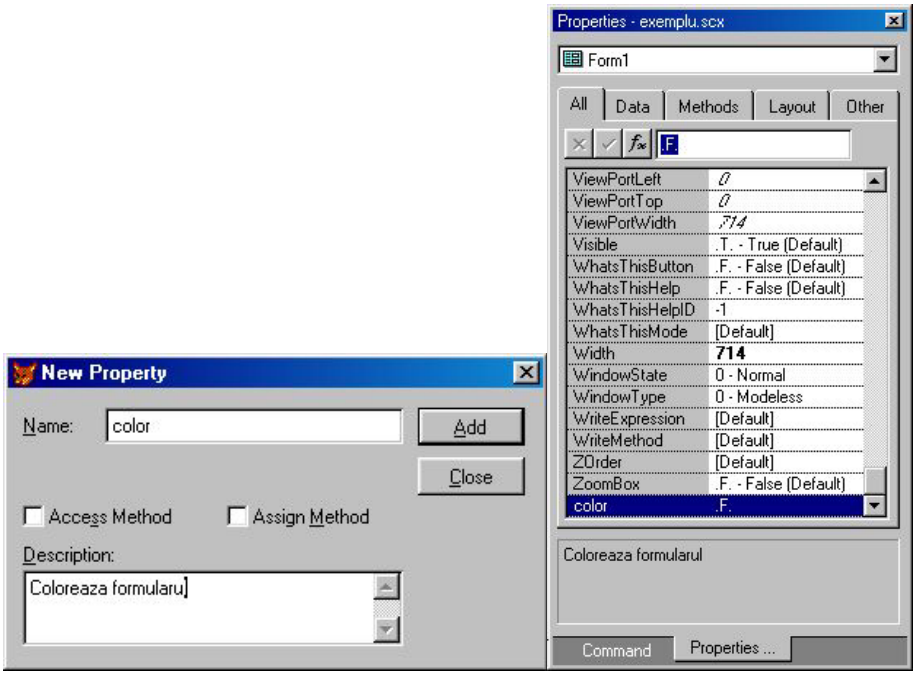


Fig.6. 21 Optiunea New Property

Remarcați faptul că deocamdată nu puteți specifica decât numele proprietății şi descrierea. După ce introduceți aceste informații şi executați clic pe OK, VFP adaugă proprietatea în caseta de dialog **New Property**, la sfârşitul paginilor **All** sau **Other.** Remarcați faptul că valoarea prestabilită a propietății este .F. Puteți înlocui această valoare cu oricare alta, executând clic pe ea şi introducând o valoare prestabilită în caseta de editare a proprietății. De asemenea, atunci când noua proprietate este pusă în evidență, descrierea introdusă anterior apare în zona de descrieri situată la baza casetei de dialog **Properties**.

Opțiunea **Form, Edit Property/Method**... vă permite să modificați descrierea unei proprietăți, însă nu şi ortografia. Dacă faceți o greşeală şi vreți să modificați ortografia unei noi proprietăți sau metode, trebuie să o eliminați şi să o adăugați din nou.

Dacă adăugați o metodă, ea apare în caseta de dialog Properties Puteți să deschideți codul asociat formularului executând dublu clic pe formular sau selectând **View, Code**.

6.10 Definirea metodelor

Programarea condusă de evenimente înseamnă că utilizatorul este acela care controlează programul. Practic, programul stă pur şi simplu şi aşteaptă ca utilizatorul să facă ceva. Acest ceva poate fi apăsarea unei taste sau un clic cu mouse-ul. Odată declanşat evenimentul, programul poate rula o singură metodă sau poate executa mai multe. De exemplu, executarea unui clic cu un buton al mouse-ului declanşează un simplu eveniment clic. În schimb, acest eveniment poate determina programul să încarce un formular. Există o serie de evenimente asociate încărcării formularelor, cum ar fi **INIT şi WHEN** atât pentru formular, cât şi pentru obiectele acestuia.

Acelaşi lucru este valabil şi atunci când vreţi să definiţi o acţiune pentru un

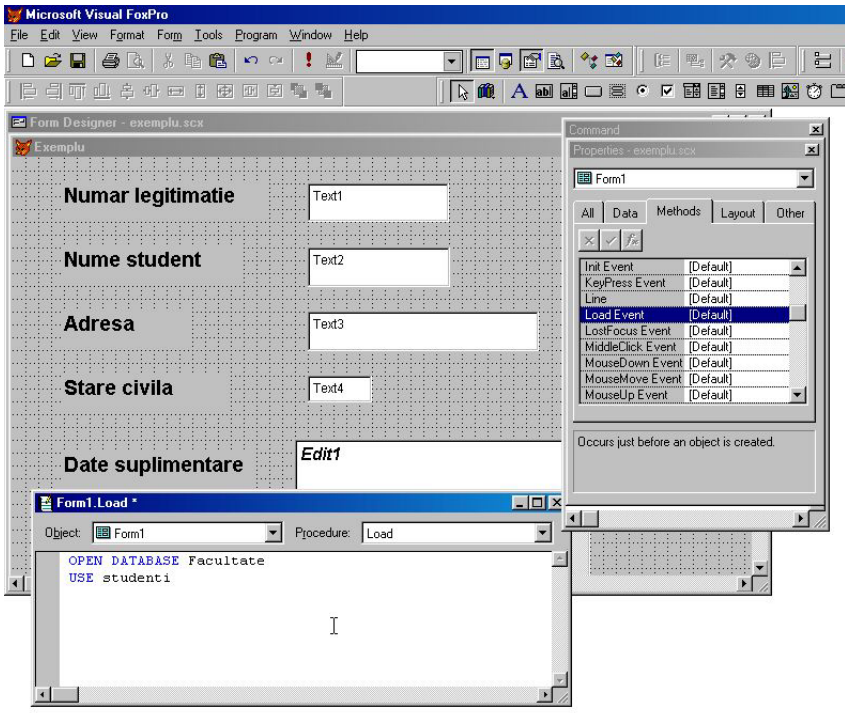


Fig.6.22 Definirea unei metode

obiect al formularului. Vom considera un formular existent care vă permite să introduceţi personal. În continuare, programul foloseşte valoarea selectată pentru a afişa o listă a candidatilor admisi. Figura 6.22 de mai sus ilustrează formularul deschis. Acest formular constă din mai multe obiecte. O metodă are loc în formularul însuşi. Atunci când formularul se deschide, el va trebui să deschidă baza de date Facultate, astfel încât să poată accesa tabelul ce are denumirea Studenti, ca şi baza de date. Puteti realiza acest lucru cu caseta de dialog **Data Environment**, însă ca să vedem cum arată codul unui eveniment, ne vom ocupa de evenimentul **Load** al formularului.

Pentru a deschide codul, puteţi să executaţi dublu clic pe evenimentul **Load** din lista **Methods** a casetei de dialog **Properties**, pentru obiectul Form1. De asemenea, puteţi să executaţi clic cu butonul drept oriunde pe formularul propriu-zis (nu pe un alt obiect) şi să selectaţi **Code** din meniul derulant

Figura 6.22 prezintă şi caseta de dialog care apare. Ea are două casete cu listă în partea superioară, iar dedesubt, o zonă albă. Caseta cu listă **Object** vă permite să selectaţi orice obiect care face parte la momentul curent din formular sau din setul de formulare, inclusiv formularul propriu-zis. Caseta cu listă **Procedure** afişează procedurile şi metodele disponibile, pe care le puteţi defini pentru obiectul respectiv. În cazul de faţă, asiguraţi-vă că în caseta cu listă Object este afişat Form1, iar în caseta cu listă **Procedure** este afişat **Load**, unde este scris textul următor:

```
* Deschide baza de date Facultate şi selecteaza tabelul Studenti
CLOSE DATABASES
OPEN DATABASE Facultate
USE Studenti
```

Acesta este întregul cod necesar pentru deschiderea tabelului Personal pentru această aplicaţie simplă. Este, de asemenea, singurul cod introdus în obiectul Form1.

6.11 Alinierea obiectelor

Chiar dacă este activată opţiunea **Snap-to-Grid** din meniul derulant **Format**, s-ar putea să nu fiţi mulţumit de modul în care sunt aliniate obiectele de pe formular. Pentru a le realinia pe toate, deschideţi bara cu instrumente **Layout** selectând-o din meniul derulant **View**. Ea conţine 13 opţiuni pentru alinierea obiectelor, opţiuni descrise în tabelul 6.2.

Tabelul 6.2

Control	Pictogramă	Descriere
Align Left Sides		Aliniază obiectele selectate la marginea cea mai din stânga
Align Right Sides		Aliniază obiectele selectate la marginea cea mai din dreapta
Align Top Edges		Aliniază obiectele selectate la marginea situată cel mai sus
Align Bottom Edges		Aliniază obiectele selectate la marginea situată cel mai jos.
Align VerticalCenters		Aliniază centrele obiectelor selectate după o axă verticală.
Align Horizontal Centers		Aliniază centrele obiectelor selectate după o axă orizontală.
Same Width		Ajustează lăţimile obiectelor selectate aducându-le la valoarea celui mai lat.
Same Height		Ajustează înălţimile obiectelor selectate aducându-le la valoarea celui mai înalt.
Same Size		Ajustează dimensiunile obiectelor selectate aducându-le la dimensiunea celui mai mare.
Center Horizontally		Aliniază centrele obiectelor selectate după o axă verticală care trece prin centrul formularului

Center Vertically		Aliniază centrele obiectelor selectate după o axă orizontală care trece prin centrul formularului
Bring to Front		Plasează obiectele selectate în fața tuturor celorlalte obiecte
Send to Back		Plasează obiectele selectate în spatele tuturor celorlalte obiecte

Înainte de selectarea unui stil de aliniere, selectați obiectele pe care vreți să le aliniați. Le puteți alinia relativ la o latură sau o axă de simetrie. Puteți, de asemenea, să centrați obiectele pe orizontală sau pe verticală relativ la formular.

Dacă selectați un set vertical de obiecte, nu încercați să le centrați pe verticală relativ la formular. Visual FoxPro centrează fiecare obiect în mod individual. Rezultatul este că Visual FoxPro va plasa obiectele unele peste altele. Aceleaşi precauții trebuie avute în vedere şi în cazul obiectelor dispuse pe orizontală. Nu încercați să le centrați pe orizontală.

Puteți, de asemenea, să redimensionați un grup de obiecte pentru a le aduce pe toate la aceeaşi lățime, aceeaşi înălțime sau ambele.

În sfârşit, dacă obiecte se suprapun (cazul liniilor şi al formelor care se suprapun peste alte câmpuri), puteți modifica ordinea de suprapunere cu opțiunile **Send to Back şi Bring to Front.** Aceste două opțiuni există şi în meniul derulant **Format.**

Folosiți bara cu instrumente **Layout** pentru a ordona obiectele formularului după ce le-ați adăugat pe toate. Un formular cu obiecte de aceeaşi dimensiune şi aliniate de-a lungul unei laturi este mai atractiv decât unul în care toate obiectele par să fie dispuse aleator.

6.12 Definirea ordinii de parcurgere cu tasta Tab

Nu întotdeauna utilizarea mouse-ului pentru deplasarea de la un câmp la următorul este cea mai convenabilă metodă. Dacă utilizatorii trebuie să introducă date în majoritatea câmpurilor sau chiar în toate, ei nu vor dori să-şi ridice mâinile de pe tastatură după completarea fiecărui câmp pentru a selecta câmpul următor. Ar fi preferabil să poată apăsa tastele **Enter sau Tab** pentru a comunica programului Visual FoxPro că au completat câmpul curent şi sunt gata să treacă la următorul.

În mod prestabilit, Visual FoxPro defineşte ordinea de parcurgere a câmpurilor cu tasta ⇆ ca fiind ordinea în care adăugați câmpurile în formular. Majoritatea utilizatorilor aşteaptă ca deplasarea cursorului prin câmpuri să se facă fie pe verticală, fie pe orizontală. Dacă apăsarea tastei Tab⇆ deplasează cursorul dintr-o parte în cealaltă a formularului, se pierde timp pentru a localiza următorul câmp de introducere a datelor.

Există două metode de modificare a ordinii de parcurgere: interactiv şi cu ajutorul unei liste. Puteți selecta metoda dorită alegând **Tools, Options, Forms** şi deschideți lista derulantă **Tab Ordering** pentru a selecta metoda dorită: **Interactive sau By List.**

Dacă optați pentru schimbarea interactivă (figura 6.23) a ordinii de parcugere, selectați **View**, **Tab Order**. În colțul din stânga-sus al fiecărui obiect este afişată o

casetă micuță, iar în interiorul fiecărei casete se află un număr reprezentând secvența de parcurgere cu tasta ⇆.

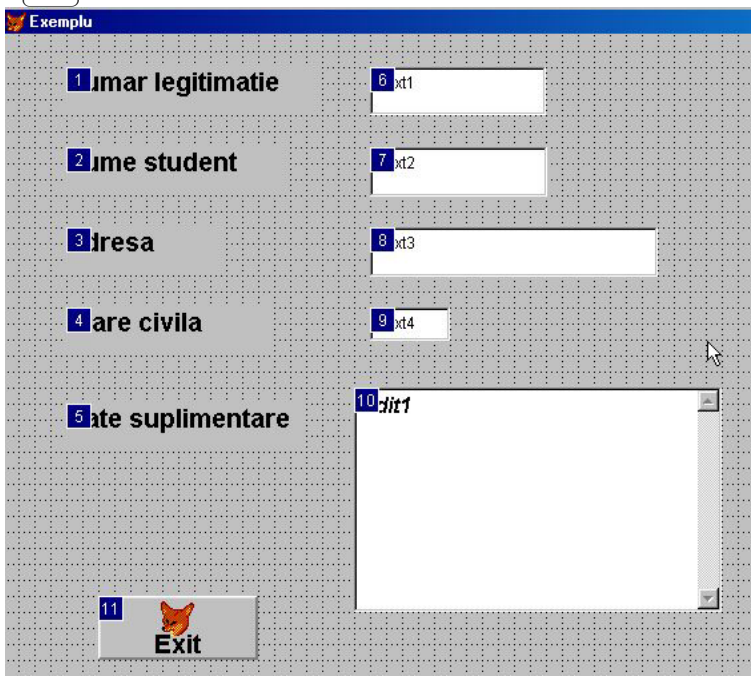


Fig.6. 23 Parcurgere cu tasta Tab interactivă

Pentru a schimba secvența de parcurgere cu tasta ⇆, mențineți tasta ⇧ Shift apăsată şi executați clic pe caseta obiectului pe care vreți să-l modificați. Aceasta duce la eliminarea numărului şi la renumerotarea tuturor celorlalte obiecte. Dacă executați încă o dată ⇧ Shift+clic pe aceeaşi casetă, obiectul este plasat la sfârşitul secvenței

O tehnică mai rapidă presupune executarea unui clic pe primul obiect al secvenței de parcurgere, fără apăsarea tastei ⇧ Shift. Aceasta duce la dezactivarea tuturor numerelor şi plasează cifra 1 în caseta obiectului curent. În continuare, folosind metoda ⇧ Shift+clic, executați clic pe fiecare câmp în ordinea în care vreți ca utilizatorii să le completeze. Când ați terminat, executați clic pe butonul **Reorder**.

O altă metodă de a stabili ordinea de parcurgere este cu ajutorul unei liste (By List din **Options, Forms**. Acum, când selectați **View**, **Tab Order**, apare caseta de dialog prezentată în figura 6.24.

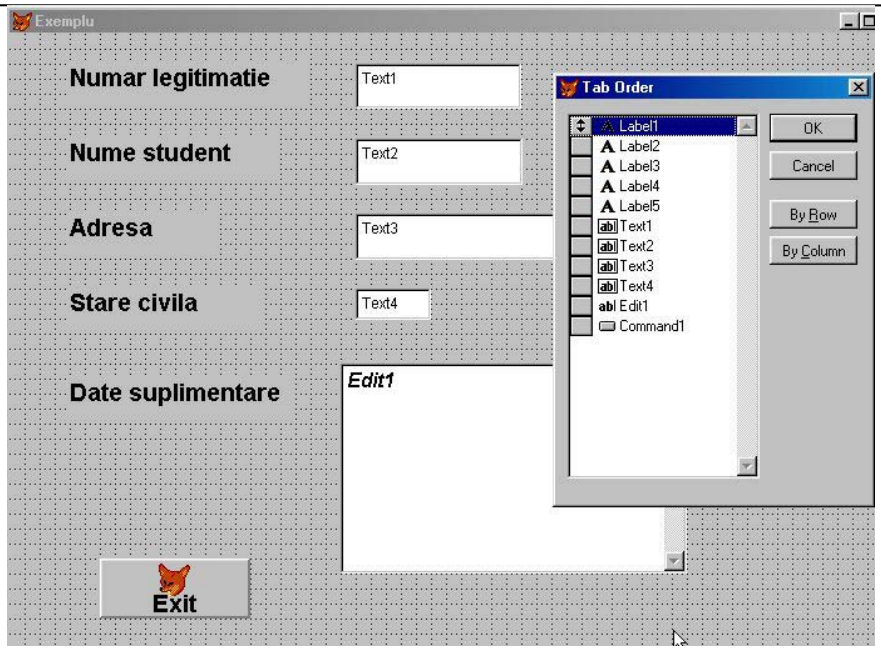


Fig.6. 24 Ordine de parcurgere cu Tab cu ajutorul unei liste

Fiecare obiect din formular apare în lista derulantă din stânga. Executând clic pe butoanele de deplasare din stânga numelor obiectelor şi trăgându-le, puteți adapta ordinea de parcurgere după cum doriți. Observați că, Visual FoxPro afişează, o pictogramă, reprezentând tipul obiectului, pentru a vă ajuta să identificați obiectele. Puteți, de asemenea, să selectați butoanele **By Row sau By Column** pentru a defini rapid ordinea câmpurilor. Totuşi, în cazul multor ecrane complexe, aceste opțiuni s-ar putea să nu dea rezultatele dorite. După se ați fixat ordinea de parcurgere dorită, executați clic pe butonul OK.

6.13 Metode echivalente clauzelor comenzii READ

Un obiect este citit într-un formular. Transferul de la un obiect la altul se poate face folosind tasta Tab. Comanda **READ** poate exista până apare o comandă **CLEAR EVENTS** sau o proprietate **TerminateRead** are valoarea .T.

Utilizatorii activează formularele fie executând clic pe obiectul formular, fie apelând metoda **SHOW** a formularului.

Visual FoxPro foloseşte metoda **Show** atât pentru a deschide formularul, cât şi pentru a stabili dacă acesta este nemodal sau modal. Comanda următoare afişează formularul nemodal, ceea ce înseamnă că atunci când formularul este deschis, execuția codului continuă:

```
Form1.Show (0)
F1 = CREATEOBJ('Test_fer')
F1.Caption =' Acesta este un formular de testare'
F1.Show (0)
= MessageBox ('Testare program!')
DEFINE CLASS test_fer AS FORM
```

```
    ADD OBJECT Exit AS COMMANDBUTTON ;
      WITH Caption = ‘QUIT’ ;
           Top = 5
           Left = 10
           Height = 2
           Width = 10
     PROCEDURE Exit.CLICK
       ThisForm.Release
     ENDPROC
ENDDEFINE
```

Comanda Form1.Show(1) deschide formularul MODAL. În acest fel, i se interzice utilizatorului să comute la o altă fereastră şi se suspendă continuarea execuției programului, până în momentul în care formularul este închis.

```
F2 = CREATEOBJ(‘Test_fereastra')
F2.Caption = ‘Aceasta fereastra este MODAL'
F2.Show(1)
= MessageBox (`Testare program!')
DEFINE CLASS Test_fereastra AS FORM
   ADD OBJECT Quit AS COMMANDBUTTON ;
      WITH Caption = `QUIT'
           Top = 5
           Left = 10
           Height = 2
           Width = 10
     PROCEDURE Quit.CLICK
       ThisForm.Release
     ENDPROC
ENDDEFINE
```

Dacă nu furnizați nici un parametru metodei Show, VFP utilizează valoarea proprietății WindowType.

Metoda **Show** *a unui formular afişează formularul*. Atunci când apelați metoda **Refresh** a formularului, aceasta reîmprospătează formularul şi toate obiectele acestuia. Comanda este:

Form1.Refresh

Acest lucru înseamnă reapelarea formularului şi a obiectelor acestuia, precum şi actualizarea valorilor obiectelor.

Să presupunem însă că s-a modificat valoarea unui singur obiect. Reapelarea întregului formular durează prea mult. O metodă mai rapidă presupune apelarea metodei **Refresh** numai pentru obiectul respectiv, după cum urmează:

Nume_formular.nume_obiect.REFRESH

Pentru a abandona un formular se poate folosi comanda

Nume_formular.RELEASE

În lumea de obiecte a mediului Visual FoxPro, metoda **SetFocus** este utilizată în scopul activării unui obiect. Fiecare obiect care permite interacţiunea utilizatorului acceptă această metodă. Pentru a vă abate de la ordinea prestabilită de parcurgere cu tasta Tab, lansaţi pur şi simplu comanda:

ThisForm.text1.SetFocus

Această comandă cedează focalizarea unui obiect numit Text1.

Obiectul **Timer** poate fi utilizat pentru simularea efectului comenzii TIMEOUT.

Similar clauzei **Lock** a comenzii **READ**, Visual FoxPro introduce proprietatea **BufferMode**. Această proprietate stabileşte când vor fi blocate înregistrările, după cum este definită în tabelul 6.3.

Tabelul 6.3

Valoare	Metodă
0	Nici una: Înregistrărite sunt blocate şi câmpurile sunt scrise atunci când sunt editate.
1	Pesimistă: Înregistrările sunt blocate de îndată ce începeţi să editaţi orice câmp. Câmpurile sunt scrise în tabel în momentul deplasării indicatorului de înregistrări.
2	Optimistă: Blochează înregistrările numai atunci când se încearcă scrierea datelor câmpurilor în tabel.

Proprietăţile unui formular nu permit specificarea unei scheme de culori. În schimb, ele permit stabilirea culorilor de prim-plan, de fundal şi de umplere. Pe de altă parte, obiectele individuale acceptă culori de prim-plan şi de fundal, precum şi o schemă de culori şi o sursă de culori.

Atribuirea valorii 0 proprietăţii **ColorScheme** comunică programului Visual FoxPro să utilizeze celelalte proprietăţi standard referitoare la culoare (**ForeColor** şi **BackColor**). De asemenea, această proprietate acceptă valori între 1 şi 24 pentru a asigura compatibilitatea cu schemele de culori ale versiunilor FoxPro 2.x. Pentru a stabili o schemă de culori, trebuie să folosiţi comanda SET COLOR SET, care caută setul de culori în fişierul curent de resurse.

Proprietatea ColorSource are patru valori posibile, după cum se poate vedea în tabelul 114.

Tabelul 6.4

Valoare	Utilizare
0	Utilizează proprietăţile obiectului referitoare la culoare (BackColor şi ForeColor)
1	Utilizează proprietăţile părintelui obiectului referitoare la culoare
2	Utilizează schema de culori a obiectului
3	Utilizează proprietăţile prestabilite ale obiectului din schema de culori

6.14 Lucrul cu mai multe formulare active

Puteți lucra cu formulare active multiple prin gruparea lor într-un obiect de grup numit *set de formulare* (**Form Set**). Avantajele utilizării seturilor de formulare față de apelarea formularelor individuale cu instrucțiuni DO FORM independente sunt următoarele:

- Sincronizarea indicatorilor de înregistrări utilizați de fiecare formular atunci când definiți mediul de date la nivelul setului de formulare.
- Posibilitatea de a afişa (Show) şi de a ascunde (Hide) toate formularele simultan.
- Posibilitatea de a aranja vizual pe ecran formularele multiple în locul stabilirii pozițiilor prin încercare şi eroare.

Pentru a crea un set de formulare, deschideți sau creați primul formular pe care intenționați să-l includeți în set. Odată activat, instrumentul **Form Designer** adaugă meniul derulant Form în meniul principal. Una dintre opțiunile acestui meniu este **Create Form Set**.

Selectarea comenzii **Form**, **Create Form Set** duce la adăugarea automată a formularului în set. Pentru a adăuga formulare suplimentare, alegeți **Add New Form** din meniul Form. Această comandă deschide un formular gol.

6.14.1 Referirea obiectelor situate în alt formular al setului de formulare

Se poate stabili proprietatea unui obiect din formularul curent prin preluarea proprietății din alt formular al setului, folosind o instrucțiune de genul următor:

ThisForm.Label.Caption = ThisFormSet.Form2.Text3.Value

Această expresie preia proprietatea **Value** a unei casete de text numite Text3, aparținând unui formular numit Form2 din setul curent de formulare, apoi o atribuie proprietății **Caption** a unei etichete numite Labell din formularul curent.

Tot la fel, puteți configura proprietatea oricărui obiect aparținând unui alt formular dintr-un set de forrnulare, folosind instrucțiuni de genul următor:

ThisFormSet.Form2.Label2.Caption = 'Angajat'

sau

ThisFormSet.Form2.Label2.Caption = ThisForm.Text3.Value

Prima dintre aceste două expresii atribuie un şir invariabil proprietății Caption a unei etichete aparținând formularului Form2 din setul curent de formulare. A doua expresie foloseşte valoarea dintr-o casetă de text numită Text3 aparținând formularului curent pentru a actualiza proprietatea Caption a aceleiaşi etichete.

Este simplu să transferați date dintr-un formular în altul în cadrul unui set de formulare, atât timp cât aveți în vedere ierarhia numelor utilizate pentru a face referire la fiecare obiect.

6.14.2 Transferul controlului între formulare

Atunci când rulați un set de formulare, Visual FoxPro utilizează ordinea în care au fost definite obiectele fiecărui formular drept secvență prestabilită de parcurgere cu tasta Tab. De asemenea, foloseşte ordinea în care formularele au fost adăugate în setul de formulare pentru a defini ordinea de parcurgere a formularelor distincte cu ajutorul tastei Tab. Atunci când utilizatorul completează toate obiectele primului formuiar, Visual FoxPro transferă focalizarea în următorul formular şi aşa mai departe. Atunci când utilizatorul ajunge la ultimul obiect al ultimului formular, focalizarea este transferată ciclic primului obiect. Acest proces continuă la infinit, cu excepția cazului în care proprietatea **TerminateRead** a unui formular are valoarea .T. sau dacă este întâlnită o comandă de genul:

ThisFormSet.Release

Această comandă elimină toate formularele setului de formulare.

6.14.3 Gestionarea formularelor

Proprietatea *WindowType* a setului de formulare poate avea una dintre cele patru valori enumerate în tabelul 12.12.

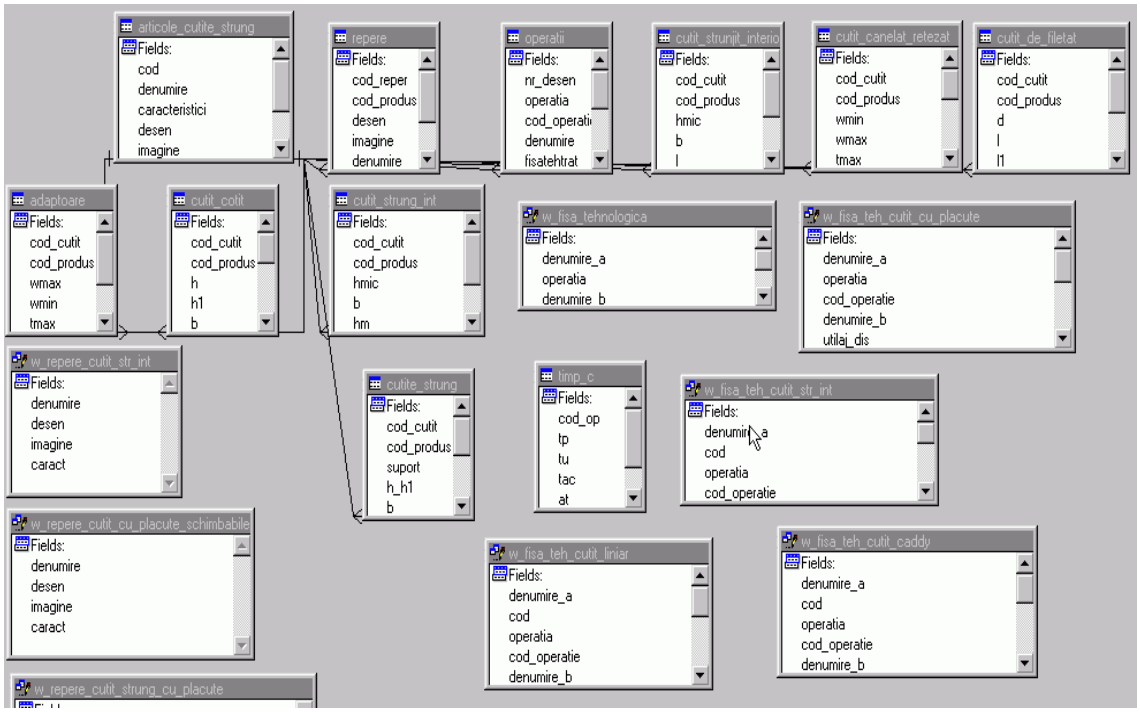
Tabelul 6.9 Valorile proprietății WindowType pentru seturile de formulare

Valoare	Utilizare
0	**Modeless**. Utilizatorul poate selecta orice obiect din orice formular al setului de formulare. De asemenea, poate selecta din meniu sau din orice alt formular activ în momentul respectiv. Această opțiune permite utilizatorului un maximum de control asupra aplicației
1	**Modal**. Utilizatorul poate selecta orice obiect din orice formular, cu condiția ca acesta să facă parte din setul de formulare. Nu este posibilă selectarea din meniu sau din alte formulare active în momentul respectiv.
2	**Read.** Execuția se opreşte la comanda care activează setul de formulare. Execuția programului se reia de îndată ce utilizatorul închide setul de formulare.

6.15 Exemplu de realizare a unei aplicații folosind *form-uri*.

Se consideră baza de date ***CUȚITE.*** Se va realiza o aplicație prin care vor fi vizualizate datele despre cuțite, reperele care intră pe acestea precum şi operațiile aferente prelucrării unui anumit tip de cuțit. Totodată se vor elabora documente economice: document de “lansare”, document “bon de material”

Baza de date CUTITE



Principalele fişiere de date sunt:

Fişierul de date Articole_cutite

Cod	Denumire	Caracteristici	Desen	Imagine
CSG1R/L5002.1	CUTIT DE STRUNG CU PLACUTE DIN CARBURI METALICE SCHIMBABILE	Gen	Gen	Gen
CSG1R/L5004.1	cutit de strung cu placute din carburi metalice schimbabile	Gen	Gen	Gen
CSG1R/L5005.1	cutit de strung cu placute din carburi metalice schimbabile	Gen	Gen	Gen
CSG1R5006.1	cutit de strung cu placute din carburi metalice schimbabile	Gen	Gen	Gen
CSG1R5003.2	cutit de strung cu placute din carburi metalice schimbabile	Gen	Gen	Gen
CSG1R5004.2	CUTIT DE STRUNG CU PLACUTE DIN CARBURI METALICE	Gen	Gen	Gen
CSG2R/L5023.1	cutite pentru strunguri automate si revolver cu placute schimbabile din carburi metalice	Gen	Gen	Gen
CSG2R/L5024.1	cutite pentru strunguri de copiat cu placute schimbabile din carburi metalice	Gen	Gen	Gen
CSG2R/L5026.1	cutite de strung pentru interior cu placute schimbabile din carburi metalice	Gen	Gen	gen
CSG2R/L5013.3	cutite pentru strunguri automate si revolver cu placute schimbabile din carburi metalice	Gen	Gen	Gen
CSG75D	cutit de strung drept	Gen	Gen	Gen
CSG30F	cutit de strung frontal	Gen	Gen	Gen

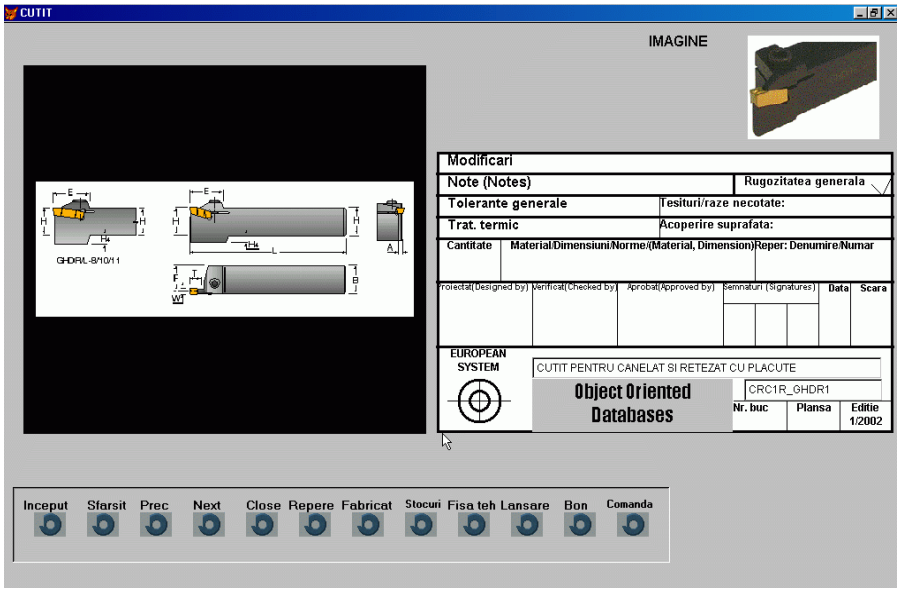
Fişierul de date Repere

Cod_reper	Cod_produs	Desen	Imagine	Denumire	Codul
SP_CSG1R5006.1	CSG1R5006.1	Gen	gen	suport	Gen
BR_CSG1R5006.1	CSG1R5006.1	Gen	gen	brida	Gen
SU_CSG1R5006.1	CSG1R5006.1	Gen	gen	surub	Gen
PS_CSG1R5006.1	CSG1R5006.1	Gen	gen	placuta de sprijin	Gen
ST_CSG1R5006.1	CSG1R5006.1	Gen	gen	stift	Gen
CH_CSG1R5006.1	CSG1R5006.1	Gen	gen	cheie hexagonala	Gen
SP_CSG1R/L5004.1	CSG1R/L5004.1	Gen	gen	suport	Gen
BR_CSG1R/L5004.1	CSG1R/L5004.1	Gen	gen	brida	Gen
SU_CSG1R/L5004.1	CSG1R/L5004.1	Gen	gen	surub	Gen
PS_CSG1R/L5004.1	CSG1R/L5004.1	Gen	gen	placuta de sprijin	Gen

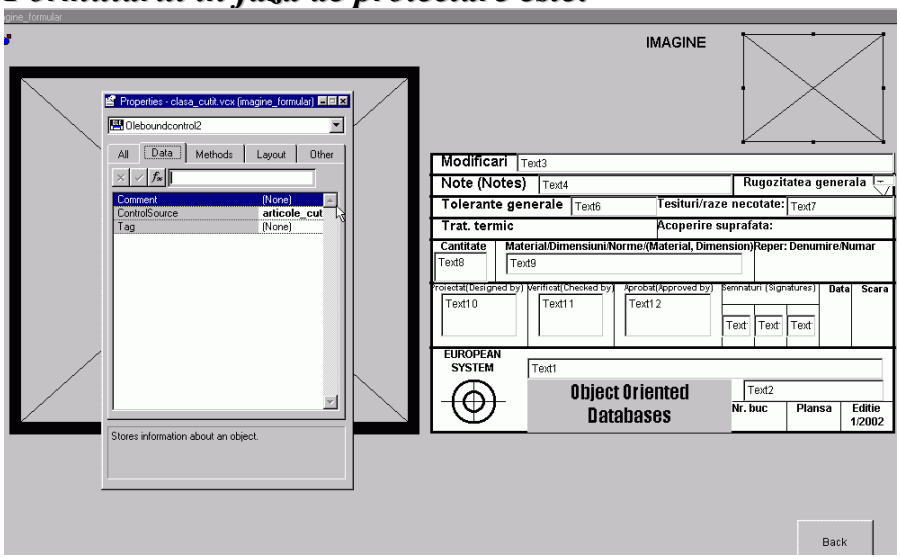
Fişierul de date Operaţii

Nr_desen	Operatia	Codul operatiei	Denumirea operatiei	Fisa tehnolog	Fisa tehnol	Scule necesar	Utilaj disponibil	Desen
CSG1R/L5002.1	1	6193	ascutirea corpului cutitului pe fata de asezare principala	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	2	6193	ascutirea corpului cutitului pe fata de asezare secundara	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	5	6193	ascutirea placutei de carburi metalice pe fata de asezare principala	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	6	6193	ascutirea fetei de asezare secundara	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	7	6193	ascutirea fetei de degajare	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	8	6193	ascutirea fatetei pe fata de degajare	memo	memo	Piatra abraziva 1	3010	Gen
CSG1R/L5002.1	10	5080	rotunjirea virfului cutitului r>= 1mm	memo	memo	Piatra abraziva 1	40500	Gen
CSG1R/L5002.1	15	4030	netezirea cutitului pe fata de asezare principala	memo	memo	Disc de fonta cu	3030	Gen
CSG1R/L5002.1	16	4030	netezirea fetei de asezare secundara	memo	memo	Disc de fonta cu	3030	Gen
CSG1R/L5002.1	17	4030	netezirea fatetei pefeta de degajare	memo	memo	Disc de fonta cu	3030	Gen
CSG1R/L5002.1	20	4030	netezirea virfului cutitului r>= 1mm	memo	memo	Disc de fonta cu	3030	Gen
CUSTRINTSL1299	10	7905	atentie respectati NTS	memo	memo		90500	gen
CUSTRINTSL1299	20	1530	debitat pe fereastra alter	memo	memo		100	gen
CUSTRINTSL1299	30	7621	lacatuserie sculer trasat ajustat dupa lipire	memo	memo		90270	gen
CUSTRINTSL1299	40	3018	strunjit la cote conf desen	memo	memo		2020	gen

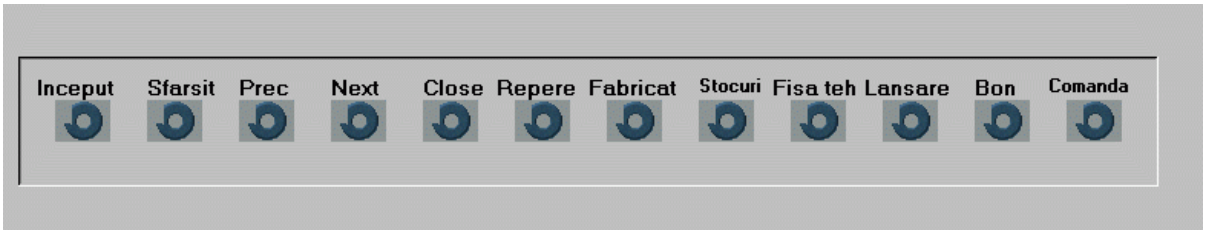
Formularul pentru lansarea în execuţie a aplicaţiei este:



Formularul în faza de proiectare este:



Cu ajutorul butoanelor se realizează mişcarea în interiorul aplicaţei



La execuția unui click pe unul dintre butoanele inceput, sfârşit, prec, next, close se realizează deplasarea înainte, înapoi, la sfârşit, respectiv început de fişier şi se părăseşte aplicația. Totodată butoanele permit vizualizarea reperelor a fişei tehnologice, scrierea unui formular de lansare, bon de material sau comanda unei cuțit

Procedura de poziționare la începutul fişierului

```
GO TOP
THISFORM.REFRESH()
```

Procedura de poziționare la sfârşitul fişierului

```
SKIP 1
IF eof()
    ??chr(7)
    WAIT WINDOW nowait "la sfarsit de fisier"
    GO bottom
ENDIF
    THISFORM.REFRESH()
GO bottom
THISFORM.REFRESH()
```

Procedura de poziționare pe înregistrarea următoare

```
SKIP 1
IF eof()
    ??chr(7)
    WAIT WINDOW nowait "la sfarsit de fisier"
    GO bottom
endif
    THISFORM.REFRESH()
```

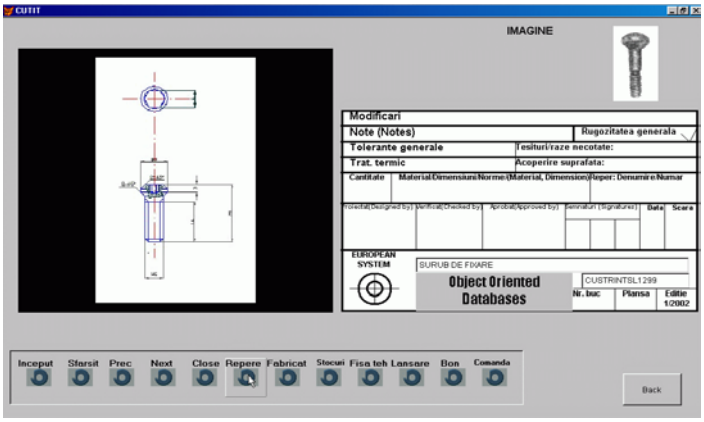
Procedura de poziționare pe înregistrarea precedenta

```
SKIP -1
IF bof()
    ??chr(7)
    WAIT WINDOW nowait "la inceput de fisier"
    GO top
ENDIF
    THISFORM.REFRESH()
```

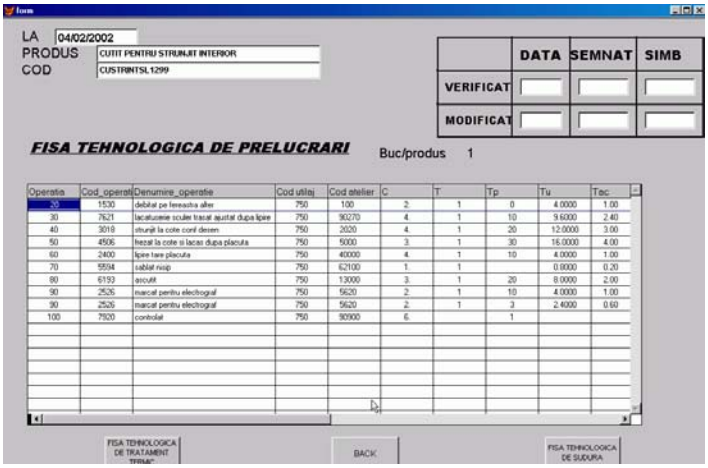
Procedura de părăsire a aplicației

```
RELEASE classlib flatbtn
RELEASE classlib clasa_cutit
RELEASE all
CLEAR EVENTS
CLOSE ALL
```

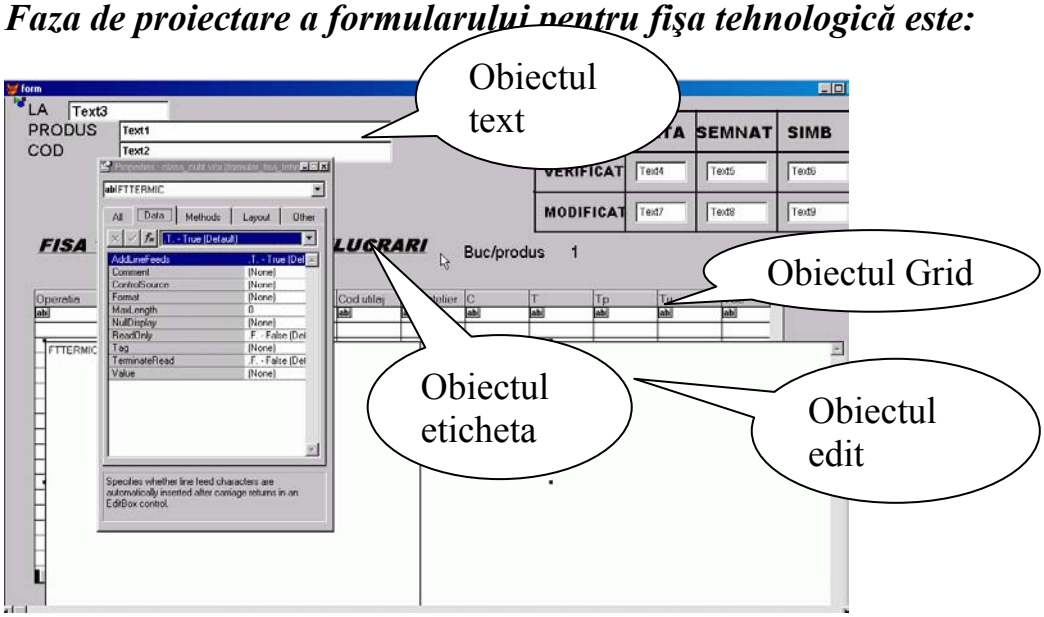
Formularul pentru prezentarea reperelor este:



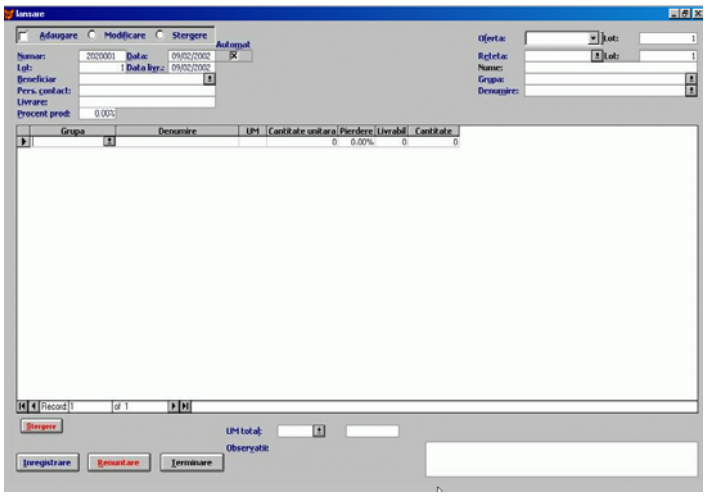
Formularul pentru fisa tehnologica este:



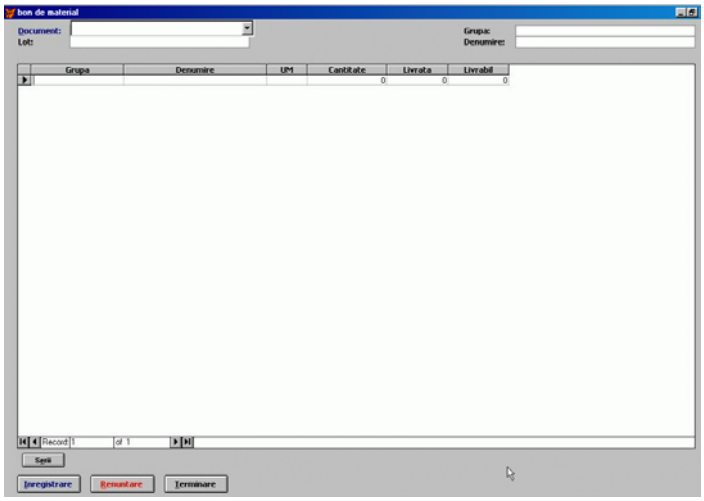
Faza de proiectare a formularului pentru fişa tehnologică este:



Formularul de lansare



Formularul pentru bon material



Concluzii

Formularele sunt entități foarte importante în mediile software vizuale. Ele permit realizarea relativ uşoară a unei interfețe pentru o aplicație utilizator, folosind obiectele şi clasele predefinite. Un formular poate exista ca o entitate separată putând fi lansat în execuție prin comanda: DO FORM. Într-un formular pot exista mulțimi de formulare, astfel încât se pot realiza aplicații complexe numai din formulare. Formularul se poate salva sub forma unei clase de obiecte, lucrul important din punct de vedere al programării orientate obiect.

Capitolul

7

GENERATORUL DE RAPOARTE

Rapoartele sunt programe care afişează pe ecranul moitorului sau la imprimantă, diverse informaţii obţinute din unul sau mai multe tabele. Strctura generală a unui raport cuprinde trei etape:

- de preluare a parametrilor necesari construirii raportului;
- de prelucrare a datelor din bazele de date, adică de extragere a datelor din baza de date şi prelucrarea lor într-o formă cât mai apropiată de cea a raportului, de obicei se crează o tabelă temporară sau o interogare pe baza căreia este construit raportul;
- de prezentare exterioară a raportului, datele fiind aranjate conform cerinţelor utilizatorului.

În VisualFoxPro există două metode de a realiza rapoarte:

1. metoda clasică prin care se construiesc programe de listare sau vizualizare folosind instrucţiuni de intrare ieşire Fox;
2. folosind generatorul de rapoarte cu ajutorul căruia se realizează raportul într-un fişier cu extensia .FRX.

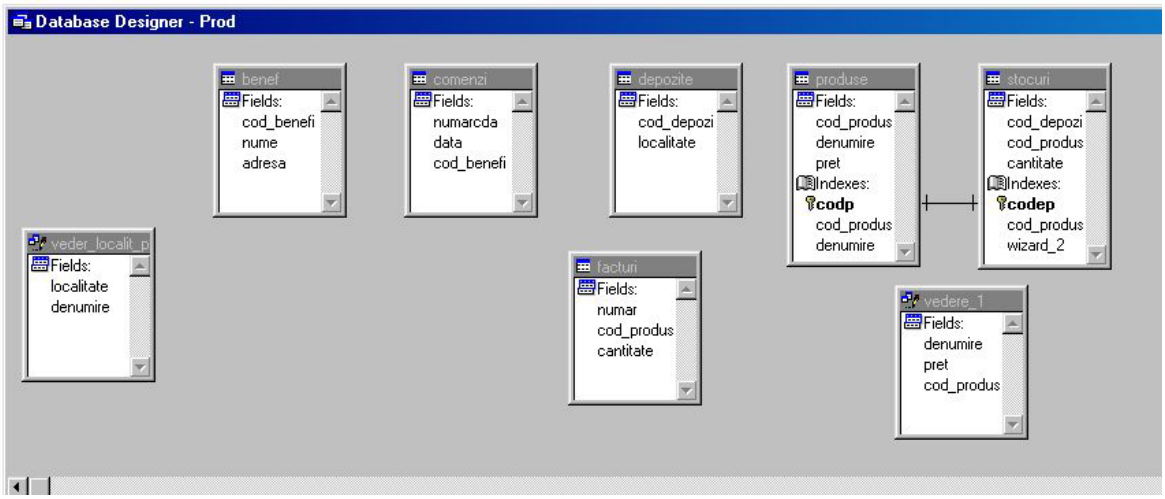
În general un raport poate fi construit pe lângă o formă dar în acest caz aceasta trebuie să conţină un buton prin care este apelat raportul. Un raport conţine datele existente în una sau mai multe tabele.

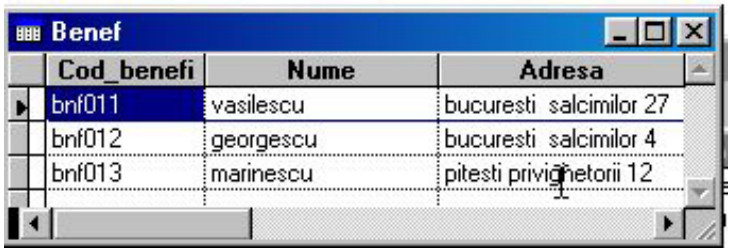
Folosind generatorul de rapoarte, un raport poate fi construit în două moduri şi anume folosind modul Design sau utilitarul Wizard. În cazul în care se foloseşte utilitarul Wizard se pot construi trei tipuri derapoarte şi anume:

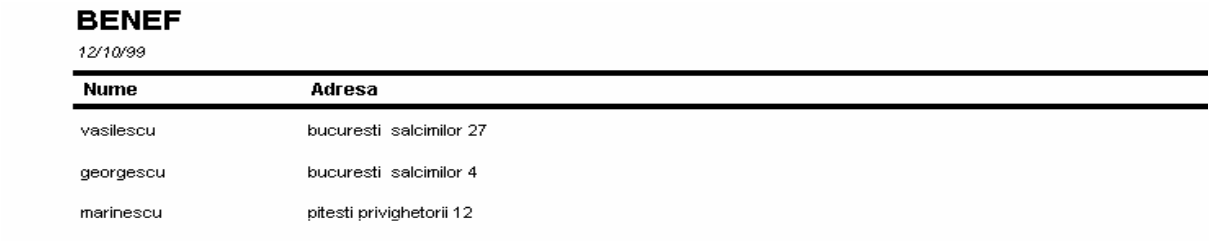
- raport totalizator, care conţine câmpuri totalizatoare sau subtotalizatoare;
- raport one to many în care se alege o tabelă părinte şi una sau mai multe tabele copil;
- raport clasic în care se prezintă datele dintr-o tabelă.

7.1 Structura unui raport

Structura unui raport poate diferi foarte mult de la caz la caz. Un exemplu este raportul wizard pe fişierul Benef (figura 7.1) din baza de date Prod.







BENEF

12/10/99

Nume	Adresa
vasilescu	bucuresti salcimilor 27
georgescu	bucuresti salcimilor 4
marinescu	pitesti privighetorii 12

Fig 7.1 Raport Wizard pe o bază de date

Raportul conține un titlu, un cap de tabel (nume, adresa) şi informațiile existente în tabelă. Fizic raportul există sub forma unui fişier cu extensia .FRX. Structura raportului este prezentată în figura 7.2:

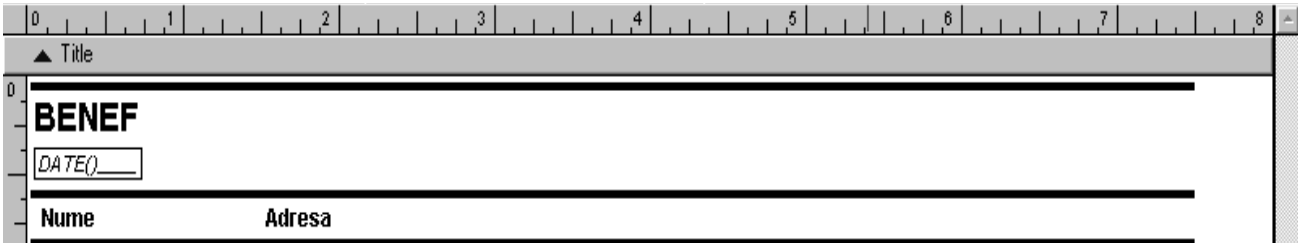


Fig. 7.2

Nume şi adresa sunt câmpuri alese din fişierul Benef, sub titlu este pusă implicit data curentă iar în zona footer este pus numărul de pagină. În general un raport conține trei zone:

- ***Page Header*** unde este introdus titlul raportului, şi capul de tabel. Titlul raportului poate apare doar pe prima pagină;
- ***Zona Detail*** în care sunt introduse informațiile din tabele'

- ***Page Footer*** în care sunt introduse informații care se vor afla în partea de jos a paginii.

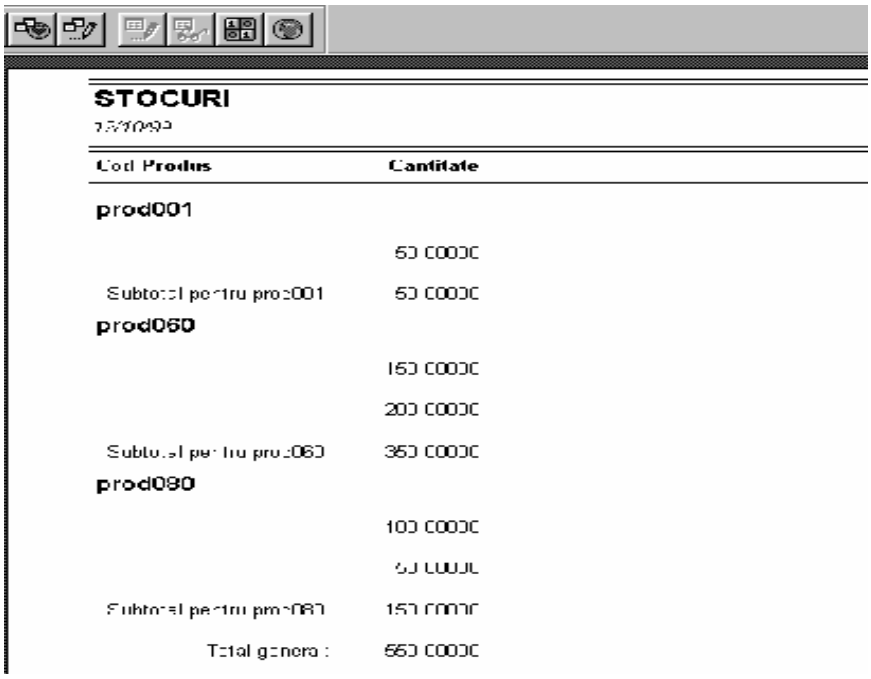


Fig. 7.3 Raport totalizator

Un raport totalizator este prezentat în figura 7.3. El a fost realizat pe baza tabelei STOCURI fig. 7.3_1 din baza de date Prod (figura 7.1).
Structura unui astfel de raport este prezentată în figura 7.4

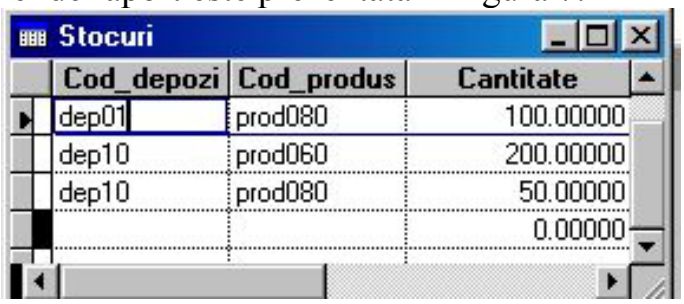


Fig 7.3_1 Datele din tabela Stocuri

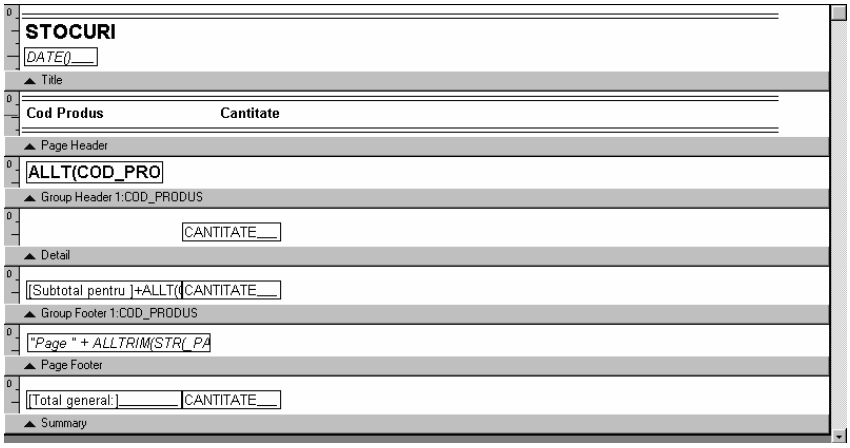


Fig 7.4 Structura raportului din tabela Stocuri

În cazul unui raport totalizator vor apare secţiuni noi, datorită faptului că de data aceasta informaţiile trebuie grupate pentru a se obţine situaţii totalizatoare. Gruparea s-a făcut după câmpul cod_produs şi s-a obţinut un subtotal pe câmpul cantitate pentru fiecare produs şi pentru toate produsele.
Comanda de creare a unui raport este CREATE REPORT <nume raport> sau din meniul FILE->NEW-. REPORT din fereastra New care apare.
Efectul este apariţia pe ecran a ferestrei din figura 7.5

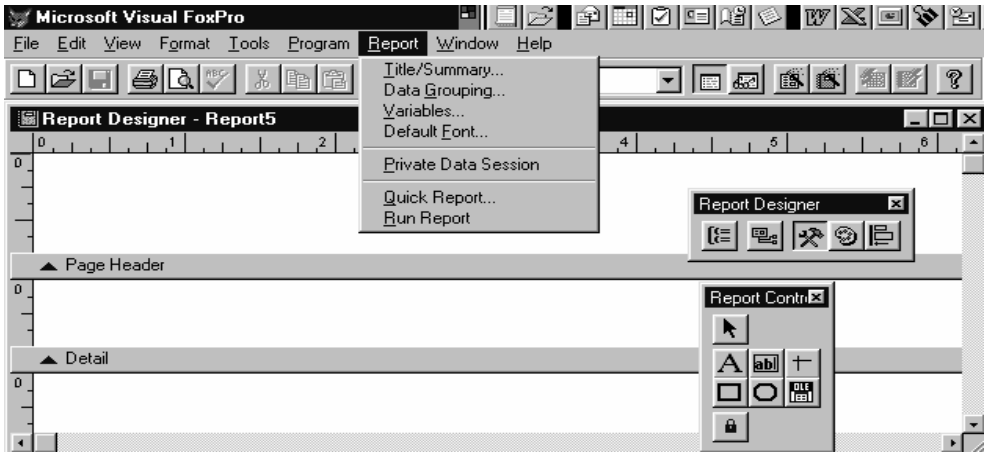


Fig. 7.5 Proiectarea raportului

După cum se observă din figură raportul construit cu Constructorul de rapoarte este format din trei benzi şi anume:

- Antet de pagină (**Page Hader**);
- Detalii (**Detail**);
- Subsolul paginii (**Page Footer**).

Fiecare bandă va genera în raportul afişat mai multe instanţe ale sale sau obiecte ale sale. Antetul de pagină va conţîne instanţe care vor apare pe fiecare pagină. Ea are atâtea instanţe câte pagini are raportul.

În Visual FoxPro benzile unui raport sunt următoarele:

- **Detail** este banda de bază, ea conţine informaţiile din tabelă;
- **Title** conţine titlul raportului care va apare doar pe prima pagină şi eventual capul de tabel dacă acesta nu trebuie să apară pe fiecare pagină.
- **Summary** sau rezumat se foloseşte pentru a specifica zona de sfârşit a raportului şi apare o singură dată pe ultima pagină a raportului şi poate conţine informaţii referitoare la totaluri grupate pe anumite informţii;
- **Page Header** specifică textul care apare pe fiecare pagină;
- **Page Footer** specifică tectul care apre la subsolul fiecărei pagini;
- **Group Header n** antetul grupului de nivel n şi **Group Footer n** subsolul grupului de nivel n apar datorită faptului că fiecare criteriu de grupare generează mai multe grupuri în raportul final, funcţie de valorile cheii de grupare. La începutul fiecărui grup va pare antetul iar la sfârşit subsolul de grup;

- **Column Header** antetul de coloană şi **Column Footer** subsolul de coloană araqtă faptul că la nivel de coloană poate fi specificat un antet şi un subsol

Aşa cum se vede din figura 7.5 nu toate benzile apar la pornirea constructorulu de rapoarte dar ele pot fi generate ulterior. La limită o bandă poate avea înălțimea zero, ceea ce înseamnă că ea nu va apare în raportul generat. Banda de titlu şi de concluzii (Summary) sunt generate prin activarea opțiunii **Title/ Summary din meniul Report** (bara de titlu).

7.2 Proprietățile benzilor raportului

Dacă se execută dublu clic pe o anumită bandă va apare o fereastră corespunzătoare fig 7.6

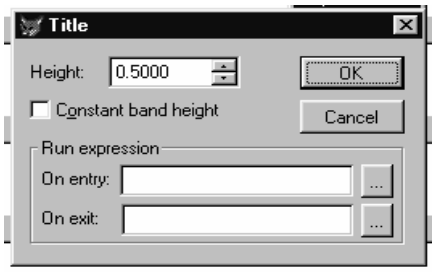


Fig. 7.6 Proiectarea unei benzi

- Dimensiunea benzii este stabilită de către caseta **Height.**
- Dacă se doreşte ca toate instanțele benzii să aibă aceaşi dimensiune se poate activa butonul **Constant band Height**;
- **On entry** permite evaluarea unei anumite expresii înainte de afişarea benzii;
- **On exit** determină evaluarea unei expresii după afişarea benzii**.**

7.3 Proprietățile în ansamblu ale unui raport

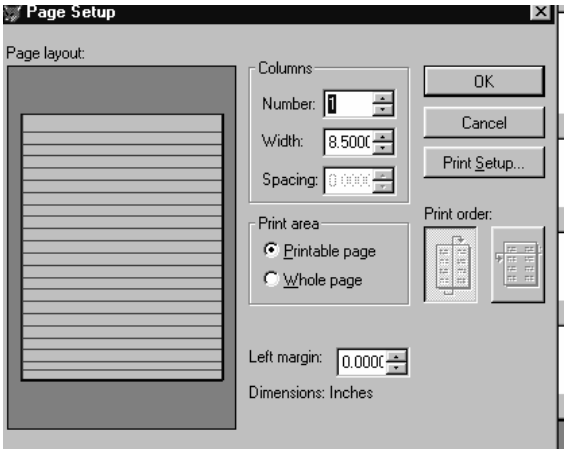


Fig. 7.7 Opțiunea Page Setup

Setarea caracteristicilor paginii raportului se poate face activând opțiunea **Page Setup** din meniul **File** figura 7.7

- **Number** defineşte numărul de coloane;
- **Width** lătimea unei coloane;
- **Spacing** distanța dintre coloane. În cazul în care avem o singură coloană atunci valoarea Spacing este inhibată;
- **Whole Page** permite alegerea întregii pagini fizice de hârtie pentru listare;
- **Printable Page** permite alegerea numai a zonei tipăribile a paginii;
- **Left Margin** defineşte un spațiu suplimentar care este rezervat pentru marginea din stânga a raportului

7.4 Mediul de date al raportului

Mediul de date al unui raport, se configurează în fereastra mediului deschisă prin alegerea opțiunii **Data Environment** a meniului **View**. Automat va apare în meniul principal opțiunea **Data Environment**. Folosind opțiunea **ADD** se adaugă tabele vederi sau interogări noi la raport. Acestea vor fi deschise automat în momentul în care este deschis raportul**. Proprietățile ferestrei Data Environment** se pot seta activând opțiunea **Properties** din meniul care apare dacă se execută click pe butonul din dreapta al mouse-ului, în timp ce acesta se găseşte în fereastră (fig. 7.8). Metodele permit apelarea dinamică a tabelelor în timpul rulării unui raport.

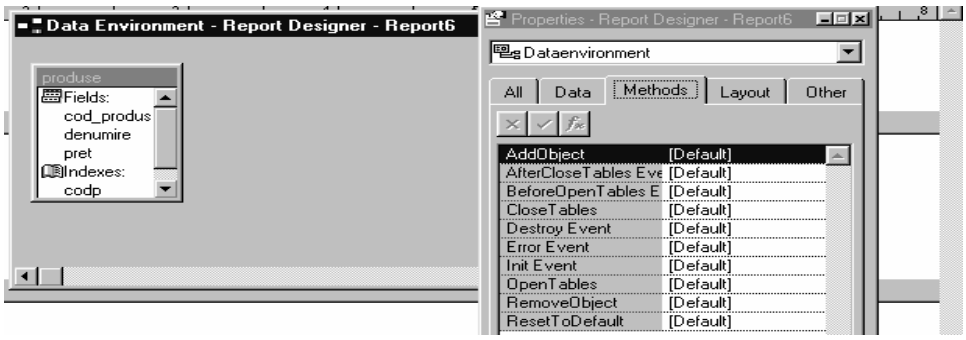


Fig. 7.8 Fereastra Data Environment

Principalele metode (Methods) sunt:

- **Before Open Tables, Open Tables**, stabilesc fişierele care vor fi deschise înainte de rularea raportului, comenzile pentru deschiderea efectivă a fişierelor şi selecta rea tabelei curente;
- **AddObject, Remove Obiect** permite adăugarea respectiv ştrgerea unui nou obiect la fereastra de stare;
- **Close Tables** închide efectiv tabelele specificate;
- **After Close Table** acțiunile care se intreprind după înhiderea tabelelor.

Opțiunea Data definește proprietățile mediului care sunt:

- **InitialSelectedAlias** stabileşte tabela selectată inițial;
- **AutoOpenTables** determină deschiderea automată a tabelelor specificate în mediul de date dacă are valoarea .T.;
- **AutoCloseTables** determină închiderea automată a tabelelor specificate în mediul de date al formei

În cazul în care se defineşte un mediu de date specific unui anumit raport este indicat să se definească o sesiune privată de lucru pentru respectivul raport alegând opțiunea **Private Data Session** a meniului **Report.**

7.5 Fontul implicit al raportului

O altă setare globală care se poate efectua asupra unui raport este alegerea fontului implicit, care se realizează din meniul **Report** opțiunea **Default Font**.

7.6 Controalele Raportului

Controalele care pot fi folosite într-un raport sunt active prin acționarea butonului respectiv de pe bara **Controls Report**. Butoanele din bara de stare **Controls Report** sunt prezentate în figura 7.9.

Fig 7. 9 Butoanele Controls Report

Butonul **etichetă** sau **Text** permite introducerea unui text explicativ în raport. Specific acestui tip de instanță este faptul că poate fi ales un anumit Font din meniuil Format sau din meniul care apare executând click pe butonul din dreapta al mouse-ului.

Butonull **Field** este unul dintre cele mai importante şi permite alegerea unor câmpuri din tabela de date pentru a fi vizualizate în raport. Proprietățile acestui obiect se pot defini în fereastra **Report Expression** care apare implicit atunci când se crează un obiect **Field** nou sau în cazul în care se execută dublu click pe un obiect câmp existent sau se alege opțiunea **Properties** din meniul ce apare prin click, pe butonul din dreapta cu mouse-ul poziționat pe obiect (fig 7.10)

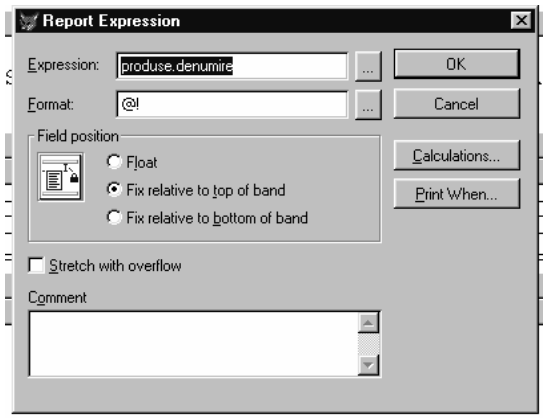


Fig. 7.10 Alegerea câmpurilor într-un raport

În caseta **Report expression** se defineşte câmpul dorit sau o expresie de orice tip admis de către mediul VisualFox. Poziția câmpului este determinată de către casetele:

- **Float** este poziția mobilă adică un obiect se poziționează în pagină funcție de obiectele definite deasupra lui;
- **Fix relative to top band** este poziție relativă la marginea superioară a benzii adică, poziția câmpului se calculează funcție de marginea superioară a benzii în cazul în care aceasta este variabilă;
- **Fix Relative to bottom of band** pozziția câmpului este considerată fixă dar ea se calculează relativ la marginea inferioară a benzii;

Comutatorul **Strech with overflow** determină extinderea unui obiect atât cât este necesar pentru a putea fi vizualizat

Butonul **Calculation** defineşte câmpul calculat Funcția după care se calculează poate fi:

- **Nothing** nu se calculează;
- **Count** în câmp se afişează numărul de Înregistrări ale tabelei
- **Sum** calculează suma valorilor dintr-un anumit câmp;
- **Average** calculează media aritmetică;
- **Lowest** calculează cea mai mică valoare;
- **Highest** calculează cea mai mare valoare.

În cazul în care datele sunt grupate momentul aducerii la 0 al unui câmp este stabilit cu ajutorul listei derulante **Reset** din fereastra **Calculate Field** şi pot fi la sfârşitul paginii, la sfârşitul raportului sau la sfârşitul coloanei în cazul în care raportul se derulează pe mai multe coloane.

Modul în care apare câmpul este definit în fereastra format (fig 7.11)

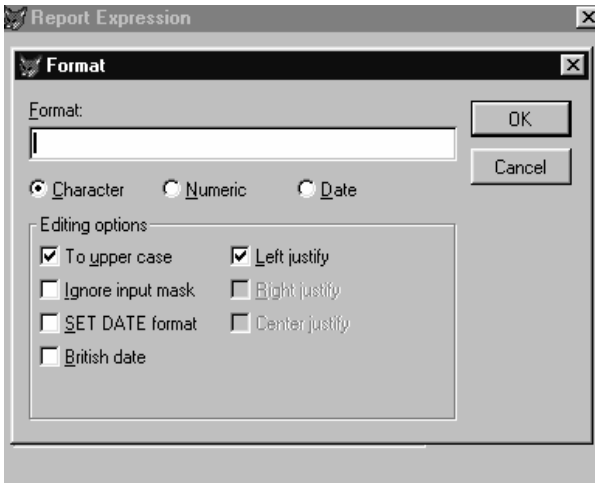


Fig. 7.11 Fereastra Format

În această fereastră se alege tipul datelor din câmp şi modul de tipărire al câmpului: cu litere mari, cu ignorarea măştii de intrare, cu setarea datei calendaristice, modul de aliniere etc.

7.7 Gruparea datelor în raport

Obținerea unor totaluri sau subtotaluri într-un raport este un lucru deosebit de important Într-un raport pot exista mai multe niveluri de grupare. Fiecare criteriu de grupare este caracterizat printr-o expresie, care se evaluează pentru fiecare instanță a benzii. Vor forma un grup acele instanțe care corespund aceleiaşi valori ale expresiei de grupare. Un nivel de grupare adăugat la raport aduce cu sine o bandă de antet de grup, afişată la începutul fiecărui grup de nivelul respectiv şi una de subsol de grup, afişată după fiecare grup de nivelul respectiv.

Exemplul 7.1

Să se creeze un raport care să returneze prețul tuturor produselor de acelaşi tip din fişierul Produse.dbf al bazei de date Prod. Dbc şi valoarea totală a acestora.

Gruparea datelor se realizează folosind opțiunea Data Grouping din meniul **Reset**. Lista **Group Expression** conține câte o linie pentru fiecare nivel de grupare (fig 7.12).

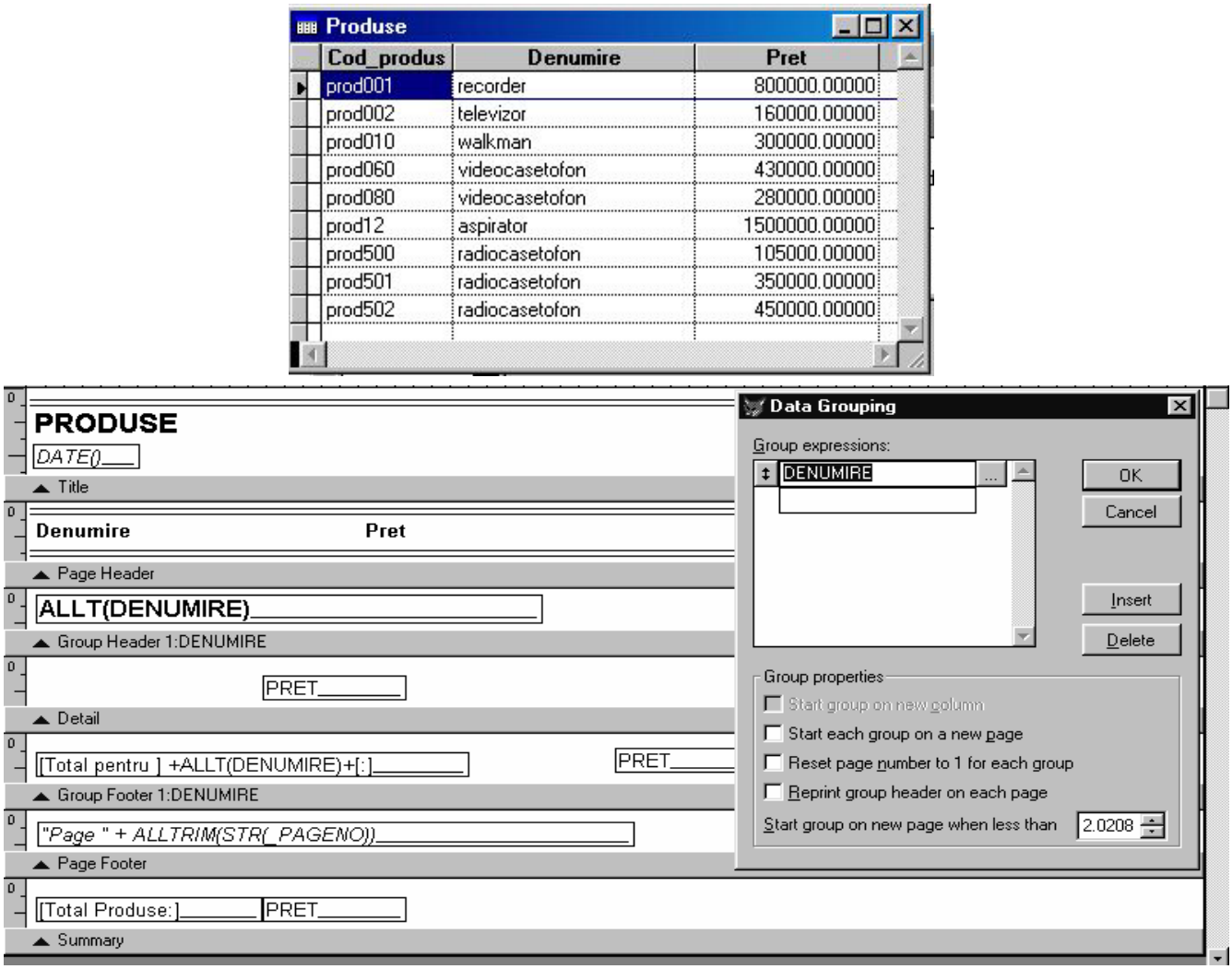


Fig 7.12 Fereastra de grupare Data Grouping

Se alege câmpul DENUMIRE ca şi criteriu de grupare. Inserarea unui nou criteriu de grupare se poate realiza folosind butonul Insert. Ştergerea unui criteriu de grupare se realizează folosind butonul Delete. Funcție de dorința utilizatorului se poate lista fiecare grup pe pagină nouă, se poate reseta numărul de pagină la valoarea 1 după fiecare grup sau se poate lista antetul de grup pe fiecare pagină. Subtotalurile , adică totalurile pentru fiecare denumire de produs se calculează în câmpul PRET din banda **Group Footer** (fig 7.13)

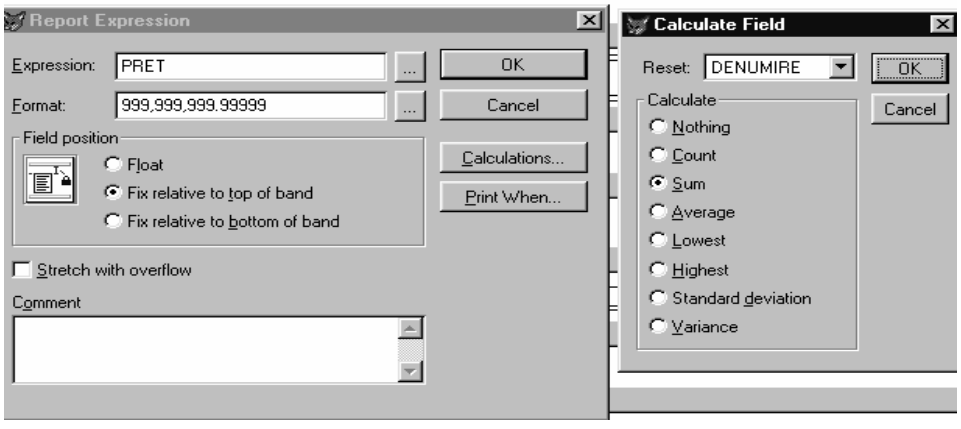


Fig. 7.13 Fereastra Calculate field

În caseta **Reset** se introduce numele câmpului de grupare după care se va face totalul. Se va folosi funcția SUM(). În câmpul pentru totalul general, din banda Summary, în caseta Reset, din fereastra Calculation se va folosi opțiunea END OF REPORT.

Structura raportului este prezentată în figura 7.14 iar rezultatele execuției raportului în figurile 7.15,7.16.

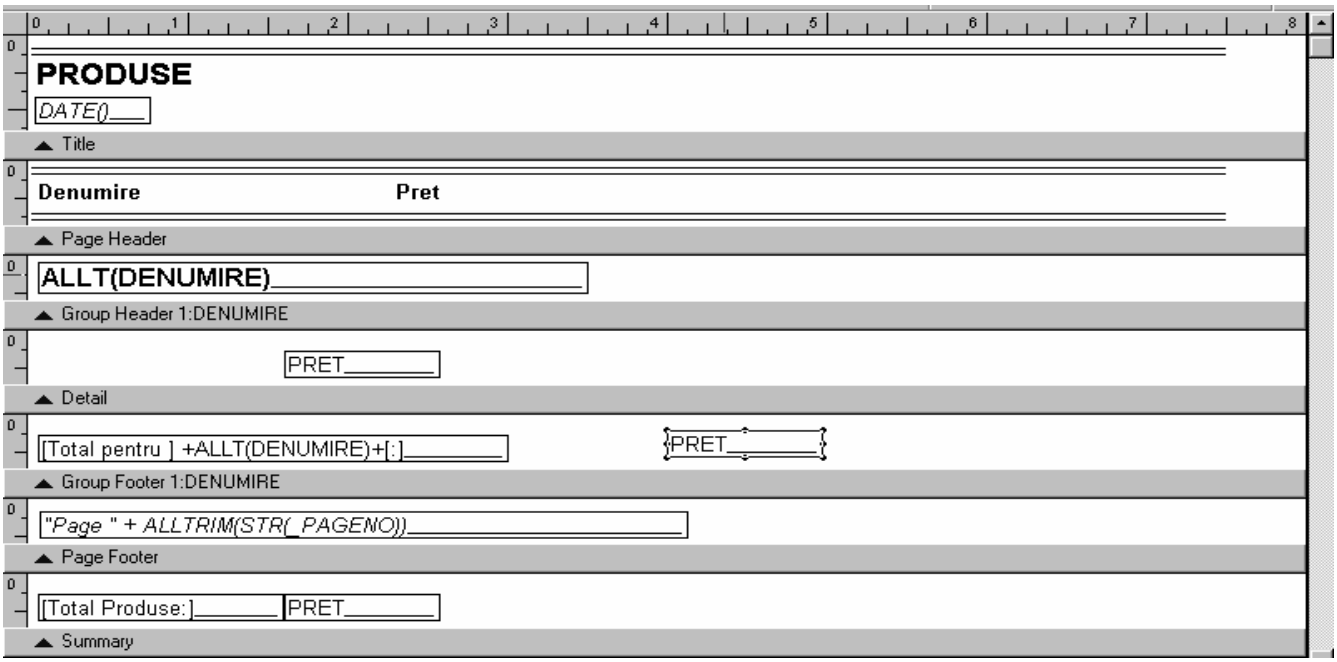


Fig 7.14 Structura Raportului

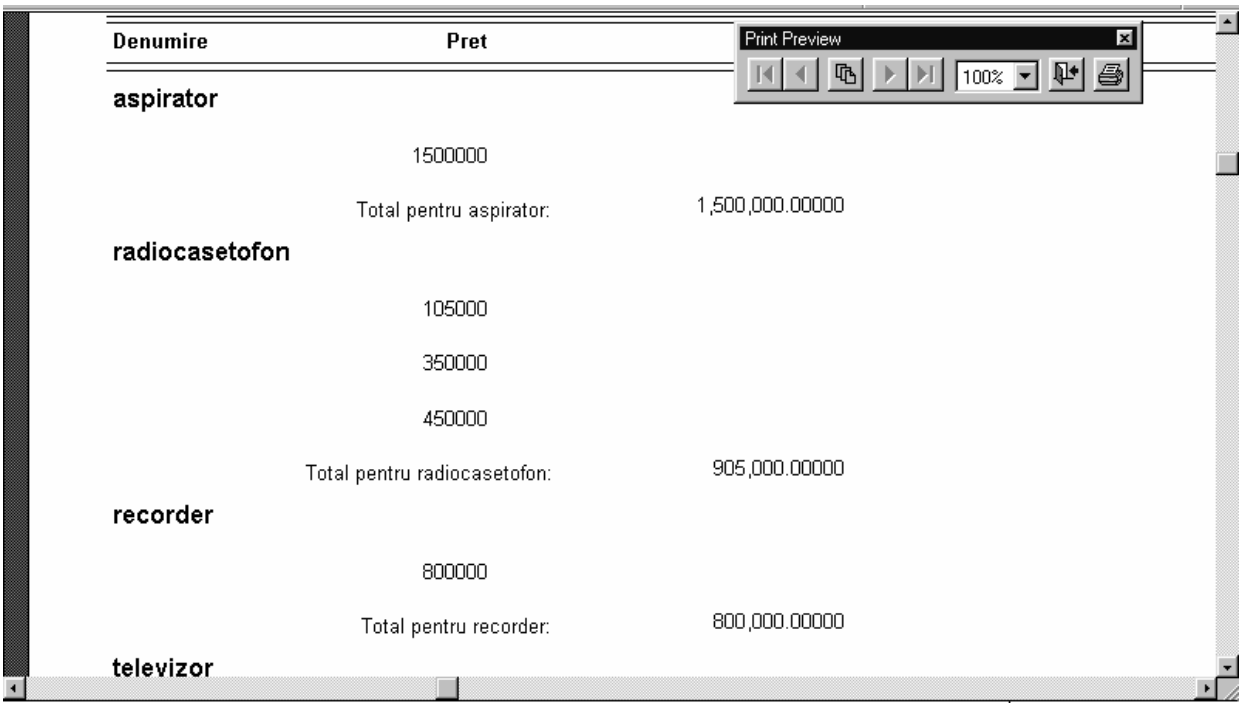


Fig 7.15 Raportul în faza de execuție

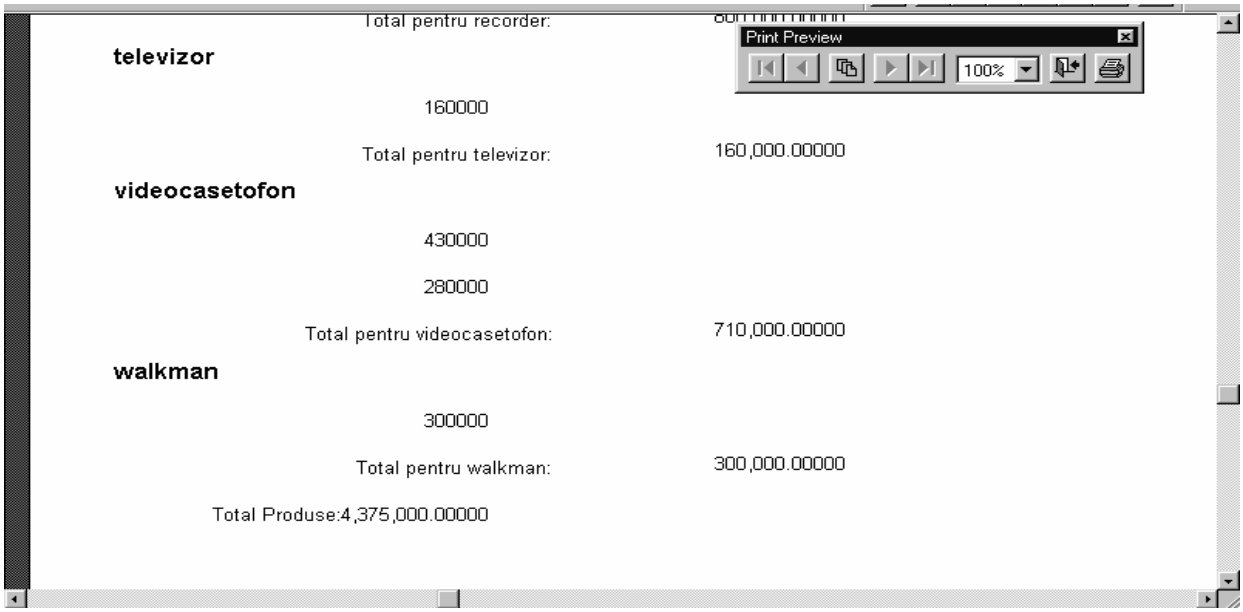


Fig 7.16 Raportul în faza de execuție

7.8 Folosirea variabilelor în construirea rapoartelor

În cadrul unui raport poate fi necesară folosirea variabilelor. De exemplu în cazul în care este calculată o sumă şi aceasta va fi folosită de mai multe ori în tr-un raport, este de preferat ca această sumă să apară într-uo variabilă. Definirea unei variabile se realizează în fereastra Report Variables obținută prin alegerea opțiunii Variabiles a meniului Report.

Exemplul 7.2
Să se realizeze un raport pe fişierul Stocuri.dbf din baza de date Prod.dbf, în care

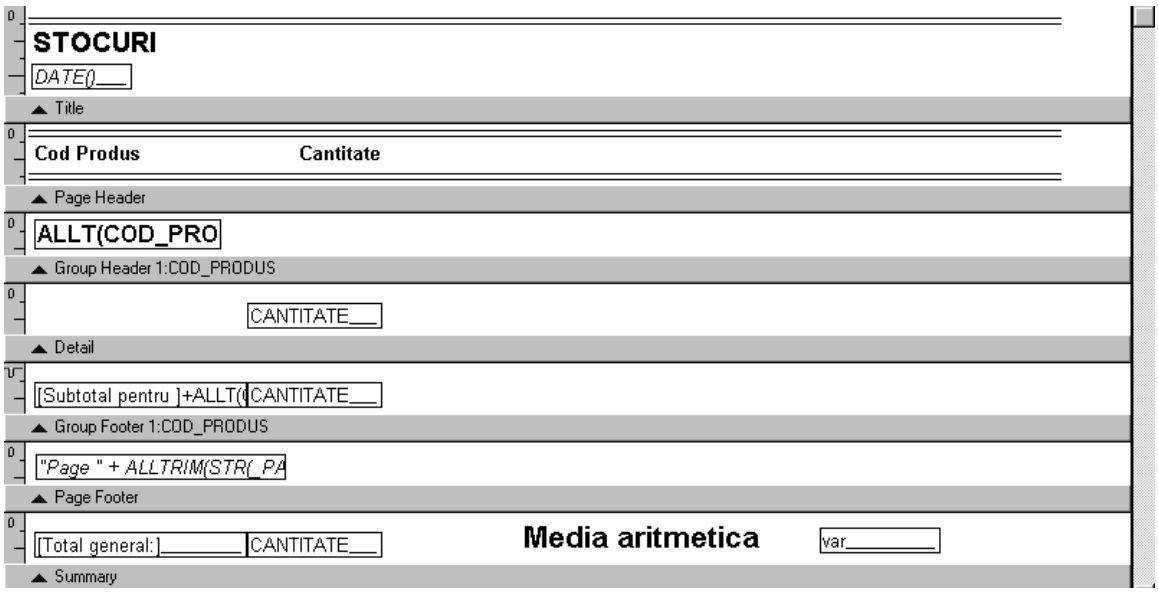


Fig. 7.17 Structura raportului pe fişierul Stocuri

să se vizualizeze cantitatea totală pe produs şi cantitatea totală de produse existentă în

fişier. Totodată să se vizualizeze cantitatea medie pe articol existentă în fişier figura 7.17.

Gruparea înregistrărilor se face după cod produs. Pentru a afişa media pe articole se alege o variabilă cu numele VAR figura 7.18

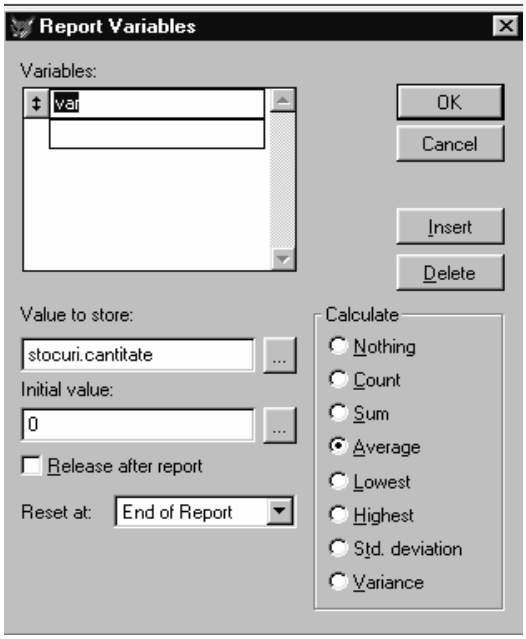


Fig 7.18 Fereastra de definire a variabilelor

Se va alege funcţia **AVG** pe câmpul cantitate. Explicarea câmpurilor se va face din figură.

Pentru a afişa un raport pe ecran poate fi folosită comanda:

REPORT FROM <nume_raport> **PREVIEW**

sau poate folosi butonul cu acelaşi nume din fereastra **Report**. Rezultatele din

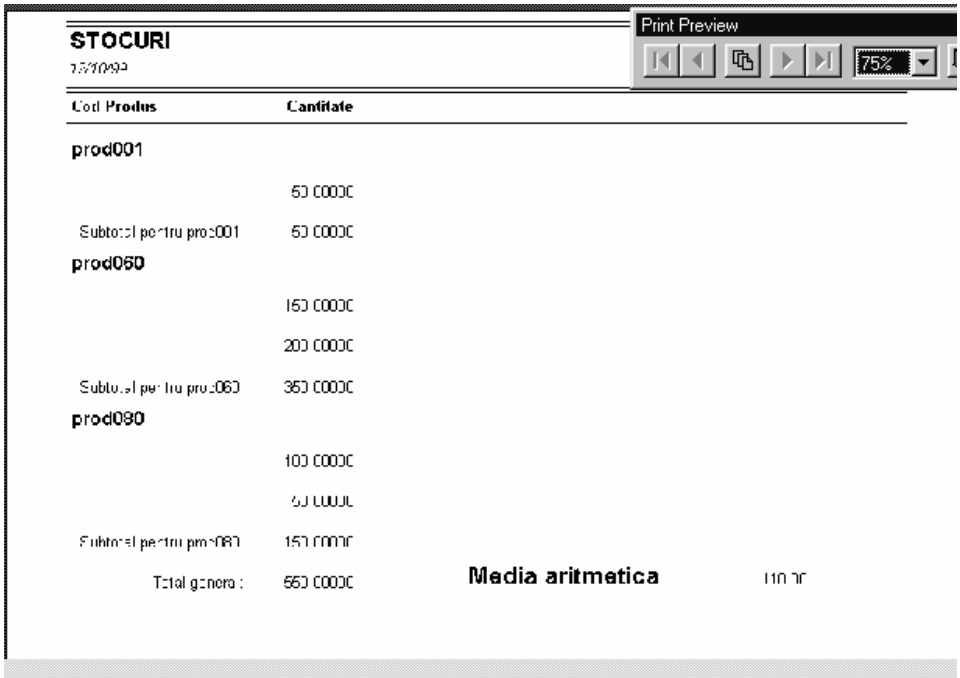


Fig. 7.19 Rezultatele raportului pe fişierul Stocuri

acest raport sunt ilustrate în figura 7.19

Elementele semigrafice linie, dreptunghi, elipse se vor folosi acționând butoanele cu acelaşi nume.

Pentru introducerea unei imagini se foloseşte tehnologia OLE de incorporare şi legare ale obiectelor. În fereastra care apare opțiunile:

- **Clip picture** păstrează dimensiunile originale ale imaginii;
- **Scale picture**, **retain shape** se redimensioneazaginea păstrându-se proporțiile inițiale astfel ca imaginea să încapă cât mai bine în cadru;
- **Scale picture**, **fill the frame** se redimensionează imaginea dar nu mai sunt păstrate proporțiile originale, imaginea fiind eventual deformată pentru a umple complet cadrul rezervat în raport.

Concluzii

În general orice aplicație are nevoie de programe prin care sunt vizualizate datele prelucrărilor, fie pe ecran, fie la imprimantă. Un instrument deosebit de util pentru realizarea acestui lucru este Genertorul de Rapoarte prin care poate fi realizată rapid şi corect în mod interactiv orice situație dorită. Practic cu Generatorul de Rapoate, se poate elimina scrierea de cod obiect în programele de listare a unei aplicații